МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
РАДИОЭЛЕКТРОНИКИ

РАДИОТЕХНИКА

Всеукраинский межведомственный научно-технический сборник

Тематический выпуск
«ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Основан в 1965 г.

ВЫПУСК 134

Харківський національний
університет радіоелектроніки

2003

УДК 621.3

Сборник включен в список специальных зданий ВАК Украины по физико-математическим и техническим наукам.

Регистрационное свидетельство КВ № 4486 от 22.08.2000 г.

Настоящий сборник посвящен решению ряда проблемных задач в сфере информационной безопасности. Основные статьи в этом направлении написаны сотрудниками кафедры Безопасности информационных технологий ХНУРЭ и АО «Институт информационных технологий». Обсуждается ряд противоречий, которые возникли в Украине, предлагаются методы, системы и средства их разрешения. Ряд статей носит методологический характер, в них делаются попытки определить направления развития этой очень динамичной области.

В сборнике представлены статьи по четырем направлениям:

- Проблемы теории и практики создания и развития перспективных систем защиты информации;
- Перспективные блочные шифры и их свойства;
- Методы, протоколы и средства криптографических преобразований;
- Системы и средства защиты информации.

Настоящий выпуск предназначен для специалистов в области защиты информации. Представленный материал может быть использован в учебном процессе по специальностям направления «Информационная безопасность».

Ответственность за содержание статей несут авторы.

Ответственный за выпуск д-р техн. наук *И.Д. Горбенко*

Рекомендовано Ученым советом Харьковского национального университета радиоэлектроники, протокол № 50 от 26.09.03.

Адрес редакционной коллегии: Харьковский национальный университет радиоэлектроники (ХНУРЭ), просп. Ленина, 14, Харьков, 61166, тел. (0572) 7021-397.

Выпуск подготовлен при поддержке и участии сотрудников АО «Институт информационных технологий».

СОДЕРЖАНИЕ ЗМІСТ

ПРОБЛЕМЫ ТЕОРИИ И ПРАКТИКИ СОЗДАНИЯ И РАЗВИТИЯ ПЕРСПЕКТИВНЫХ СИСТЕМ ЗАЩИТЫ ИНФОРМАЦИИ

ПЕРСПЕКТИВНЫЕ БЛОЧНЫЕ СИММЕТРИЧНЫЕ ШИФРЫ И ИХ СВОЙСТВА

МЕТОДЫ, ПРОТОКОЛЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

СИСТЕМЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

CONTENTS

ПРЕДИСЛОВИЕ

Уважаемые читатели, коллеги!

М. Бондаренко

Сегодня в Украине, как и во всем мире широко и интенсивно во все сферы жизни внедряются новейшие информационные технологии, информационно-телекоммуникационные системы, информационно-аналитические системы и системы автоматизированного управления. Анализируя состояние информационной безопасности в этих системах, можно прийти к выводу, что объектами защиты, которые вызывают наибольшее беспокойство и аккумулируют все проблемы информационной безопасности, являются различного рода информационно-телекоммуникационные системы и информационные технологии. Обратимся к конкретным фактам.

В соответствии с данными ФБР США в 2001 г. финансовые потери от компьютерного воровства составили около 400 млн. дол. По оценкам отдела науки и информационных технологий при президенте США ежегодные потери, которые наносятся американскому бизнесу компьютерными мошенниками, в 2001 году составили около 100 млрд. дол. Потери от несанкционированного доступа в том же году составили не менее 1 млрд. дол. И это несмотря на то, что США выделяют на решение задач обеспечения информационной безопасности наибольшие финансово-материальные ресурсы в мире. Так, только государственное финансирование исследований в этой области составит на 2003 – 2008 гг. более 1,5 млрд. дол.

И. Горбенко

В то же время число злоумышленных и мошеннических действий, например в компьютерных сетях, возрастает. В США их число возросло в 2002 г. в сравнении с 2001 г. с 58 до 80 тысяч. Большую угрозу составляют «политические хакеры», их необходимо отнести к наиболее опасным злоумышленникам. Жертвами политических атак стали правительственные сайты. Так в США за три последних года взломаны сайты Белого Дома, Пентагона и Госдепартамента. В соответствии с данными МВД Российской федерации в 1997 г. было зарегистрировано 309 компьютерных преступлений, в 1998 – более 500, и в последующем сохранилась тенденция их ежегодного увеличения на 30 %.

Существенные материально-технические и финансовые ресурсы направляются на создание информационного оружия и целой системы поддержания его боеготовности и совершенствования.

Сказанное вынуждает даже технологически среднеразвитые государства направлять значительные ресурсы и концентрировать усилия на обеспечении информационной

безопасности. Бесспорно, что национальная безопасность любого государства во многом определяется информационной безопасностью. Украина в этом направлении делает существенные шаги и имеет достижения.

В 2003 г. приняты законы «Про електронні документи та електронний документообіг» і «Про електронний цифровий підпис». Введен в действие стандарт ДСТУ 4145-2002 на электронную цифровую подпись.

В результате выполнения Европейского проекта созданы стандартные криптографические примитивы по основным видам криптографических преобразований – симметричные и направленные шифры, цифровые подписи, функции хеширования, коды аутентификации, схемы идентификации и алгоритмы генерации псевдослучайных последовательностей. Сегодня актуальны задачи освоения этих примитивов, дальнейшие исследования, прежде всего в части реальной криптостойкости, статистической безопасности и надежности математической базы.

Настоящий тематический выпуск позволит довести до специалистов ряд интересных результатов, особенно в части блочных симметричных шифров, обсудить состояние вопроса, продолжить ведущиеся в Украине дискуссии.

Большинство статей прошли обсуждение на кафедре «Безопасности информационных технологий» ХНУРЭ и рекомендованы к опубликованию Ученым Советом университета. Считаем, что представленные материалы необходимо использовать и в учебном процессе, прежде всего по специальностям направления «Информационная безопасность».

С уважением, признательностью и благодарностью к коллегам, специалистам и читателям, которые остаются неравнодушными к проблемам информационной безопасности.

М.Ф. Бондаренко,
ректор ХНУРЭ,
д-р техн. наук, профессор

И.Д. Горбенко,
зав. кафедрой БИТ,
д-р техн. наук, профессор

ПРОБЛЕМЫ ТЕОРИИ И ПРАКТИКИ СОЗДАНИЯ И РАЗВИТИЯ ПЕРСПЕКТИВНЫХ СИСТЕМ ЗАЩИТЫ ИНФОРМАЦИИ

УДК 681.3.06:519.248.681

М. Ф. БОНДАРЕНКО, д-р. техн. наук , О. В. ПОТІЙ, канд. техн. наук, В. Г. ЛАВРІНЕНКО, Ю. І. ГОРБЕНКО

ВИЗНАЧЕННЯ ТА ОБҐРУНТУВАННЯ СУТІ ПОЛІТИКИ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Фундаментальну роль у забезпеченні безпеки інформаційних технологій (ІТ-безпеки) відіграє політика безпеки інформації (ПБ). Політика безпеки це основа створення та ефективного функціонування комплексної системи захисту інформації (КСЗІ). Відсутність сильної, добре продуманої політики безпеки приводить до відсутності методологічного та організаційного фундаменту забезпечення безпеки інформації та інших цінних ресурсів, інформаційної інфраструктури систем інформаційних технологій (ІТ-систем), комп'ютерних та автоматизованих систем, а також до безсистемності розв'язання задач захисту інформації. Реальність показує, що на сьогодні інформаційні технології настільки глибоко інтегровані в телекомунікаційні системи, комп'ютерні та автоматизовані системи, що у практику впроваджується термін інформаційно-телекомунікаційні системи (ІТС). У подальшому ми будемо використовувати саме цей термін.

Згідно з уже майже усталеними поглядами політика безпеки розробляється на основі моделі забезпечення безпеки інформації. Проведений аналіз та практичний досвід показують, що в якості базової змістовної моделі забезпечення безпеки інформації необхідно використовувати модель, що визначається міжнародним стандартом ISO/IEC 15408 «Єдині критерії оцінки безпеки систем інформаційних технологій» [1 – 3].

Згідно з даною моделлю, при розгляданні проблеми забезпечення безпеки інформації в ІТС необхідно виходити з того, що існують дві основні протиборчі (конфліктуючі) сторони – *власник* ресурсів ІТС, що мають певну цінність і вимагають свого захисту, і *порушник(зловмисник)*, який має мотиви і можливість для незаконного використання ресурсів ІТС, що може привести до нанесення збитку (морального, матеріального, економічного і т. ін.) власнику ресурсів. Третьою, незалежною стороною є арбітр (система арбітражу), основними задачами якого є розгляд суперечок між користувачами ІТС, перш за все коли один із них є внутрішнім зловмисником, а також між користувачами та власниками інформації та ресурсів. Крім того, арбітр може бути як довіреною стороною, так і зловмисником.

За збереження ресурсів відповідає їх власник, для якого ці ресурси мають цінність. У такому випадку ресурси можуть розглядатися як активи. Безпека ІТС пов'язана перш за все із захистом інформаційних ресурсів системи. Необхідно чітко розуміти, що інформація (інформаційний ресурс) – це найбільш цінний актив власника ІТС.

Зловмисник або агент погроз розглядається як джерело погроз безпеці інформації та ресурсам. Під погрозою будемо розуміти можливі події, дії (впливи), процеси чи явища, реалізація яких може привести до нанесення збитку (втрат) власнику та користувачам ресурсів. Агенти погроз (порушники) мають певну зацікавленість у несанкціонованому використанні ресурсів ІТС і прагнуть їх використати, незважаючи на інтереси власника. Власник ресурсів повинен сприймати подібні погрози як потенціал впливу на ресурси, що призводить до пониження їх цінності і до різних видів збитків для власника. Основними загрозами в ІТС є:

– втрата (порушення) конфіденційності інформації та ресурсів, тобто розкриття змісту інформаційного ресурсу несанкціонованим користувачем;

– втрата (порушення) цілісності інформації та ресурсів внаслідок впливів як природного, так і штучного характеру;

– втрата або неякісна доступність до інформації та ресурсів користувачів, а також можливість доступу до інформації зловмисників;

– неякісна спостережність власників та/або користувачів за інформацією та ресурсами.

Власник автоматизованої системи аналізує можливі погрози з метою виявлення, які з них дійсно мають місце у середовищі експлуатації автоматизованої системи. Як результат такого аналізу визначаються ризики інформаційної безпеки. Аналіз може допомогти при виборі контрзаходів для протистояння погрозам та зниженню ризиків до придатного рівня.

Контрзаходи застосовуються для зменшення вразливості та реалізації політики безпеки власника ресурсів. Однак і після введення таких контрзаходів можуть зберігатися залишкові вразливості. Такі вразливості можуть використовуватися агентами погроз (порушниками), становлячи рівень залишкового ризику для активів. Власник ресурсів повинен мінімізувати цей ризик, задаючи додаткові обмеження.

Під ресурсами слід розуміти, в широкому розумінні, все, що має цінність з точки зору власника автоматизованої системи. Виділяють наступні класи ресурсів:

– обладнання автоматизованої системи (фізичні ресурси);

– інформаційні ресурси (бази даних, файли, всі види документів, дані, що передаються каналами зв'язку);

– програмне забезпечення (системне, прикладне, утиліти, інші допоміжні програми);

– сервіс та підтримуюча інфраструктура (обслуговуючі засоби обчислювальної техніки, енергопостачання, забезпечення необхідних умов експлуатації і т.ін.).

Традиційно під *політикою безпеки розуміють низку правил безпеки, котрі регламентують порядок обробки інформації та направлені на захист інформації від визначеної множини погроз безпеці* [4 – 8]. На наш погляд, з точки зору системного підходу таке визначення не повністю відображає сутність політики безпеки. Зокрема при такому визначенні не враховується діяльнісний аспект політики безпеки. Метою даної статті є уточнення самого визначення політики безпеки, з'ясування її змісту та сутності, а також формулювання підходу до розробки політики безпеки на основі діяльнісного підходу як альтернативи традиційному (морфологічному).

1 Сутність традиційного підходу до визначення політики безпеки

Морфологічний підхід до визначення політики безпеки відображає традиційні, класичні погляди на сутність політики безпеки і зручний для її аналізу з точки зору складу та змісту правил безпеки, тобто які конкретно правила безпеки повинні бути включені до політики, на основі яких принципів здійснюється забезпечення безпеки інформації в ІТС, які основні напрямки захисту інформації обрані у системі. Таким чином, з позиції морфологічного підходу політика безпеки повинна розглядатися як деяка організована сукупність правил безпеки, а саме *політика безпеки – це система взаємопов'язаних та погоджених правил безпеки, що регламентують порядок обробки інформації в ІТС, і направлених на відвертання визначеної множини погроз безпеці.* Принциповою відмінністю сформульованого визначення від раніше наведених є те, що правила безпеки повинні являти собою не просто деякий набір, а систему з усіма системними властивостями. Такий погляд на політику безпеки дозволяє нам сформулювати загальний підхід до розробки змісту політики безпеки і визначити властивості, які повинна мати політика безпеки як сукупність правил безпеки.

Політика безпеки може бути подана у вигляді дворівневої моделі. Перший рівень моделі – множина цілей безпеки і задач захисту, тобто цільова множина $T \in T_0$, де T_0 – вихідна множина задач захисту та цілей безпеки. Другий рівень – це множина правил безпеки $R \in R_0$,

де R_0 – вихідна множина правил захисту безпеки. Вихідна множина правил безпеки та вихідна множина задач захисту міститься у нормативних документах та стандартах зі інформаційної безпеки, наприклад у таких документах як [7 – 9]. Множина правил безпеки R розробляється для досягнення цілей безпеки шляхом розв'язання задач захисту, а їх сукупність є унікальною для конкретної автоматизованої системи.

Морфологічна модель політики безпеки розглядає правила безпеки з позиції вимог власника об'єкту захисту (інформації), тобто політика безпеки визначає, що необхідно зробити для задоволення потреб у захисті інформації. В основу формування правил безпеки закладаються загальноприйняті принципи забезпечення безпеки ІТС, які застосовуються незалежно від призначення системи, її розмірів та критичності.

Необхідно як мінімум виділити вісім основоположних принципів, на яких будь-яка організація базує свою програму щодо забезпечення безпеки інформації [10, 11].

1. Система безпеки інформації повинна підтримувати головну мету власника ІТС. Мета забезпечення безпеки інформації полягає у захисті ресурсів ІТС (активів власника ІТС) шляхом вибору та застосування відповідних заходів безпеки.
2. Забезпечення безпеки інформації – це елемент єдиного управління. В цьому виявляється ієрархічність управління інформаційною безпекою, підпорядкованість єдиній меті управління.
3. Рентабельність та ефективність процесів забезпечення безпеки інформації. Необхідно відмітити, що впровадження заходів безпеки інформації потребує виділення додаткових коштів. Рівень безпеки повинен відповідати і бути пропорційним цінності ресурсів, що захищаються, та рівню можливого збитку, який може бути нанесено у випадку порушення цілісності, конфіденційності або доступності інформації.
4. Відповідальність власника ІТС за безпеку ресурсів перед зовнішніми сторонами. Якщо в системі циркулює інформація, яка належить стороннім організаціям, то власник ІТС повинен забезпечувати адекватний захист цієї інформації і нести відповідальність за порушення конфіденційності, цілісності та доступності даної інформації під час її обробки засобами ІТС.
5. Попередній розподіл та встановлення відповідальності всіх сторін (власників та користувачів) за фактичний стан безпеки інформації. Даний принцип передбачає, що в ІТС на всіх рівнях чітко визначені права, обов'язки та відповідальність усіх осіб, що приймають участь у процесі обробки інформації, відносно розв'язання задач захисту інформації. З одного боку це означає визначення правових та адміністративних норм, які регулюють взаємовідношення та обов'язки різних учасників відносно забезпечення безпеки інформації. З іншого боку це передбачає реалізацію спеціальних заходів нагляду (спостережності), що дозволяють визначити порушення політики безпеки та ідентифікувати особу, що причетна до дій, які привели до порушення безпеки.
6. Комплексний підхід до розв'язання задач безпеки, за якого реалізація заходів захисту здійснюється на правовому, адміністративному, процедурному, програмно-технічному рівнях з комплексним застосуванням засобів захисту інформації, взаємодії всіх елементів та служб ІТС.
7. Безперервність забезпечення режиму інформаційної безпеки передбачає реалізацію заходів безпеки на постійній основі, періодичну переоцінку рівня захищеності ІТС, адаптацію системи безпеки ІТС до умов експлуатації, що змінюються.
8. Дотримання вимог та положень існуючого національного та міжнародного законодавства, врахування соціальних чинників, у тому числі тих, що впливають на обслуговуючий персонал та користувачів. Експлуатація систем забезпечення інформаційної безпеки повинна бути психологічно сприйнятною і не викликати напруги власників та користувачів ІТС.

З точки зору морфологічної моделі або моделі складу політика безпеки повинна задовольняти наступним вимогам та наступним властивостям:

1. Узгодженість та несуперечність головної мети функціонування ІТС та цілей безпеки і задач захисту, що направлені на досягнення головної мети функціонування ІТС за рахунок впровадження політики безпеки.
2. Повнота сукупності правил безпеки (необхідність та достатність). Тут необхідно врахувати, що умова достатності є важко досяжною. Достатність можна забезпечити тільки на деякому проміжку часу функціонування системи захисту інформації. У ході функціонування системи захисту будуть відбуватися зміни зовнішніх умов функціонування ІТС, зміни моделі погроз безпеці, що потягне за собою необхідність внесення змін до політики безпеки. Умова необхідності передбачає включення до політики безпеки мінімально необхідної множини правил безпеки, що як мінімум відображає вимоги нормативних документів щодо забезпечення безпеки інформації. Умова необхідності передбачає реалізацію базового рівня захищеності об'єкта захисту.
3. Відповідність положень політики безпеки вимогам законодавства та нормативних документів, що регламентують порядок забезпечення безпеки інформації в ІТС (національних та міжнародних стандартів, рекомендації органів державного управління, відомчих нормативних документів).
4. Законність розробки та прийняття політики безпеки.
5. Взаємна узгодженість цілей безпеки, задач захисту та правил безпеки.

2 Сучасне визначення політики інформаційної безпеки

Згідно з новітніми поглядами, подальше вдосконалення суті політики безпеки може бути виконане на основі використання моделі практичної діяльності власника ІТС із забезпечення інформаційної безпеки. Використання моделі практичної діяльності власника ІТС щодо забезпечення безпеки інформації є суттю діяльнісного підходу до визначення політики безпеки Забезпечення безпеки інформації в ІТС, з точки зору практичної діяльності власника, є видом організаційно-управлінської діяльності, котра буде підпорядковуватися загальним законам здійснення такої діяльності з акцентом на особливості і специфіку розв'язання задач забезпечення безпеки інформаційних технологій. З цих позицій *політика безпеки це систематична, стабільна, організована та цілеспрямована ДІЯЛЬНІСТЬ власника ІТС відносно розв'язання проблем забезпечення безпеки інформації в ІТС, яка здійснюється або безпосередньо самим власником ІТС, або непрямо – через відповідні механізми і органи управління, і має вплив на функціонування ІТС і на управління підприємством (технологічними процесами, бізнес-процесами і т. ін.) в цілому.* Така діяльність дозволяє за виділених фінансових та матеріально-технічних затратах мінімізувати витрати в конкретних умовах функціонування ІТС.

Політика безпеки в цілому це не тільки сукупність правил безпеки – це план високого рівня, в якому описуються цілі та задачі заходів щодо забезпечення безпеки інформації в ІТС. Вона забезпечує планування і виконання програми безпеки.

Після розробки правил безпеки виникає проблема впровадження, реалізації цих правил у конкретній системі, на конкретному об'єкті. Тут виникає безліч питань, а саме: яким чином можна оцінити ступінь рішення проблем безпеки інформації, чи є питання політики безпеки адекватним даній ситуації, яким чином можна оцінити альтернативні стратегії забезпечення безпеки інформації. Відповіді на ці та подібні питання можна одержати лише за розгляду політики безпеки з позиції діяльнісного підходу.

Аналіз низки джерел та проведені дослідження показали, що сучасна ефективна ПБ може бути здійснена на базі таких базових принципів.

1. Принцип цілеспрямованості, який передбачає, що в основу забезпечення безпеки закладені конкретні цілі безпеки, на досягнення яких направлена практична діяль-

ність власника ІТС. Уся діяльність власника ІТС має основний системостворюючий чинник – спрямування до основної мети щодо забезпечення безпеки.

2. Принцип цілісності (інтегрованості) вимагає внутрішню єдність складових елементів політики безпеки. Всі правила політики безпеки повинні неухильно виконуватись усіма об'єктами та суб'єктами процесу забезпечення безпеки інформації, у противному випадку це порушує цілісність політики.
3. Принцип структурованості передбачає чіткість та суворість взаємного розподілення функцій, задач, прав, обов'язків та відповідальності між усіма учасниками процесів забезпечення безпеки інформації.
4. Принцип організованості вимагає, щоб діяльність здійснювалась систематично, підпорядковувалась визначеному порядку виконання практичних робіт, залучені до процесу особи діяли у відповідності з раніше встановленими планами (спланованість діяльності). Крім того, ця діяльність піддається постійному контролю за результативністю і управляємістю.
5. Принцип узгодженості (координованості) діяльності передбачає гармонійне поєднання всіх видів діяльності і заходів щодо забезпечення безпеки інформації, взаємну зумовленість та взаємозв'язок практичних робіт.
6. Принцип мотивованості та усвідомленості передбачає формування активного, усвідомленого, зацікавленого та діяльнісного відношення всіх учасників процесів забезпечення безпеки інформації на всіх рівнях і напрямках до виконання правил безпеки.
7. Принцип стабільності, який передбачає стабільність рівня вимог, правил безпеки та зусиль власника щодо реалізації правил безпеки в часі.
8. Принцип безперервності, що передбачає здійснення практичної діяльності на постійній основі й протягом усього життєвого циклу ІТС.

Використання діяльнісного підходу до визначення політики безпеки дозволяє ввести практичні критерії оцінки політики безпеки. Ці критерії дозволяють оцінити та порівняти різні альтернативи здійснення політики безпеки з точки зору її впливу на функціонування ІТС у цілому, на досягнення поставлених цілей безпеки і розв'язання задач захисту, а також оцінити діяльність власника стосовно впровадження у життя положень політики безпеки і реалізації принципів забезпечення безпеки інформації. В якості таких критеріїв необхідно використовувати:

- критерій результативності;
- критерій ефективності;
- критерій адекватності;
- критерій реагованості (гнучкості);
- критерій доцільності;
- критерій здійснюваності:
- критерій простоти в адміністративному забезпеченні.

Критерій результативності та критерій ефективності є самостійними оціночними критеріями, останні можуть бути віднесені до практичних.

Критерій результативності визначає, в якій мірі може і чи може взагалі впровадження політики безпеки й окремих її положень або правил привести до досягнення поставлених цілей безпеки та розв'язання визначених задач захисту. Основною проблемою використання даного критерію на сьогодні є визначення показників результативності політики безпеки. Бажано мати в розпорядженні кількісні показники результативності, однак цілком можливо

застосування і якісних показників. Використовуючи показники результативності можна ввести *шкалу результативності*, яка буде показувати рівень досягнення цілей безпеки.

Під ефективністю політики безпеки будемо розуміти співвідношення результатів, досягнутих за впровадження політики безпеки, і витрат, необхідних для досягнення цих результатів. Таким чином, у даному випадку ефективність є синонімом економічної раціональності політики безпеки. Ефективність в основному вимірюється грошовим еквівалентом. Основним підходом до визначення ефективності є підрахунок витрат на реалізацію сукупності правил безпеки, направлених на забезпечення розв'язання визначеної задачі захисту. Кожне правило безпеки, що прописане у нормативній політиці безпеки, має на увазі виконання комплексу організаційних заходів, проведення технічних робіт, закупку технічних та інших засобів, які мають безпосереднє відношення до захисту інформації. Усе це вимагає вкладання коштів, що і дає підставу на введення та використання критерію ефективності політики безпеки.

При визначенні показників ефективності виникає ряд проблем, а саме:

- відсутність показників результативності політики безпеки;
- різноманітність варіантів реалізації політики безпеки;
- наявність широкого спектру технічних засобів захисту інформації, що володіють різною технічною ефективністю;
- необхідність обліку побічних витрат, що виникають як при розробці самої політики безпеки, так і при впровадженні її у повсякденну практику.

Політика безпеки є ефективною, якщо вона досягає максимальної результативності за мінімальних затрат.

Таким чином, критерії результативності й ефективності тісно взаємопов'язані і є базовими оціночними критеріями політики безпеки.

Розглянемо практичні критерії оцінки політики безпеки.

Критерій адекватності визначає, що даний рівень результативності дійсно відповідає ситуації, яка склалася відносно існуючих погроз безпеці і задовольняє потребам власника інформаційних ресурсів відносно забезпечення безпеки інформації. Політика безпеки є адекватною, якщо правила безпеки і рівень їх реалізації адекватні погрозам безпеки для даного об'єкту захисту і сприяє надійному запобіганню виявлених погроз та зниженню ризиків до придатного рівня.

Критерій доцільності пов'язаний із визначенням, чи є протиріччя між задачами і основними положеннями політики безпеки та загальними задачами об'єкту захисту. Критерій доцільності уточнює питання, чи необхідні цілі безпеки і задачі захисту в конкретній системі, чи нема протиріч між загальними задачами, що стоять перед нею.

Для оцінки політики безпеки за даним критерієм необхідно врахувати всі критерії одночасно, виразити відношення між їхніми кількісними формами. Використання для розробки політики безпеки стандартів безпеки або залучання для її розробки і оцінки експертів – це один з основних доводів доцільності політики безпеки. Таким чином, політика безпеки – доцільна, якщо положення безпеки і задачі захисту не мають протиріч між основними задачами об'єктів захисту, узгоджені з цілями функціонування об'єкту.

Критерій гнучкості (реагованості) пов'язаний з оцінкою здатності політики безпеки задовольнити потреби власника інформаційних ресурсів у відношенні безпеки інформації в умовах обстановки, що змінюється. Іншими словами, політика безпеки повинна бути здатною адекватно реагувати на зміни умов функціонування об'єкту інформації, зміни цілей і задач функціонування, інтересів і потреб власника інформаційних ресурсів. Політика безпеки називається гнучкою, якщо вона здатна задовольнити потреби в безпеці інформації у будь-яких умовах функціонування ІТС.

Критерій здійсненності пов'язаний з визначенням умов здійсненності конкретної політики безпеки в конкретних умовах при заданих обмеженнях у конкретній ІТС. Даний критерій враховує умови здійсненності політики безпеки і впровадження її на різних рівнях

забезпечення безпеки інформації і у зв'язку з цим має різні аспекти. Безпека інформації забезпечується на чотирьох рівнях – законодавчому (правовому), адміністративному, процедурному і програмно-технічному. Таким чином, необхідно розглядати здійсненність політики безпеки на цих рівнях.

На правовому рівні необхідно оцінювати політику безпеки з точки зору її легітимності. Чи можуть конкретні положення політики безпеки бути реалізовані на даному об'єкті з точки зору правового поля держави в області захисту інформації. Чи має право керівництво приймати такого роду рішення? Чи відповідають положення політики безпеки нормам законів та інших нормативних актів в області захисту інформації.

Здійсненність політики безпеки на адміністративному рівні залежить від ступеню розуміння керівництвом цілей безпеки і задач захисту, усвідомлення реальності погроз безпеці, реалізація яких може нанести збиток, рівня сформованості потреб у розв'язанні таких задач захисту.

На процедурному рівні здійсненність політики безпеки залежить від ступеня технологічної і організаційної готовності об'єкта інформатизації до впровадження правил безпеки. Тут важливе місце набуває готовність персоналу виконувати вимоги безпеки. Така готовність залежить від багатьох чинників: розуміння персоналом необхідності виконання цих вимог, рівня усвідомлення і дисциплінованості персоналу, рівня його професійної підготовленості.

Іншими чинниками, що мають істотний вплив на здійсненність політики безпеки, це якість системи менеджменту (управління) безпекою на об'єкті інформатизації. Якість практичних робіт щодо реалізації положень політики безпеки і досягнення цілей безпеки залежить не тільки від придбання ефективних засобів захисту інформації, але й від ефективного управління безпекою на об'єкті, планування захисту, визначення конкретного переліку робіт щодо реалізації правил безпеки, впровадження системи контролю та оцінки виконання цих робіт.

На програмно-технічному рівні на здійсненність політики безпеки має вплив можливість закупки необхідних заходів захисту і технічні можливості їх застосування на об'єкті інформатизації. На здійсненність впливають розмір фондів, що виділяються на реалізацію програми забезпечення безпеки інформації і планів захисту, наявність на ринку відповідних засобів захисту потрібної якості, технологічний рівень процесів обробки інформації на об'єкті інформатизації (стан парку обчислювальної та іншої спеціалізованої техніки, рівень комп'ютеризації та інформатизації технологічних процесів і т. ін.).

Таким чином, політика безпеки є здійсненною, якщо на правовому, адміністративному, процедурному та програмно-технічному рівнях забезпечення безпеки інформації створені всі умови для здійснення правил безпеки.

Критерій простоти в адміністративному забезпеченні розглядає політику безпеки з точки зору її придатності для адміністрування, тобто враховує наявність достатнього адміністративного персоналу для впровадження політики безпеки, рівень професіоналізму, організаторських здібностей та навичок персоналу, що відповідає за реалізацію політики безпеки і організацію ефективного захисту інформації. Політика безпеки є простою в адміністративному забезпеченні, якщо вона потребує мінімальних витрат на організаційно-штатні зміни в структурі організації.

Таким чином, з точки зору діяльності політика безпеки повинна бути цілеспрямованою, стабільною, неперервною, організованою, керованою, узгодженою та мотивованою, забезпечувати максимальну результативність і ефективність при досягненні цілей безпеки і розв'язанні задач захисту, бути адекватною погрозам безпеки, доцільною та гнучкою в реалізації, здійсненною на різних рівнях забезпечення безпеки інформації, достатньо простою у адміністративному забезпеченні. З позиції діяльнісного підходу від політики безпеки вимагається значно більше, ніж це передбачається за традиційного підходу до аналізу політики безпеки.

3 Взаємозв'язок морфологічного та діяльнісного підходу. Системне визначення політики безпеки

Викладене в першому та другому розділах дозволяє стверджувати, що політика безпеки має двоякі властивості. З одного боку це сукупність документів, що містять систематизоване викладення цілей безпеки, задач захисту та правил безпеки. З іншого – це практична діяльність власника ІТС, яка здійснюється у відповідності до правил безпеки і спрямована на досягнення цілей безпеки. Таким чином, політика безпеки складається з двох складових – пасивної та активної. Пасивна складова – це нормативна (документальна) частина політики безпеки, тобто взаємопов'язана сукупність правил безпеки, які визначають, що повинно бути захищеним і які обмеження накладаються на управління процесами забезпечення безпеки інформації. У подальшому для позначення пасивної складової політики безпеки будемо використовувати термін *нормативна політика безпеки.*

Активна складова політики безпеки – це діяльність власника інформаційних ресурсів, що спрямована на досягнення цілей безпеки на основі і через реалізацію встановлених правил безпеки. По суті це *практична політика безпеки.*

Аналіз показує, що обидві складові знаходяться у тісній взаємодії і не можуть розглядатися у відриві одна від одної. Якщо формувати нормативну частину політики безпеки без підтримки власника ІТС, без урахування можливостей наступної реалізації, то політика безпеки виродиться у мертві документи.

Здійснення ж практичної діяльності без нормативного забезпечення перетвориться в латання дірок у системі захисту, у вирішення окремих задач, які не позв'язані єдиним задумом, концепцією і планом. У результаті не буде досягнуто необхідний рівень захищеності всієї інформаційно-телекомунікаційної системи.

Узагальнена системна модель політики безпеки подана на рис. 1.

Таким чином, сформулюємо визначення політики безпеки. **Політика безпеки це систематична, стабільна, організована та цілеспрямована ДІЯЛЬНІСТЬ власника ІТС відносно розв'язання проблем забезпечення безпеки інформації, що здійснюється на основі і через реалізацію встановлених правил безпеки і впливає на функціонування ІТС у цілому.**

Тісний внутрішній зв'язок пасивної та активної складових політики безпеки і обумовлює взаємопов'язаність властивостей, критеріїв та показників цих складових. Цей взаємозв'язок представлено на рис. 2.

Подана модель дозволяє визначити наступні базові елементи політики безпеки, які складають стратегію забезпечення безпеки інформації (рис. 3):

- створення організаційно-методологічних основ забезпечення безпеки інформації, що виражається у розробці Концепції безпеки інформації в АСУ і стратегічної програми щодо забезпечення інформаційної безпеки у результаті створюється організаційно-методологічні основи забезпечення ІБ в ІТС;
- здійснення ефективного менеджменту в області безпеки інформації;
- здійснення практичної діяльності щодо реалізації правил політики безпеки (інжиніринг безпеки);
- створення ефективної системи аудиту безпеки (контролю ефективності комплексної системи захисту інформації).

Реалізація всіх перелічених вище елементів політики безпеки дійсно дозволяє сформувати і проводити в життя ефективну політику безпеки.

Рис. 1

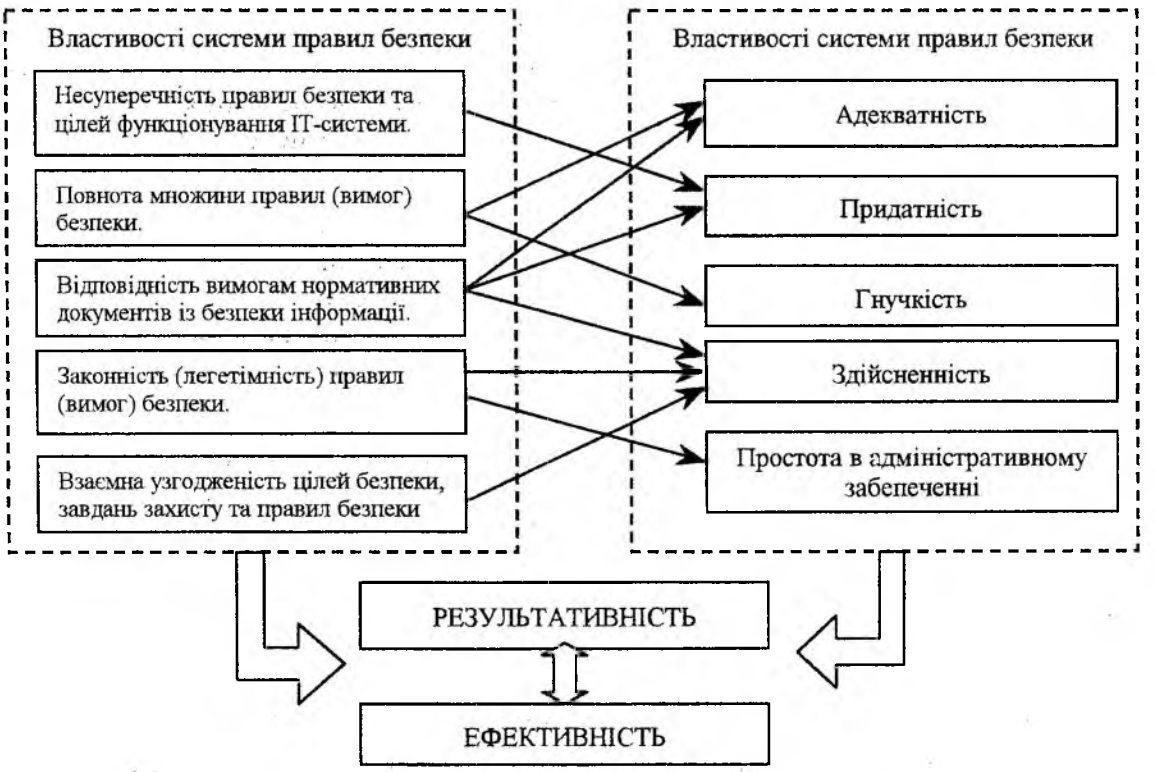


Рис. 2

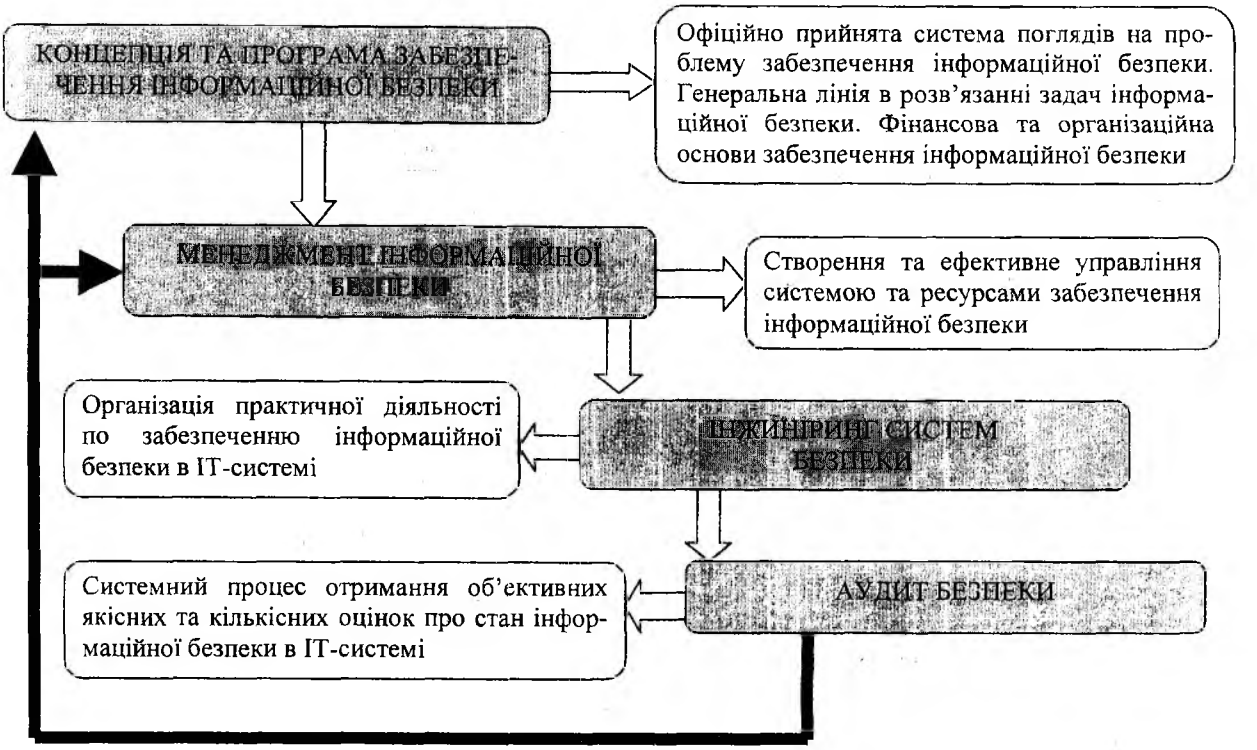


Рис. 3

4 Загальний порядок розробки політики безпеки

З практичної точки зору під час розробки змісту політики безпеки (ПБ) важливо чітко уявляти склад і зміст вихідних даних для її розробки (рис. 4).

Вихідними даними, що безпосередньо впливають на зміст ПБ є:

- характеристика об'єкту застосування ПБ. Політика безпеки як нормативний документ застосовується тільки до конкретного об'єкту. Об'єктами застосування політики безпеки можуть бути організації в цілому, окрема проблема забезпечення інформаційної безпеки і конкретна система;
- сукупність вимог, що містяться в законах і нормативних актах держави, міжнародних, національних та промислових стандартах у галузі інформаційної безпеки, нормативних документах державного і відомчого характеру. Політика безпеки повинна розроблятися у відповідності до нормативно-правової бази, що діє на території держави (держав) і відомства (відомств), у рамках якого існує, здійснює діяльність (функціонує) об'єкт застосування політики безпеки;
- групи осіб, для яких призначена політика безпеки. Цілі, задачі, структура та зміст ПБ суттєво залежать від того, для якого рівня керівників та фахівців об'єкта застосування розробляється політика.

Перелічені вище категорії інформації визначають наступні практичні аспекти розробки політики безпеки:

- визначення цілей і задач політики безпеки;
- розробка структури і конкретного змісту політики безпеки (розробка правил безпеки);
- визначення шляхів проведення в життя і реалізації положень, правил, норм та вимог політики безпеки, ступінь відповідальності за порушення вимог ПБ і контроль за їх виконанням.

Наостанку, не можна забувати про те, що розробка ПБ і діяльність щодо реалізації її положень можуть бути ефективними лише в умовах реального взаємозв'язку з іншими документами і діяльністю організації в області забезпечення ефективного функціонування і безпеки.

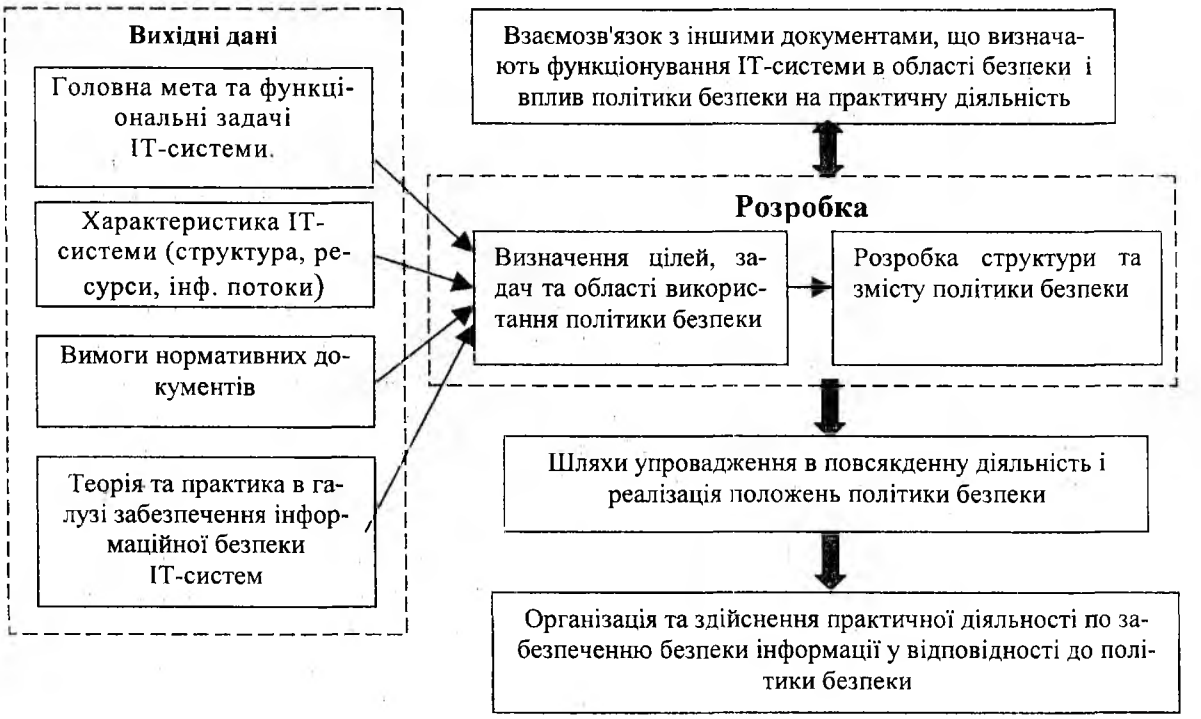


Рис. 4

Нормативна політика безпеки це сукупність документів, які охоплюють досить широке коло питань і є документами загального характеру. Для опису правил безпеки розроблюються і використовуються різні документи:

- методичні вказівки щодо здійснення практичної діяльності;
- порядок робіт, інструкції, керівництва;
- міжнародні, національні, галузеві та промислові стандарти;
- настанови, розпорядження, директиви, накази;
- регламенти (технічні регламенти) та інші документи.

Розробка вищеперелічених документів повинна здійснюватися на єдиній методологічній основі із забезпеченням наслідування і не мати протиріч із правилами та вимогами, що містяться у нормативних документах різного рівня.

5 Типи політики безпеки

Сьогодні розрізняють три основні типи політики безпеки [8, 12]:

- програмна політика безпеки;
- проблемно-орієнтована політика безпеки;
- політика безпеки конкретної системи чи об'єктова політика безпеки.

Програмна політика безпеки

Програмна політика безпеки це політика верхнього рівня. Об'єктом застосування програмної політики безпеки є відомство (організація) в цілому як систематизоване і свідоме об'єднання по місцю та часу дій людей, що мають досягти визначених цілей. За організацію і здійснення розробки програмної політики безпеки безпосередню відповідальність несе керівник. Програмна політика безпеки розробляється з метою визначення (реструктуризації)

основних компонентів і реалізації (впровадження в дію) програми забезпечення інформаційної безпеки (ПЗІБ) відомства.

Програмна політика безпеки визначає:

- цілі і задачі ПЗІБ, сферу її діяльності всередині відомства;
- осіб, що відповідають за реалізацію різних напрямків ПЗІБ.

Таким чином, програмна політика безпеки визначає множину стратегічних напрямків забезпечення інформаційної безпеки, види і обсяг ресурсів, які виділяються для реалізації ПЗІБ. У даному випадку програмна політика безпеки визначає стратегію керівництва в області забезпечення інформаційної безпеки. Саме тому цей документ розробляється на довгий строк (5 і більше років).

Основними компонентами програмної політики безпеки є:

1. Ціль і призначення. Програмна політика безпеки визначає, встановлює і на адміністративному рівні закріплює стратегічний напрямок діяльності керівництва відомства в області забезпечення інформаційної безпеки. Вона містить обґрунтування необхідності реалізації програми забезпечення інформаційної безпеки, визначає цілі програми. В даній частині визначається і обґрунтовується потреба в забезпеченні цілісності, доступності, конфіденційності ресурсів. Ці потреби формулюються у формі основних (базових) цілей (задач) захисту, котрі встановлюються політикою безпеки.

2. Область та об'єкт застосування положень політики безпеки. Програмна політика повинна чітко визначати, які ресурси, включаючи обладнання, апаратуру, апаратне та програмне забезпечення, інформацію та персонал, охоплює програма із забезпечення безпеки. У більшості випадків програма буде охоплювати всю систему і весь персонал організації. Іноді область застосування програми може бути обмеженою.

3. Встановлення прав та обов'язків. Програмна політика безпеки повинна створювати організаційно-технічну основу формування відповідних структур (служб, підрозділів), що повинні здійснювати управління та контроль над процесами забезпечення режиму інформаційної безпеки в організації. Документ повинен чітко визначати права та обов'язки керівників різного рівня, технічного персоналу, користувачів та інших осіб у відношенні підтримки, реалізації цілей та задач забезпечення інформаційної безпеки. Визначення змісту прав та обов'язків повинно здійснюватись у відповідності до принципів конкретності (призначається конкретна особа, яка несе відповідальність за конкретні дії) і принципу відповідності вимогам чинного законодавства та інших нормативних документів.

4. Визначення принципів контролю (нагляду) за підтриманням режиму інформаційної безпеки. В програмній політиці приділяється увага двом основним задачам забезпечення контролю. По-перше, здійснення загального контролю за реалізацією вимог безпеки, що визначаються програмою із забезпечення безпеки і керівниками різних підрозділів організації. З цією метою може бути призначений генеральний інспектор, який відповідає за здійснення постійного моніторингу стану інформаційної безпеки в організації, включаючи перевірку якості здійснення процесів управління безпекою і реалізації положень програми щодо забезпечення інформаційної безпеки. Іншою задачею є визначення штрафів та дисциплінарних стягнень. Оскільки політика безпеки – це документ високого рівня, то визначення конкретного змісту покарань за різні порушення в документі не деталізується. Політика безпеки може встановити необхідність створення контролюючої структури (служби), яка відповідає за розробку конкретного переліку порушень і визначення стягнень за них. Даний розділ політики безпеки повинен ураховувати норми законодавства в даній сфері і бути узгодженим із ними. Покарання, що передбачені за визначені дії законами, не повинні дублюватися у внутрішніх документах організації. Але з метою навчання та інформованості персоналу перелік цих дій і форми відповідальності за них можуть наводитися в політиці безпе-

ки. Розробник даної частини політики безпеки повинен пам'ятати, що порушення політики можуть бути і не навмисними.

Проблемно-орієнтована політика безпеки

Об'єктом застосування проблемно-орієнтованої політики безпеки є окрема проблема або задача в області забезпечення безпеки інформації в організації. Частіше за все проблемно-орієнтована політика безпеки розробляється для розв'язання знову виникаючих проблем, наприклад для реалізації і обліку нових вимог, що вводяться прийняттям нового закону чи іншого нормативного документа. Необхідність розробки проблемно-орієнтованої політики безпеки часто вимагає у відповідь як на появу та використання в організації нових технологій, так і на виникнення нових погроз та слабкостей. Проблемно-орієнтована політика безпеки може часто піддаватися перегляду в залежності від змін, які відбуваються в технологіях, законодавстві країни, структурі організації, та інших чинників. Звідси витікає одна з її властивостей – змінність.

Серед галузей діяльності організації, для яких повинна розроблятися проблемно-орієнтована політика безпеки, є політика здійснення Internet-доступу в організації; політика захисту електронної пошти; управління ризиками та планування безперебійної роботи; правила використання неавторизованого програмного забезпечення; організація виконання робіт з використанням мобільних та стаціонарних засобів обчислювальної техніки; правила використання засобів зберігання даних та систем резервного копіювання і т. ін.

У залежності від вирішуваної проблеми проблемно-орієнтована політика безпеки може бути подана у вигляді як окремого, інколи досить об'ємного документа, так і у вигляді окремих правил безпеки. Частіше за все проблемно-орієнтована політика безпеки уточнює, конкретизує положення програмної політики безпеки чи об'єктової політики безпеки.

Основними компонентами проблемно-орієнтована політика безпеки є:

1. Формулювання та опис роботи. Чітке визначення проблеми, що підлягає розв'язанню, її системний опис повинні передувати безпосередній розробці правил безпеки. Розробник політики повинен визначити важливість, відмінні ознаки та умови існування проблеми. Усвідомлення проблеми дозволить сформулювати цілі проблемно-орієнтованої політики безпеки і обґрунтувати необхідність її розробки.

2. Формулювання позиції організації. Даний компонент містить чітке та прозоре викладення позиції організації у відношенні до даної проблеми.

3. Обґрунтованість застосування. Правила, які вводяться проблемно-орієнтованою політикою безпеки, повинні містити обґрунтування свого застосування. Це означає, що положення політики повинні містити пояснення що, де, як, коли та ким конкретно застосовується правило.

4. Повноваження та відповідальність. Як і будь-яка політика безпеки, проблемно-орієнтована політика безпеки повинна чітко формулювати повноваження та відповідальність осіб, що мають відношення до реалізації положень політики.

5. Порядок взаємодії. У будь-якій проблемній політиці безпеки називаються конкретні посадові особи в організації, які уповноважені вирішувати ті чи інші проблеми, що виникли при реалізації конкретних положень політики безпеки. Оскільки позиція організації з даного питання змінюється не частіше, ніж конкретні співробітники, то краще вказати посаду (роль) співробітника, якому впроваджено вирішення таких проблем посадовими обов'язками. Для вирішення практичних проблем в реалізації конкретних положень політики (правил) користувачі можуть контактувати з керівниками різного рангу, економістами, інженерно-технічним складом, системними адміністраторами і т.ін. Співробітник повинен знати, до кого звернутися з питаннями і за додатковою інформацією – до безпосереднього начальника, системного адміністратора чи співробітника служби захисту інформації.

Проблемно-орієнтована політика безпеки часто супроводжується керівництвами, настановами та інструкціями. Наприклад, політика використання неофіційного ПЗ може включати в себе керівництва з перевірки дисків, їх реєстрації та обліку і т.ін.

Системно-орієнтована політика безпеки

Системно-орієнтована політика безпеки перш за все визначає напрямок, методи та процедури забезпечення інформаційної безпеки у конкретній ІТС. Даний тип політики обмежений рамками окремої ІТС, а також областю взаємодії самої системи і середовища її експлуатації. Політика безпеки ІТС – це частина програмної політики безпеки, яка наслідує основні принципи політики інформаційної безпеки організації (відомства). Системна політики безпеки розробляється з одного боку для керівників старшої ланки, які приймають фінансові та технічні рішення відносно використання в тій чи іншій ІТС. З іншого боку вимоги політики безпеки доводяться до відома всіх робітників, які мають безпосереднє відношення до закупівлі обладнання та експлуатації ІТС, а також до інформації, котра обробляється в даній системі.

В самому загальному випадку для опису системної політики безпеки можна використовувати дворівневу модель: цілі (задачі) захисту та практичні правила безпеки. Для розробки пов'язаного та повного набору правил безпеки розробник повинен використовувати спеціальні прийоми, за допомогою яких на основі аналізу задач захисту формулюються правила безпеки.

Визначення цілей безпеки та формулювання задач захисту є важливим етапом розробки системної політики безпеки. Крім того, даний етап – неперервний, тобто спеціалісти служби захисту інформації ІТС здійснюють постійний моніторинг актуальності сформульованих та уточнених задач захисту в процесі експлуатації системи. Відносно безпеки ІТС задачі захисту повинні бути конкретно та прозоро сформульовані. Тут краще застосовувати вимоги міжнародних стандартів ISO/IEC 15408 «Єдині критерії оцінки безпеки IT-систем» та ISO/IEC 15446 «Керівництво з розробки профілю захисту та проекту безпеки». Останній стандарт досить докладно описує методику визначення цілей безпеки та формулювання задач захисту IT-систем. Задачі захисту містять послідовність тверджень, які описують чіткі цілеспрямовані дії над конкретними ресурсами. В цілому задачі захисту допомагають точно визначити, що повинно бути захищеним у IT-системі.

Після того як будуть визначені задачі захисту, необхідно сформулювати правила безпеки. Загальна модель типового правила – хто, що і за яких умов. Тобто хто (особа, категорія осіб, організація) має право здійснювати будь-які дії (записати, модифікувати, купувати, знищити, увійти і т.ін.) і за яких умов ця дія може бути здійснена. При розробці правил політики безпеки основною перешкодою є забезпечення необхідного рівня деталізації правила. Необхідно пам'ятати, що політики безпеки не є ні директива, ні норматив, ні інструкція. Вона описує безпеку в узагальнених термінах без специфічних деталей. У правилах безпеки описано, що повинно бути захищеним і які обмеження накладаються на управління. Надмірна деталізація правил безпеки на рівні системи приведе до істотного адміністративного тягаря, негнучкості політики безпеки, яка розробляється на досить тривалий строк і повинна давати можливість використовувати широкий спектр різних методів та засобів захисту для забезпечення вимог безпеки.

Основними компонентами системно-орієнтованої політики безпеки є:

1. Концепція інформаційної безпеки ІТС, яка відображає систему поглядів, основних принципів та основних направлень забезпечення режиму інформаційної безпеки в системі. В концепції на основі аналізу сучасного досягнутого рівня і динаміки розвитку інформаційних технологій, очікуваних погроз інформаційній безпеці, джерел цих погроз та чинників, що сприяють їх реалізації, подається систематизоване викладання цілей, задач та принципів досягнення рівня безпеки, що вимагається. Концепція визначає генеральну лінію у вирішенні проблем інформаційної безпеки та викладає шляхи досягнення поставлених цілей безпеки.

2. Опис процесів управління ризиками. Даний компонент містить результати аналізу погроз, аналізу ризиків та вразливість і оцінку ризиків. Управління ризиками передбачає вивчення моделі погроз і моделей джерел погроз, оцінку можливих наслідків від реалізації погроз, формування моделі захисту інформації в ІТС і прийняття рішення на знешкодження ризиків. Основними елементами управління ризиками є визначення компонентів і ресурсів (активів) ІТС, ідентифікація погроз та зв'язок їх з об'єктом захисту, оцінка ризиків та величини можливого збитку, вибір варіанту побудови системи захисту інформації, оцінка витрат на реалізацію засобів захисту та створення системи захисту.

3. Загальні вимоги до безпеки інформації та рішення щодо забезпечення режиму інформаційної безпеки. Тут безпосередньо містяться правила безпеки відносно фізичної безпеки, автентифікації, ідентифікації та управління доступом, правила застосування криптографічних засобів, правила забезпечення антивірусного захисту та інші питання.

4. Обов'язки в області інформаційної безпеки розробляються з метою одержати ясне розуміння ролей та обов'язків окремих осіб у відношенні до безпеки ІТС.

5. План забезпечення безперебійної роботи ІТС містить опис процедур реагування на нештатні та надзвичайні ситуації, процедур переходу системи в аварійний режим функціонування, процедури відновлення функціонування системи після збоїв та інших ситуацій.

Тут наведені основні компоненти політики безпеки, перелік яких може бути збільшено. Структура та зміст системної політики безпеки визначаються у кожному конкретному випадку.

Політика безпеки повинна бути активним компонентом всієї діяльності щодо створення, реалізації, виготовленню, експлуатації ІТС. Вона буде і повинна впливати на різні аспекти організаційно-технічної діяльності з організації циклу управління підприємством. На рис.5 подано взаємозв'язок політики безпеки з різними аспектами експлуатації ІТС.

Рис. 5

Висновки

1. Політика безпеки має складну двояку природу та має пасивну і активну складові. З одного боку політика безпеки – це система взаємопов'язаних та узгоджених правил безпеки, які регламентують порядок обробки інформації в ІТС і направлені на запобігання визначеної множини погроз безпеці (пасивна складова). З цих позицій політика безпеки повинна мати властивості не протиріччя цілей безпеки та головної цілі функціонування ІТС, повноти вимог безпеки, не протиріччя правил безпеки вимогам нормативних документів і стандартів, законності, узгодженості. Пасивна складова відповідає за формування правил безпеки і по суті відповідає за формальне формування політики безпеки. Правила безпеки не замінюють інструкції та стандарти, не є директивами і засобами управління, описують безпеку в загальних термінах і не дають вказівки, яким чином здійснюються конкретні заходи безпеки.

З іншого боку політика безпеки – це систематична, стабільна, організована та цілеспрямована ДІЯЛЬНІСТЬ власника ІТС відносно розв'язання проблем забезпечення безпеки інформації, що здійснюється або безпосередньо самим власником, або через відповідні механізми і органи управління та впливає на функціонування ІТС у цілому (активна складова). З позиції діяльнісного підходу політика безпеки повинна бути цілеспрямованою, стабільною, неперервною, організованою, керованою, узгодженою, мотивованою, забезпечувати максимальну результативність і ефективність при досягненні цілей безпеки і розв'язанні задач захисту, бути адекватною погрозам безпеки, придатною та гнучкою в реалізації, здійснюваною на різних рівнях забезпечення безпеки інформації, досить простою в адміністративному забезпеченні.

2. Основними складовими політики безпеки є: концепція і програма забезпечення безпеки інформації в ІТС, менеджмент, інжиніринг та аудит безпеки.

3. Правила безпеки важливі для забезпечення якісного управління безпекою, вибору номенклатури засобів захисту інформації, демонстрації активної підтримки власником процесів забезпечення безпеки інформації в ІТС, знищення організаційних і економічних перешкод створенню комплексної системи захисту інформації в ІТС, забезпечення послідовного і повного захисту інформаційних ресурсів на систематичній основі.

4. Загальний порядок розробки пасивної складової політики безпеки включає обробку вихідних даних відносно об'єкта захисту і середовища його експлуатації, визначення цілей і задач політики безпеки, розробку структури і конкретного змісту політики безпеки, визначення шляхів впровадження в життя та реалізації положень, правил, норм та вимог політики безпеки, а також визначення ступеня відповідальності за порушення вимог ПБ та контроль за їх виконанням.

5. Загальними задачами, на вирішення яких направлена політика безпеки, є:

- забезпечення конфіденційності, цілісності та доступності інформації в ІТС;
- безпечне функціонування ІТС із придатним рівнем ризику;
- управління інформацією та ресурсами з метою вдоволення експлуатаційних вимог до ІТС;
- здіснення цілеспрямованої та структурованої діяльності з тестування й оцінки безпеки за реалізації розроблюваних та запропонованих до застосування функцій та механізмів безпеки;
- здійснення аудиту безпеки, сертифікації та акредитації компонентів ІТС для прийняття рішення про можливість функціонування системи в захищеному режимі з вимагаємим рівнем ризику.

Список літератури: 1. *ISO/IEC 15408:2000* – Information technology – Security techniques – Evaluation criteria for IT security. – Part 1: Introduction and general model. 2. *ISO/IEC 15408:2000* – Information technology – Security techniques – Evaluation criteria for IT security. – Part 2: Security functional requirements. 3. *ISO/IEC 15408:2000* – Information technology – Security techniques – Evaluation criteria for IT security. – Part 3: Security assurance requirements. 4. *Бондаренко М.Ф., Черных С.П., Горбенко И.Д., Замула А.А., Ткач А.А.* Методологические основы концепции и политики безопасности информационных технологий // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 119. С. 5 – 17. 5. *НД ТЗИ 1.1 – 003 –99.* Терминология в области защиты информации в компьютерных системах от несанкционированного доступа. 6. *НД ТЗІ 2.5 – 004 – 99.* Критерии оценки защищенности информации в компьютерных системах от несанкционированного доступа. 7. *НД ТЗИ 3.7. – 001 – 99.* Методические указания по разработке технического задания на создание комплексной системы защиты информации в автоматизированной системе. 8. *NIST SP 800-12.* An Introduction to Computer Security: The NIST Handbook. 1998. 9. *ISO/IEC 17799:2001* – Information technology. – Information Security Management – Code of Practice for Information Security Management. 10. *OECD* Guidelines for Security of Information Systems and Networks. – OECD, 2002. 11. *NIST SP 800-14.* Generally Accepted Principles and Practices for securing Information Technology Systems (Principles and Practices). 2000. 12. *Потий А.В.* Политика безопасности: её типы и оценка // Служба безопасности. 2002. № 4, 5, 6. С. 18 – 20, 23 – 25, 27 – 31.

Харківський національний
університет радіоелектроніки
Державне управління справами
АО «Інститут інформаційних технологій»

Надійшла до редколегії 15.05.2003

УДК 681.3.06:519.248.681

И. Д. ГОРБЕНКО, д-р техн. наук, С. А. ГОЛОВАШИЧ, канд. техн. наук, А. Н. ЛЕПЕХА

СРАВНИТЕЛЬНЫЙ АНАЛИЗ БЛОЧНЫХ СИММЕТРИЧНЫХ ШИФРОВ, ПРЕДСТАВЛЕННЫХ В ПРОЕКТЕ NESSIE

Сегодня в области криптографических средств защиты информации происходит процесс обновления используемых алгоритмов криптозащиты: большинство наиболее распространённых алгоритмов уже морально устарели, в то время как новые алгоритмы требуют тщательного исследования и стандартизации. В связи с этим с 1998 в США был организован конкурс AES (Advanced Encryption Standard), направленный на выбор нового стандарта блочного симметричного шифрования (БСШ). В результате выполнения этого проекта в качестве нового стандарта был выбран алгоритм Rijndael (и на его основе принят стандарт США FIPS-197). В ноябре 2000 года прошел первый открытый семинар NESSIE (New European Schemes for Signatures, Integrity, and Encryption – Новые европейские схемы для электронных подписей, обеспечения целостности информации и шифрования, www.cryptonessie.org). Проект был начат под эгидой Европейской комиссии. Основными задачами проекта NESSIE является отбор лучших десяти криптографических примитивов. В этот набор входят алгоритмы блочного и поточного шифрования, генераторы случайных чисел, схемы быстрой аутентификации данных (MAC), хэш-функции и алгоритмы цифровой подписи. В качестве основных критериев отбора претендентов выбраны реальная безопасность, производительность, гибкость и требования рынка.

Проект NESSIE запланирован на три года и состоит из двух этапов. На перовой конференции в январе 2000 года на рассмотрение был вынесен список кандидатов для открытого обсуждения. В сентябре 2001 года завершилась первая фаза конкурса NESSIE и были объявлены первые результаты отбора. В течение второй фазы конкурса, которая завершится осенью 2002 года, планируется отобрать набор криптопримитивов для возможной стандартизации некоторых из них в будущем.

Целью данной статьи является изложение и анализ промежуточных результатов проекта NESSIE относительно блочных симметричных криптоалгоритмов. Это позволит сформулировать требования, которые сегодня предъявляются к БСШ различного уровня стойкости, а также определить направления развития симметричных шифров.

Исследования, проводимые криптологами мира, позволят обнаружить слабые и сильные стороны различных блочных симметричных шифров.

1 Основные требования

В связи с прогрессом в усовершенствовании методов выполнения криптоаналитических атак к кандидатам были предъявлены повышенные требования [1]. Для БСШ рассматривается три класса безопасности:

1. Высокий уровень безопасности – длина блока $l_б$=128 бит, длина ключа $l_к$=256 бит.
2. Нормальный уровень безопасности – длина блока $l_б$=128 бит, длина ключа $l_к$=128 бит.
3. Удовлетворительный уровень безопасности – длина блока $l_б$=64 бит, длина ключа $l_к$=128 бит.

Криптоалгоритм должен строиться на основе использования криптографии симметричных ключей. То есть ключи источника криптограмм и получателя должны или совпадать, или рассчитываться один из другого не выше чем с полиномиальной сложностью.

Каждый подаваемый на конкурс алгоритм должен содержать его описание, математическое описание, таблицы, диаграммы и описание параметров (размер слова, размер ключа,

требования к памяти, объем кода). Выбор используемых компонентов и параметров алгоритма разработчики должны обосновать и предоставить в том же документе.

Участвующие в проекте алгоритмы должны допускать как программную, так и аппаратную реализацию. Для оценки сложности аппаратной реализации рекомендуется использовать в качестве показателя – число логических элементов (вентилей); для программной реализации – тип процессора, тактовую частоту процессора и реализации на различных современных языках программирования и т.п.

В данной статье представлены результаты, касающиеся криптографических алгоритмов высокого и нормального уровня стойкости. Некоторые сведения о первоначально допущенных алгоритмах 1-го класса стойкости приведены в табл. 1.

Скорость (сложность) криптоалгоритма рекомендуется оценивать в виде числа тактов работы процессора, необходимых для:

— зашифрования одного блока данных;
— расшифрования одного блока данных;
— разворачивания ключа;
— настройки алгоритма или его части (например, формирования таблиц).

Таблица 1

Наименование алгоритма	Размер блока,бит	Размер слова,бит	Размер ключа,бит	Размер подключа,бит	Табл.подст./разм. табл. (бит х бит)	Сдвиг/перестановка + умножение, бит	XOR, ADD (бит)	AND, OR, NOT (бит)	Суммарная таблица подст.	Размер кода прогр., кб
Anubis	128	8	128-320	128* (R-1)	*	180+15* (R-12)	128	128	*	32
Camellia	128	8	128	1664	144 (8x64)	130	2(128 бит) 162(64 бит) 8(32 бит)	144(64 бит) 4(32 бит) 4(32 бит)	144	8,2
Camellia	128	8	192-256	2176	192 (8x64)	174	2(128 бит) 216(64 бит) 128(32 бит)	192(64 бит) 6(32 бит) 6(32 бит)	192	8,2
Grandcru	128	8	128	4224	50(5x8) 672(8x8)	144(8 бит) 160(8 бит)	1372(8 бит) 32(8 бит)	144(8 бит)	722	16
Hierocrypt-3	128	8	128	1792	16(8x8) 96(8x32) 80 (8x128)	144(32 бит)	120(32 бит) 76(128 бит)	192(32бит)	182	15
Hierocrypt-3	128	8	192	2048	16(8x8) 112(8x32) 96(8x128)	168(32 бит)	140(32 бит) 91(128 бит)	224(32бит)	224	15
Hierocrypt-3	128	8	256	2304	16(8x8) 128(8x32) 112(8x128	192(32бит)	160(32бит) 106(128бит)	256(32бит)	256	15
Noekeon	128	32	128	2048		128	16(128бит), 304(32бит)	32(32бит), 32(32бит), 32(32бит)	0	10,5

Продолжение таблицы 1

Nush128	128	32	128, 192, 256	4608		68(32бит)	212(32бит)	68(32бит)	0	15
Q	128	32		1280	128(8x8)	24	26(128бит), 144(32бит)	56(32бит) 16(32бит)	128	11,3
RC6	128	32	128	1408		40(32бит), 80(32бит) +40(32бит) умножен.	40(32бит), 84(32бит)		0	8
Safer++/128	128	8	128	1920	112(8x8)		15(64бит), 456(8бит)		112	35
Safer++/256	128	8	256	2688	160(8x8)		21(64бит) 648(8бит)		160	35
SC2000 (128)	128	32	128	1792	24(10x10) 48(11x11)	48(32бит)	87(32бит), 12(64бит), 14(128бит)	145(32бит)	72	20
SC2000 (192, 256)	128	32	192-256	2048	28(10x10) 56(11x11)	56(32бит)	98(32бит), 14(64бит), 16(128бит)	168(32бит)	84	20
Rijndael	128	8	128	1408	160(8x32)	30	11(128бит), 120(32бит)		160	
SHACAL-1	160	32	512	2560		224(32бит)	272(32бит), 320(32бит)	180(32бит)	0	8

* – 192+ 16*(R-12)

** – 6144+512*(R-12)

2 Критерии и показатели оценки качества

В первой фазе конкурса NESSIE к алгоритмам при отборе предъявлялись следующие критерии:

1. Защищенность алгоритмов от криптоаналитических атак. При этом основными методами криптоанализа являются: дифференциальный криптоанализ, расширения для дифференциального криптоанализа, поиск наилучшей дифференциальной характеристики, линейный криптоанализ; интерполяционное вторжение; вторжение с частичным угадыванием ключа; вторжение с использованием связанного ключа; вторжение на основе обработки сбоев; поиск лазеек.

2. Особенности конструкции и открытость структуры. Представленные криптоалгоритмы должны обладать понятной, легкоанализируемой структурой и основываться на надежных математических и криптографических принципах.

3. Устойчивость при модификации. Все кандидаты проверяются на устойчивость к различного рода модификациям: устойчивость к криптоаналитическим атакам при уменьшении числа циклов, сокращении компонентов используемых алгоритмом и т.п.

4. Статистическая безопасность криптографических алгоритмов.

5. Вычислительная сложность (скорость) зашифрования/расшифрования.

6. Сложность программной, аппаратной и программно-аппаратной реализации должна оцениваться объемом памяти как при программной, так и аппаратной реализации. В том числе, при программной – количеством необходимой оперативной памяти, размером исходного кода, скоростью работы программы на различных платформах при реализации на известных языках программирования. При аппаратной оценивается количеством вентилей и скоростью в Мб/с.

7. Универсальность криптоалгоритма:

— возможность работы с различными длинами начальных ключей и информационных блоков;

— безопасность реализации на различных платформах и приложениях.

— возможность использования криптографического алгоритма в основных режимах работы БСШ.

3 Защищенность БСШ от криптоаналитических атак

Стойкость против криптоаналитических атак при оценке криптографического алгоритма является самой важной характеристикой. Сравнительный анализ криптоалгоритмов по этой характеристике является достаточно сложной проблемой. В настоящее время наибольшую угрозу для БСШ представляют аналитические атаки, постороенные на основе дифференциального и линейного криптоанализа. Криптоаналитические атаки, которые сегодня представляют основную угрозу для БСШ, представлены во втором разделе данной статьи. Существует несколько критериев оценки стойкости криптоалгоритма к указанным выше атакам. Первым из них является критерий «идеальной безопасности». В данном случае предполагается, что криптоаналитик располагает неограниченными ресурсами как по времени, так и по объему памяти. Для того, чтобы криптоалгоритм удовлетворял данному критерию, необходимо, чтобы количество бит зашифрованных открытых текстов не превышало K/N, где K – количество бит ключа, N – количество бит блока. Однако на практике такой критерий не применим. Критерий доказуемой стойкости. На практике это означает, что криптоалгоритм может быть атакован при помощи решения какой-либо математической проблемы. Например, дискретного логарифма или факторизации. Такие модели чаще всего используются для оценки шифров, использующих открытые ключи. Для блочных симметричных криптоалгоритмов наибольшую опасность представляют атаки, построенные на видоизменениях дифференциального и линейного криптоанализа. Как правило, оценивается сложность по временным затратам, памяти и количестве необходимых данных. Алгоритмы, если для них не были обнаружены криптоаналитические атаки, оцениваются по нижней границе вычислительных затрат, необходимых на реализацию той или иной атаки. Другая модель, используемая при оценке криптопримитивов, базируется на безопасном времени. Данная модель позволяет оценить стойкость криптоалгоритма на протяжении некоторого времени.

Основной моделью, которая использовалась в конкурсе NESSIE, была практическая модель [2]. В ней оценивалось три параметра. Первый – это время как количество тактов, необходимых для успешного криптоанализа. Второй – это память как среднее число некоторых данных, которые необходимо получить для работы криптоаналитического алгоритма. Третий – это количество текстов, которые необходимо получить для успешной работы криптоаналитического алгоритма. Затраты на память считаются более дорогими, чем затраты на время. На практике при выборе криптоаналитического алгоритма жертвуют увеличением времени с уменьшением исходных данных. Атака считается успешной теоретически, если K бит ключа могут быть вычислены за время, меньшее времени прямого перебора ключа. Блочный симметричный криптоалгоритм можно считать безопасным, если не найдено атаки со сложностью «время»-»данные», меньшей, чем 2^K и 2^N соответственно. Найти эффективную атаку на новые криптоалгоритмы БСШ практически невозможно. Поэтому при сравнении БСШ реализуют атаки на криптопримитивы с уменьшенным количеством циклов или без криптографических преобразований.

В табл. 2 приведены результаты оценки атак на представленные в конкурсе NESSIE БСШ.

Таблица 2

Криптоалгоритм	Циклы	Потенциальная уязвимость	Наилучшая атака			
			Циклы	Техника	Время	Данные
Криптоалгоритмы прошедшие во второй тур						
Camellia	18	x^{-1} S-box потенциально открыты для алгебраических атак.	5	Невыполнимые дифференциалы		
			6	Square (с FL блоками)		2^{16}
			6	Square (без FL блоков)	2^{112}	$2^{11,7}$
			7	Невыполнимые дифференциалы (без FL)	$2^{55,6}$	2^{83}
			8	Усеченные дифференциалы (без FL блоков)	2^{202}	2^{60}
			9	Square + ключ (с FL блоками)		2^{105}
			9	Дифференциальный криптоанализ (без FL блоков)	2^{255}	2^{21}
			11	Дифференциалы высокого порядка (без FL блоков)	2^{256}	2^{93}
RC6	20	Малая пограничная полоса стойкости, уязвим через зависящие от данных сдвиги	16	Дифференциальный криптоанализ		2^{190}
			16	Линейный криптоанализ		2^{119}
			8	Линейный криптоанализ	2^{186}	2^{76}
			14	Мультилинейный криптоанализ		2^{120}
			18	Мультилинейный криптоанализ + 2^{-90} слабых ключей	2^{193}	2^{127}
			15	χ^2 - аттака (статистическая)		
			17	χ^2 - для 2^{-80} слабых ключей		
Rijndael	10 12 14	Маленькая граница безопасности. Потенциально открыт для алгебраических атак. Некоторые слабости в схеме разворачивания ключей.	5	Невыполнимые дифференциалы	2^{31}	$2^{29,5}$
			6	Square	2^{72}	2^{32}
			7	Коллизии (192, 256 бит ключ)	2^{140}	2^{32}
			7	Коллизии (128)	2^{128}	2^{32}
			7	Square (192 бит ключ)	2^{176}	2^{32}
			7	Square (256 бит ключ)	2^{192}	2^{32}
			6	Square	2^{44}	2^{32}
			7	Square (192 бит ключ)	2^{155}	2^{32}
			7	Square (256 бит ключ)	2^{172}	2^{32}
			7	Square	2^{120}	2^{128}
			8	Square (192 бит ключ)	2^{188}	2^{119}-2^{128}
			8	Square (256 бит ключ)	2^{204}	2^{119}-2^{128}
			9	Связанные ключи/ Square 256 связанных ключей (192 бит ключ)	2^{224}	2^{77}

Продолжение таблицы 2

Safer++	8	Негомоморфические линейные аппроксимации. Мало бит и байт количества ветвей для рассеивания.	3,5	Линейный криптоанализ (слабые ключи)		2^{121}
Криптоалгоритмы, не прошедшие во второй тур						
Anubis	8 + N		5	Невыполнимые дифференциалы	2^{31}	$2^{29,5}$
			6	Saturation (все ключи)	$6x2^{48}$	$6x2^{32}$
			7	Saturation (все ключи)	2^{120}	2^{128}
			7	Gilbert-Minier (ключ > 140)	2^{140}	2^{32}
			8	Saturation (ключ > 204)	2^{204}	2^{128}
Hierocrypt-3	6-8	Схема разворачивания ключей	2,5	Интеграл	2^{168}	2^{13}
			3	Интеграл	2^{40}	$6x2^{32}$
			3,5	Интеграл	2^{168}	$22x2^{32}$
Noekeon	16	S-box содержат идентичные функции. Много связанных ключей		Дифференциальный/линейный криптоанализ		
Nush128		Высоколинейный отклонения		Линейный криптоанализ	2^{K-1}	—
Q	8	Высокие дифференциальный и линейные вероятности	8	Дифференциальный		
			8	Линейный (128 битный ключ)	2^{77}	2^{105}
			9	Линейный (192 бит ключ)	2^{96}	2^{125}
			9	Линейный (256 бит ключ)	2^{128}	2^{125}
SC2000	6,5-7,5	Высокие диф. вероятности	3,5	Дифференциальный криптоанализ		
			4,5	Дифференциальный криптоанализ		
			4,5	Дифференциальный криптоанализ		

На алгоритм SAFER++ был осуществлен линейный криптоанализ [3]. Полученные результаты все еще не представляют угрозы для этого шифра (с любым блоком или ключом). Наилучшие результаты приведены в табл. 2. Атаки на 3-х и 4-х цикловый SAFER++ реально осуществимы для $2^6 - 2^{12}$ групп слабых ключей 256 бит длиной.

На SC2000 были осуществлены три типа атак (таблица 2). Дифференциальный криптоанализ был осуществлен на 4,5 – цикловый вариант SC2000 с 2^{110} парами откры-

тый/шифрованный текст и было получено 32 бита ключа первого и последнего циклового ключа, оставшиеся 96 бит были определены полным перебором [4].

В документе к криптоалгоритму SHACAL были представлены результаты дифференциального и линейного криптоанализа. Были обнаружены криптоаналитические характеристики на различном числе циклов. Для успешного осуществления атаки необходимо 2^{116} пар зашифрованный/открытый текст, а для линейного криптоанализа это число колеблется около 2^{80} [7].

Разработчики Hierocrypt первоначально представили успешную атаку на 2,5-цикловый вариант шифра, однако затем был найден улучшенный вариант этой атаки на 3,5 цикла с возможным расширением ее на большее число циклов [3].

При анализе Noekeon были обнаружены большие группы связанных ключей. Причем существование этих групп ключей одинаково для обеих схем распределения ключей. В колонке «вероятность» указаны вероятности появления дифференциалов, в колонке «ключи» указано количество ключей, в которых существует данный дифференциал (для любого ключа).

Криптоалгоритм Q был успешно атакован при помощи дифференциального криптоанализа. Eli Biham с учениками нашел дополнительную дифференциальную характеристику [5] кроме описанных разработчиками. Был предложен улучшенный вариант атаки для данного алгоритма, сокращающий сложность криптоаналитической атаки на основе дифференциального криптоанализа. При этом сложность анализа уменьшается пропорционально количеству выбранных пар открытого и зашифрованного текста и наоборот.

Проведенные исследования криптоалгоритма Anubis показали, что его стойкость против известных криптоаналитических атак не превышает заявленных авторами. Однако исследования также показали, что линейный криптоанализ дал эффективные результаты для алгоритма с большим числом циклов, чем было заявлено авторами. Eli Biham в своих исследованиях [6] высказал свое мнение о необходимости изучения выявленных отклонений для расширения данной атаки на большее число циклов.

Авторы Camellia представили 18-цикловый вариант алгоритма и утверждали, что даже 10-цикловый вариант безопасен. При исследованиях была обнаружена 3- и 7-цикловые характеристики с вероятностями 2^{-52} и 2^{-130} соответственно. Была атакована также 9-цикловая версия Camellia при помощи дифференциального криптоанализа. Линейный криптоанализ алгоритма не дал результатов [6].

RC6 успешно атакован при помощи дифференциального и линейного криптоанализа с уменьшенным числом циклов. Максимальная характеристика была обнаружена на 18-цикловом варианте алгоритма [7]. На данный момент RC6 является безопасным алгоритмом. Его безопасность основана на независимых циклических сдвигах.

4 Статистическая безопасность криптографических алгоритмов

Представленные на конкурс алгоритмы блочного шифрования были исследованы по критерию статистической безопасности. Это было сделано с целью выявления аномалий и зависимостей криптоалгоритмов, а также использования усовершенствованных атак. Для статистического исследования криптоалгоритмов использовался пакет NIST RIPE. Специально для конкурса NESSIE этот пакет был расширен дополнительными тестами и инструментами [8]. Для БСШ использовались наборы таких тестов, как корреляционный тест, тест линейной аппроксимации и тест корреляционного иммунитета (для S-box), а также тест линейных факторов.

1. Корреляционный тест вычисляет матрицу зависимостей и матрицу расстояний для блочного симметричного шифра.Также определяется степень «совершенства» (d_c), степень «лавинообразного» эффекта (d_a), строгий лавинный критерий (d_{sa}). Эти параметры должны принимать при тестировании криптоалгоритма на полном количестве циклов значения: $d_c=1$; $d_a \approx 1$; $d_{sa} \approx 1$. При таких значениях алгоритм считается прошедшим данный тест:

$$d_c = 1 - \frac{\#\{(i,j) \mid a_{ij} = 0\}}{nm}, \tag{1}$$

где: n, m – размерности матрицы зависимости (n x m);

a_{ij} – элементы матрицы зависимости,

$$a_{ij} = \#\left\{x \in (GF(2))^n \mid (f(x^{(i)}))_j \neq (f(x))_j\right\} \text{ для } i=1,\ldots,n;\ j=0,\ldots,m.$$

$$d_a = 1 - \frac{\sum_{i=1}^{n} \left| \frac{1}{\#X} \sum_{j=1}^{m} 2jb_{ij} - m \right|}{nm}, \tag{2}$$

где: $b_{ij} = \#\{x \in (GF(2))^n \mid w(F(x^{(i)}) - f(x)) = j\}$ для $i=1,\ldots,n;\ j=0,\ldots,m.$

$$d_{sa} = 1 - \frac{\sum_{i=1}^{n} \sum_{i=1}^{m} \left| \frac{2a_{ij}}{\#X} - 1 \right|}{nm}. \tag{3}$$

Полнота проявляется в том случае, если каждый выходной бит зависит от каждого входного. Лавинообразный эффект должен приводить к изменению в среднем половины выходных битов. Критерий распространения (строгий лавинный критерий) характеризует зависимость выходных значений от входных векторов с различным весом Хемминга.

2. Тесты линейной аппроксимации и корреляционного иммунитета основаны на быстром преобразовании Уолша. Эти тесты используются для тестирования S-блоков БСШ. Пусть имеется некоторая функция f на V_n, где $V_n = (0,1)^n$. Функция f имеет корреляционный иммунитет порядка k, если ее преобразование Уолша $F(w)$ функции f над V_n определяется как принимающая действительные значения функция (4)

$$F(w) = 2^{-n} \sum_x (-1)^{f(x) \oplus \langle w,x \rangle}, \tag{4}$$

где $w \in V_n$.

3. Тест линейных факторов определяет линейную характеристику (зависимость), которая сохраняется между битами открытого текста, ключа и криптограммы.

Все криптоалгоритмы тестировались в режиме обратной связи по выходу. Для тестирования использовался статистический пакет RIPE, который специально для конкурса NESSIE был видоизменен. Кроме того, криптоалгоритмы прошли тестирование в режиме счетчика.

Далее приводятся результаты тестирования для БСШ высокого уровня стойкости, которые прошли первый отборочный тур. Это криптоалгоритмы SAFER++, Camellia, RC6, SHACAL и Rijndael.

При проведении первого этапа статистического исследования получены следующие результаты (табл. 3).

Таблица 3

Тип теста	Camellia	SAFER++	RC6	Rijndael	SHACAL
Корреляционный:					
кол.вх.векторов	1000	1000	1000	1000	1000
кол.бит кот.измен	63,995267	64,003016	64,00608	63,994571	63,993161
d_c	1,000000	1,0000000	1,000000	1,000000	1,000000
d_a	0,999239	0,999250	0,999269	0,999246	0,999321
d_{sa}	0,992012	0,992005	0,991951	0,991925	0,991965

Результаты анализа для всех алгоритмов показали, что отклонений от случайной последовательности не наблюдается.

Тест на поиск линейных комбинаций (тест линейных факторов) между зашифрованным текстом, открытым текстом и ключами не выявил каких-либо зависимостей среди оставшихся БСЩ.

На втором этапе производилось статистическое исследование представленнх криптоалгоритмов на версиях с меньшим числом циклов. Целью такого исследования было определение, на каком числе циклов перестанут появляться статистические зависимости. Результаты этого исследования сведены в табл. 4, где указывается количество циклов, после которого данный критерий удовлетворяет необходимому условию.

Таблица 4

Наименование шифра	Число бит, которые изменились	Полнота, d_c	Критерий распространения, d_a	Обнаруженные линейные факторы, d_{sa}
Camellia	0 +0,5 – 1,00000	0,007812	0,015625	0,000000
	1 +0,5– 11,958142	0,179688	0,186846	0,165106
	2 +0,5– 38,416694	0,597656	0,600261	0,578002
	3 +0,5– 58,952955	0,921875	0,920817	0,909080
	4 + 0,5– 63,555463	1,000000	0,999271	0,991995
	5 + 0,5– 64,004316	1,000000	0,999254	0,991951
SAFER++	0 + 0,5 – 1,241775	0,021484	0,019403	0,002794
	1 + 0,5 – 61,752857	0,953125	0,951608	0,900645
	2 + 0,5 – 63,996523	1,000000	0,999224	0,991944
	3 + 0,5 – 63,996395	1,000000	0,999276	0,991985
	4 + 0,5 – 64,009339	1,000000	0,999288	0,991959
	5 + 0,5 – 63,995669	1,000000	0,999291	0,991968
RC6	0,5– 1,934266	0,086853	0,030223	0,010645
	1+0,5–14,334177	0,417847	0,223972	0,213940
	2+0,5–41,784901	0,875000	0,652889	0,651275
	3+0,5–60,301572	1,000000	0,942141	0,937845
	4+0,5–63,923282	1,000000	0,998479	0,991661
	5+0,5–63,994611	1,000000	0,999273	0,992041
Rijndael	0 + 1 – 4,040025	0,062500	0,063125	0,059129
	1 + 1 – 16,064059	0,250000	0,251001	0,247672
	2 + 1– 64,247014	1,000000	0,996140	0,991466
	3 + 1– 64,001791	1,000000	0,999350	0,992043
SHACAL	0 + 0,5 – 4,170949	0,252136	0,065171	0,045457
	1 + 0,5 – 7,846366	0,448914	0,122599	0,097094
	2 + 0,5– 13,574798	0,688721	0,212106	0,180149
	3 + 0,5– 20,962220	0,861023	0,327535	0,286454
	4 + 0,5– 29,419141	0,951843	0,459674	0,418031
	5 + 0,5– 38,451393	0,993286	0,600803	0,560878
	6 + 0,5– 46,865140	0,999817	0,732268	0,698457
	7 + 0,5– 53,556009	1,000000	0,836813	0,813627
	8 + 0,5– 58,392201	1,000000	0,912378	0,896313
	9 + 0,5– 61,327378	1,000000	0,958240	0,948508
	10 + 0,5– 62,932188	1,000000	0,983313	0,975846
	11+ 0,5– 63,635425	1,000000	0,994199	0,987382
	12+ 0,5– 63,900275	1,000000	0,998219	0,990990
	13+ 0,5– 63,980032	1,000000	0,999127	0,991881

Анализ полученных результатов показывает, что для Camellia полнота (d_c) наступает после 3 + 0,5 цикла; критерий распространенности выше 0,98 после 3 + 0,5 циклов; линейные факторы исчезают после 4+ 0,5 цикла.

Исследования БСШ SAFER++ показали, что полнота (d_c) наступает после 2 + 0,5 цикла; критерий распространенности выше 0,98 после 2 + 0,5 циклов; линейные факторы исчезают после 2 + 0,5 цикла.

Для RC6 получены следующие данные. Полнота (d_c) наступает после 4 + 0,5 цикла; критерий распространенности выше 0,98 после 4 + 0,5 циклов; линейные факторы исчезают после 3 + 0,5 цикла.

Полнота (d_c) для Rijendael наступает после 2 + 1 цикла; критерий распространенности выше 0,98 после 3 + 1 циклов; линейные факторы исчезают после 3 + 1 цикла.

Анализ статистических исследований для SHACAL показывает, что полнота (d_c) наступает после 8 + 0,5 цикла; критерий распространенности выше 0,98 после 12 + 0,5 циклов; линейные факторы исчезают после 12 + 0,5 цикла.

Было проведено также тестирование криптоалгоритмов в режиме обратной связи по выходу. В рамках этого теста применялись следующие тесты. Частотный (монобитный) тест, тест на выявление коллизий, тест m-мерных перекрытий, тест на наличие интервалов (нулей), тест на непрерывающиеся последовательности, универсальный тест Маурера, тест Покера, быстрый спектральный тест, корреляционный тест, проверка ранга матрицы, проверка линейной сложности, определение максимального порядка сложности, проверка сжатия по алгоритму Лемпеля-Зива, тест двухэлементной сложности (dyadic complexity test), персоляционный тест, тест постоянных серий. В виду большого объема полученных данных, они здесь не приводятся. Исследования показали, что представленные последовательности не отличаются от случайных.

Также было проведено тестирование БСШ Camellia, SAFER++, RC6, Rijndael и SHACAL в режиме «счетчика». Для исследований применялись описанные выше тесты. Результаты исследования показали, что гаммы шифрующие и криптограммы не отличаются от случайных, что подтверждает их хорошие криптографические свойства.

5 Вычислительная сложность криптографических преобразований при программной и аппаратной реализации

Производительность БСШ после требований к безопасности криптоалгоритма является второй по важности характеристикой. По требованиям конкурса NESSIE криптоалгоритм должен удовлетворять требованиям программной и аппаратной реализации, сложность которых была бы сопоставима со сложностью DES. Как уже отмечалось выше при программной реализации все криптоалгоритмы сравниваются по размеру кода: на различных языках программирования, размеру исполняемого файла, объему потребляемой программой памяти и скорости выполнения криптографических операций. При аппаратной реализации учитываются площадь, занимаемая реализацией алгоритма на кристалле, и скорость выполнения криптографических операций. Под криптографическими операциями понимаются операции разворачивания ключа, а также выполнения прямых и обратных криптографических преобразований.

Тестировались только криптоалгоритмы, относящиеся к высокому уровню безопасности и прошедшие первый тур конкурса NESSIE. Проверка программной реализации проводилась в два этапа. На первом было сгенерировано 100000 случайных ключей и 100000 открытых текстов. На втором этапе генерировалось 100 случайных ключей, которые потом разворачивались при использовании схем разворачивания ключей тестируемых криптоалгоритмов. Оценивалось время разворачивания. Далее производилось зашифрование сгенерированных открытых текстов с использованием этих ключей. Оценивалось время, затрачиваемое на за-

шифрование информации. Потом производилось расшифрование и производилась оценка времени, затрачиваемого на эту операцию. Полученные после расшифровании тексты сверялись с исходными на соответствие. Время измерялось в количестве тактов, необходимых процессору для выполнения криптографического преобразования над одним байтом.

Тестирование выполнялось на процессорах типа Pentium III – 850, 256 MB ОЗУ, Windows 2000 with Visual C++; на Pentium III Xeon @550 MHz, Visual C++ 6.0. Результаты сведены в табл. 5.

Таблица 5

Наименование криптоалгоритма	Разворачивание ключа, тыс.тактов/ключ		Зашифрование, циклов/байт		Расшифрование, циклов/байт	
	Xeon	PIII	Xeon	PIII	Xeon	PIII
Safer++/128	2,9	6,1	89	160	93	314
Camellia/128	3,0	4,2	172	162	238	212
RC6/128	2,9	6,1	49	53	49	52
Rijndael/256	0,53/2,1	0,68/2,0	59	54	69	56
Shacal/512	1	2,7	62	66	нет	нет
SC2000 (128)	1,2	1,5	475	440	480	450
Nush128	2,8	2,3	45	110	42	90
Grandcru/128	150	181	3400	2600	5400	3900
Anubis	4,2	5,5	50	53	51	54
DES	16	17	103	117	98	111
Hierocrypt-3/128	183	605	63	150	70	170

Как видно из таблицы, криптоалгоритмы Grandcru и SC2000 имеют низкие показатели скорости, вследствие чего они не прошли во второй тур. Rijndael имеет две схемы разворачивания ключей – одну для зашифрования, другую для расшифрования. Наиболее быстрым среди алгоритмов являются БСШ RC6 и Rijendael, которые положительно оценены еще при проведении конкурса AES.

Важным является исследование представленных криптоалгоритмов в аппаратном исполнении для выявления возможности реализации в ASIC- и FPGA-исполнении. В табл. 6 приведены некоторые из криптоалгоритмов в «быстром исполнении», для сравнения приведены некоторые кандидаты из конкурса AES.

Таблица 6

Наименование алгоритма	Площадь на кристалле (вентилей)		Время разворачивания ключа (нс)	Скорость (Мб/с)
	Зашифрование/расшифрование	Общая логика		
DES	42,204	54,405	—	1161,31
Triple-DES	124,888	128,147	—	407,40
MARS	690,654	2,935,754	1740,99	225,55
RC6	741,641	1,643,037	2112,26	203,96
Rijndael	518,508	612,834	57,39	1950,03
Twofish	200,165	431,857	16,38	394,08
Camellia	216	272,819	24,36	1170,50

В таблице 7 приведены не оптимизированные данные для NESSIE конкурсантов в ASIC- и FPGA-исполнении.

Таблица 7

Криптоалгоритм	Тип аппаратной реализации	Скорость, Мб/с	Детали
Camellia	ASIC 0,35μ ASIC 0,35μ FPGA XC4000XLμ	1170 220 122	273 тыс. вентилей 11 тыс. вентилей
Hierocrypt-3	ASIC 0,14μ 126MHz ASIC 0,14μ 185MHz FPGA 7,4MHz	897 85 53	81.5 тыс. вентилей 26.7 тыс. вентилей
RC6-128	FPGA XCV1000 14MHz FPGA XCV1000 38MHz ASIC 0,5μ	127 2400 104 2200 204	
Rijndael-128	FPGA XCV1000 14MHz FPGA XCV1000 32MHz ASIC 0,5μ	300 1940 524 5100 1950	 613 тыс. вентилей

Анализ представленных в табл. 6 и 7 данных, позволяет сделать вывод, что при аппаратной реализации имеют предпочтение Camellia и Rijndael.

6 Описание криптографических схем среди БСШ 1-го класса стойкости, прошедших первый тур конкурса NESSIE

Первоначально к рассмотрению были допущены следующие криптоалгоритмы: CS-Cipher, Hierocrypt-L1, IDEA, Khazad, MISTY1, Nimbus (алгоритмы, относящиеся к 3-й категории), Grand Cru, Noekeon, SHACAL (алгоритмы, относящиеся ко 2-й категории), NUSH, SAFER++, Hierocrypt-3, Q, RC6, SC2000, Anubis, Camellia, Rijndael-256 (алгоритмы, относящиеся к 1-й частично ко 2-й категории).

К сентябрю 2001 года была завершена первая фаза конкурса и отобраны следующие криптоалгоритмы: нормального и высокого уровня стойкости – SAFER++, Camellia, RC6, Rijndael-256, SHACAL. Удовлетворительного уровня стойкости – IDEA, Khazad, MISTY1.

На основе представленных в предыдущих разделах материалов можно выявить причины, по которым не прошли конкурс те или иные криптоалгоритмы. CS-Cipher был отброшен вследствие низкой скорости (для всех кандидатов предъявлялись требования по скорости, сопоставимые со скоростями участников AES), Hierocrypt-L1 и Hierocrypt-3 имеют в основе одинаковую схему, но на них была проведена SQUARE-подобная атака, которая прошла на 3,5- и 6-цикловом вариантах данного криптоалгоритма. Также проведенные исследования показали, что алгоритм имеет плохие показатели скорости и были найдены слабые места в схеме разворачивания ключей. Криптоалгоритм Nimbus был успешно атакован при помощи атаки на основе выбранных открытых текстов. Росийский криптоалгоритм NUSH, предложенный компанией ЛАН-Крипто, показал предельно минимальный уровень безопасности и атака с помощью линейного криптоанализа имеет меньшую сложность, чем атака «грубая

сила», вследствие чего криптоалгоритм не прошел во второй тур. Среди высокостойких блочных криптоалгоритмов сложилась следующая ситуация. Grand Cru, базируется на AES, однако показал плохие результаты по скорости. На Q была найдена атака намного быстрее, чем атака «грубая сила». Noekeon показал плохие результаты против атаки на связанных ключах. Криптоалгоритмы SC2000 и Anubis качественно не отличаются от AES Rijendael.

Рассмотрим кратко оставшиеся криптоалгоритмы высокого уровня стойкости, их конструктивные особенности, преимущества и недостатки.

В криптоалгоритме SAFER++ не используется структура типа цепи Фейстеля. Шифр работает с блоками длинной 128 бит и длинами ключей 128, 256 и 512 бит. Процедуры зашифрования/расшифрования представляют собой последовательность итераций, число которых зависит от длины ключа и равно 8, 12 и 16 соответственно. В общем, работу шифра можно описать следующим образом: итерация зашифрования состоит из слоя подмешивания подключа, слоя нелинейной замены и еще одного слоя подмешивания ключа и линейного обратимого преобразования [10]. Интересным является решение линейного обратимого перемешивания, которое реализовано в виде умножения текстового блока на специальную невырожденную матрицу. Это преобразование называется точечным псевдопреобразованием Адамара. При этом все операции выполняются побайтно по модулю 256. Достоинствами криптоалгоритма является то, что он не использует сложных вычислительных операций, а использует побайтовые операции, что является предпочтительным при аппаратной реализации. Однако алгоритм не является инволютивным, т.е. при расшифровании необходимы другие операции (вычитание), что увеличивает затраты на реализацию. Криптоалгоритм имеет открытую и легкоанализируемую структуру. В частности, проведенные Nakahara J.Jr [11] исследования на устойчивость данной структуры против линейного криптоанализа подтвердили стойкость. К недостаткам данного алгоритма можно отнести следующее. Он имеет низкие показатели по скорости при реализации на 32- и 64-битных процессорах. SAFER++ имеет запутанную и сложную схему разворачивания ключа, что приводит к большим затратам времени при разворачивании ключа, а также может стать причиной ошибок при реализации. Это приводит к тому, что криптографическая безопасность алгоритма основана только на надежности используемой математической базы. Потому для данного алгоритма отсутствуют доказательства полной защиты. SAFER++ является дальнейшим развитием давно известного SAFER-K64 и представленного на конкурсе AES SAFER+. Как показали исследования на схему разворачивания ключа SAFER+ [10], была успешно осуществлена атака со связанными ключами. В настоящий момент схема претерпела некоторые несущественные изменения, однако в самой основе она аналогична SAFER+.

Другим претендентом среди БСШ является RC6, который выдвигался еще на AES [12]. Этот криптоалгоритм является полностью параметризованным. Во всех вариантах RC6 работает с четырьмя n-битными блоками, используя 6 базовых операций: сложение и вычитание по модулю 2^w, сложение по модулю 2, целочисленное умножение по модулю 2^w и циклические сдвиги вправо и влево. В целом криптоалгоритм имеет легкоанализируемую и наглядную схему. Алгоритм обладает высокой скоростью при реализации на широкораспространенных процессорах PII, PIII, Athlon (см. табл. 5). Он подходит также и для программной реализации, в частности на языке Java. Однако использование модулярных операций сложения и умножения плохо реализуется на некоторых 8-битовых процессорах, используемых в смарт-картах, что также играет немаловажную роль (табл. 1).

Структура алгоритма Camellia выбрана, исходя из возможности эффективной как программной, так и аппаратной реализации [13]. Цикловое преобразование построено на основе хорошо известной схемы Фейстеля, что позволило разработчикам получить «инволютивный» шифр и значительно сократить затраты на его аппаратную реализацию. Однако в отличие от классической цепи Фейстеля, через каждые 6 циклов итеративного преобразования в обеих ветвях цепи выполняется Фейстель-подобное линейное преобразование над «четвер-

тушками» блока. Такая модификация не нарушает свойство «инволютивности» шифра, и по утверждению разработчиков, позволяет увеличить сложность атак линейного и дифференциального криптоанализа за счёт того, что линейные и дифференциальные характеристики (и соответствующие им вероятности) становятся ключезависимыми. Алгоритм является байт-ориентированным и может быть реализован на основе байтовых таблиц подстановки (реализация S-блоков) и простых логических операций, которые могут быть легко реализованы на большинстве современных платформ. Поэтому алгоритм может быть эффективно реализован как на высокопроизводительных 32- и 64-битных процессорах, так и на 8-битных процессорах, используемых в смарт-картах. Все S-блоки могут быть легко получены непосредственно из основной таблицы подстановки во время работы алгоритма, что значительно снижает затраты на память и что важно при аппаратной реализации. Интересное решение используется разработчиками в схеме разворачивания ключей – все подключи генерируются налету, т.е. нет необходимости генерировать их заранее с последующим сохранением в памяти. Хотя Camellia незначительно уступает некоторым другим конкурсантам в производительности на мощных 32-, 64-битных процессорах, он обеспечивает значительные результаты при реализации на спецпроцессорах (табл. 1).

Алгоритм Rijndael, представлен вариантом с длиной блока 256 бит и достаточно описан в различной литературе. Все операции, используемые в криптоалгоритме, проводятся в поле Галуа GF (2^8). Данный алгоритм достаточно хорошо исследован и показал приемлемые результаты по всем тестам. Алгоритм обладает приемлемой скоростью работы как при программной, так и при аппаратной реализации. Он незначительно проигрывает при аппаратной реализации Camellia по занимаемой на кристалле площади. Недостатками алгоритма можно считать не совсем удачную схему разворачивания ключа. Алгоритм не имеет открытой легко анализируемой структуры, что не исключает в нем закладок или ошибок. Другой недостаток – это восприимчивость к атакам, связанным с реализацией. Также исследования NESSIE указали, что данный криптоалгоритм имеет низкую границу безопасности. Это может снизить время безопасности криптоалгоритма до появления улучшенных методов и средств криптоанализа.

8 Выводы

Стойкость БСШ против криптоаналитических атак является определяюшей его характеристикой. В настоящее время наибольшую угрозу для БСШ представляют аналитические атаки, постороенные на основе дифференциального и линейного криптоанализа, как правило на их видоизменениях. В проекте NESSIE принята практическая модель криптоанализа, при которой основными являются следующие параметры: время в виде количества тактов, необходимый объем памяти и количество текстов, которые необходимы для выполнения криптоанализа.

Атака на БСШ может считаться успешной, если ее сложность не меньше сложности атаки типа «грубая сила». Найти аналитическую атаку на полноцикловый БСШ сегодня практически невозможно. Поэтому сравнение БСШ по стойкости можно проводить на уменьшенном числе циклов.

Лучшими по показателю реальной и статистической безопасности могут быть БСШ Camellia, Rijndael, SAFER++, RC6 и SHACAL.

Статистическая безопасность названных БСШ проверялась с использованием корреляционного теста, а также тестов линейной аппроксимации и линейных факторов. Было также проведено статистическое тестирование БСШ в режимах «счетчика» и обратной связи по шифртексту. Результаты исследования показали, что гаммы шифрующие и криптограммы не отличаются от случайных, что подтверждает их хорошие свойства.

Основным недостатком БСШ SAFER++ является то, что он не является инволютивным, имеет худшие показатели по скорости при реализации на 32- и 64-битных процессорах. Кроме того, SAFER++ имеет запутанную и сложную схему разворачивания ключа. Поэтому вряд ли БСШ SAFER++ станет победителем, хотя он обеспечивает высокую стойкость.

БСШ RC6 обеспечивает высокую стойкость, имеет легко анализируемую и наглядную схему, обеспечивает высокую скорость при реализации на широко распространенных процессорах. Он эффективно может быть реализован программно. Однако модулярные операции сложения и умножения плохо реализуются на некоторых 8-битовых процессорах.

Проведенные исследования и изложенное выше позволяют сделать вывод о предпочтительности БСШ Rijndael (AES), Camellia и SHACAL.

Список литературы: 1. *Report* on the NESSIE call. 1999. 2. *Description* of Methodology for Security Evaluation // NESSIE home page. http://www.cryptonessie.org, 1999. 3. *Bart VanRompay, Vincent Rijment, Jorge Nakahara.* A first report on CS-Ciper, Hierocrypt, Grand Cru, SAFER++ and SHACAL /NESSIE home page. http://www.cryptonessie.org, 2001. 4. *Lars R. Knudsen* A Differential Attack on Reduced-Round SC2000, Department of Informatics, University of Bergen, 2001. 5. *Ali Biham.* Differential cryptoanalisis of Q //NESSIE home page. http://www.cryptonessie.org, 2001. 6. *Ali Biham.* Preliminary report on the NESSIE submission Anubis, Camellia, Q, Nimbus. // NESSIE home page. http://www.cryptonessie.org, 2001. 7. *NESIE security report* // NESSIE home page. http://www.cryptonessie.org, 2003. 8. *Потий А.В., Орлова С.Ю. и др.* Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NIST STS. // Радиотехника: Всеукр межвед. науч.-техн. сб. 2000. Вып. 114. С. 14. *9. Nomination* of SAFER++ as candidate algorithm for the NESSIE, 2000. 10. *Nakahara J. Jr.* Cryptanalysis of reduced-round SAFER++, 2001. *11. Schneier B., Kelsey J.* Key-Schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, 2001. *12. RC6 – Block Cipher*, 2000. 13. *Camellia:* A 128 bit block cipher suitable for multiple platforms, 2000.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 19.05.2003

УДК 621.3.06

Е. В. ПОПОВИЧ, А. В. ПОТИЙ, канд. техн. наук

МЕТОДИКА ОПЕРАТИВНОГО СТАТИСТИЧЕСКОГО ТЕСТИРОВАНИЯ СЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ БОЛЬШОЙ ДЛИНЫ

Шифры с одноразовой гаммой, как единственные теоретически стойкие шифры, всегда привлекали внимание разработчиков. Теоретическая стойкость таких шифров была доказана Шенноном [1], однако сложность их практической реализации сдерживает их активное использование на практике. Рассматривая особенности данных систем, следует отметить, что их стойкость полностью определяется качеством одноразовой гаммы [2].

Достижения в технологиях хранения данных большого объема и использование их в современных информационных технологиях несколько облегчает решение практических проблем реализации схем типа шифр Вернама. Однако остается нерешенной задача получения доказательств «качества» одноразовой гаммы большой длинны, записанной на носитель.

На практике возникает проблема создания качественного ключа – случайной гаммы заданного качества. Если учитывать значительный объем гаммы (десятки и сотни мегабайт), то наиболее очевидны такие проблемы:

1. Каково максимально допустимое время тестирования последовательности?
2. Какие статистические тесты должны применяться для тестирования?
3. Сколько тестов необходимо применить для получения валидной оценки?

Ответы на эти вопросы позволяют уточнить технические характеристики систем защиты: пропускную способность, период смены ключей, количество ключей, необходимое для поддержки заданной конфигурации сети.

Сегодня уже предложен ряд пакетов статистического тестирования псевдослучайных последовательностей (ПСП). Это пакеты NIST STS, DIEHARD, RIPE, Crypt-X, которые предлагают достаточно обоснованные методику и набор статистических тестов. Однако, они ориентированы прежде всего на, если так можно выразиться, аттестацию источника псевдослучайной последовательности. Проведенный авторами поиск позволил сделать вывод о том, что средств и методик проверки качества отдельно взятой ПСП большого периода (более 10^9 бит) на сегодняшний день не существует. Необходимость в ряде случаев практической реализации систем, подобных системе Вернама, делает задачу создания таких методик достаточно актуальной.

В данной работе предлагается методика оперативного статистического тестирования ПСП большой длинны. Методика основана на использовании стандартного пакета тестов NIST STS. Для выбора минимально необходимого набора тестов предлагается новый показатель оценки результатов тестирования – коэффициент крутизны статистической характеристики.

1 Этапы жизненного цикла формирования ПСП большого периода

В процессах формирования случайных последовательностей большой длины (СПБД) можно выделить четыре основных этапа:

1. Проектирование и создание генератора (источника) СПБД.
2. Приемка разработанных (оценка существующих) генераторов и постановка на эксплуатацию.
3. Получение случайной последовательности и оперативное тестирование работоспособности генератора во время эксплуатации.
4. Периодическое, детальное тестирование генератора.

Каждый из этих этапов характеризуется перечнем основных решаемых задач и требований к порядку и условиям их решения.

1. Проектирование и создание генераторов. Основная задача – создание генератора. Характеризуется отсутствием жестких требований к временным параметрам. Это дает возможность использовать большое (неограниченное) количество разнообразных тестов. Набор тестов и методика тестирования (дальше методика тестирования) должны давать детальную (инструментальную) картину генератора, а также позволять детально сравнивать несколько результатов тестирования с целью выявления изменений в ходе разработки.

2. Приемка разработанных (оценка существующих) генераторов и постановка на эксплуатацию. Основная задача – определить, отвечает ли генератор заданным требованиям. Характеризуется отсутствием жестких требований к временным характеристикам, что дает возможность использовать большое количество тестов. В отличие от первого этапа, методика тестирования должна дать оценку генератора, по которой принимается решение о пригодности.

3. Получение заданной последовательности и оперативное тестирование работоспособности генератора во время эксплуатации. Основная задача – получить случайную последовательность с заданными параметрами. Вторая задача – осуществление оперативного контроля работоспособности и исправности генератора. Характеризуется, как правило, жесткими временными ограничениями. На данном этапе вступают в противоречие требования по скорости получения случайной последовательности и ее качеству. Методика тестирования должна обеспечить отбраковку последовательностей, не удовлетворяющих заданным требованиям. При изменении количества брака принимается решение о работоспособности генератора – выходе генератора из строя.

4. Периодическое, детальное тестирование генератора. Основная задача – оценить отклонения в свойствах генератора и оценить его пригодность к дальнейшей эксплуатации. Характеризуется нежесткими требованиями по времени, что дает возможность использовать расширенное количество тестов по сравнению с третьим этапом. Методика тестирования должна выявлять отклонения и сбои в работе генератора с целью принятия решения об исправности и прогнозирования линии его дальнейшего поведения.

Очевидно, что набор тестов и методик тестирования на каждом этапе должны коррелировать с назначением генератора. Следует обратить внимание на то, что методики тестирования на первом и четвертом этапах направлены на выявление технических особенностей генераторов. В то время как методики тестирования на втором и третьем этапах направлены на решение конечной задачи – получение случайных последовательностей с заданным качеством.

Анализ и практическое использование существующих пактеов и методик статистического тестирования СПБД показал, что пакеты NIST STS, DIEHARD, RIPE и Crypt-X предназначены для оценки существующих генераторов, т.е. применимы на втором этапе жизненного цикла. В американском стандарте FIPS-140-2 предложена методика для оперативного тестирования работоспособности генератора во время работы [3]. Однако методика FIPS-140-2 непригодна для тестирования СПБД, поскольку по требованиям стандарта тестированию подлежат последовательности длиной $L = 2*10^4$. Принятие решения осуществляется по интервальному критерию, который обладает недостаточной гибкостью. Это ограничивает возможности данной методики относительно формирования надежного решения о качестве СПБД.

2 Методика оперативного тестирования СПБД

Рассматривая подходы к созданию методик оперативного тестирования, можно выделить несколько основных этапов:

1. Определение требований к случайной последовательности.
2. Определение критериев принятия решения о соответствии последовательности требованиям.

3. Определение перечня тестов и порядка их использования для реализации критерия.

При построении методик оперативного тестирования необходимо учитывать, что качество последовательности определяется количеством и содержанием тестов, которым она подвергается. С другой стороны, чем больше тестов, тем больше ошибка первого рода и соответственно меньше КПД[1] генератора. Хотя ошибка второго рода в этом случае тоже уменьшается, что может являться определяющим для некоторого класса систем.

В [4] приведена оценка ошибки первого A рода в зависимости от количества независимых тестов n и уровня значимости α:

$$A = 1 - (1 - \alpha)^n. \qquad (1)$$

Исходя из (1) видно, что для пакетов NIST STS, DIEHARD, RIPE ошибка первого рода является достаточно большой, что приводит к практической неприменимости данных методик тестирования для получения последовательности с заданным качеством и оперативного тестирования работоспособности генератора во время работы. В частности для NIST STS ошибка первого рода составляет 0.85. Результаты практических испытаний NIST STS показывают, что из генеральной выборки, состоящей из 100 последовательностей[2], все 189 тестов проходят примерно 16-18 (КПД = 16 – 18%), при этом вся генеральная выборка проходит тестирование данным пакетом.

Для снижения ошибки первого рода необходимо уменьшать количество тестов. Исходя из формулы [4]

$$n = \left[-\alpha^{-1} \cdot \ln(1 - A) \right], \qquad (2)$$

при ошибке первого рода A=0,1 и уровне значимости α = 0,01 получаем количество тестов n = 10.

Для разработки методики оперативного тестирования в качестве базового набора тестов предлагается использовать тесты NIST STS. Такой выбор сделан в связи с тем, что по сравнению с остальными пакетами для NIST STS существует доказательство независимости используемых тестов [5].

Как отмечалось, в состав данного пакета входит 16 базовых тестов, некоторые из которых имеют несколько «подтестов». Общее количество проверок составляет 189. В связи с этим при разработке методики оперативного тестирования на базе NIST STS необходимо определить тесты, которые необходимо будет исключить. В качестве показателя, на основе которого будет приниматься решение об исключении того или иного теста, предлагается выбрать чувствительность каждого теста к определенным искажениям в тестируемых последовательностях. Для определения данного показателя предлагается следующая методика оценки:

1. Выбирается генеральная выборка длиной 10^8 бит, которая проходит тестирование пакетом NIST STS. В качестве параметров тестов выбираются параметры, рекомендуемые в [5].

---

[1] Под КПД генератора будем понимать отношение количества последовательностей заданной длины, прошедших тестирование N_t, к общему количеству протестированных последовательностей N_o. $КПД = \frac{N_t}{N_o} \cdot 100\%$

[2] В качестве датчика случайных чисел использовалось изделие "Гряда-31".

2. В каждую последовательность длиной 10^6 бит вносятся $I, I = \{1,10,100\}$ блоков искажений. Суммарный процент искажений составляет от 0,1 – 1%[3] с шагом 0,1%. Для испытаний, исходя из предложенного варианта криптоанализа [1], использовались два типа искажающих последовательностей:

– сплошные нули[4] 000000…;

– меандр 01010101….

3. Каждая генеральная выборка подвергается тестированию.

4. По результатам тестирования оценивается количество последовательностей N, которые прошли тестирование конкретным тестом.

Для тестов с более чем одним «подтестом» N вычислялось как среднее арифметическое.

С целью качественной оценки результатов тестирования введем коэффициент крутизны статистической характеристики K_S, который определяется согласно выражению

$$K_S = \frac{\left(N_0 - N_1\right)}{P_1}, \tag{3}$$

где: N_0 – начальное значение количества пропущенных последовательностей при отсутствии искажений;

N_1 – количество пропущенных последовательностей при одном проценте искажений;

P_1 – значение процента искажений, при котором количество пропущенных последовательностей равно нулю, в противном случае P_1=1.

В качестве порогового значения для критерия принятия решения возьмем значение $K = 100$. Т.е. значение крутизны характеристики, при котором количество последовательностей, прошедших тестирование, устанавливается равным 0 при 1% искажений.

По результатам тестирования получены данные, приведенные на рис. 1-6[5], где используются следующие сокращения для названий тестов (табл. 1):

Таблица 1

1. Frequency (monobit) Test – «Frequency».	9. Maurer's «Universal Statistical» Test – «Universal».
2. Frequency Test within a Block – «Bl. Frequency».	10. Lempel-Ziv Compression Test – «Lemp.-Ziv».
3. Runs Test – «Runs».	11. LINEAR COMPLEXITY TEST – «LIN. COMPL.».
4. Test for the Longest Run of Ones in a Block – «L. Run».	12. Serial Test – «Serial» (2).
5. Binary Matrix Rank Test – «Rank».	13. Approximate Entropy Test – «apen».
6. Discrete Fourier Transform (Spectral) Test – «fft».	14. Cumulative Sums (Cusum) Test – «Cusum» (2).
7. Non-overlapping Template Matching Test – «Non-ov.Templ.» (148).	15. Random Excursions Test – «Rnd. Exc.» (8).
8. Overlapping Template Matching Test – «Ov. Templ.».	Random Excursions Variant Test – «Rnd. Exc.V» (18).

На рисунке 1 представлены результаты, полученны для искажения типа «00000000…» и одного искаженного блока. На рисунке 2 – для искажений типа «00000000…» и 10 искаженных блоков. На рисунке 3 – для искажений типа «00000000…» и 100 искаженных блоков. На рисунке 4 – для искажений типа «01010101…» и 1 искаженного блока. На рисунке 5 – для искажений типа «01010101…» и 10 искаженных блоков. На рисунке 6 – для искажений типа «01010101…» и 100 искаженных блоков.

[3] Для последовательности длиной 10^6 бит, 1% составляет 1250 байт.

[4] При искажениях «сплошные единицы» и «сплошные нули» результаты тестирования практически совпадают.

[5] В скобках после названия указано количество проводимых тестов.

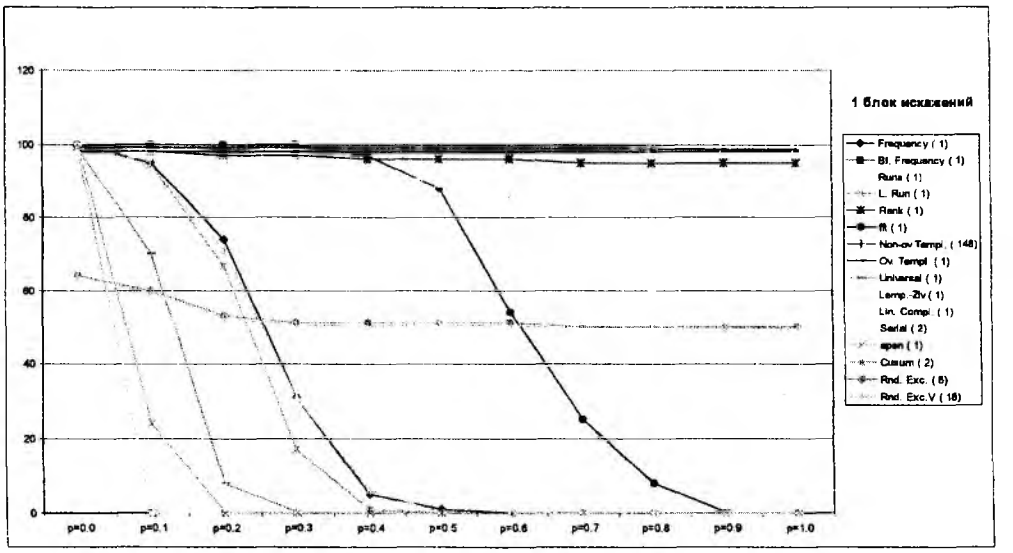


Рис. 1

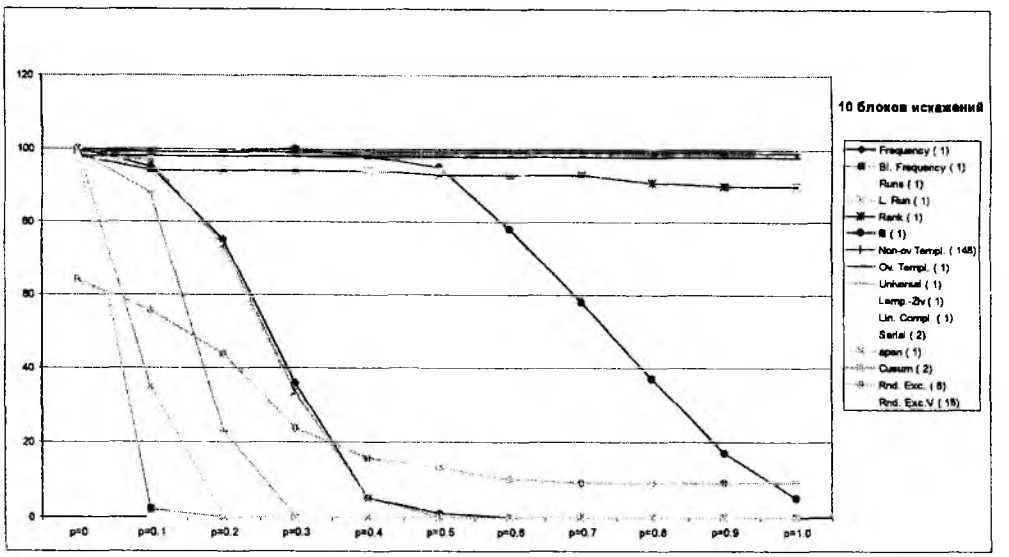


Рис. 2

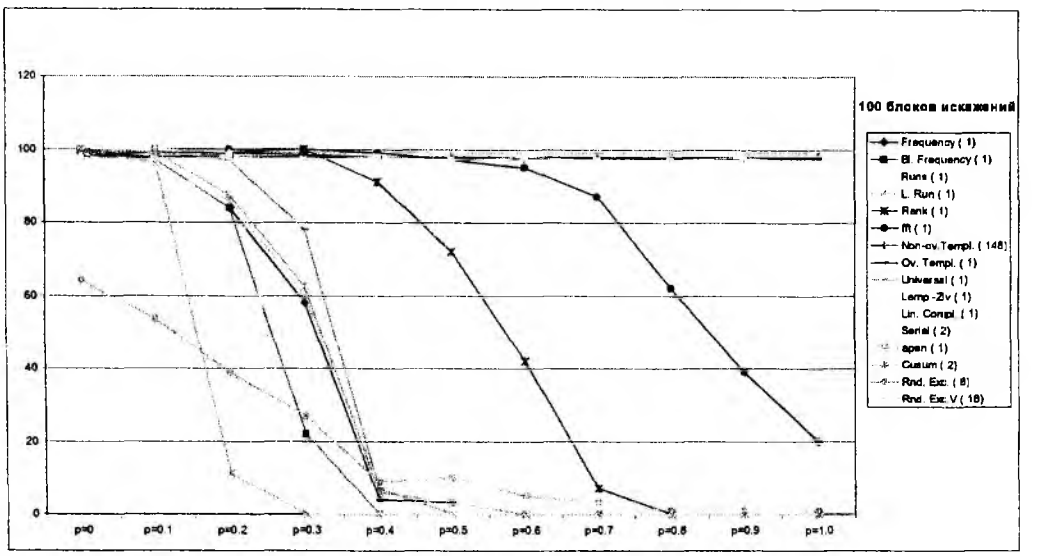


Рис. 3

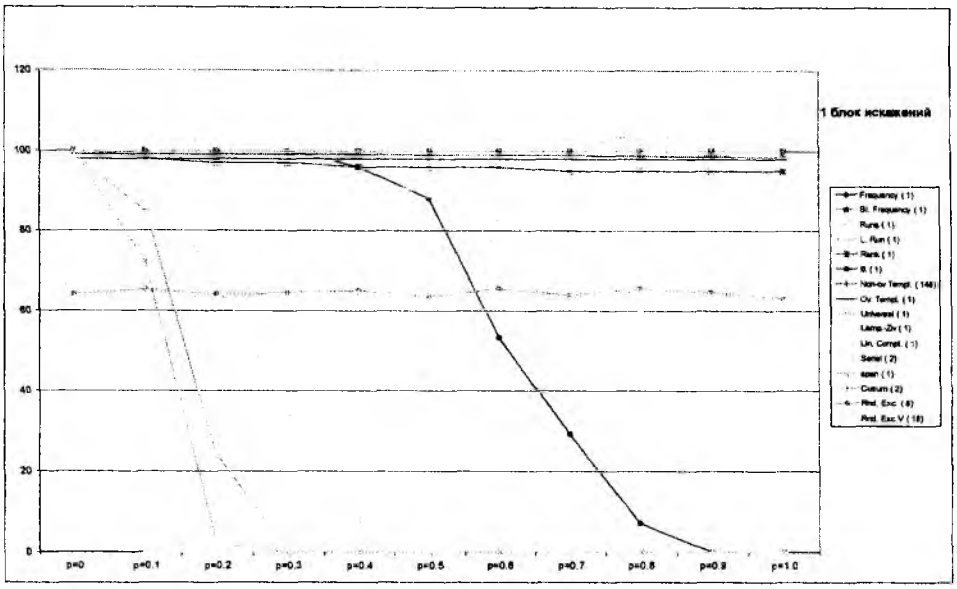


Рис. 4

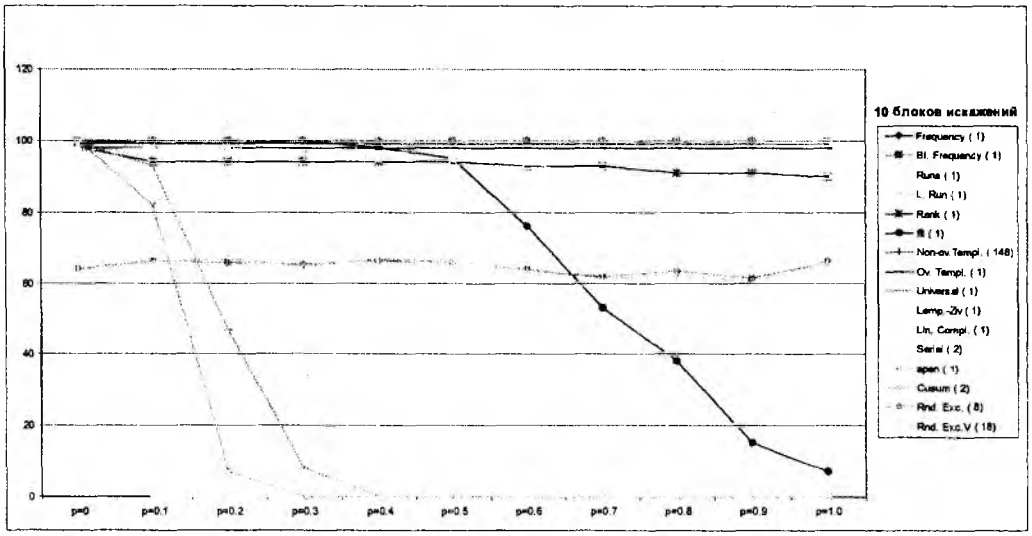


Рис. 5

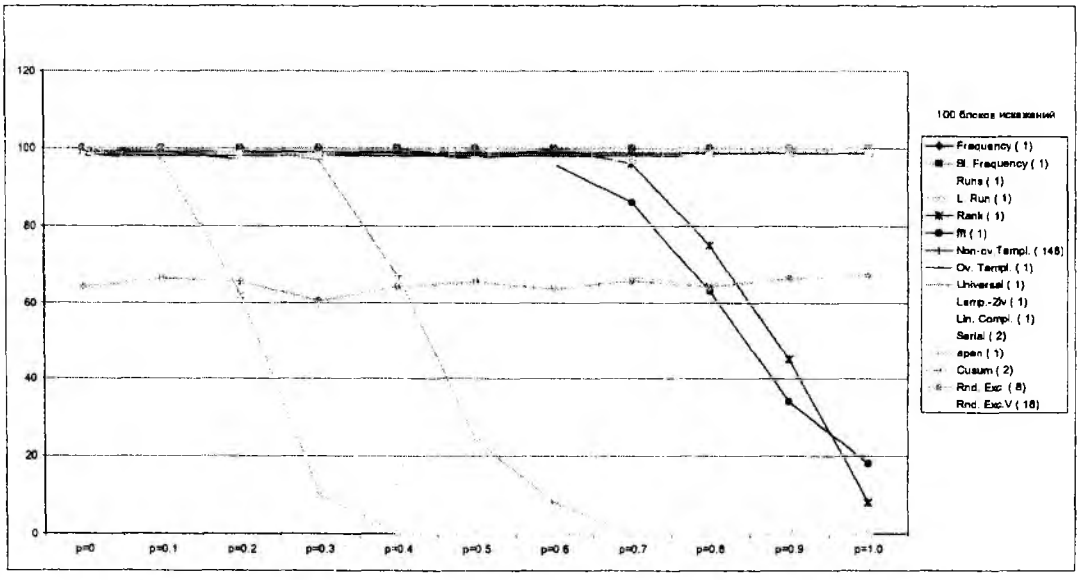


Рис. 6

Обобщенные результаты тестирования коэффициента крутизны статистической характеристики K_S сведены в табл. 2.

Таблица 2

Тип искажений	00000000			01010101		
Количество искаженных блоков	1	10	100	1	10	100
Frequency (1)	**166,667**	**166,667**	**166,667**	0,000	0,000	0,000
Bl. Frequency (1)	**990,000**	**495,000**	**247,500**	-1,000	-1,000	-1,000
Runs (1)	**198,000**	**198,000**	**165,000**	**165,000**	**165,000**	**141,429**
L. Run (1)	0,000	0,000	0,000	0,000	0,000	0,000
Rank (1)	3,000	8,000	**122,500**	3,000	8,000	90,000
fft (1)	**111,100**	95,000	80,000	**111,111**	93,000	82,000
Non-ov.Templ. (148)	0,595	0,074	1,601	0,576	0,094	0,621
Ov. Templ. (1)	0,000	0,000	0,000	0,000	0,000	0,000
Universal (1)	**330,000**	**330,000**	**198,000**	**330,000**	**247,500**	**141,429**
Lemp.-Ziv (1)	**1000,000**	**1000,000**	**333,333**	**1000,000**	**1000,000**	**250,000**
Lin. Compl. (1)	69,000	28,000	1,000	69,000	28,000	-1,000
Serial (2)	**328,333**	**328,333**	**246,250**	**328,333**	**328,333**	**246,250**
apen (1)	**500,000**	**500,000**	**333,333**	**333,333**	**333,333**	**250,000**
Cusum (2)	**200,000**	**200,000**	**166,667**	0,000	0,000	0,000
Rnd. Exc. (8)	14,000	55,000	63,125	1,000	-2,000	-3,125
Rnd. Exc.V (18)	13,945	54,445	62,333	0,889	-2,389	-4,333

Применив описанный выше критерий, разделим все тесты на группы в зависимости от значения K_S:

1. Тесты, для которых K_S: всегда больше 100: Apen (1), Serial (2), Lemp.-Ziv (1), Universal (1) Runs (1).
2. Тесты, для которых K_S иногда больше 100: Cusum (2) fft (1), Rank (1), Bl. Frequency (1), Frequency (1).
3. Тесты, для которых K_S всегда меньше 100: Rnd. Exc.V (18), Rnd. Exc. (8), Lin. Compl. (1), Ov. Templ. (1), Non-ov.Templ. (148), L. Run (1).

Таким образом, можно сделать вывод, что для построения методики оперативного тестирования случайных последовательностей из набора тестов NIST можно использовать следующие тесты:

1. Approximate Entropy Test.
2. Serial Test (2).
3. Lempel-Ziv Compression Test.
4. Maurer's «Universal Statistical» Test.
5. Runs Test.
6. Cumulative Sums (Cusum) Test (2).
7. Frequency Test within a Block.
8. Frequency (monobit) Test.

С учетом того, что Serial Test и Cumulative Sums (Cusum) Test реально выполняют по 2 теста, получаем общее количество тестов 10, что совпадает с оценкой, полученной по формуле (2).

Проведя тестирование данным набором тестов, получили, что из 100 тестируемых последовательностей тест прошло 95 (КПД генератора – 95%). Данное значение попадает в интервал ошибки первого рода 0,1.

Выводы

В данной статье предложена методика отбора тестов для формирования пакета оперативного тестирования, основанная на проверке чувствительности ряда независимых тестов. Предложен показатель и критерий отбора статистических тестов из пакета NIST STS. Обосновано количество и содержание тестов, отобранных по согласно предложенному показателю и критерию. Экспериментально подтверждена сходимость теоретической и эмпирической оценок необходимого количества тестов для оперативного тестирования. Использование полученных результатов позволяет построить методику оперативного тестирования, которая удовлетворяет временным ограничениям на тестирования СПБД на специализированных АРМах генерации таких последовательностей. Направлением дальнейших исследований является разработка пакета оперативного тестирования и расчет технических характеристик АРМа тестирования.

Список литературы: 1. *Шеннон К.Э.* Теория связи в секретных системах // Работы по теории информации и кибернетике (сб. статей). М.: Изд-во иностр. лит., 1963. 2. *Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В.,* Основы криптографии: Учебное пособие. М.: Гелиос АРВ, 2001. 3. FIPS PUB 140-2 Security Requirements For Cryptographic Modules, 1999. 4. *Гулак Г.М., Ковальчук Л.В.,* Підходи до визначення випадкових послідовностей // Правове, нормативне та метрологічне забезпечення систем захисту інформації в Україні. 2001. Вип. 3. 5. NIST Special Publication 800-22. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications, 2000.

Департамент специальных телекоммуникационных систем и защиты информации
Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 15.04.2003

УДК 621.3.06

А. В. ПОТИЙ, канд. техн. наук, Ю. А. ИЗБЕНКО

СИСТЕМА ПОКАЗАТЕЛЕЙ ОЦЕНКИ ЭФФЕКТИВНОСТИ ФУНКЦИОНИРОВАНИЯ СХЕМ ПОТОЧНОГО ШИФРОВАНИЯ

Сегодня можно с уверенностью утверждать, что в международной практике в качестве основного подхода к решению разработки новых алгоритмов криптографического преобразования информации утвердился системный подход. Суть подхода заключается в том, что разработчики уделяют внимание не только обеспечению криптографической стойкости алгоритма, но и обеспечению эффективности функционирования алгоритма в целом. Действительно, при современном уровне развития вычислительной техники и средств связи, переходе к эксплуатации информационно-телекоммуникационных систем обеспечение эффективности функционирования алгоритмов шифрования становится таким же естественным требованием, как и обеспечение их криптографической стойкости.

К сожалению, зачастую эффективность и стойкость являются конфликтующими целями. С одной стороны, постоянно происходит ужесточение требований по безопасности, поскольку алгоритм должен обладать запасом стойкости не только к известным криптонападениям, но и к новым методам криптоанализа. С другой стороны, повышаются требования по производительности средств шифрования. Полное разрешение противоречия между стойкостью и эффективностью получить невозможно, однако если рассматривать стойкость как составляющую эффективности алгоритма, согласовать различные показатели, остроту решения этого вопроса можно снизить. Такой подход к разработке алгоритмов шифрования хорошо зарекомендовал себя при проведении конкурса на новый стандарт блочного шифрования AES [1]. Очевидно, что такой подход будет эффективен и при разработке схем поточного шифрования (ПШ).

Эффективность функционирования схем ПШ есть степень соответствия полученных результатов функционирования схемы R_p требуемым R_{mp}:

$$P = < R_p , R_{mp} >. \qquad (1)$$

Требуемым результатом функционирования схемы R_{mp} является обеспечение функции конфиденциальности путем реализации механизма шифрования. Полученным результатом функционирования схемы R_p является реальный уровень обеспечения функции конфиденциальности. Оценка того, насколько полно реализована функция конфиденциальности или, другими словами, насколько R_p соответствует R_{mp}, осуществляется с помощью показателей, отражающих степень выполнения схемой функциональной задачи.

На сегодняшний день не существует систематизированного множества показателей, которые могли бы охарактеризовать эффективность функционирования схем ПШ. Как показал анализ [1], разработчики схем ПШ предлагают различные показатели, которые характеризуют отдельные стороны схем, но не позволяют представить картину в целом. В данной работе ставится задача обоснования и определения минимального множества требований и частных показателей эффективности схем ПШ. Другой задачей является формулировка критериев выбора схем ПШ.

В первой части работы представлена декомпозиция требований к схемам ПШ, общая классификация показателей и критериев эффективности, формируется векторный показатель эффективности схем ПШ.

Вторая часть работы содержит описание множества показателей и критериев эффективности схем ПШ, разбитых на группы показателей стойкости, показателей эффективности реализации, конструктивно-технологические показатели.

В заключении обсуждаются дальнейшие направления исследований в данной области.

1 Классификация критериев и показателей эффективности функционирования схем ПШ

Эффективность функционирования схем криптографического преобразования данных оценивается с точки зрения стойкости схем к аналитическим методам вскрытия информации. Анализ работ [3-8] показал, что при оценке стойкости используются следующие критерии:

1. Вычислительная сложность методов криптоанализа должна быть не меньше вычислительной сложности универсальных методов (полный перебор, парадокс дней рождения и т.п.).

2. Рассматриваемые схемы оцениваются согласно утверждениям разработчиков о стойкости схем. В случае нахождения метода анализа, имеющего вычислительную сложность меньшую, чем заявлено разработчиками, схема считается скомпрометированной.

3. Рассматриваемые схемы оцениваются в пределах заявленной области применения. Таким образом, рассмотрение уязвимости к побочным методам нападениям (например timing attack, силовые атаки) должно быть соответствующим.

Очевидно, что первоочередной задачей схем криптографического преобразования данных является обеспечение стойкости схем к известным на сегодняшний день методам криптоанализа. Однако в силу интенсивного развития средств телекоммуникаций, возрастания объемов циркулирующей в информационных системах информации, многообразия сред приложений, требующих выполнения криптопреобразований, неотъемлемыми требованиями к схемам криптопреобразований стали быстродействие, объемы требуемой памяти, многоплатформенность реализации и т.д. В рамках конкурса NESSIE для отбора кандидатов на европейский стандарт поточного шифрования предлагаются следующие показатели [2]:

– стойкость к методам анализа. Любая рассматриваемая схема должна быть стойкой к атакам заявленного уровня стойкости. Подверженность анализу каким-либо методом со сложностью меньше заявленной дисквалифицирует рассматриваемую схему;

– общая концепция конструкции схемы. Важным элементом оценки стойкости схемы криптопреобразований является используемая концепция построения и прозрачность конструкции анализируемой схемы. Ясность и прозрачность конструкторских решений, основанных на хорошо понятных и изученных математических и криптографических принципах, позволяют с бóльшим доверием относиться к результатам оценки стойкости исследуемых схем. Кроме того, данные принципы особенно уместны при проведении сравнений между исследуемыми схемами;

– стойкость модифицируемых схем. Под модифицированной схемой понимают схему, в которой модифицируют или удаляют некоторые ее составные части. В этом случае выводы относительно стойкости схемы делают на основе оценки стойкости модифицированной схемы;

– показатели относительной защиты. При оценке разработанных схем с идентичным функциональным назначением возникает задача сравнительного анализа стойкости предоставляемого ими уровня защиты. При проведении таких сравнений следует быть крайне осторожным в выборе критериев сравнения. Так, показателем, определяющим нижнюю границу стойкости блочных симметричных алгоритмов, является разница между количеством циклов алгоритма и количеством взломанных циклов. Однако при этом у аналитиков нет никакого общего согласия об определении или использовании такого показателя. Имеют место случаи, когда оцениваемые схемы имеют различные конструкции и концепции построения, при их анализе используются отличные друг от друга показатели. Все это затрудняет осуществление оценки схем с единых позиций. Таким образом, необходимо для всех схем определить некоторую нижнюю границу стойкости к методам анализа;

– характеристики среды эксплуатации и особенности применения схем. В определенных криптографических средах представленные схемы могут обладать свойственными им достоинствами и недостатками. В качестве примера можно рассматривать схемы, обладаю-

щие стойкостью по отношению к силовым атакам и timing attack, выполненным на смарт-картах. Рассмотрение других типов атак на данные схемы было бы нецелесообразным;

– статистическая стойкость. Рассматриваемые схемы подвергаются статистическому тестированию. Целью тестирования является выявление некоторых статистических отклонений в генерируемых последовательностях, что может указать на некоторую криптографическую слабость и потребует дальнейшего исследования схемы;

– быстродействие схем. Быстродействие представленных схем рассматривается на нескольких платформах с целью определения собственно быстродействия, а также гибкости реализации и многоплатформенности приложений;

– объем используемой памяти. В некоторых приложениях данный показатель является критичным. Любая конструкция должна гарантировать минимальный объем памяти без нанесения ущерба стойкости рассматриваемой схемы.

Таким образом, на данный момент в различных источниках рассмотрены показатели, относящиеся к оценке разных сторон схем преобразования. Анализ документов, представленных на конкурсы AES, NESSIE [2, 9], показывает, что наряду с показателями стойкости немаловажную роль имеют и аппаратно-реализационные показатели.

При определении эффективности функционирования схем ПШ целесообразно рассмотрение следующих основных групп показателей (рис. 1).

Рис. 1

Требуемый результат функционирования схемы R_{mp} может быть представлен в виде

$$R_{mp} = < П_{СТ}, П_{Р}, П_{КТ} >, \quad (2)$$

где $П_{СТ}$ – показатели стойкости схем ПШ; $П_{Р}$ – программно- и аппаратно- реализационные показатели; $П_{КТ}$ – конструктивно-технологические показатели.

Показатели стойкости схем ПШ, $П_{СТ}$, характеризуют стойкость схем ПШ к аналитическим методам анализа и являются основными показателями, поскольку целесообразность применения любых схем криптопреобразования информации основывается, в первую очередь, на оценке стойкости данных схем к известным на сегодняшний день методам (атакам) криптоанализа, применимым к исследуемому классу схем.

Программно- и аппаратно- реализационные показатели, $П_{Р}$, характеризуют эффективность программной и аппаратной реализаций и являются показателями, характеризующими практичность схем криптопреобразований. Схема может обладать высокой стойкостью к различного рода методам криптоанализа, но тем не менее иметь ограничения на практическое использование из-за высокой стоимости аппаратной реализации, низкого быстродействия и т.п. Одним из важнейших преимуществ методов поточного шифрования информации, благодаря которому они получили широкое распространение, является высокая скорость преобразования информации. Как следствие, схемы, имеющие высокие показатели эффективности реализации, с наибольшей вероятностью будут отобраны для практических приложений. Применимость схем часто рассматривается с позиций унификации (универсальности): например, одна и та же схема должна быть одинаково успешно реализуема и в программном обеспечении, и в аппаратном, на смарт-картах и т.п. Некоторые области приложений налагают очень высокие требования к скорости связи (гигабитные трафики), для других областей применения приложения приемлют минимальные аппаратные платформы и/или компактные аппаратные средства (сотовые телефоны, смарт-карты). В идеале рассматриваемые схемы должны быть гибкими, т.е. быть реализуемыми более чем на одной платформе. Таким образом, при определении эффективности реализации рассматриваемых схем целесообразно производить отбор показателей, отражающих эффективность функционирования схем в заданных областях приложений, поскольку, очевидно, одна и та же схема вряд ли может быть эффективно реализована на всех платформах и во всех приложениях.

Конструктивно-технологические показатели, $П_{КТ}$, характеризуют прозрачность конструкции, перспективность схем, запас их стойкости, пригодность известных методов анализа к оценке характеристик схем криптопреобразований, возможность проведения сравнительного анализа схем данного класса и, как следствие, степень доверия пользователей.

2 Показатели эффективности функционирования схем ПШ

Каждая группа предложенных выше показателей включает в себя множество частных показателей. В данном разделе обосновывается и предлагается минимально необходимое множество показателей эффективности схем ПШ, формулируются критерии оценки эффективности. Показатели и критерии эффективности могут носить как количественный, так и качественный характер.

2.1 Показатели стойкости схем ПШ

На основе исследований, проведенных в [3 – 8, 10 – 12], для схем ПШ определим следующее множество показателей стойкости.

Показатели, характеризующие параметры регистров и точек съема для нелинейной функции и обратных связей, $П_{ДРР}$:

- *Примитивность образующего полинома.* Примитивный многочлен степени n – это неприводимый многочлен, который делит x, но не делит $x^{2^{n-1}} + 1$ для любого такого d, которое делит 2^n-1. Степень многочлена – это длина регистра сдвига. Только регистры, использую-

щие в качестве образующего полинома примитивный полином, являются регистрами максимального периода (проходят через все возможные 2^n-1 состояний) и генерируют последовательности максимального периода, так называемые *ml* – последовательности.

Критерием отбора является логическая переменная Да\Нет.

- *Взаимно простые степени образующих полиномов* (для комбинирующих генераторов). При выборе образующих полиномов, имеющих взаимно простые степени, линейная сложность (см. показатель *линейная сложность*) генерируемой последовательности z является максимальной $\Lambda(z) = f(\Lambda_1, \ldots, \Lambda_N)$, где Λ_j – линейная сложность j-го фильтр-генератора, $j = 1, \ldots, N$.

Критерием отбора является логическая переменная Да\Нет.

- *Степень образующего полинома.* При правильно подобранном образующем полиноме степень полинома n определяет период генерируемой последовательности. На сегодняшний день для противостояния аналитическим атакам минимальная длина регистра должна составлять не менее 128 бит.

Критерием отбора будем полагать $n \geq 128$ бит.

- *Плотность образующего полинома.* Для противостояния корреляционным атакам, основанным на низковесовых проверках четности, количество ненулевых коэффициентов k в образующем полиноме должно быть около $n/2$.

Критерием отбора будем полагать $k \rightarrow n/2$.

- *Правильность определения количества точек съема для нелинейной функции* (для фильтр-генераторов). Для противостояния аналитическим атакам количество точек съема r должно быть $r` \leq \sqrt{2n}$.

Критерием отбора является $r \rightarrow r`$.

- *Соответствие множества точек обратных связей B полному множеству положительных разностей ΔB.* Множество ΔB называют *полным множеством положительных разностей*, если все положительные попарные разности между его элементами различны. Требуемое соответствие обеспечивает противостояние аналитическим атакам.

Критерием отбора является логическая переменная Да\Нет.

- *Соответствие множества точек съема для нелинейной функции Г полному множеству положительных разностей* $\Delta\ \Gamma$ (для фильтр-генераторов). Требуемое соответствие обеспечивает противостояние аналитическим атакам.

Критерием отбора является логическая переменная Да\Нет.

- *Наибольший общий делитель двух парных (соседних) положительных разностей должен быть равен* 1.

Критерием отбора является логическая переменная Да\Нет.

Показатели, характеризующие стойкость нелинейной функции, $\mathbf{П_{НЛФ}}$:

- *Сбалансированность выходной последовательности.* Функция f над V_n является *сбалансированной* функцией, если ее выходные значения являются равновероятными:

$$| \{x \mid f(x) = 0 \} | = | \{x \mid f(x) = 1 \} | = 2^{n-1}. \qquad (3)$$

Сбалансированность функции является показателем, отражающим стойкость гаммы к статистическим атакам.

Критерием отбора является логическая переменная Да\Нет.

- *Нелинейность. Нелинейность* функции f – минимальное расстояние Хэмминга N_f между функцией f и всеми аффинными функциями над V_n [12]:

$$N_f = \min \{d(f, \varphi)\}, \qquad (4)$$

где φ – множество аффинных функций.

Для сбалансированной функции f над V_n ($n \geq 3$) нелинейность N_f может достигать [12]:

$$N_f \le \begin{cases} 2^{n-1} - 2^{n/2-1} - 2, & n = 2k, \\ \left\lfloor\!\left\lfloor 2^{n-1} - 2^{n/2-1} \right\rfloor\!\right\rfloor, & n = 2k+1, \end{cases} \tag{5}$$

где $\lfloor\!\lfloor x \rfloor\!\rfloor$ – максимальное четное целое, меньшее либо равное x.

Верхняя граница нелинейности для произвольной функции f над V_n может достигать

$$N_f \le 2^{n-1} - 2^{n/2-1}. \tag{6}$$

Нелинейность функции является важным показателем, поскольку несоблюдение данного показателя делает возможным проведение корреляционных атак, использующих корреляцию данной функции со множеством слабых функций. При построении криптографически стойких булевых функций необходимо обеспечить ее минимальную корреляцию со множеством всех слабых функций, то есть стремиться, чтобы нелинейность данной функции стремилась к верхней границе нелинейности, определенной согласно (6).

Критерием отбора является $N_f = \max\limits_{N_j} \{N_1, \ldots, N_r\}$, где N_j – нелинейность j – й функции; $j = 1, \ldots, r$.

- *Алгебраическая степень.* Алгебраическая степень *deg(f)* является степенью самого длинного слагаемого функции, представленной в алгебраической нормальной форме. Алгебраической нормальной формой называется выражение вида

$$f(x_1,\ldots,x_n) = a_0 \oplus \bigoplus_{i=1}^{n} a_i x_i \oplus \bigoplus_{1\le i<j\le 1} a_{ij} x_i x_j \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \cdots x_n. \tag{7}$$

Высокая алгебраическая степень позволяет противостоять различным аналитическим атакам, призванным свести данную функцию к криптографически слабой (линейной).

Критерием отбора является $deg(f) = \max\limits_{deg(f_j)} \{ deg(f_1), \ldots, deg(f_r)\}$, где $deg(f_j)$ – нелинейность j – й функции; $j = 1, \ldots, r$.

- *Коэффициент равномерной минимизации кросс-корреляции.* Характеризует равномерную минимизацию коэффициентов кросс-корреляции

$$k_{рм} = \frac{k_{гр}}{r \cdot c_{ср}}, \tag{8}$$

где $k_{гр}$ – граничный коэффициент кросс-корреляции; r – удельный вес ненулевых значений коэффициентов кросс-корреляции; $c_{ср}$ – среднее значение коэффициента кросс-корреляции. Приведенные значения имеют следующий вид:

$$r = \left| \frac{B}{2^n} - 2^{n-1} \right|, \tag{9}$$

$$c_{ср} = \frac{\sum\limits_{i=1}^{2^n} c_i}{2^n}, \tag{10}$$

$$k_{гр} = \frac{\left|1 - 2^n\right| \cdot (c_n + c_{n+2})}{2} \text{ для } V_{n+1} \quad \text{и} \quad k_{гр} = \left|1 - 2^n\right| \cdot c_n \text{ для } V_n, \tag{11}$$

где B – общее количество ненулевых значений коэффициентов кросс-корреляции; c_i – значение коэффициента кросс-корреляции; c_n – коэффициент корреляции бент-функции над V_n; c_{n+2} – коэффициент корреляции бент-функции над V_{n+2}.

Критерием отбора является $k_{рм} = \min\limits_{k_{рм\,j}} \{ k_{рм\,1}, ..., k_{рм\,r}\}$, где $k_{рм\,j}$ – коэффициент равномерной минимизации кросс-корреляции j – й функции; $j = 1, ..., r$.

- Абсолютное значение кросс-корреляции функции

$$C_f = \max\limits_{l_i} |c(f, \ell_i)|. \tag{12}$$

Критерием отбора является $C_f = \min\limits_{l_i} \{ C_{f\,1}, ..., C_{f\,r}\}$, где $C_{f\,j}$ - абсолютное значение кросс-корреляции j-й функции; $j = 1, ..., r$.

- Количество векторов num_1, при которых функция не удовлетворяет критерию распространения

$$num_1 = !PC(k). \tag{13}$$

Критерием отбора является $num_1 = \min\limits_{num_{1\,j}} \{ num_{11}, ..., num_{1\,r}\}$, где $num_{1\,j}$ - количество векторов num_1, при которых j-я функция не удовлетворяет критерию распространения; $j = 1, ..., r$.

- Количество векторов num_2, при которых функция имеет линейную структуру

$$num_2 = PC_{ЛС}(k). \tag{14}$$

Критерием отбора является $num_2 = \min\limits_{num_{2\,j}} \{ num_{21}, ..., num_{2\,r}\}$, где $num_{2\,j}$ - количество векторов num_2, при которых j-я функция имеет линейную структуру; $j = 1, ..., r$.

Показатели, характеризующие стойкость процедуры ключевой инициализации, **$П_{КИ}$**:

- Нелинейность операций ключевой загрузки. Нелинейность операций повышает стойкость к алгоритмам, основанным на использовании ключевой загрузки.

Критерием отбора является логическая переменная Да\Нет.

- Каждый бит инициализированного регистра является результатом нелинейных преобразований всех бит ключа. Желательной является реализация, при которой каждый бит начального состояния регистра является функцией от нелинейного преобразования всех бит ключа, в таком случае

$$I(K_{out}^{n} ; K_{in}^{n-k}) \to 0, \tag{15}$$

где n – длина ключа; k – количество бит инициализированного ключа, введенных нелинейными преобразованиями, $k \to n$; K_{in}^{n-k} и K_{out}^{n} – вводимый и инициализированный ключи соответственно. Таким образом, знание вводимого ключа исключает знание инициализированного ключа (при гарантированной стойкости нелинейных преобразований).

Критерием отбора является логическая переменная Да\Нет.

Общие показатели, **$П_О$**:

- Период гаммы. Периодом гаммы шифрующей T именуется количество бит до того момента, когда последовательность начнет повторяться. Период – одна из наиболее важных характеристик при изучении свойств гаммы шифрующей. Если период гаммы шифрующей окажется слишком коротким, то различные части открытого текста окажутся преобразованными идентичным образом, что составляет серьезную слабость схемы. Зная фрагмент

открытого текста, аналитик может восстановить соответствующий фрагмент гаммы шифрующей. Тот факт, что данный участок гаммы шифрующей появляется и во множестве других мест гаммы шифрующей, позволяет аналитику успешно вскрывать другие зашифрованные тексты. Кроме того, если имеются только преобразованные тексты, зашифрованные одной и той же гаммой шифрующей, то их можно попарно складывать друг с другом для получения последовательности, которая равноценна сумме двух открытых сообщений и не зависит от гаммы шифрующей. Имея необходимую статистику открытого текста, на этой основе можно восстановить как открытые тексты сообщений, так и саму гамму шифрующую [3].

Поскольку ЛРР длиной n бит может находиться в одном из $2^n - 1$ внутренних состояний, то теоретически он может иметь период $2^n - 1$. Вопрос о том, насколько большим должен быть период шифрующей последовательности, в открытой литературе является дискуссионным, и решение его зависит от конкретной области применения. Как отмечено в [3] , для схемы ПШ, работающей на скорости 1 Мбайт/сек, последовательность с периодом 2^{32} повторит сама себя всего через 2^9 секунд или 8,5 минут, поэтому средства засекречивания информации с подобной длиной периода не могут считаться адекватной защитой.

На проходящем в настоящее время открытом европейском криптографическом конкурсе NESSIE (New European Schemes for Signature, Integrity and Encryption), целью которого является определение европейских стандартов криптографического преобразования данных, для схем ПШ определено два уровня стойкости:

- «высокий», при котором схема должна обладать длиной ключа $\ell_{кл} \geq 256$ бит и иметь внутреннюю память $P_в \geq 256$ бит;
- «нормальный», при котором схема должна обладать длиной ключа $\ell_{кл} \geq 128$ бит и иметь внутреннюю память $P_в \geq 128$ бит.

Таким образом, согласно европейскому стандарту, длины регистров для обеспечения требуемой стойкости должны составлять по меньшей мере 256 и 128 бит соответственно. Как следствие, можем записать, что длины периодов $T_в$ для схем с «высоким» уровнем стойкости и $T_н$ для схем с «нормальным» уровнем стойкости должны составлять

$$T_в \geq 2^{256}, \text{ бит},$$
$$T_н \geq 2^{128}, \text{ бит}.$$

Тщательная оценка периода гаммы шифрующей, порождаемой схемой, совершенно необходима при разработке любой схемы ПШ. С практической точки зрения, гамма шифрующая должна быть достаточно длинной для того, чтобы в подавляющем большинстве случаев было маловероятным повторное использование одного и того же фрагмента в процессе шифрования информации.

Критерием отбора является $T_f = \max\limits_{T_j} \{T_1, \ldots, T_r\}$, где T_j – период j – й схемы; $j = 1, \ldots, r$.

- Линейная сложность последовательности. Линейная сложность $\Lambda(s^l)$ последовательности $s^l = s_0, s_1, \ldots, s_{l-1}$ – длина L самого короткого регистра сдвига, порождающего заданную периодическую последовательность s^l, когда первые L цифр последовательности s^l являются начальным заполнением регистра.

Данный критерий является основополагающим критерием. Как отмечено в [4], любая последовательность, которую можно сгенерировать конечным автоматом (линейным или нелинейным) над конечным полем, имеет конечную линейную сложность. Следовательно, возможно построение алгоритмов, определяющих линейную сложность любой последовательности вне зависимости от способа ее генерации. Так, существует эффективный алгоритм Берлекампа-Месси [5], который быстро находит такой кратчайший регистр после изучения всего $2L$ бит шифрующей последовательности.

По своей сути линейная сложность является мерой сложности сгенерированной последовательности. Рассматривая некоторый отрезок последовательности, аналитик пытается на основе имеющейся последовательности сконструировать свою собственную последовательность. Определение линейной сложности рассматриваемой последовательности дает возможность на основе найденного линейного эквивалента построить схему, которая бы генерировала последовательность, аналогичную рассматриваемой, при этом знания о структуре схемы, сгенерировавшей заданную последовательность, являются излишними. Таким образом, большая линейная сложность гаммы шифрующей – необходимое, но не достаточное условие практической стойкости схем ПШ.

Критерием отбора является $\Lambda_f = \max\limits_{\Lambda_j} \{\Lambda_1, \ldots, \Lambda_r\}$, где Λ_j – линейная сложность последовательности j – й схемы; $j = 1, \ldots, r$.

- *Длина используемых ключей.* Длина ключа определяет стойкость к силовым атакам и составляет $k_в \geq 256$ бит для схем с «высоким» уровнем стойкости и $k_н \geq 128$ бит для схем с «нормальным» уровнем стойкости.

Критерием отбора является $k_f = \max\limits_{k_j} \{k_1, \ldots, k_r\}$, где k_j – длина ключа j – й схемы с заданным уровнем стойкости; $j = 1, \ldots, r$.

- *Внутреннее состояние генератора (внутренняя память) M должно быть не менее 2k бит при длине ключа k бит.* Данный показатель отражает стойкость к атакам «время-память».

Критерием отбора является логическая переменная Да\Нет.

- *Длина генерируемой последовательности $\ell_п$ не должна превышать некоторого порогового значения N_{max},* $\ell_п < N_{max}$. Данный показатель отражает стойкость к корреляционным атакам, позволяющим по имеющемуся фрагменту гаммы шифрующей восстановить начальное заполнение ЛРР. Целью введения N_{max} является недопущение генерации последовательности, длина которой гарантировала бы использование корреляции последовательности с последовательностью ЛРР [7].

Критерием отбора является логическая переменная Да\Нет.

- *Показатель, характеризующий стойкость схемы к аналитическим методам анализа,* $П_{АН}$. В целом, именно стойкость схемы к методам анализа определяет эффективность функционирования схем ПШ.

Критерием отбора является логическая переменная Да\Нет.

- *Показатель, характеризующий стойкость схемы к статистическим методам анализа,* $П_{СТАТ}$.

Статистические свойства шифрующей гаммы являются одной из составляющих, определяющей стойкость схемы шифрования. Стойкость схемы зависит от того, насколько близко она аппроксимирует генератор случайных чисел, то есть насколько гамма шифрующая будет вычислительно непредсказуемой и неотличимой от действительно случайной последовательности. Общепризнанной методикой оценки статистических свойств криптографических преобразований на сегодняшний день является методика, предложенная NIST [10]. В данной методике по выборке строится статистический портрет источника, который содержит значения вероятности $P_{ij} \in [0,1]$ – значения вероятности, полученной в результате тестирования i-й последовательности j-м тестом; $i = 1, \ldots, m, j = 1, \ldots, q$. Рекомендуемые значения: m=100 последовательностей по 10^6 бит (генеральная совокупность составляет 10^8 бит), q = 189. Затем при анализе статистического портрета применяются такие критерии. Первым критерием отбора полагается коэффициент прохождения тестов последовательностями

$$r_j = \frac{\#\{P_{ij} \geq \alpha \mid i = 1,2,\ldots,m\}}{m}, \qquad (16)$$

попавший в доверительный интервал $[r_{max}, r_{min}]$,

$$r_{\max(\min)} = \hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}, \qquad (17)$$

где $\hat{p} = 1 - \alpha$, α – уровень значимости, равный 0,01.

Второй критерий строится на основе расчета значения χ^2:

$$\chi_j^2 = \sum_{k=1}^{10} \frac{\left(F_k - m/10\right)^2}{m/10}. \qquad (18)$$

Считается, что схема прошла статистическое тестирование, если значения коэффициентов r_j для всех $j = 1, \ldots, q$ находятся внутри доверительного интервала $[r_{\max}, r_{\min}]$ и для (18) соблюдается условие $\chi_j^2 > 0{,}0001$ для всех $j = 1, \ldots, q$.

Таким образом, комплексный показатель стойкости $П_{СТ}$ схем ПШ имеет вид:

$$П_{СТ} = (П_{ЛРР}, П_{НЛФ}, П_{КИ}, П_{О}). \qquad (19)$$

2.2 Программно- и аппаратно- реализационные показатели

На основе исследований, проведенных в [2], для схем ПШ определим следующее множество показателей эффективности реализации $П_Р$:

- *Ключевая инициализация*, $П_{КИ}$, циклов/байт – отображает скорость выполнения инициализации схемы – количество загруженных байт $n_{бз}$ ключа за один полный цикл сдвига регистра.

Критерием отбора является $n_{бз} = \max_{n_j} \{ n_1, \ldots, n_r \}$, где n_j – количество загруженных байт $n_{бз}$ ключа за один полный цикл сдвига регистра j – й схемы; $j = 1, \ldots, r$.

- *Генерация выходной последовательности*, $П_{ВП}$, циклов/байт – отображает скорость генерации выходной последовательности схемы — количество сгенерированных байт $m_{бв}$ выходной последовательности за один полный цикл сдвига регистра.

Критерием отбора является $m_{бв} = \max_{m_j} \{ m_1, \ldots, m_r \}$, где m_j – количество сгенерированных байт $n_{бв}$ выходной последовательности за один полный цикл сдвига регистра j – й схемы; $j = 1, \ldots, r$.

- *Размер используемой памяти*, $П_{ПАМ}$, Кб. Отображает размер используемой памяти, ОЗУ и ПЗУ.

Критерием отбора является $v_{mem} = \min_{v_j} \{v_1, \ldots, v_r\}$, где v_j – размер памяти, используемый j – й схемой; $j = 1, \ldots, r$.

- *Скорость шифрования/дешифрования информации*, $П_{СШ}$, Мб/сек.

Критерием отбора является $s = \max_{s_j} \{s_1, \ldots, s_r\}$, где s_j – скорость шифрования/ дешифрования информации j – й схемой; $j = 1, \ldots, r$.

- *Количество используемых различных арифметических операций*, $П_{РО}$.

Критерием отбора является $op = \min_{op_j} \{ op_1, \ldots, op_r \}$, где op_j – количество используемых различных арифметических операций j – й схемой; $j = 1, \ldots, r$.

- *Переносимость на другие платформы*, $П_{ПЕР}$.

Критерием отбора является логическая переменная Да\Нет.

При аппаратной реализации в качестве дополнительных показателей могут рассматриваться:

1. Габариты изделия.
2. Стоимость изделия.

3. Возможность сопряжения с другими типами аппаратуры.
4. Возможность реализации на различного рода микросхемах.

Таким образом, комплексный показатель эффективности программной и аппаратной реализации $П_Р$ схем ПШ имеет вид:

$$П_Р = (П_{КИ}, П_{ВП}, П_{ПАМ}, П_{СШ}, П_{РО}, П_{ПЕР}). \quad (20)$$

2.3. Конструктивно-технологические показатели

Определим следующее множество показателей эффективности реализации $П_{КТ}$:
- Прозрачность конструкции, $П_п$.

Критерием отбора является логическая переменная Да\Нет.
- Возможность проведения сравнительного анализа, $П_А$.

Критерием отбора является логическая переменная Да\Нет.
- Перспективность, $П_П$.

Критерием отбора является логическая переменная Да\Нет.
- Запас стойкости, $П_З$.

Критерием отбора является логическая переменная Да\Нет.

На сегодняшний день для оценки конструктивно-технологических показателей можно использовать методы экспертных оценок.

Таким образом, требуемый результат функционирования схемы R_{mp} определяется следующим множеством показателей

$$R_{mp} = < (П_{ЛРР}, П_{НЛФ}, П_{КИ}, П_О), (П_{КИ}, П_{ВП}, П_{ПАМ}, П_{СШ}, П_{РО}, П_{ПЕР}), (П_п, П_А, П_П, П_З) >.$$

Заключение

Приведенные в статье показатели достаточно полно характеризуют конкретную схему поточного шифрования. Введенные критерии эффективности в совокупности с методиками оценки позволяют осуществить сравнение различных схем поточного шифрования. Использование количественных и качественных показателей приводит к снижению неопределенности эксперта относительно оцениваемой схемы. В целом, предложенное множество показателей и критериев эффективности, на наш взгляд, позволяют построить эффективный инструментарий оценки схем ПШ. Ниже представлена сводная таблица показателей и критериев оценки эффективности (табл. 1).

Таблица 1

Показатели эффективности функционирования схем ПШ	Критерии оценки эффективности функционирования схем ПШ
Показатели стойкости схем ПШ	
Показатели, характеризующие параметры регистров и точек съема для нелинейной функции и обратных связей	
Примитивность образующего полинома	Да/нет
Взаимно простые степени образующих полиномов	Да/нет
Степень образующего полинома, n	$n \geq 128$ бит
Плотность образующего полинома, k	$k \to n/2$
Правильность определения количества точек съема для нелинейной функции	Да/нет
Соответствие множества точек обратных связей B полному множеству положительных разностей ΔB	Да/нет
Соответствие множества точек съема для нелинейной функции Γ полному множеству положительных разностей $\Delta \Gamma$	Да/нет
Наибольший общий делитель двух парных (соседних) положительных разностей должен быть равен 1	Да/нет

Продолжение табл. 1

Показатели, характеризующие стойкость нелинейной функции	
Сбалансированность	Да/нет
Нелинейность, N_f	$N_f = \max\limits_{N_j} \{N_1, \ldots, N_r\}$
Алгебраическая степень, $deg(f)$	$deg(f) = \max\limits_{deg(f_j)} \{deg(f_1), \ldots, deg(f_r)\}$
Коэффициент равномерной минимизации кросс-корреляции, $k_{рм}$	$k_{рм} = \min\limits_{k_{рм\,j}} \{k_{рм\,1}, \ldots, k_{рм\,r}\}$
Абсолютное значение кросс-корреляции функции, C_f	$C_f = \min\limits_{l_i} \{C_{f1}, \ldots, C_{fr}\}$
Количество векторов, при которых функция не удовлетворяет критерию распространения, num_1	$num_1 = \min\limits_{num_{1j}} \{num_{11}, \ldots, num_{1r}\}$
Количество векторов, при которых функция имеет линейную структуру, num_2	$num_2 = \min\limits_{num_{2j}} \{num_{21}, \ldots, num_{2r}\}$
Показатели, характеризующие стойкость процедуры ключевой инициализации	
Нелинейность операций ключевой загрузки	Да/нет
Каждый бит инициализированного регистра является результатом нелинейных преобразований всех бит ключа	Да/нет
Общие показатели	
Период гаммы, T_f	$T_f = \max\limits_{T_j} \{T_1, \ldots, T_r\}$
Линейная сложность последовательности, Λ_f	$\Lambda_f = \max\limits_{\Lambda_j} \{\Lambda_1, \ldots, \Lambda_r\}$
Длина используемых ключей, k_f	$k_f = \max\limits_{k_j} \{k_1, \ldots, k_r\}$
Внутреннее состояние генератора (внутренняя память) M должно быть не менее $2k$ бит при длине ключа k бит	Да/нет
Длина генерируемой последовательности $\ell_п$ не должна превышать некоторого порогового значения N_{max}	Да/нет
Показатель, характеризующий стойкость схемы к аналитическим методам анализа	Да/нет
Показатели, характеризующие стойкость схемы к статистическим методам анализа, r_j и χ^2	$r_j = \dfrac{\#\{P_{ij} \ge \alpha \mid i = 1,2,\ldots,m\}}{m}$ $\chi^2_j = \sum\limits_{k=1}^{10} \dfrac{\left(F_k - m/10\right)^2}{m/10}$
Программно- и аппаратно- реализационные показатели	
Ключевая инициализация, $n_{бз}$	$n_{бз} = \max\limits_{n_j} \{n_1, \ldots, n_r\}$
Генерация выходной последовательности, $m_{бв}$	$m_{бв} = \max\limits_{m_j} \{m_1, \ldots, m_r\}$
Размер используемой памяти, v_{mem}	$v_{mem} = \min\limits_{v_j} \{v_1, \ldots, v_r\}$
Скорость шифрования/дешифрования информации, s	$s = \max\limits_{s_j} \{s_1, \ldots, s_r\}$

Окончание табл. 1

Количество используемых различных арифметических операций, *op*	$op = \min_{op_j} \{op_1, \ldots, op_r\}$
Переносимость на другие платформы	Да/нет
Конструктивно-технологические показатели	
Прозрачность конструкции	Да/нет
Возможность проведения сравнительного анализа	Да/нет
Перспективность	Да/нет
Запас стойкости	Да/нет

Как видно из таблицы, вектор показателей эффективности содержит достаточно много частных показателей. На сегодняшний день до конца остаются неисследованными вопросы взаимного влияния этих показателей. Исследование сложных зависимостей и разработка методов построения оптимальных схем ПШ являются актуальными на сегодняшний день задачами.

Другим интересным направлением исследований является разработка комплексных показателей, которые интегрируют несколько частных показателей и характеризуют сразу несколько различных свойств схем. В этом случае, при обеспечении заданного уровня качества оценки (точности, доверия и т.д.) сокращается количество показателей и упрощается методика оценки эффективности схем.

Наконец, актуальной остается задача разработки методик оценки эффективности на основе использования методов системного анализа. В области оценки криптографических схем достаточно долго доминировал параметрический подход к оценке эффективности на основе показателей стойкости. Очевидно, что такой подход сегодня уже исчерпал себя. Более перспективным является использование методов определения общесистемных показателей, совместное использование методов расчета количественных показателей и определения качественных показателей (например, использование методов декомпозиции с последующими экспертными оценками). Исследования в данном направлении позволят разработать эффективные методики оценки схем ПШ.

Список литературы: 1. *А.В. Потий, О.И. Олешко.* Новые требования и принципы разработки современных алгоритмов блочного шифрования (по результатам анализа алгоритмов-кандидатов в AES) // Открытые информационные и компьютерные технологии. 2000. Вып. 12. С. 30 – 45. 2.*NESSIE security report*, deliverable D20, October 21, 2002, version 1.0, *http: //www.cryptonessie.org*. 3. *M.J. Robshaw.* Stream Ciphers. Technical Report TR-401, RSA Laboratories, revised July 1995. 4. *J.L. Massey.* «Cryptography and System Theory», Proc. 24th Allerton Conf.Commun., Control, Comput., Oct. 1-3, 1986. 5. *J.L. Massey.* «Shift-register synthesis and BCH decoding», IEEE Trans. Inform. Theory, vol. IT-15, pp. 122 – 127, Jan. 1969. 6. *J.D. Golic.* On the Security of Nonlinear Filter Generators. In Fast Software Encryption – Third International Workshop, Cambridge, February 1996, pp. 173 – 188, Springer-Verlag, berlin, 1996. 7. *V. Chepyzhov, T. Johansson, B. Smeets.* A simple algorithm for fast correlation attacks on stream ciphers. *http: //www.it.lth.se./thomas/*, 2000. 8. *Горбенко И, Потий А, Избенко Ю, Орлова С.* Анализ схем поточного шифрования, представленных на европейский конкурс NESSIE // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні: ДСТСЗІ/СБУ; НТУ «КПІ». 2002. Вып. 5. С. 92 – 110. 9. *FIPS PUB 197:2001.* Advanced Encryption Standard (AES). 10. *Andrew Rukhin, Juan Soto.* A Statistical Test Suite for Random and Pseudorandom Number Generator for Cryptographic Application. NIST Special Publication 800-22, September 2001. 11. *Потий А.В, Орлова С.Ю, Гриненко Т.А.* Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NIST STS // Правове, нормативне та метрологічне забезпечення захисту інформації в Україні: Наук.-техн. зб. 2001. Вип.2. С. 206 – 213. 12. *J. Seberry, X.-M. Zhang and Y. Zheng.* Nonlinearity and Propagation Characteristics of Balanced Boolean Functions. Information and Computation, Vol. 119, No 1, pp. 1 – 13, 1995.

Харьковский национальный
университет радиоэлектроники
Харьковский военный университет

Поступила в редколлегию 14.05.2003

ПЕРСПЕКТИВНЫЕ БЛОЧНЫЕ СИММЕТРИЧНЫЕ ШИФРЫ И ИХ СВОЙСТВА

УДК 681.3.06:519.248.681

И. Д. ГОРБЕНКО, д-р техн. наук, С. А. ГОЛОВАШИЧ, канд. техн. наук

АЛГОРИТМ БЛОЧНОГО СИММЕТРИЧНОГО ШИФРОВАНИЯ «ТОРНАДО». СПЕЦИФИКАЦИЯ ПРЕОБРАЗОВАНИЯ

На протяжении последнего десятилетия в нашей стране проходили процессы интенсивной интеграции в мировое информационное сообщество. Следствием этих процессов стало, с одной стороны, повсеместное внедрение средств вычислительной техники, а с другой – массовое использование сети Internet, как наиболее доступного средства коммуникаций и доступа к неограниченным информационным ресурсам. Сформировавшаяся за это время инфраструктура вычислительных систем использует вычислительную технику зарубежного производства и базируется на открытых каналах телекоммуникаций. При этом информация, обрабатываемая в этих системах, часто носит конфиденциальный характер и требует применения средств криптографической защиты. Неотъемлемым элементом криптографической защиты информации в компьютерных системах стали блочные симметричные шифры (БСШ).

В настоящее время в Украине отсутствует национальный стандарт БСШ, и временно разрешён к применению стандарт бывшего СССР ГОСТ 28147-89, принятый в 1989 году. Однако, учитывая значительный прогресс в сфере методов и средств криптоанализа, на протяжении последних 10–15 лет этот алгоритм уже переходит в класс "морально" устаревших. Косвенным подтверждением этому факту является проведение в США и Европе конкурсов (проекты AES [1] и NESSIE [2] соответственно), направленных на выбор новых криптопримитивов, удовлетворяющих возросшим требованиям безопасности. Так, в соответствии с требованиями проекта NESSIE, алгоритм ГОСТ 28147-89 относится только к третьему (наименьшему) классу стойкости. Поэтому проблема разработки отечественного стандарта БСШ, удовлетворяющего наиболее жёстким требованиям безопасности и учитывающего последние достижения в области методов криптоанализа и криптозащиты симметричных алгоритмов, является весьма актуальной для Украины. Основной целью разработки алгоритма «Торнадо» было решение этой проблемы.

Предлагаемый алгоритм является итеративным «инволютивным» шифром [3] и предусматривает три варианта длины блока (для применения в различных классах безопасности) и переменную длину ключа шифрования для каждого варианта длины блока.

Структура алгоритма «Торнадо» идентична для различных длин блока, отличие заключается только в необходимом числе циклов шифрования. Поэтому описание алгоритма приводится в общем виде для блока длиной **128*n*** бит, где ***n*** – *коэффициент кратности длины блока*, который может принимать значения **1**, **2** или **4**, то есть алгоритм поддерживает блоки длиной **128**, **256** и **512** бит. Для каждой длины блока алгоритм поддерживает 4 значения длины ключа в диапазоне $256n \div 448n$ с шагом $64n$ бита (то есть значения $(4 \div 7) \times 64n$). Отметим, что в данной статье приводится версия 3.0 алгоритма «Торнадо».

1 Обозначения и соглашения

Префикс **0x** будем использовать для обозначения шестнадцатеричных чисел. Например, 0x1AF = 431.

1.1 Список символов и обозначений

$\mathbf{Z}_p$ – кольцо целых чисел по модулю p;
$\mathbf{F}_p$ – поле порядка p (запись эквивалентна GF(p));

$\mathbf{B}$ – векторное пространство 8-битных элементов (байтов), $\mathbf{B} = \mathrm{GF}(2)^8 = \{0,1\}^8$;
$\mathbf{W}$ – векторное пространство 64-битных элементов (слов), $\mathbf{W} = \mathbf{B}^8 = \mathrm{GF}(2)^{64} = \{0,1\}^{64}$;
$\mathbf{H}$ – векторное пространство $64n$-битных элементов (полублоков), $\mathbf{H} = \mathbf{W}^n = \mathrm{GF}(2)^{64n} = \{0,1\}^{64n}$;
$\mathbf{T}$ – векторное пространство $128n$-битных элементов (блоков), $\mathbf{T} = \mathbf{H}^2 = \mathrm{GF}(2)^{128n} = \{0,1\}^{128n}$;
$\mathbf{P}$ – множество 2^{12} «равновероятных» перестановок из 8 элементов. Элементы множества кодируются 12-битными векторами, то есть $\{0,1\}^{12}$. Правило формирования множества приведено в пп. 3.7;
$\mathbf{P}'$ – подмножество перестановок множества $\mathbf{P}$ размерностью 2^8. Элементы множества кодируются 8-битными векторами, то есть $\{0,1\}^8$. Правило формирования подмножества приведено в пп. 3.7;

$\oplus$ – побитовая операция «исключающее ИЛИ» (XOR, сложение по модулю 2);
$\wedge$ – побитовая операция логическое «И» (AND);
$\vee$ – побитовая операция логическое «ИЛИ» (OR);
$\overline{x}$ – операция побитового инвертирования («НЕ», NOT);
$\times$ – операция умножения матрицы на вектор (элементы принадлежат полю $\mathrm{GF}(2^8)$);
$+$ – сложение в кольце $\mathbf{Z}_{2^m}$;
$-$ – вычитание в кольце $\mathbf{Z}_{2^m}$;
$<< l$ – логический сдвиг на l бит влево;
$>> l$ – логический сдвиг на l бит вправо;
$<<< l$ – циклический сдвиг на l бит влево;
$>>> l$ – циклический сдвиг на l бит вправо;
$\mathrm{div}\ m$ – целая часть от деления на m;
$\mathrm{Mod}\ m$ – остаток от деления на m;
$x \| y$ – конкатенация операндов x и y.

Для обозначения совокупностей элементов, представляющих единое целое, будем использовать два способа записи:

$(x_0, \ldots, x_m)$ – способ записи множества либо последовательности элементов, для которых порядок взаимного расположения в памяти шифратора значения не имеет (с точки зрения применяемых операций);
$\langle x_m \| \ldots \| x_0 \rangle$ – способ записи вектора, который отражает необходимый порядок расположения его элементов в памяти шифратора («младшие» справа).

Отдельные разряды будем обозначать b_i, байты – B_i, а слова – W_i, используя при этом *Little Indian* нотацию, то есть нумерация разрядов, байтов, слов и так далее выполняется от младших «весов» к старшим: разряд b_i имеет «вес» 2^i, а байт B_i – «вес» 2^{8i} и так далее.

Например:

$X \in \mathbf{H}=\mathbf{W}^4$ $(n = 4)$, $W_i \in \mathbf{W}$, $B_i \in \mathbf{B}$, $b_i \in \mathrm{GF}(2) \Rightarrow$

$X = \langle W_3 \| W_2 \| W_1 \| W_0 \rangle = \langle B_{31} \| \ldots \| B_1 \| B_0 \rangle = \langle b_{255} \| \ldots \| b_1 \| b_0 \rangle$,

$W_0 = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle = \langle b_{63} \| \ldots \| b_1 \| b_0 \rangle$,

$W_1 = \langle B_{15} \| \ldots \| B_9 \| B_8 \rangle = \langle b_{127} \| \ldots \| b_{65} \| b_{64} \rangle$,

$B_0 = \langle b_7 \| \ldots \| b_1 \| b_0 \rangle$,

$B_{15} = \langle b_{127} \| \ldots \| b_{121} \| b_{120} \rangle$;

$X = 0\text{x}123456789\text{FFEEDDCCBBAA}9988776655443322110 0 \Rightarrow$

$W_0 = 0\text{x}7766554433221100$, $W_1 = 0\text{xFFEEDDCCBBAA9988}$,

$B_0 = 0\text{x}00$, $B_1 = 0\text{x}11$, $B_{15} = 0\text{xFF}$.

Символы операций $\oplus$, $+$, $-$, $<<$, $>>$, $<<<$, $>>>$, заключённые в квадратные скобки (например $[+]$), будем использовать для обозначения операций, выполняемых над словами (64 бита).

Далее будет использоваться следующая нотация:

- верхний индекс (например, Fun^n) – обозначает количество повторений (n) некоторого преобразования (Fun) для векторных аргументов (длиной n элементов);
- нижний индекс (например, X_i) – обозначает номер (позицию) элемента в некоторой последовательности (векторе);
- верхний индекс в круглых скобках (например, $X^{(i)}$) – обозначает номер итерации (или цикла);
- нижний индекс в угловых скобках (например, $X_{<64n>}$) – обозначает разрядность вектора (X);
- символы с 1–3 штрихами (например, X'') – обозначают промежуточные результаты.

1.2 Основные определения

Блоком будем называть битовый кортеж, длина которого соответствует разрядности входа шифратора, то есть $128n$ бит.

Полублоком будем называть битовый кортеж длиной $64n$ битов. Блок (T) состоит из двух полублоков: *левого* L («старшего») и правого R («младшего»), то есть $T = L \| R$.

S-блоком будем называть группу совместно вычисляемых нелинейных булевых функций от общего ограниченного множества аргументов (8 бит), реализуемых, как правило, табличным способом.

Слоем будем называть последовательность идентичных преобразований, выполняемых «параллельно» для всех элементов вектора-аргумента.

Циклической MDS–матрицей будем называть квадратную матрицу размерностью 8×8 (с элементами из поля $\mathrm{GF}(2^8)$), все строки которой получены циклическим сдвигом полинома 7-й степени над полем $\mathrm{GF}(2^8)$, образующего *циклический разделимый код максимального расстояния* (*циклический МДР- или MDS-код*).

Числом ветвей активизации линейного преобразования будем называть минимальное суммарное количество активных S-блоков на входе и выходе этого преобразования при условии фактической активизации входа.

Под одним *циклом* шифрования будем понимать две итерации цепи Фейстеля. Выбор такой нотации обусловлен двумя причинами:

- чётные и нечётные итерации имеют незначительное отличие, в то время как пары соседних итераций полностью идентичны;
- изменение каждого бита блока данных возможно после применения не менее чем 2 итераций цепи Фейстеля.

Исходным (или пользовательским) ключом будем называть битовый вектор, подаваемый на ключевой вход шифратора.

Рабочим ключом будем называть ключевую последовательность, непосредственно используемую процедурой шифрования и полученную в результате работы схемы «разворачивания ключа».

Подключом (рабочего ключа) или *рабочим подключом* будем называть элемент рабочего ключа, используемый на одном цикле, итерации или в отдельном преобразовании. Рабочий подключ, используемый на одной итерации, также будем называть *ключом итерации*.

2 Элементарные преобразования алгоритма «Торнадо»

Все элементарные преобразования в алгоритме «Торнадо» имеют векторную структуру, то есть блок либо полублок данных следует интерпретировать как последовательность слов, которые, в свою очередь, могут рассматриваться как последовательность из 8 байтов.

2.1 Векторные операции над словами

Ниже приведены обозначения элементарных операций, выполняемых над словами, блоками либо полублоками, то есть первый либо оба аргумента имеют вид последовательности из 1, $2n$ либо n слов соответственно. Результат этих операций имеет ту же размерность, что и первый аргумент.

$x\,[\oplus]^n\,y$ – сложение по модулю 2 двух векторов x и y длиной по n слов;

$x\,[+]^n\,y$ – сложение по модулю 2^{64} одноимённых слов, составляющих векторы x и y (длиной по n слов);

$x\,[-]^n\,y$ – вычитание по модулю 2^{64} слов, составляющих вектор y из одноимённых слов, составляющих вектор x (длина по n слов);

$x\,[<<]^n\,l$ – логический сдвиг каждого из n слов, составляющих вектор x, на l бит влево ($0 \le l \le 63$);

$x\,[<<<]^n\,l$ – циклический сдвиг каждого из n слов, составляющих вектор x, на l бит влево ($0 \le l \le 63$);

$x\,[>>>]^n\,l$ – циклический сдвиг каждого из n слов, составляющих вектор x, на l бит вправо ($0 \le l \le 63$).

Для обозначения векторных (биективных) преобразований общего вида над словами будем использовать аналогичную нотацию:

$$Y = \mathrm{Fun}^n(X, K).$$

Подобная запись означает, что входной (X) и выходной (Y) векторы данных состоят из n слов, а вектор управляющего сигнала K (для управляемых отображений) состоит из n последовательных кортежей одинаковой длины, и каждое слово результата Y получается в результате независимого применения преобразования Fun к соответствующему слову аргумента X с использованием соответствующего управляющего кортежа ключа K.

2.2 Элементарные операции над байтами

Кроме перечисленных выше бинарных операций, к элементарным операциям также будем относить унарную операцию над байтами – ***нелинейную подстановку*** 8-битных векторов (***S-блок***). Применение этой операции к отдельному байту будем обозначать: $y = \mathrm{S}_{\nu}(x)$; $x, y \in \mathbf{B}$.

Для обозначения векторных преобразований над байтовыми строками будем использовать обозначения, аналогичные введенным выше, например:

$$Y = [\mathrm{S}_{\nu}]^{8n}(X); \quad X, Y \in \mathbf{H}.$$

Наиболее эффективно преобразования этого класса реализуются табличным способом. В этом случае реализация S-блока может быть эффективно совмещена с реализацией последующего фиксированного преобразования, то есть оба преобразования могут быть выполнены с помощью одной общей таблицы.

В алгоритме используется три различных S-блока 8 в 8 бит (обозначены S_0, S_1, S_2), которые выбираются из множества, так называемых, предельно-нелинейных биективных преобразований [4]. Они строятся на основе конструкции Ниберг-Динга, то есть являются аффинно-эквивалентными функции вычисления обратного элемента в поле $\mathrm{GF}(2^8)$:

$$\mathrm{S}(X) = \mathbf{M}_{\mathrm{y}} \times \left[\left(\mathbf{M}_{\mathbf{x}} \times X \oplus \mathbf{V}_{\mathbf{x}} \right)^{2^8 - 2} \right]_B \oplus \mathbf{V}_{\mathrm{y}}; \quad X, \mathbf{V}_{\mathbf{x}}, \mathbf{V}_{\mathrm{y}} \in \mathbf{F}_2^8; \quad \mathbf{M}_{\mathbf{x}}, \mathbf{M}_{\mathrm{y}} \in \mathbf{GL}\left(8, \mathbf{F}_2\right),$$

где B – некоторый базис над $\mathrm{GF}(2^8)$, определяемый образующим (неприводимым) полиномом;

$\mathrm{M_X}$, $\mathrm{M_Y}$ – квадратные невырожденные матрицы размерностью (8×8).

В последнем соотношении, для упрощения записи, векторы, участвующие в матричном умножении, рассматриваются как вектор–столбцы.

В данной спецификации не приводятся конкретные значения B, $\mathrm{M_X}$, $\mathrm{M_Y}$, $\mathrm{V_X}$, $\mathrm{V_Y}$, что позволяет использовать S-блоки в качестве долговременного секретного ключа шифрования. Применение указанной конструкции позволяет достигнуть предельно низких значений вероятностей заданного трансформирования дифференциальной разности ($p_{\mathrm{S}}^{\mathrm{DC}} = 2^{-6}$) и линейной аппроксимации ($p_{\mathrm{S}}^{\mathrm{LC}} = 2^{-6}$) S-блока, а также получить максимальную алгебраическую степень булевого полинома ($\deg_y = 7$). Кроме того, выбор входного ($\mathrm{M_X}$) и выходного ($\mathrm{M_Y}$) аффинных преобразований может осуществляться на основе дополнительных требований к S-блокам [4].

3 Структура алгоритма «Торнадо»

3.1 Преобразования зашифрования и расшифрования

Алгоритм «Торнадо» является «инволютивным» шифром, то есть зашифрование и расшифрование выполняются на основе одной общей процедуры, отличие заключается только в противоположной последовательности применения ключей итераций $K^{(i)}$, а также ключей начального и конечного преобразований (подключи K^{IT} и K^{FT} соответственно).

Преобразования зашифрование / расшифрования состоит из двух фаз:

1. Процедура разворачивания ключа KShed на основе исходного ключа *UK* (с учётом направления преобразования – зашифрование или расшифрование (e\d)) выполняет формирование рабочего ключа *WK*.
2. Процедура шифрования EF выполняет преобразование входного блока данных (*P* либо *C*) на рабочем ключе *WK*.

$$WK = \mathrm{KShed}(UK,\ mode = \mathrm{e}\backslash \mathrm{d}):$$
$$C = \mathrm{E}_{UK}(P) = \mathrm{EF}_{WK_{\mathrm{E}}}(P),$$
$$P = \mathrm{D}_{UK}(C) = \mathrm{EF}_{WK_{\mathrm{D}}}(C);\quad P, C \in \mathbf{T};$$
$$WK_{\mathrm{E}} = \left(K^{\mathrm{IT}}, K^{(0)}, K^{(1)}, \ldots, K^{(2r-1)}, K^{\mathrm{FT}}\right),$$
$$WK_{\mathrm{D}} = \left(K^{\mathrm{FT}}, K^{(2r-1)}, K^{(2r-2)}, \ldots, K^{(0)}, K^{\mathrm{IT}}\right),$$

где KShed – процедура разворачивания ключа;
mode – режим преобразования: зашифрование (e) / расшифрование (d);
UK – исходный (пользовательский) ключ;
WK_{E} – рабочий ключ зашифрования;
WK_{D} – рабочий ключ расшифрования;
$K^{(i)}$ – ключ для i-й итерации;
E_k – преобразование зашифрования на исходном ключе k;
D_k – преобразование расшифрования на исходном ключе k;
EF_k – процедура шифрования на рабочем ключе k;
P – блок входного («открытого») текста;
C – блок выходного текста («криптограммы»).

3.2 Процедура шифрования EF

Алгоритм «Торнадо» является итеративным шифром, то есть основу его процедуры шифрования составляет цикловое преобразование, которое повторяется заданное число раз (обозначим число циклов шифрования символом r). Кроме повторяемого циклового преобразования процедура шифрования включает начальное (IT) и конечное (FT) преобразования. Свойство инволютивности шифра достигается за счёт применения обобщённой конструкции полублоковой цепи Фейстеля. Структура процедуры шифрования EF приведена на рис. 1.

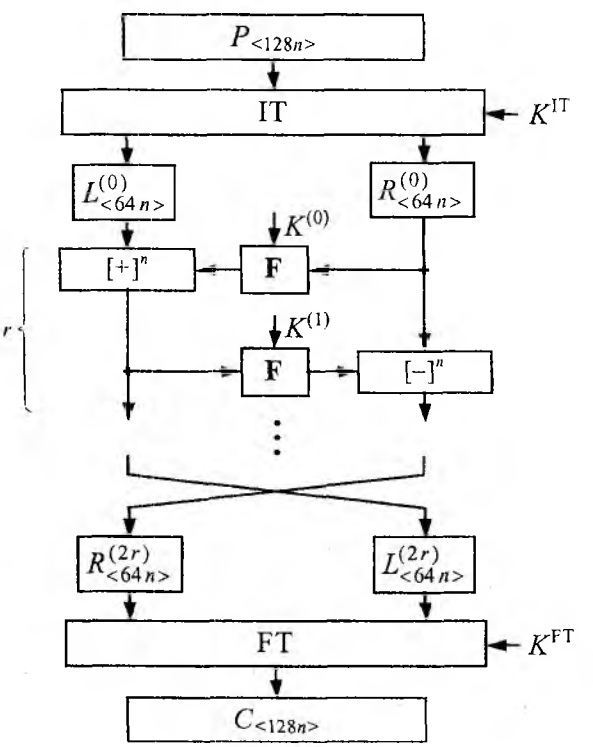


Рис. 1

Количество циклов шифрования (r) определяется длиной блока текста, соответствующие значения приведены в табл. 1. Выбранные значения рассчитаны в соответствии с практическим критерием оценки стойкости БСШ [5].

Таблица 1

Длина блока $l_Б$, бит	Коэффициент n	Число циклов r
128	1	4
256	2	6
512	4	10

Процедура шифрования EF состоит из трёх шагов:

1. Исходный блок данных P (длиной $128n$ бит) обрабатывается начальным (IT) преобразованием на ключе K^{IT}.
2. Результат 1-го шага разбивается на два *полублока* длиной по $64n$ бита: левый L («старший») и правый R («младший»). Полученная пара полублоков преобразуется на r циклах, каждый из которых состоит из двух итераций. Обе итерации используют общее, управляемое подключом $K^{(i)}$, нелинейное преобразование – F-функцию. Единственное отличие этих двух итераций заключается в том, что на первой итерации выход F-функции объединяется с левым полублоком операцией сложения по модулю 2^{64} (обозначение [+]), а на второй итерации – операцией вычитания по модулю 2^{64} (обозначение [–]).
3. Два полублока L и R, полученные в результате r-циклового итеративного преобразования, меняются местами и обрабатываются конечным (FT) преобразованием на ключе K^{FT}. Полученный после преобразования двоичный вектор C (длиной $128n$ бит) является результирующим блоком («криптограммой»).

Аналитически процедура EF может быть представлена следующим образом:

$$C = \mathrm{EF}_{WK}(P):$$

$$\left\langle L^{(0)} \parallel R^{(0)} \right\rangle = \mathrm{IT}\left(P, K^{\mathrm{IT}}\right);$$

$$\begin{matrix} i = 2j, \\ j = \overline{0, r-1} \end{matrix}: \quad \begin{cases} L^{(i+1)} = L^{(i)} \,[+]^n\, \mathrm{F}\left(R^{(i)}, K^{(i)}\right), & R^{(i+1)} = R^{(i)} \\ L^{(i+2)} = L^{(i+1)}, & R^{(i+2)} = R^{(i+1)} \,[-]^n\, \mathrm{F}\left(L^{(i+1)}, K^{(i+1)}\right) \end{cases};$$

$$C = \mathrm{FT}\left(\left\langle R^{(2r)} \parallel L^{(2r)} \right\rangle, K^{\mathrm{FT}}\right); \quad P, C \in \mathbf{T}; \quad L^{(i)}, R^{(i)} \in \mathbf{H}.$$

В приведенных выше соотношениях символом i обозначается номер итерации, а символом j – номер цикла (в обоих случаях нумерация выполняется с 0).

На каждой итерации F-функция использует $148n$-битный подключ $K^{(i)}$, длина ключей начального (K^{IT}) и конечного (K^{FT}) преобразований составляет по $128n$ бит каждый.

3.3 Базовая F*-функция*

Конструкция F-функции алгоритма «Торнадо» построена в соответствии с принципами, позволяющими обеспечить свойства рассеивания и размножения активизации [6].

F-функция условно может быть разбита на четыре преобразования:

- сложение полублока с подключом по модулю 2;
- нелинейное ключезависимое «смешивающее» преобразование SCL;
- нелинейное ключезависимое «рассеивающее» преобразование SCP;
- фиксированная байтовая перестановка P_{byte}.

Подробно эти преобразования будут рассмотрены ниже.

F-функция имеет следующее аналитическое представление:

$$Y = \mathrm{F}\left(X, K^{(i)}\right) = \mathrm{SCP}^n\left(\mathrm{P}_{\mathrm{byte}}\left(\mathrm{SCL}^n\left(X \oplus xk_1^{(i)}, zk_2^{(i)}\right) \oplus xk_2^{(i)}\right), zk_2^{(i)}\right);$$

$$K^{(i)} = \left(xk_1^{(i)}, zk_1^{(i)}, xk_2^{(i)}, zk_2^{(i)}\right); \quad X, Y, xk_1^{(i)}, xk_2^{(i)} \in \mathbf{H}; \quad zk_1^{(i)} \in \mathbf{P}'^n; \quad zk_2^{(i)} \in \mathbf{P}^n.$$

В общем виде F-функция может быть представлена схемой, приведенной на рис. 2. На одной итерации F-функция использует подключ длиной $148n$ бит, который состоит из трёх составляющих:

- $xk_1^{(i)}$ и $xk_2^{(i)}$ – ключи сложения по модулю 2, длиной по $64n$ бита каждый;
- $zk_1^{(i)}$ и $zk_2^{(i)}$ – ключи байтовой перестановки (пространство значений 2^{8n} и 2^{12n} соответственно).

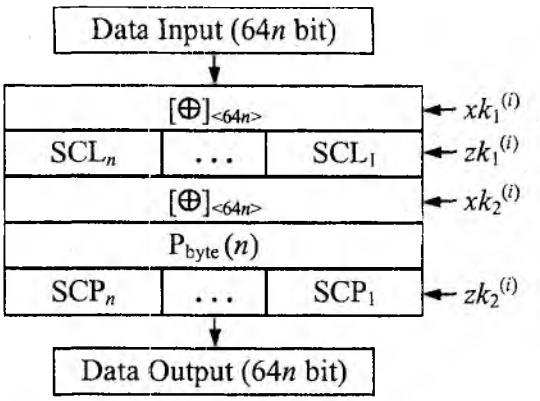


Рис. 2

Частный случай F-функции, когда $n = 1$ (то есть длина полублока составляет 64 бита), представлен на рис. 3. В этом случае ($n = 1$) преобразование $\mathrm{P}_{\mathrm{byte}}$, как будет показано ниже, является тождественным, поэтому на рисунке оно отсутствует.

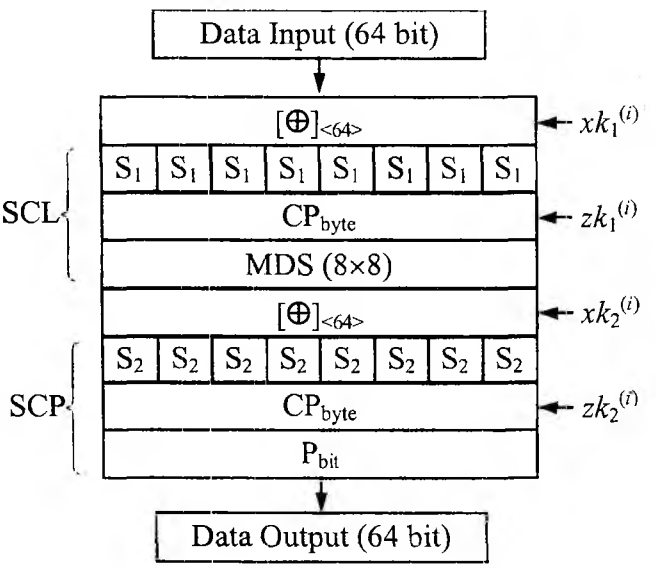


Рис. 3

При выполнении векторных преобразований SLM^n и SCP^n полублок данных рассматривается как последовательность из n слов (по 64 бита), к каждому из которых независимо применяется соответствующее преобразование. Аналогичным образом подключ, используемый для «управления» векторным преобразованием, разбивается на n последовательных кортежей равной длины:

$$X, Y \in \mathbf{H}; \quad zk \in \mathbf{P}^n; \quad W_j, W'_j \in \mathbf{W}; \quad zw_j \in \mathbf{P}; \quad j = \overline{0, n-1};$$
$$X = \langle W_{n-1} \| \ldots \| W_1 \| W_0 \rangle,$$
$$Y = \langle W'_{n-1} \| \ldots \| W'_1 \| W'_0 \rangle,$$
$$zk = \langle zw_{n-1} \| \ldots \| zw_1 \| zw_0 \rangle;$$
$$Y = \mathrm{SCL}^n(X, zk) \Rightarrow W'_j = \mathrm{SCL}(W_j, zw_j),$$
$$Y = \mathrm{SCP}^n(X, zk) \Rightarrow W'_j = \mathrm{SCP}(W_j, zw_j).$$

3.4 Операция обмена битов между двумя полублоками $\mathbf{CX_{bit}}$

В составе начального (IT) и конечного (FT) преобразований используется операция *управляемого «обмена» одноимёнными битами* (CX_{bit}) между левым (L) и правым (R) полублоками. Под одноимёнными понимаются разряды, расположенные в идентичных позициях полублоков-аргументов. Подмножество битов, подлежащих перестановке, определяется единичными битами управляющей маски X (длиной один полублок). В алгоритме «Торнадо» используются *управляющие маски* только специального вида – маска всегда получается путём тиражирования $32n$ раза некоторого 4-битного кортежа, поэтому мы будем использовать сокращённую форму записи (например, $X = 0x55...55$).

Преобразование CX_{bit} может быть представлено следующим образом:

$$\mathrm{CX}_{\mathrm{bit}}(\langle L \| R \rangle, X) = \langle (((L \oplus R) \wedge X) \oplus L \| (((L \oplus R) \wedge X) \oplus R \rangle; \quad L, R, X \in \mathbf{H}.$$

3.5 Начальное IT *и конечное* FT *преобразования*

Начальное IT (рис. 4) и конечное FT (рис. 5) преобразования включают сложение блока данных с рабочим подключом (длиной $128n$ бит) и фиксированное преобразование блока данных. Фиксированные (то есть не зависящие от ключа) составляющие преобразований IT и FT являются взаимно обратными и включают два «слоя»:

- нелинейная подстановка («S–слой»);
- битовая перестановка («P–слой»).

«P–слой» обоих преобразований удобно рассматривать как последовательность из двух операций:

- циклический сдвиг каждого слова блока влево либо вправо на ($32n$ + 4) разряда;
- операция «обмена» одноимёнными битами между полублоками (CX_{bit}).

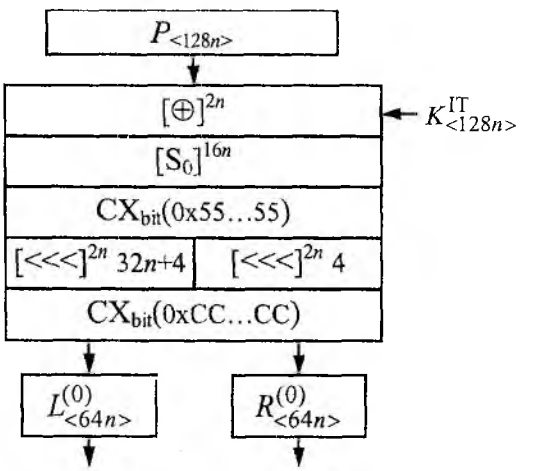


Рис. 4

Начальное преобразование (IT) может быть представлено следующей последовательностью операций:

$$\langle L^{(0)} \| R^{(0)} \rangle = \mathrm{IT}\left(P, K^{\mathrm{IT}}\right): \quad L, R \in \mathbf{H}; \quad P, K^{\mathrm{IT}} \in \mathbf{T};$$

$$1)\ \langle L' \| R' \rangle == [\mathrm{S}_0]^{16n}\left(P\ [\oplus]^{2n}\ K^{\mathrm{IT}}\right);$$

$$2)\ \langle L'' \| R'' \rangle = \mathrm{CX}_{\mathrm{bit}}\left(\langle L' \| R' \rangle, 0\mathrm{x}55...55\right);$$

$$3)\ \langle L''' \| R''' \rangle = \left\langle \left(L''\ [<<<]^{2n}\ (32n+4)\right) \| \left(R''\ [<<<]^{2n}\ 4\right) \right\rangle;$$

$$4)\ \left\langle L^{(0)} \| R^{(0)} \right\rangle = \mathrm{CX}_{\mathrm{bit}}\left(\langle L''' \| R''' \rangle, 0\mathrm{xCC}...\mathrm{CC}\right),$$

где P – исходный блок текста на входе шифратора («открытый текст»);
$L^{(0)}$, $R^{(0)}$ – соответственно левый и правый полублоки на выходе IT-преобразования.

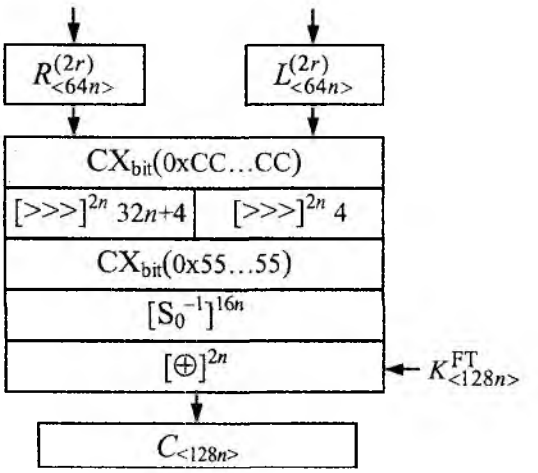


Рис. 5

Конечное преобразование (FT) может быть представлено следующей последовательностью операций:

$$C = \mathrm{FT}\left(\left\langle L^{(2r)} \| R^{(2r)} \right\rangle, K^{\mathrm{FT}}\right): \quad L, R \in \mathbf{H}; \quad C, K^{\mathrm{FT}} \in \mathbf{T};$$

$$1)\ \langle L' \| R' \rangle = \mathrm{CX}_{\mathrm{bit}}\left(\left\langle R^{(2r)} \| L^{(2r)} \right\rangle, 0\mathrm{xCC}...\mathrm{CC}\right);$$

$$2)\ \langle L'' \| R'' \rangle = \left\langle \left(L'\ [>>>]^{2n}\ (32n+4)\right) \| \left(R'\ [>>>]^{2n}\ 4\right) \right\rangle;$$

$$3)\ \langle L''' \| R''' \rangle = \mathrm{CX}_{\mathrm{bit}}\left(\langle L'' \| R'' \rangle, 0\mathrm{x}55...55\right);$$

$$4)\ C = [\mathrm{S}_0^{-1}]^{16n}\left(\langle L''' \| R''' \rangle\right)\ [\oplus]^{2n}\ K^{\mathrm{FT}},$$

где C – результирующий блок текста на выходе шифратора («криптограмма»);
$L^{(2r)}$, $R^{(2r)}$ – соответственно левый и правый полублоки на входе FT-преобразования.

3.6 Фиксированная перестановка байтов $\mathbf{P_{byte}(n)}$

Вид байтовой перестановки $\mathrm{P}_{\mathrm{byte}}(n)$ определяется количеством 64-битных слов в полублоке (то есть коэффициентом n). Байтовая перестановка выполняется в соответствии со следующим соотношением:

$$\mathrm{P}_{\mathrm{byte}}(n):\quad B'_i = B_{n\times(i \bmod 8)+(i \operatorname{div} 8)},\quad B_i \in \mathbf{B},\quad i = \overline{0, 8n-1},$$

где B_j – байт-источник (j – позиция байта в исходном полублоке);
B_i' – байт-получатель (i – позиция байта в результирующем полублоке).

На рис. 6 – 8 показаны схемы перестановки байтов ($\mathrm{P}_{\mathrm{byte}}$) для всех допустимых значений n (то есть 1, 2 и 4). На приведенных схемах каждая ячейка соответствует одному байту, «большое» число в ячейке – исходная позиция (64-битного) слова в полублоке, а нижний индекс – исходная позиция байта внутри соответствующего (64-битного) слова.

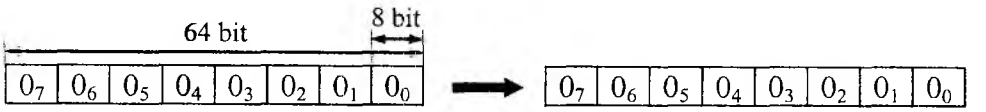


Рис. 6 (случай $n = 1$)

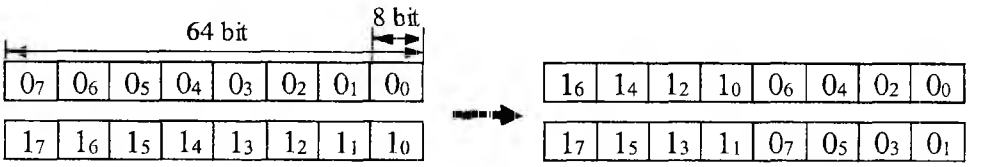


Рис. 7 (случай $n = 2$)

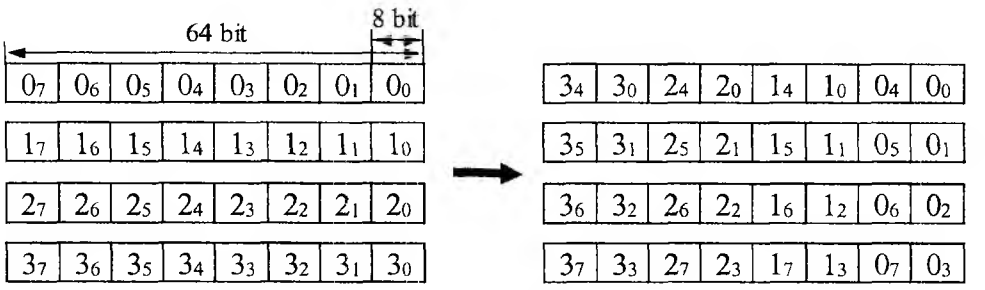


Рис. 8 (случай $n = 4$)

3.7 Функции α и β

Как было показано выше, F-функция алгоритма «Торнадо», кроме операции "сложения по модулю 2" (для ввода циклового ключа), использует управляемую перестановку байтов внутри слова (преобразование $\mathrm{CP}_{\mathrm{byte}}$), то есть "операцию" перестановки на множестве из 8 элементов, соответствующих позициям байтов внутри слова. При этом размерность пространства допустимых перестановок составляет 2^{12}. Перестановки, принадлежащие этому пространству, будем обозначать π_z, где z – индекс или ключ, выбирающий одну из допустимых перестановок.

Ключ перестановки удобно рассматривать как три кортежа по 4 бита:

$$z = \langle k4 \| k2 \| k1 \rangle,\quad z \in \{0,1\}^{12} = \mathbf{P};\quad k1, k2, k4 \in \{0,1\}^4;$$

$$z' = \langle k2 \| k1 \rangle,\quad z' \in \{0,1\}^8 = \mathbf{P}',\quad k4 = 0.$$

Перестановка π_z может быть представлена композицией трёх "элементарных" управляемых перестановок $\sigma_1, \sigma_2, \sigma_4$:

$$\sigma_t = \begin{pmatrix} a_7 & \dots & a_1 & a_0 \\ a'_7 & \dots & a'_1 & a'_0 \end{pmatrix}.$$

Каждая из этих "элементарных" перестановок представляет собой композицию четырёх независимых транспозиций τ, управляемых одним битом ключа. Отдельную управляемую транспозицию обозначим:

$$\tau(i,j,k) = \begin{cases} a'_i \leftarrow a_i,\ a'_j \leftarrow a_j, & \text{если } k = 0 \\ a'_i \leftarrow a_j,\ a'_j \leftarrow a_i, & \text{если } k = 1 \end{cases}, \quad k \in \{0,1\},$$

где i, j – позиции двух элементов, подлежащих "управляемой транспозиции";
k – бит ключа, определяющий, выполнять ($k = 1$) или не выполнять ($k = 0$) транспозицию.

Каждая из перестановок $\sigma_1, \sigma_2, \sigma_4$ в качестве ключа использует соответствующий 4-битный кортеж:

σ_1: $\left\{ i = \overline{0,3}:\ \tau(2i+0,\ 2i+1,\ k1_i) \right\}$;

σ_2: $\left\{ i = \overline{0,3}:\ \tau(j+0,\ j+2,\ k2_i),\quad j = 4 \times (i \operatorname{div} 2) + (i \bmod 2) \right\}$;

σ_4: $\left\{ i = \overline{0,3}:\ \tau(i+0,\ i+4,\ k4_i) \right\}$,

где kt_i – бит i ключевого кортежа kt (t = 1, 2, 4).

Наглядное представление перестановки π_z приведено на рис. 9.

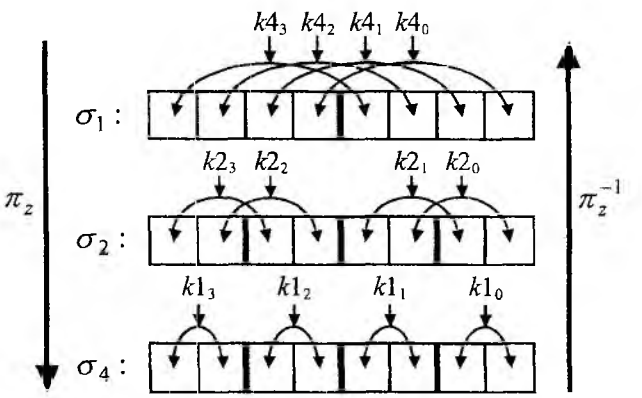


Рис. 9

Для удобства последующего описания преобразования CP_{byte}, на основе рассмотренных управляемых перестановок $\sigma_1, \sigma_2, \sigma_4$, определим две функции – α и β. Эти функции будут выполнять формирование соответственно прямой π_z и обратной π_z^{-1} перестановок на множестве чисел {0, 1, … , 7} по заданному 12 (или 8) –битному ключу z (или z').

Функция α: $(zd_0, zd_1, \dots, zd_7) \leftarrow \alpha(z),\quad 0 \le zd_j \le 7;$

$$zd_i = \pi_z(i),\quad i = \overline{0,7};\quad \pi_z = \sigma_4 \circ \sigma_2 \circ \sigma_1,$$

где i – позиция (индекс) байта-источника;
zd_i – позиция (индекс) байта-получателя для "источника" i.

Функция β:
$$(zs_0, zs_1, \ldots, zs_7) \leftarrow \beta(z), \quad 0 \le zs_j \le 7;$$
$$zs_i = \pi_z^{-1}(i), \quad i = \overline{0,7}; \quad \pi_z^{-1} = \sigma_1 \circ \sigma_2 \circ \sigma_4,$$

где i – позиция (индекс) байта-получателя;
zs_i – позиция (индекс) байта-источника для "получателя" i.

3.8 Управляемая перестановка байтов $\mathbf{CP}_{\text{byte}}$

Как было отмечено выше, преобразование CP_{byte} выполняет управляемую ключом перестановку байтов внутри слова. В качестве управляющей информации данное преобразование использует 12-битный ключ перестановки zw. В соответствии с функциями $\alpha(zw)$ либо $\beta(zw)$ этот ключ «разворачивается» в последовательность из 8 кортежей по 3 бита, каждый из которых определяет позицию соответственно байта-получателя или байта-источника.

$$\text{CP}_{\text{byte}}: \quad \mathbf{W} \times \mathbf{P} \to \mathbf{W}$$
$$Y = \text{CP}_{\text{byte}}(X, zw),$$
$$X = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle, \quad Y = \langle B'_7 \| \ldots \| B'_1 \| B'_0 \rangle,$$
$$B_j, B'_j \in \mathbf{B}; \quad X, Y \in \mathbf{W}; \quad zw \in \mathbf{P}.$$

Перестановка может быть выполнена двумя способами:

Вариант 1:
$$zw \xrightarrow{\alpha} (zd_0, zd_1, \ldots, zd_7), \quad 0 \le zd_j \le 7;$$
$$j = \overline{0,7}: \quad B'_{zd_j} = B_j.$$

Вариант 2:
$$zw \xrightarrow{\beta} (zs_0, zs_1, \ldots, zs_7), \quad 0 \le zs_j \le 7;$$
$$j = \overline{0,7}: \quad B'_j = B_{zs_j}.$$

3.9 «Расширяющая» битовая перестановка $\mathbf{P}_{1\text{bit}}$

Перестановка $\text{P}_{1\text{bit}}$ является вспомогательной и приводится только для демонстрации одного из способов эффективной реализации шифра.

$$\text{P}_{1\text{bit}}: \quad \mathbf{B} \to \mathbf{W}$$
$$\langle B'_7 \| \ldots \| B'_1 \| B'_0 \rangle = \text{P}_{\text{bit}}(\langle b_7 \| \ldots \| b_1 \| b_0 \rangle),$$
$$B'_j = \langle 0 \| \ldots \| 0 \| b_j \rangle, \quad b_j \in \{0,1\}, \quad B'_j \in \mathbf{B}, \quad j = \overline{0,7}.$$

3.10 «Равномерная» битовая перестановка $\mathbf{P}_{\text{bit}}$

Преобразование P_{bit} определяет фиксированную перестановку битов внутри слова. Перестановка имеет следующий вид:

$$\mathrm{P_{bit}}: \quad \mathbf{W} \to \mathbf{W}$$
$$Y = \mathrm{P_{bit}}(X); \quad X, Y \in \mathbf{W};$$
$$X = \langle b_{63} \| \ldots \| b_1 \| b_0 \rangle, \quad Y = \langle b'_{63} \| \ldots \| b'_1 \| b'_0 \rangle;$$
$$b'_{8\times j+i} = b_{8\times i+j}; \quad i, j = \overline{0,7}; \quad b_t, b'_t \in \{0,1\}.$$

Эквивалентно преобразование $\mathrm{P_{bit}}$ может быть представлено с помощью «расширяющей» перестановки $\mathrm{P_{1bit}}$:

$$Y = \bigvee_{j=0}^{7} \left(\mathrm{P_{1bit}}(B_j) << j\right).$$

3.11 Умножение байта на MDS-код специального вида

Это преобразование определяет операцию умножения элемента поля $\mathrm{GF}(2^8)$, соответствующего байту-аргументу, на фиксированный полином 7 степени с коэффициентами из $\mathrm{GF}(2^8)$. Этот полином должен формировать циклический код максимального расстояния (МДР или MDS -код), а также обладать рядом дополнительных свойств, приведенных ниже. Преобразование имеет следующий вид:

$$\mathrm{LCM}: \quad \mathbf{B} \to \mathbf{W}$$
$$Y = \mathrm{LCM}(x), \quad x \in \mathbf{B}, \quad Y \in \mathbf{W};$$
$$Y = \langle g_7 x \| \ldots \| g_1 x \| g_0 x \rangle; \quad x, g_i, (g_i x) \in \mathrm{F}_{2^8}; \quad Y \in \mathrm{F}_{2^8}^8;$$
$$\forall i, j, \ (i \neq j): \quad g_i \neq g_j, \quad 0 \le i, j \le 7,$$

где g_i – коэффициенты полинома, образующего циклический MDS–код;
x – входной байт;
Y – выходное слово.

Кроме того, коэффициенты полинома g_i должны быть выбраны таким образом, чтобы все булевы функции выхода преобразования были уникальны, то есть удовлетворяли следующим условиям:

$$Y = \langle y_{63} \| \ldots \| y_1 \| y_0 \rangle, \quad y_i = f_i(x), \quad y_i \in \mathrm{GF}(2);$$
$$\forall i, j, \ (i \neq j): \quad f_i(x) \neq f_j(x), \quad 0 \le i, j \le 63,$$

где y_i – i –й бит слова-результата Y;
f_i – булева функция i –го выхода преобразования.

Отметим, что все функции f_i являются линейными, то есть $\deg(f_i) = 1$.

3.12 Умножение слова на MDS-матрицу

Это преобразование определяет биективное линейное отображение слов, составляющих полублок. Преобразование имеет следующий вид:

$$\mathrm{MDS}: \quad \mathbf{W} \to \mathbf{W}$$
$$Y = \mathrm{MDS}(X); \quad X, Y \in \mathbf{W}; \quad B_j \in \mathbf{B};$$
$$X = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle;$$
$$Y = \bigoplus_{j=0}^{7} \mathrm{LCM}\left(B_j\right) <<< (8\times j).$$

Это линейное преобразование может быть представлено в виде матричного умножения – квадратная матрица размерностью 8×8 байт, образованная циклическим MDS–кодом (рассмотренным в п.п. 3.11), умножается справа на вектор-столбец длиной 8 байт (соответствующий слову-аргументу). Полученный вектор-столбец (длиной 8 байт) соответствует слову–результату. Байты, составляющие матрицу и оба вектора, интерпретируются как элементы поля $GF(2^8)$, образованного *выбранным* неприводимым полиномом 8-й степени.

3.13 Преобразование SCL

Преобразование SCL выполняет управляемое биективное отображение слов. В качестве управляющей информации данное преобразование использует 8-битный ключ перестановки *zw*. Преобразование состоит из следующей последовательности шагов:

$$\mathrm{SCL}: \quad \mathbf{W}\times\mathbf{P}'\to\mathbf{W}$$
$$Y=\mathrm{SCL}(X, zw);$$
$$X=\langle B_7 \| \ldots \| B_1 \| B_0\rangle; \quad B_j\in\mathbf{B}; \quad X,Y\in\mathbf{W}; \quad zw\in\mathbf{P}';$$
$$1)\ X'=\langle \mathrm{S}_2(B_7)\| \ldots \| \mathrm{S}_2(B_1)\| \mathrm{S}_2(B_0)\rangle;$$
$$1)\ X''=\mathrm{CP}_{\mathrm{byte}}(X', zw);$$
$$2)\ Y=\mathrm{MDS}(X''),$$

где X – входное слово (64 бита);
Y – выходное слово (64 бита).

Таким образом, SCL преобразование может быть реализовано двумя способами:

Вариант 1:
$$zw\xrightarrow{\alpha}(zd_0, zd_1, \ldots, zd_7), \quad 0\le zd_j\le 7;$$
$$Y=\bigoplus_{j=0}^{7}\mathrm{LCM}\left(\mathrm{S}_1(B_j)\right)<<<(8\times zd_j).$$

Вариант 2:
$$zw\xrightarrow{\beta}(zs_0, zs_1, \ldots, zs_7), \quad 0\le zs_j\le 7;$$
$$Y=\bigoplus_{j=0}^{7}\mathrm{LCM}\left(\mathrm{S}_1(B_{zs_j})\right)<<<(8\times j).$$

Отметим, что применение циклической матрицы позволяет реализовать операцию умножения на матрицу в виде одной таблицы размером 256 слов (то есть 256×8 байт = 2 КБайта), которая, в свою очередь, может быть совмещена с таблицей вычисления S-блока S_1.

3.14 Преобразование SCP

Преобразование SCP выполняет управляемое биективное отображение слов. В качестве управляющей информации данное преобразование использует 12-битный ключ перестановки *zw*.

SCP–преобразование имеет следующую структуру:

$$\mathrm{SCP}: \quad \mathbf{W}\times\mathbf{P}\to\mathbf{W}$$

$$Y=\mathrm{SCP}(X,zw);$$

$$X=\langle B_7\,\|\,\ldots\,\|\,B_1\,\|\,B_0\rangle;\quad B_j\in\mathbf{B};\quad X,Y\in\mathbf{W};\quad zw\in\mathbf{P};$$

1) $X'=\langle \mathrm{S}_2(B_7)\,\|\,\ldots\,\|\,\mathrm{S}_2(B_1)\,\|\,\mathrm{S}_2(B_0)\rangle;$

2) $X''=\mathrm{CP}_{\mathrm{byte}}(X',\,zw);$

3) $Y=\mathrm{P}_{\mathrm{bit}}(X'').$

Это преобразование может быть реализовано двумя способами:

Вариант 1:

$$zw\xrightarrow{\alpha}(zd_0,zd_1,\ldots,zd_7),\quad 0\le zd_j\le 7;$$

$$Y=\bigvee_{j=0}^{7}\mathrm{P}_{1\,\mathrm{bit}}\left(\mathrm{S}_2(B_j)\right)<<zd_j.$$

Вариант 2:

$$zw\xrightarrow{\beta}(zs_0,zs_1,\ldots,zs_7),\quad 0\le zs_j\le 7;$$

$$Y=\bigvee_{j=0}^{7}\mathrm{P}_{1\,\mathrm{bit}}\left(\mathrm{S}_2(B_{zs_j})\right)<<j.$$

Отметим, что применение идентичной битовой перестановки $\mathrm{P}_{1\mathrm{bit}}$ для всех байтов, составляющих слово, позволяет реализовать это преобразование в виде одной таблицы размером 256 слов (то есть 256×8 байт = 2 КБайта), которая, в свою очередь, может быть совмещена с таблицей вычисления S-блока S_2.

3.15 Процедура «разворачивания ключа»

Процедура «разворачивания ключа» алгоритма «Торнадо» построена на базе криптографического генератора псевдослучайных последовательностей (КГПСП), его структурная схема приведена на рис. 10.

КГПСП, используемый процедурой разворачивания ключа алгоритма «Торнадо», построен по схеме циклического шифрования пяти *«слов» регистра состояния генератора* в режиме «связки шифроблоков» (CBC-режим). По разрядности каждое *«слово» регистра состояния* эквивалентно полублоку (то есть 64*n* бит). Шифрование *«слов» регистра состояния* выполняется с помощью базовой F-функции, однако в качестве управляющих, используются только ключевые входы xk_1 и xk_2, а на входы zk_1 и zk_2 (ключи перестановки) всегда подаётся постоянное значение $zw^{\mathrm{fix}}=0$. То есть перестановки $\mathrm{CP}_{\mathrm{byte}}$ внутри F-функции КГПСП являются фиксированными и определяют тождественную перестановку.

Введём определение *«сырой» ключевой последовательности*, под которой будем понимать последовательность ключевых полублоков, сформированную на выходе КГПСП, до "усечения" ключей перестановки (шаги процедуры WKGen 1–7, см. ниже).

Таким образом, на каждом шаге формирования *«сырой» ключевой последовательности* текущее состояние КГПСП определяется содержимым *«пятисловного» регистра состояния* $S_{\mathrm{ks}}^{(i)}=(s_0^{(i)},s_1^{(i)},s_2^{(i)},s_3^{(i)},s_4^{(i)})$, а также *«двухсловного» управляющего ключа* $K^{\mathrm{ks}}=(xk_1^{\mathrm{ks}},xk_2^{\mathrm{ks}})$. Разрядность каждого *«слова»* составляет 64*n* бит (полублок).

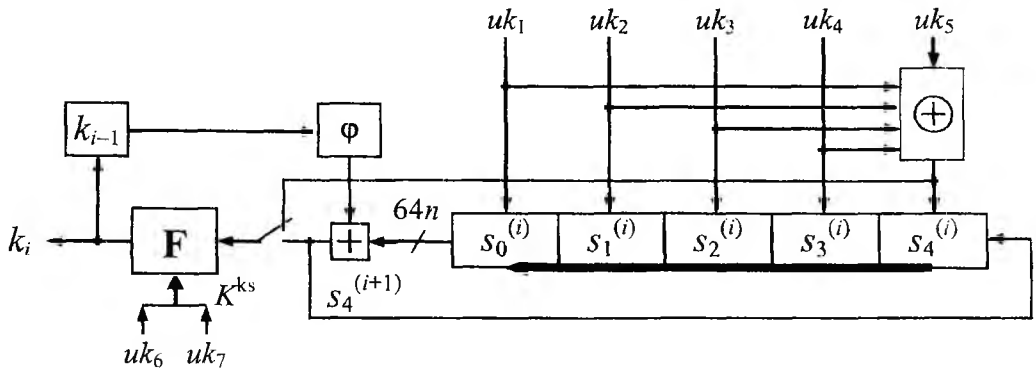


Рис. 10

Алгоритм «Торнадо» поддерживает переменную длину исходного ключа *UK*. Длина l_{UK} исходного ключа измеряется в полублоках и может находиться в диапазоне от 4 до 7 включительно (то есть $4 \le l_{\mathrm{UK}} \le 7$).

Для описания процедуры разворачивания ключа воспользуемся следующими обозначениями:

$$K^{(i)} = \left(xk_1^{(i)}, zk_1^{(i)}, xk_2^{(i)}, zk_2^{(i)}\right),$$

$$k_i = \left\langle kw_{i,n-1} \| \ldots \| kw_{i,1} \| kw_{i,0} \right\rangle,$$

$$zk_t^{(i)} = \left\langle zw_{t,n-1}^{(i)} \| \ldots \| zw_{t,1}^{(i)} \| zw_{t,0}^{(i)} \right\rangle;$$

$$k_i^{(i)}, xk_1^{(i)}, xk_2^{(i)} \in \mathbf{H}; \quad kw_{i,p} \in \mathbf{W};$$

$$zk_1^{(i)} \in \mathbf{P}'^n; \quad zw_{1,p}^{(i)} \in \mathbf{P}'; \quad zk_2^{(i)} \in \mathbf{P}^n; \quad zw_{2,p}^{(i)} \in \mathbf{P}.$$

где $K^{(i)}$ – рабочий подключ i-ой итерации;

$xk_1^{(i)}, xk_2^{(i)}$ – подключи «сложения» i-ой итерации;

k_i – i-й полублок «сырого» развёрнутого ключа (i-й выход КГПСП);

$kw_{i,p}$ – p-ое (64-битное) слово i-го полублока «сырого» развёрнутого ключа;

$zw_{1,p}^{(i)}$ – ключ SCL перестановки для слова p итерации i (длина 8 бит);

$zw_{2,p}^{(i)}$ – ключ SCP перестановки для слова p итерации i (длина 12 бит).

Начальное состояние КГПСП определяется значением исходного ключа $UK = \left\langle uk_{l_{\mathrm{UK}}} \| \ldots \| uk_2 \| uk_1 \right\rangle$, где uk_i соответствует одному полублоку.

Процедура разворачивания ключа может быть представлена следующей последовательностью шагов.

WKGen:

INPUT: $l_{\mathrm{UK}}, uk_1, uk_2, \ldots, uk_{l_{\mathrm{UK}}} \quad (4 \le l_{\mathrm{UK}} \le 7)$.

OUTPUT: $K^{(0)}, K^{(1)}, \ldots, K^{(2r-1)}, K^{\mathrm{IT}}, K^{\mathrm{FT}}$.

$k_i, s_j^{(i)}, uk_t, xk^{\mathrm{ks}}, xk^{\mathrm{fix}}, g_\varphi \in \mathbf{H}; \quad zk_t^{(i)}, zk^{\mathrm{fix}} \in \mathbf{P}^n$.

Если длина пользовательского ключа меньше 7, то выполняется его доопределение:

1) $if\,(l_{\mathrm{UK}} < 5)\ then\ uk_5 = 0;$

2) $if\,(l_{\mathrm{UK}} < 6)\ then\ uk_6 = xk_1^{\mathrm{fix}};$

3) $if\,(l_{\mathrm{UK}} < 7)\ then\ uk_7 = xk_2^{\mathrm{fix}}.$

Доопределённый пользовательский ключ используется для инициализации схемы разворачивания ключа:

4) $(s_0^{(0)}, s_1^{(0)}, s_2^{(0)}, s_3^{(0)}, s_4^{(0)}) = (uk_1, uk_2, uk_3, uk_4,\ uk_1 \oplus uk_2 \oplus uk_3 \oplus uk_4 \oplus uk_5);$

5) $K^{\mathrm{ks}} = (xk_1^{\mathrm{ks}}, zk_1^{\mathrm{ks}}, xk_2^{\mathrm{ks}}, zk_2^{\mathrm{ks}}) \ \Rightarrow\ xk_1^{\mathrm{ks}} = uk_6,\quad xk_2^{\mathrm{ks}} = uk_7,\quad zk_1^{\mathrm{ks}} = zk_2^{\mathrm{ks}} = zk^{\mathrm{fix}} = 0;$

6) $k_{-1} = \mathrm{F}_{K^{\mathrm{ks}}}\left(s_4^{(0)}\right),\quad g_\varphi = \left(k_{-1} \wedge 15\right) \vee 1.$

После выполнения фазы инициализации выполняется формирование рабочего ключа:

7) for $i = \overline{0, 5r+3}$: $k_i = \mathrm{F}_{K^{\mathrm{ks}}}\left(s_0^{(i)}\right),\quad s_4^{(i+1)} = s_0^{(i)}\,[+]^n\,\varphi(k_{i-1}),$

$$s_j^{(i+1)} = s_{j+1}^{(i)},\quad j = \overline{0, 3};\quad \varphi(k_{i-1}) = \left(k_{i-1} \wedge \overline{15}\right) \vee g_\varphi.$$

Интерпретация рабочего ключа выполняется следующим образом:

8) for $i = \overline{0, r-1}$:

$xk_1^{(2i)} = k_{5i};\quad xk_2^{(2i)} = k_{5i+1};\qquad xk_1^{(2i+1)} = k_{5i+2};\quad xk_2^{(2i+1)} = k_{5i+3};$

for $p = \overline{0, n-1}$:

$zw_{1,\,p}^{(2i)} = \left(kw_{5i+4,\,p} >> 0\right) \wedge \mathrm{0xFF};\qquad zw_{2,\,p}^{(2i)} = \left(kw_{5i+4,\,p} >> 16\right) \wedge \mathrm{0xFFF};$

$zw_{1,\,p}^{(2i+1)} = \left(kw_{5i+4,\,p} >> 32\right) \wedge \mathrm{0xFF};\qquad zw_{2,\,p}^{(2i+1)} = \left(kw_{5i+4,\,p} >> 48\right) \wedge \mathrm{0xFFF};$

9) $K^{\mathrm{IT}} = \left\langle k_{5r+1} \,\|\, k_{5r+0} \right\rangle,\quad K^{\mathrm{FT}} = \left\langle k_{5r+3} \,\|\, k_{5r+2} \right\rangle,$

где i – номер итерации;

p – номер слова в полублоке (индексация с 0).

Как видно из приведенных соотношений, длина рабочего ключа определяется количеством циклов шифрования r и составляет $5r + 4$ полублока. Количество циклов шифрования определяется длиной блока данных (табл. 1).

Заключение

Алгоритм «Торнадо» удовлетворяет требованиям первого (максимального) класса стойкости в соответствии с условиями проекта NESSIE (длина блока не менее 128 бит, длина ключа не менее 256 бит). Основной сферой применения БСШ первого класса стойкости является защита информации на уровне приложений, то есть на универсальных аппаратных платформах. Поэтому алгоритм «Торнадо» учитывает ближайшие перспективы развития массовой вычислительной техники и ориентирован, в первую очередь, на 64-битные аппаратные платформы. Однако структура алгоритма позволяет реализовать его достаточно эффективно как на 32-битных микропроцессорах, так и на 8-битных. Кроме того, алгоритм построен по Фейстель-подобной схеме, что обеспечивает инволютивность шифрующего

преобразования и возможность использования общего аппаратного решения для процедур зашифрования и расшифрования.

Конструкция алгоритма «Торнадо» учитывает известные методы криптоанализа БСШ и позволяет защититься от них. В основу алгоритма были положены хорошо известные принципы построения шифров, прошедшие апробацию временем, а также сравнительно новые конструкции [3, 4, 6]. Их совместное применение позволило создать алгоритм, для успешного криптонападения на который требуется разработка новых методов криптоанализа БСШ, если они существуют. Конструкция алгоритма вместо широко распространённого принципа «"слабой" цикловой функции, повторяемой многократно», использует принцип «"сильной" цикловой функции, повторяемой ограниченное число раз» [5]. Это позволяет повысить показатели стойкости алгоритма в соответствии с теоретическим критерием оценки стойкости без снижения показателей по практическому критерию.

Алгоритм «Торнадо» может использоваться в любом стандартном режиме применения БСШ, а также в качестве базового элемента при построении «усиленных» режимов поточного шифрования [7].

Список литературы: 1. *AES discussion forum*: http://aes.nist.gov 2. *NESSIE* Call for Cryptographic Primitives, Version 2.2, 8th March 2000: http://cryptonessie.org 3. *Головашич С.А.* Принцип построения инволютивных шифров // Проблемы бионики. Харьков. 2001. Вып. 54. С. 118 – 125. 4. *Головашич С.А.* Метод построения управляемых S-блоков с предельными показателями нелинейности // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 123. С. 215 – 221. 5. *«Supporting Document on E2»*, Nippon Telegraph and Telephone Corporation, June 14, 1998. 6. *Головашич С.А.* Метод конструирования цикловых функций БСШ // Автоматизированные системы управления и приборы автоматики. Всеукр. межвед. науч.-техн. сб. 2001. Вып. 117. С. 155 – 161. 7. *Головашич С.А.* Безопасность режимов блочного шифрования // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 119. С. 135 – 145.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 25.06.2003

УДК 681.3.06: 519.248.681

В. И. ДОЛГОВ, д-р. техн. наук, С. А. ГОЛОВАШИЧ, канд. техн. наук, В. И. РУЖЕНЦЕВ

КРИПТОСТОЙКОСТЬ ШИФРА «ТОРНАДО»

Одной из первоочередных задач в области информационных технологий для Украины является создание блочного симметричного шифра (БСШ), отвечающего современным требованиям. Описание нового БСШ «Торнадо» представлено в этом же журнале в статье «Алгоритм блочного симметричного шифрования «Торнадо». Спецификация». Основной целью настоящей статьи является анализ стойкости этого шифра к известным криптоаналитическим атакам.

При рассмотрении стойкости шифра к различным атакам будем пользоваться понятием «ослабленного шифра Торнадо». Под ослабленным вариантом этого шифра будем понимать алгоритм, который получается исключением из оригинального варианта преобразований IT, FT и CP_{bit}, при этом объединение полублоков с выходом F-функции выполняется с использованием сложения по модулю 2 вместо сложения / вычитания по модулю 2^{64}. В этом случае F-функция алгоритма имеет Rijndael-подобную структуру. Формат многих операций шифра «Торнадо» – 64 бита, поэтому для удобства 64-битные слова, из которых состоит полублок шифра «Торнадо», будем называть колонками по аналогии с 32-битными колонками в шифре Rijndael [1].

1 Атаки грубой силы

Большинство атак грубой силы применимы к любому блочному шифру, и сложность конкретной атаки зависит только от длины блока n или длины ключа k, независимо от структуры алгоритма.

Атака полного перебора ключей является самым простым способом поиска ключа шифрования. Сложность такой атаки зависит от длины ключа и составляет, как известно, не менее 2^{k-1} шифрований с помощью исследуемого шифра. Для обеспечения защищенности от этой атаки в шифрах используют ключи большого размера. Поскольку минимальная длина ключа шифра «Торнадо» составляет 256 бит, исчерпывающий поиск ключа требует 2^{255} шифрований и на практике неосуществим.

К словарной атаке уязвимы шифры, обладающие недостаточной длиной блока. Для выполнения атаки требуется таблица размером 2^n блоков, а для построения такой таблицы необходимо 2^n шифрований. Минимальная длина блока «Торнадо» – 128 бит, следовательно, словарная атака также на практике неосуществима.

Далее будем говорить, что алгоритм защищен от некоторой криптоаналитической атаки, если ее реализация превышает или равна сложности атаки грубой силы.

2 Дифференциальный и линейный криптоанализ

Наиболее известными и мощными методами выполнения атак на БСШ являются предложенные в начале 90-х дифференциальный криптоанализ [2, 3] и линейный криптоанализ [4]. Известны четыре критерия стойкости n-битного шифра к этим криптоатакам [5]:

– точный критерий – максимальное значение вероятностей дифференциалов и шаблонов линейной аппроксимации ниже, чем 2^{-n};

– теоретический критерий – верхние границы значений вероятностей дифференциалов и шаблонов линейной аппроксимации ниже, чем 2^{-n};

– эвристический критерий – максимальные вероятности дифференциальных и линейных характеристик ниже, чем 2^{-n};

– практический критерий – верхние границы вероятностей дифференциальных и линейных характеристик ниже, чем 2^{-n}.

Таким образом, практический критерий стойкости шифра может быть представлен в виде следующего неравенства:

$$P_{\text{upb}}^{(r-1)} \le 2^{-n}, \tag{1}$$

где r – число циклов шифра, n – размер блока в битах, $P_{\text{upb}}^{(r)}$ – верхняя граница вероятности r-цикловой дифференциальной или линейной характеристики.

Рассмотрим ослабленные варианты шифра «Торнадо».

Поскольку цикловая функция шифра биективна, то в соответствии с [5]

$$P_{\text{upb}}^{(r)} = \begin{cases} \left(P_{\max}^{(1)}\right)^{2t}, & \text{если } r = 3t \text{ или } r = 3t+1, \\ \left(P_{\max}^{(1)}\right)^{2t+1}, & \text{если } r = 3t+2, \end{cases} \tag{2}$$

где $P_{\max}^{(1)}$ – максимальная вероятность отличной от нуля дифференциальной или линейной характеристики цикловой функции. Для вычисления верхней границы этого значения необходимо определить минимально возможное число активных S-блоков в активной цикловой функции.

Из свойства используемого в шифре MDS-преобразования следует, что число ветвей активизации равно 9, следовательно, активная цикловая функция ослабленного шифра «Торнадо» содержит не менее 9 активных S-блоков. Поэтому

$$P_{\max}^{(1)} = (p_s)^9 = \left(2^{-6}\right)^9 = 2^{-54}, \tag{3}$$

где $p_s = 2^{-6}$ максимальная вероятность отличной от нуля линейной/дифференциальной характеристики отдельного S-блока шифра «Торнадо».

С помощью формул (2) и (3) были рассчитаны верхние границы вероятностей дифференциальных/линейных характеристик $P_{\text{upb}}^{(r)}$ для ослабленных вариантов шифра «Торнадо» и для различного числа итераций цепи Фейстеля r. Результаты представлены в табл. 1 (в скобках указано минимальное число активных S-блоков a, соответствующее дифференциальной/линейной характеристике).

Таблица 1

R	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P_{\text{upb}}^{(r)}$	1	2^{-54}	2^{-108}	2^{-108}	2^{-162}	2^{-216}	2^{-216}	2^{-270}	2^{-324}	2^{-324}	2^{-378}	2^{-432}	2^{-432}	2^{-486}	2^{-540}
A	(0)	(9)	(18)	(18)	(27)	(36)	(36)	(45)	(54)	(54)	(63)	(72)	(72)	(81)	(90)

Учитывая возможность применения NR-атаки [2], в ходе которой может быть отброшено до двух итераций цепи Фейстеля, неравенство (1) следует записать как:

$$P_{\text{upb}}^{(r-2)} \le 2^{-n}. \tag{4}$$

Как видно из результатов, представленных в табл. 1, для ослабленных вариантов шифра даже без IT- и FT-преобразований выполняется неравенство (4), а значит ослабленные Торнадо-128, Торнадо-256 и Торнадо-512 являются практически стойкими к дифференциальному и линейному криптоанализу. Использование полного набора преобразований в этих шифрах увеличит их стойкость.

3 Атака усеченных дифференциалов

Одна из разновидностей дифференциальных атак была предложена в середине 90-х годов Л. Кнудсеном [6]. В своей работе он показал, что для организации атаки иногда эффективнее предсказывать не полную разность, а лишь некоторую ее часть. Такая методика им была названа атакой усеченных дифференциалов. Считается, что для байт-ориентированных

шифров естественным является изучение усеченных дифференциалов особого вида, для которых усечение заключается не в исключении из рассмотрения отдельных битов входной или выходной разности, а в рассмотрении факта активности S-блоков. Поскольку используемые в современных шифрах S-блоки чаще всего задают закон отображения байт в байт, то дифференциалы, рассматривающие активность S-блоков, часто называют байтовыми дифференциалами. В таких дифференциалах вместо разности рассматривается прохождение через преобразования шифра двоичных векторов, каждый бит которых отражает активность одного S-блока (1 – ненулевая разность – S-блок активный; 0 – S-блок пассивный). Двоичный вектор, описывающий активность всех байтов блока (полублока) далее будем называть вектором активизации. В литературе существует описание атак усеченных (байтовых) дифференциалов на ослабленные варианты байт-ориентированных шифров SAFER и E2 [7, 8].

Существует мнение, что рассмотрение усеченных (байтовых) дифференциалов для байт-ориентированных шифров позволяет получить более точную оценку стойкости к дифференциальному криптоанализу, чем при стандартном подходе, т.е. при рассмотрении дифференциальных характеристик [9, 10].

Оценка стойкости БСШ к атаке усеченных дифференциалов выполняется путем поиска байтовых дифференциалов, покрывающих достаточное для построения атаки число циклов и обладающих достаточно высокой вероятностью. Известные методы выполнения такого поиска [10, 11] были предложены для 128-битных шифров и не могут быть применены в явном виде для оценки стойкости фейстель-подобных шифров с размером блока 256 и более битов. Для того, чтобы произвести поиск байтовых дифференциальных характеристик для всех вариантов шифра «Торнадо», нами был использован метод, который позволяет значительно сократить вычислительные затраты на реализацию такого поиска для фейстель-подобных шифров с rijndael-подобной шифрующей функцией по сравнению с известным методом [11]. В основе разработанного подхода лежит идея сокращения числа рассматриваемых на каждой итерации выходных векторов активизации. Как было выяснено, среди всех векторов активизации с одинаковой комбинацией активных колонок (активная колонка содержит хотя бы один активный байт) на выходе одного цикла наиболее перспективным является тот, который содержит максимальное число активных байтов в каждой из активных колонок. При исключении из рассмотрения остальных возможных выходных векторов активизации число рассматриваемых вариантов на каждом цикле не будет превосходить числа комбинаций активных колонок $N_{комб}$, которое может вычислено по следующей формуле:

$$N_{комб} = \sum_{i=1}^{n_c-1} C_{n_c}^{i},$$

где $C_{n_c}^{i}$ – число сочетаний из i по n_c; n_c – число колонок в полублоке. Значение $N_{комб}$ зависит от числа колонок и обычно не превосходит нескольких десятков, и, следовательно, становится возможным тестирование шифров с большим размером блока.

С целью проверки справедливости предлагаемого подхода и сравнения его с известным аналогом поиск эффективных байтовых характеристик для шифра Торнадо-128 был проведен с использованием известного [11] и предлагаемого методов. Применение метода из [11] позволило найти гораздо больше эффективных характеристик. В то же время лучшие по вероятностным показателям, а значит и наиболее эффективные байтовые характеристики были найдены в обоих случаях, при этом предлагаемый метод требует значительно меньших вычислительных затрат. Все это свидетельствует о работоспособности предлагаемого подхода для фейстель-подобных шифров с «rijndael-подобной» шифрующей функцией.

Предлагаемый метод позволил также протестировать ослабленные варианты шифра «Торнадо» с размером блока 256 и 512 битов. Полученные результаты сведены в табл. 2.

Таблица 2

	Размер блока, байты (биты)	Макс. число полуциклов	Вер. усеч. дифф. хар., $P_{усд}$	Вер. обычн. диффер., $P_{диф}$
Торнадо-128	16 (128)	2	1	2^{-72}
Торнадо-256	32 (256)	5	$2^{-128.1}$	$2^{-240.1}$
Торнадо-512	64 (512)	8	$2^{-335.9}$	$2^{-495.9}$

В таблице приведены также вероятности обычных дифференциалов, вычисленные в соответствии с представленными в работе [10] соотношениями между вероятностями обычных $P_{диф}$ и усеченных дифференциалов $P_{усд}$:

$$P^{(i)}_{диф}(a,b) = p^{h(\chi(b))} P^{(i)}_{усд}(\chi(a), \chi(\beta)),$$

где $h(f)$ – вес Хемминга аргумента f; χ – функция-характеристика, которая преобразует разность в вектор активизации, т.е. ставит 1 на месте активных S-блоков в аргументе и 0 – в противном случае; p – вероятность получения на выходе активного S-блока определенного значения разности (если считать все ненулевые значения разности равновероятными, то $p = \frac{1}{255}$); a и b есть, соответственно, входная и выходная разности.

Исходя из полученных результатов и учитывая возможность использования NR-атаки [2], в ходе которой может быть отброшено до двух итераций цепи Фейстеля, сделано заключение, что ослабленные варианты шифров Торнадо-128 с 5 и более полуциклами, Торнадо-256 с 8 и более полуциклами и Торнадо-512 с 11 и более полуциклами являются стойкими к дифференциальной атаке и атаке усеченных дифференциалов.

Использование полного набора операций позволит достичь более хороших показателей стойкости шифра к рассматриваемым видам атак.

4 Атака невозможных дифференциалов

Впервые техника выполнения данного вида дифференциальных атак была предложена для ослабленных вариантов шифров Skipjack [12], IDEA [13]. В этой атаке используются дифференциалы, которые не могут выполниться, т. е. имеющие нулевую вероятность.

Методика атаки заключается в том, что на некотором цикле (обычно первом или последнем) делается предположение о примененном подключе или о его части (выполняется перебор цикловых ключей), а на остальных циклах предполагается выполнение «невозможного» дифференциала. Если на проверяемом цикловом ключе-кандидате возможно получение разностей, определенных «невозможным» дифференциалом, то этот ключ отбрасывается как ошибочный. В результате такого отбора для анализа остается ограниченное множество цикловых ключей-кандидатов.

В работе [13] также показано, что если в фейстель-подобном шифре используется биективная шифрующая функция, то всегда существует 5-цикловый невозможный дифференциал, который имеет вид $(a, 0) \rightarrow (a, 0)$ для любой ненулевой разности a. В силу того, что шифр «Торнадо» построен с использованием цепи Фейстеля, а шифрующая функция биективна, то для этого шифра существует 2,5-цикловый (5 итераций цепи Фейстеля) невозможный дифференциал. Невозможных дифференциалов, покрывающих большее число циклов, найдено не было. В подтверждение их отсутствия может служить и тот факт, что не было найдено дифференциалов, проводящих разность с вероятностью 1 через более, чем 2 итерации шифра (1 цикл). В то же время, в работе [13] говорится о том, что невозможный дифференциал обычно получается путем «стыковки» двух достоверных дифференциалов.

Приведенные соображения позволяют сделать вывод о возможности организации с помощью 2,5-циклового невозможного дифференциала атаки на шифр без начальных и конечных преобразований с 6-ю итерациями цепи Фейстеля. Варианты шифра «Торнадо» с начальными и конечными преобразованиями и со стандартным числом циклов будут неуязвимы для атаки невозможных дифференциалов.

5 Бумеранг-атака

Впервые бумеранг-атака была представлена в работе [14]. Эта атака во многом схожа с дифференциальным криптоанализом, так как использует дифференциалы. В отличие от дифференциального криптоанализа, где используется один дифференциал, покрывающий почти все циклы шифра, для организации бумеранг-атаки требуется два более «коротких» дифференциала, которые вместе покрывают все циклы шифра и обладают высокими вероятностями p_1 и p_2. Условие эффективности бумеранг-атаки, как следует из [14, 15], может быть записано в виде

$$p_1 \times p_2 \geq 2^{-n/2}, \tag{5}$$

где n – длина блока в битах.

Легко убедиться, что среди найденных в пунктах 1 и 2 дифференциалов и дифференциальных характеристик (табл. 1, табл. 2) нет таких, которые бы покрывали все циклы шифра и удовлетворяли неравенству (5). Поэтому есть основания считать все варианты шифра «Торнадо» стойкими и к этой атаке.

6 Интерполяционная атака и атака линейных сумм

Эта атака предложена Якобсеном и Кнудсеном [16]. Для ее проведения криптоаналитик выполняет построение полиномов на основе известных пар «открытый–шифрованный текст». Интерполяционная атака эффективна против шифров, позволяющих описать процедуру шифрования некоторым сравнительно простым алгебраическим выражением. Принцип интерполяционной атаки состоит в том, что если шифртекст может быть представлен как полином (или рациональное выражение) от открытого текста, в котором количество неизвестных (ключезависимых) коэффициентов равно N, то этот полином (или рациональное выражение) может быть восстановлен с использованием N пар «открытый–шифрованный текст», полученных на искомом ключе K. Для восстановления неизвестных коэффициентов обычно используется какая-либо интерполяционная формула. Выражение, определяемое полученным полиномом, будет эквивалентно зашифрованию на искомом ключе. Для защиты от этой атаки необходимо максимизировать значение N.

В работе [17] предложен алгоритм, позволяющий оценить стойкость шифра к атаке линейных сумм и интерполяционной атаке. На практике этот алгоритм реализуем для случая, когда строящийся полином $f_k(x)$ связывает значения одного байта открытого текста с одним байтом шифртекста. Полином $f_k(x)$ имеет следующий вид:

$$f_k(x) = \sum_{i=1}^{2^8} a_i(k) b_i(x),$$

где $x \in GF(2^8)$ – байт открытого текста, $a_i(k) \in GF(2^8)$ – ключезависимые коэффициенты, $\{b_i(x)\}$ – множество линейно независимых полиномов с коэффициентами из $GF(2^8)$ (атака линейных сумм эквивалентна интерполяционной атаке, когда $b_i(x) = x^{i-1}$).

Атака линейных сумм эффективна, когда число неизвестных ключезависимых коэффициентов $a_i(k)$ меньше, чем 2^8. В соответствии с предложенным в [17] алгоритмом был произведен поиск соотношений над $GF(2^8)$ между каждым байтом открытого текста и каждым байтом криптограммы. Результаты тестирования шифра «Торнадо» представлены в табл. 3.

Таблица 3

Число циклов	«Торнадо»	«Торнадо» без IT- и FT-преобразований
0	1	–
1	256	1
1,5	256	255
≥2	256	256

Представленные в табл.3 данные свидетельствуют о стойкости шифра «Торнадо» без IT- и FT-преобразований с 5 и более итерациями цепи Фейстеля к интерполяционной атаке и атаке линейных сумм. Из таблицы также видно, что использование IT- и FT-преобразований еще более увеличивает стойкость шифра к этим видам атак.

В соответствии с Теоремой 3 [17] шифр, стойкий к атаке линейных сумм, является также стойким к атаке высших дифференциалов и интегральной атаке. Поэтому есть основания считать «Торнадо» стойким и к данным видам атак.

7 Атака дифференциалов высших порядков

Применение данной атаки к БСШ было продемонстрировано Кнудсеном [6, 16]. Атака применима к шифрам, представимым в виде булевого полинома малой степени. Необходимый для проведения криптоанализа порядок дифференциала определяют по степени булевых полиномов, определяющих значения битов криптограммы как функции от битов открытого текста. Так, если биты выхода $(r-1)$-го цикла могут быть выражены через биты открытого текста в виде полиномов степени не выше d, то порядок дифференциала высшего порядка будет $d+1$, так как любой дифференциал $(d+1)$-го порядка становится равным 0 [6]. Если обозначить через b длину используемого на последнем цикле ключа, то в соответствии с Теоремой 1 [16] существует дифференциальная атака порядка $d+1$ со сложностью по времени 2^{b+d}, требующая 2^{d+1} подобранных открытых текстов и соответствующих им криптограмм, которая позволит получить ключ последнего цикла.

Для того чтобы оценить стойкость шифра к атаке высших дифферениалов, следует найти порядок полиномов, связывающих биты на выходе предпоследнего цикла с битами открытого текста. В шифре «Торнадо» единственные нелинейные элементы – это S-подстановки. Порядок полиномов, связывающих значения битов на выходе S-подстановки со значениями битов на входе, максимален и равен 7. Каждая следующая S-подстановка существенно увеличивает порядок булевых полиномов. Например, после четырех уровней S-подстановок порядок булевых полиномов будет $7^4 = 2401$. Поскольку 2401 больше, чем 128, 256 и 512, то ожидается, что все варианты шифра будут стойкими к атаке высших дифференциалов. Подтверждением этому могут служить и результаты, полученные в пункте 6.

8 Интегральный криптоанализ

Данная атака первоначально была предложена под названием square для одноименного шифра [18], который имеет байт-ориентированную структуру. Этой атаке подвержены и другие шифры, имеющие «слово»-ориентированную структуру, т.е. состоящие из последовательности линейных и нелинейных преобразований над выровненными битовыми кортежами фиксированной длины [1, 19]. В интегральной атаке используется множество открытых текстов, подобранных так, чтобы в результате суммирования (интегрирования) принадлежащих этому множеству текстов в битовых кортежах фиксированной длины получались определенные значения. Подобно тому как в дифференциальном криптоанализе производится «транспортирование» разности через преобразования шифра, в данном случае через циклы шифра проводится значение суммы. По аналогии с дифференциальной характеристикой в дифференциальной атаке в интегральном криптоанализе путь, по которому проходит значение суммы, называют интегралом. Если возможно с высокой вероятностью предсказать значение суммы в одном или нескольких битовых кортежах после r циклов шифра, то может быть организована атака на $(r+1)$-цикловый шифр. В ходе атаки перебираются возможные подключи последнего цикла и для каждого варианта производится дешифрование одного цикла для всего множества имеющихся криптограмм. Если в результате суммирования информационных блоков, полученных при одноцикловом дешифровании, в определенном битовом кортеже будет получено нужное значение, то с высокой вероятностью проверяемый подключ последнего цикла верный. Более подробно методика интегрального криптоанализа описана в работах [1, 19].

Преобразования шифра «Торнадо», за исключением битовой перестановки CP_{bit}, являются байт-ориентированными и во многом похожи на преобразования, используемые в шифрах Square, Rijndael, поэтому важно исследовать возможность организации интегральной атаки на «Торнадо». Особенности прохождения сумм через большинство преобразований рассматриваемого шифра известны из [1, 18, 19]. Рассмотрим, как различные варианты сумм будут проходить через битовую перестановку. Для этого будем использовать введенные в [19] обозначения. Символ '*C*' на определенной позиции говорит о том, что значения слов на этой позиции во всех рассматриваемых текстах равны. Символ '*A*' означает, что все слова на этой позиции в коллекции рассматриваемых текстов различны. '*S*' обозначает то, что сумма всех слов на этой позиции в коллекции рассматриваемых текстов имеет определенное значение. '?' – значение суммы не известно.

Операция CP_{bit} преобразует 64-битные блоки так, что каждый байт после преобразования содержит по одному биту из каждого байта до преобразования. Поэтому если побитовая сумма по модулю 2 во всех 64 битах равна 0, то она останется такой же и после преобразования. Естественно, если в каждом из восьми байтов в множестве рассматриваемых текстов находятся одинаковые значения, т. е. каждый из 8-ми байтов обозначается символом '*C*', то одинаковыми они останутся и после перестановки. Однако, если перед перестановкой наряду с '*C*'-байтами имеется один или несколько байтов типа '*A*' или '*S*', то на выходе будут получены значения '*S*' на всех восьми позициях. С учетом особенностей используемых в шифре «Торнадо» преобразований могут быть построены интегралы, покрывающие 3-итерации цепи Фейстеля для вариантов шифра с различным размером блока (рис. 1).

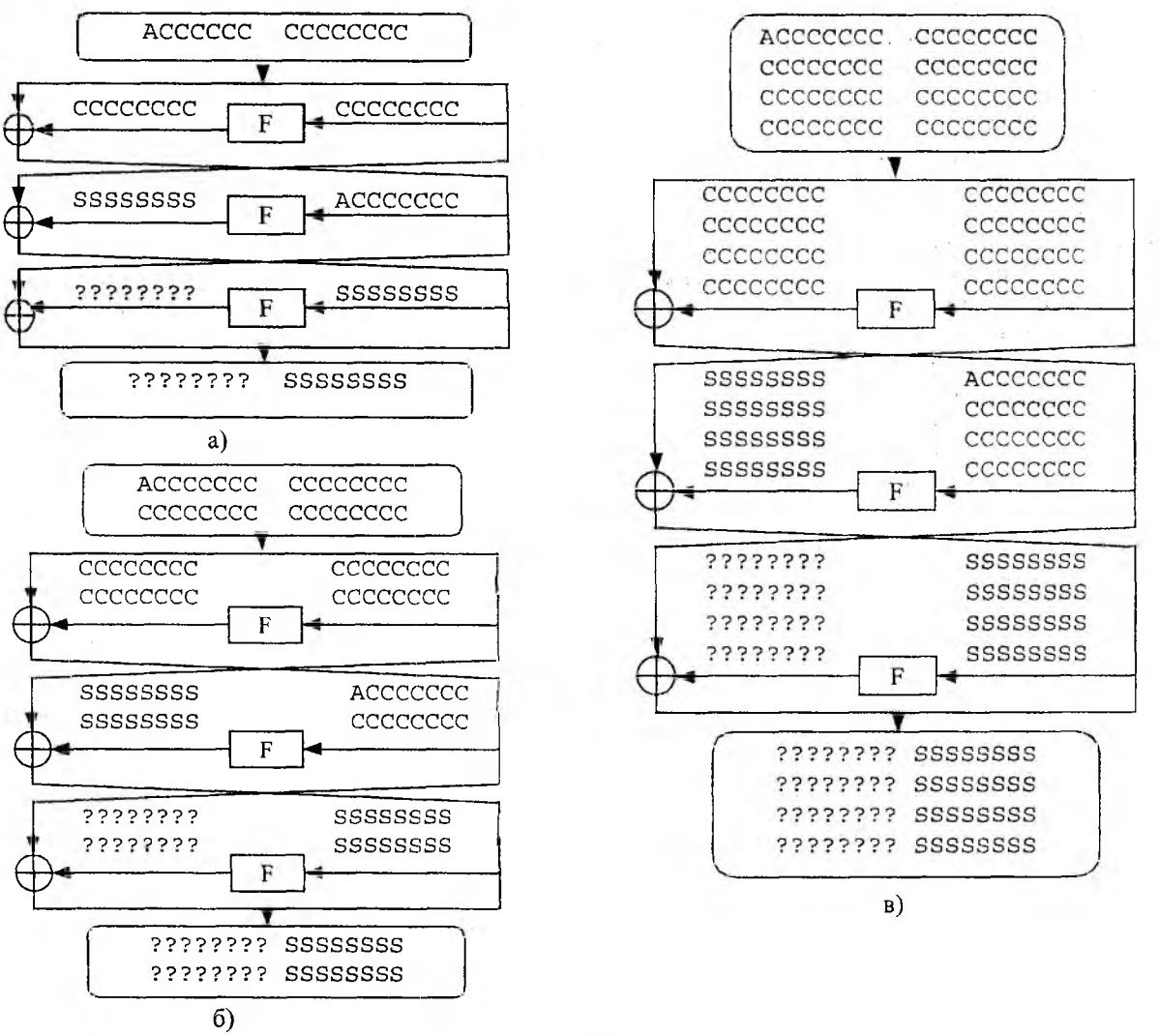


Рис. 1

С помощью представленных интегралов, вероятно, может быть организована эффективная атака на ослабленные варианты шифра «Торнадо», содержащие 4 полуцикла (2 цикла). Шифры Торнадо-128, Торнадо-256 и Торнадо-512 со стандартным числом циклов являются стойкими к интегральному криптоанализу, что подтверждается и результатами, полученными в разделе 6.

Представленные результаты позволяют сделать общий вывод о стойкости шифра «Торнадо» ко всем рассмотренным видам криптоаналитических атак. Кроме того, можно говорить о наличии определенного «запаса» стойкости, который позволит шифру оставаться безопасным на протяжении ближайшего времени и противостоять новым, ещё не известным, методам криптоанализа.

Список литературы: 1. *J. Daemen, V. Rijmen.* AES Proposal Rijndael, AES Round 1 Technical Evaluation CD-1: Documentation, National Institute of Standards and Technology, Aug 1998. See http://www.nist.gov/aes. 2. *E. Biham, A. Shamir.* Differential Cryptanalysis of DES-like Cryptosystem, Journal of Cryptology. Vol. 4. P. 3 – 72. 1991. 3. *E. Biham, A. Shamir.* Differential Cryptanalysis of the full 16-round DES. Technical Report – Computer Science Department, Technion, Israiel, 1993. 4. *M. Matsui.* Linear Cryptanalysis Method for DES Cipher, EUROCRYPT'93, pp. W112-W123, May 1993. 5. «Supporting Document on E2», Nippon Telegraph and Telephone Corporation, June 14, 1998. 6. *L. R. Knudsen.* Truncated and Higher Order Differentials. In B. Preneel, editor, Fast Software Encryption – Second International Workshop, Volume 1008 of Lecture Notes in Computer Science, pp. 196 – 211. Springer-Verlag, Berlin, Heidelberg, New York, 1995. 7. *L.R. Knudsen, T.A. Berson.* Truncated differentials of SAFER, http://www.ii.uib.no/~larsr/. 8. *M. Matsui, T. Tokita.* Cryptanalysis of reduced version of the block cipher E2, in pre-proceedings of Fast Software Encryption'99, pp. 70 – 79, 1999. 9. *K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita.* Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms, Selected Areas in Cryptography – 7th Annual International Workshop, SAC2000, Lecture Notes in Computer Scienca 2012, pp. 39 – 56, Springer-Verlag, Berlin, 2001. 10. *M. Sugita, K. Kobara.* Relationships among differential, truncated differential, impossible differential cryptanalyses against word-oriented block cipher like Rijndael, E2 // National Institute of Standards and Technology, http://www.nist.gov/aes. 11. *S. Moriai.* Security of E2 against truncated differential cryptanalysis, http://www.nist.gov/aes. 12. *E. Biham, A. Biryukov, A. Shamir.* Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials, proceedings of EUROCRYPT'99, lecture notes in computer science 1592, pp. 12 – 23, 1999. 13. *E. Biham, A. Biryukov, A. Shamir.* Miss in the Middle Attacks on Idea and Khufu, proceedings of FSE'99, lecture notes in computer science 1636, pp. 124-138, 1999. 14. *D. Wagner.* The Boomerang Attack. In L. R. Knudsen, editor, Fast Software Encryption – 6th International Workshop, FSE'99, Volume 1636 of Lecture Notes in Computer Science, pp. 156 – 170, Berlin, Heidelberg, New York, 1999. 15. *E. Biham, O. Dunkelman, N. Keller.* New Results on Boomerang and Rectangle Attacks, available from http://eprint.iacr.org/2001/070.ps. 16. *T. Jakobsen, L.R. Knudsen.* The Interpolation Attack on Block Cipher. In E. Biham, editor, Fast Software Encryption — 4th International Workshop, FSE'97, Volume 1267 of Lecture Notes in Computer Science, pp. 28 – 40, Berlin, Heidelberg, New York, 1997. Springer-Verlag. 17. *K. Aoki.* Practical Evaluation of Security against Generalized Interpolation Attack. IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences (Japan), Vol. E83-A, No. 1, pp. 33 – 38, 2000. (A preliminary version was presented at SAC'99). 18. *J. Daemen, L.R. Knudsen, V. Rijmen.* The block cipher Square, Fast Software Encryption, LNCS 1267, E. Biham, Ed., Springer-Verlag, 1997, pp. 149 – 165. 19. *L. R. Knudsen.* Integral Cryptanalysis, NESSIE internal report NES/DOC/UIB/WP5/015/1, 2001.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 04.06.2003

УДК 681.3.06

С. А. ГОЛОВАШИЧ, канд. техн. наук, А. Н. ЛЕПЕХА

СТАТИСТИЧЕСКИЙ АНАЛИЗ БСШ «ТОРНАДО»

Рассматривается блочный симметричный шифр (БСШ) «Торнадо». Алгоритм «Торнадо» поддерживает три различных длины блока (128, 256 и 512 бит) и оперирует с ключами длиной не менее 256 бит. В соответствии с условиями проекта NESSIE алгоритм относится к первому классу стойкости. Основным требованием, предъявляемым к БСШ, является устойчивость к известным криптоаналитическим атакам.

Косвенным показателем стойкости криптоалгоритма является его статистическая безопасность. Статистическое исследование шифров позволяет выявить «скрытые» (либо неизвестные) «аномалии» в работе шифратора, которые не удаётся обнаружить аналитическим путём.

Целью данной статьи является проведение исследования статистической безопасности БСШ «Торнадо» и сравнение его статистических показателей с аналогичными показателями криптоалгоритмов, победивших в конкурсах AES и NESSIE. Исследования проводились для версии алгоритма с длиной блока 128 бит.

Методы статистического анализа можно разделить на два класса: методы, основанные на анализе статистических свойств выхода шифратора (криптограмм), и методы, основанные на исследовании корреляционных зависимостей «вход-выход» шифратора.

Для исследования корреляционных свойств алгоритма (в соответствии со вторым подходом) использовалось три теста, позволяющие выявить «простые» статистические зависимости между входом и выходом криптопреобразования: лавинный эффект (AE), строгий лавинный критерий (SAC), тест частичных (однобитных) дифференциалов (TD).

С помощью перечисленных тестов исследована также процедура разворачивания ключа. В рамках этого исследования тесты лавинного эффекта (AE) и строгого лавинного критерия (SAC) использовались для анализа корреляционных зависимостей между битами исходного (пользовательского) ключа и битами развернутого (рабочего) ключа.

1 Конструкции генераторов ПСП на базе БСШ «Торнадо»

Для исследования статистических свойств алгоритма «Торнадо», в соответствии с первым подходом, использовалась методика NIST STS, разработанная Национальным институтом стандартов США (NIST) и использовавшаяся в рамках проекта AES. Этот метод рассматривает любой алгоритм как «чёрный ящик», обладающий n-битными входом и выходом, а также k-битным управляющим входом (вход ключа). Алгоритм считается «статистически безопасным», если ПСП, формируемые генератором на его основе (на случайно выбранном ключе), «выглядят» как случайная битовая последовательность. Исследование проводилось для последовательностей, сформированных в двух режимах применения БСШ: режим «с обратной связью по выходу» (ISO/IEC 10116) и режим «счетчика с плавающим периодом» [3], соответствующих схемам синхронного поточного шифрования (формирования гаммы шифрующей). Для обоих схем исследовались свойства только гаммы шифрующей, то есть текст соответствовал нулевому заполнению. Тестирование проведено для полноциклового и ряда упрощенных версий шифра (с уменьшенным числом циклов и с/без начального и конечного преобразований). Результаты сравнивались со свойствами ПСП, сформированной генератором BBS (эталонная выборка, рекомендованная NIST).

2 Тестирование по методике NIST STS

Пакет NIST STS включает в себя 16 различных статистических тестов, направленных на выявление различных «дефектов» случайности. При этом некоторые из тестов рассчитываются для нескольких различных «тестовых шаблонов» [2], проверка каждого из которых фактически является отдельным тестом. В связи с этим общее количество тестов составляет 189.

Рассмотрим методику тестирования генераторов в соответствии с NIST STS.

Методика тестирования [4].

1. С помощью тестируемого генератора формируется m двоичных последовательностей длиной по n бит: $S_i \in \{0, 1\}^n$, $i = \overline{1, m}$, т.е. $N = m \times n$ бит.

2. Для каждого теста j по результатам тестирования последовательностей $\{S_i\}$ строится вектор вероятностей $P_{i,j} \in [0, 1]$, $i = \overline{1, m}$.

4. Задаётся необходимый уровень значимости α и для каждого теста j, по соответствующему вектору $(P_{1,j}, \ldots, P_{m,j})$ определяют долю последовательностей, прошедших данный тест: $r_j = \#\{P_{i,j} \geq \alpha \mid i = \overline{1, m}\} / m$.

Считается, что генератор прошёл тестирование по j-му тесту, если значение коэффициента r_j находится в пределах доверительного интервала $[r_{min}, r_{max}]$, заданного как $\hat{p} \pm 3\sqrt{\hat{p}(1-\hat{p})/m}$, $\hat{p} = 1 - \alpha$.

5. Для каждого теста j осуществляется проверка вектора вероятностей $(P_{1,j}, \ldots, P_{m,j})$ на подчинение равномерному закону распределения в интервале [0, 1]. Для проверки этой гипотезы используется критерий χ^2 с 9 степенями свободы (интервал [0, 1] разбивается на 10 подинтервалов). По значению χ_j^2 определяется соответствующая вероятность $P_j = P(\chi_j^2, 9)$.

Значения j-го вектора вероятностей могут считаться равномерно распределёнными, если выполняется условие $P_j \geq 0{,}0001$.

6. Принимается окончательное решение о случайности последовательностей, формируемых генератором.

Последовательности, формируемые генератором, считаются «случайными», если для всех 189 тестов соблюдается условие $P_j > 0{,}0001$, а значения коэффициентов r_j находятся внутри доверительного интервала $[r_{min}, r_{max}]$.

При проведении тестирования параметры были выбраны в соответствии с рекомендациями NIST STS, то есть длина одной последовательности $n = 10^6$, количество последовательностей $m = 100$, уровень значимости $\alpha = 0{,}01$.

3 Результаты тестирования по методике NIST STS

В табл. 1 приведены результаты тестирования БСШ «Торнадо» в режиме «обратной связи по выходу шифратора». В табл. 2 сведены результаты тестирования в режиме «счетчика с плавающим периодом».

Таблица 1

Количество циклов	1	1+ IT&FT	2	2+ IT&FT	3	4	4+ IT&FT	Генератор BBS
Количество тестов, в которых тестирование прошло 99% последовательностей	138	124	135	141	138	132	139	134

Продолжение табл. 1

Количество тестов, в которых тестирование прошло более 96% последовательностей	189	188	188	188	188	188	189	189
Количество тестов, в которых значение вероятности P<=0,01	2	3	1	0	0	1	0	0
Количество тестов, в которых значение вероятности P<=0,001	0	1	0	0	0	0	0	0
Количество тестов, в которых значение вероятности P<=0,05	16	11	13	9	2	6	10	—
Непройденный тест	—	Random-Excursion-V	Random-Excursion-V	Aperiodic-Template	Aperiodic-Template	Random-Excursion	—	—

Таблица 2

Количество циклов	1	1+ IT&FT	2	2+ IT&FT	3	4	4+ IT&FT	Генератор BBS
Количество тестов, в которых тестирование прошло 99% последовательностей	123	130	141	133	139	137	133	134
Количество тестов, в которых тестирование прошло более 96% последовательностей	188	188	189	187	189	189	189	189
Количество тестов, в которых значение вероятности P<=0,01	2	1	0	2	1	4	1	0
Количество тестов, в которых значение вероятности P<=0,001	0	0	0	0	0	0	1	0
Количество тестов, в которых значение вероятности P<=0,05	6	6	7	11	13	12	11	—
Непройденный тест	Aperiodic-Template	Aperiodic-Template	Aperiodic-Template	Aperiodic-Template	—	—	—	—

По результатам проведенных испытаний можно сделать следующие частные выводы:

1) значение вероятности P_j по всем тестам для всех протестированных выборок удовлетворяет ограничению $P_j > 0{,}0001$;

2) ряд выборок не прошли ограничение $r_j \geq 0{,}96015$ по некоторому тесту, при этом максимальное отклонение от доверительного диапазона зафиксировано для режима: «с обратной связью по выходу шифратора» – $r_j = 0{,}9508$, «счетчика с плавающим периодом» – $r_j = 0{,}95$;

3) выборки, не прошедшие тестирование, были «забракованы» не более, чем 1 тестом;

4) большинство выборок прошли все тесты, при этом 99% тестов показали «прохождение» не менее чем 97% одиночных последовательностей;

5) полноцикловая версия алгоритма в данном режиме показала результаты, сопоставимые с эталонной выборкой, сформированной ГПСП BBS. Количество тестов, в которых тестирование прошло 99% последовательностей, сопоставимо или выше, чем для BBS генератора.

4 Исследование корреляционных свойств БСШ «Торнадо»

Для исследования корреляционных свойств алгоритма (в соответствии со вторым подходом) использовалось три теста, позволяющие выявить «простые» статистические зависимости между битами входного и выходного блоков криптопреобразования: лавинный эффект (AE), строгий лавинный критерий (SAC), тест частичных (однобитных) дифференциалов (TD).

Параметры тестирования:

$n = 128$ – разрядность блока данных B_t;

$m = 1000$ – количество блоков B_t, тестируемых на каждом ключе K_r;

$k = 100\text{-}10000$ – количество ключей K_r, задействованных в тестировании.

Принятые обозначения:

$w(x)$ – статистика теста;

$W_H(B)$ – вес Хемминга вектора B;

$E_K(B)$ – зашифрование / расшифрование блока данных B на ключе K;

Δ_i – двоичный вектор, в котором установлен только i-й разряд:

$$\Delta_i = (\delta_{n-1}, \ldots, \delta_0)_i : \delta_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}, 0 \le i, j < n, \Delta_i \in \{0, 1\}^n;$$

$[B]_i$ – операция «выделения» из двоичного вектора B бита с номером i:

$$[B]_i = [(b_{n-1}, \ldots, b_0)]_i = b_i, b_i \in \{0, 1\}, 0 \le i < n.$$

Предметом исследования первого теста (лавинный эффект – AE) является количество изменившихся выходных разрядов при изменении одного входного разряда y. Исследование выполняется для каждого входного разряда y:

$$w_y(x) = W_H\left(E_{K_r}(B_t) \oplus E_{K_r}(B_t \oplus \Delta_y)\right), B_t \in \{0,1\}^n,$$
$$0 \le y < n, x = r \times m + t, 0 \le r < k, 0 \le t < m.$$

Предметом исследования второго теста (строгий лавинный критерий – SAC) является вероятность изменения некоторого выходного разряда j при изменении одного входного разряда i. Исследование выполняется для каждой пары разрядов «вход–выход» $y = (i, j)$ (по всем «выходным шаблонам» v):

$$w_{y,v}(x) = \frac{1}{k \times m} \sum_{r=0}^{k-1} \sum_{t=0}^{m-1} \phi(v), \phi(v) = \begin{cases} 1, v = ov \\ 0, v \neq ov \end{cases},$$
$$ov = \left[E_{K_r}(B_t) \oplus E_{K_r}(B_t \oplus \Delta_i)\right]_j, B_t \in \{0, 1\}^n,$$
$$y = i \times n + j, 0 \le i, j < n, 0 \le v < 2, x = r \times m + t.$$

Предметом исследования третьего теста (тест частичных (однобитных) дифференциалов – TD) является вероятность определённого «изменения» ov некоторого выходного разряда j

при условии определённого «изменения» iv некоторого входного разряда i (для произвольной пары входных блоков). Исследование выполняется для каждой пары разрядов «вход–выход» $y = (i, j)$ (по всем «шаблонам вход–выход» v):

$$w_{y,v}(x) = \frac{1}{k \times m} \sum_{r=0}^{k-1} \sum_{t=0}^{m-1} \psi(v),\ \psi(v) = \begin{cases} 1, v = iv \times 2 + ov \\ 0, v \neq iv \times 2 + ov \end{cases},$$

$$iv = \left[B_t \oplus B'_t\right]_i,\ ov = \left[E_{K_r}(B_t) \oplus E_{K_r}(B'_t)\right]_j,\ B_t, B'_t \in \{0,1\}^n,$$

$$y = i \times n + j,\ 0 \le i, j < n,\ 0 \le v < 4,\ x = r \times m + t.$$

Для всех перечисленных выше тестов по набранной статистике $w(x)$ были рассчитаны математическое ожидание (M) и дисперсия (D), а также для лавинного критерия минимальное (min) и максимальное (max) зафиксированные значения. Математическое ожидание отражает количество изменившихся бит в блоке (длина блока 128 бита), а дисперсия показывает разброс значений.

Тестирование выполнялось для 1–4-циклового вариантов БСШ «Торнадо». Соответствующие результаты тестирования приведены в табл. 3.

Таблица 3

Тест	1 цикл с IT и FT блоками		2 цикл с IT и FT блоками		3 цикл с IT и FT блоками		4 цикл с IT и FT блоками	
	M	D	M	D	M	D	M	D
AE:	63,796210	33,290935	64,000622	32,005364	63,999146	32,012535	64,000443	32,015113
SAC:	0,500000	0,000006	0,500000	0,000003	0,500000	0,000002	0,500000	0,000003
TD:	0,250000	0,000003	0,250000	0,000004	0,250000	0,000004	0,250000	0,000003

Минимальное зафиксированное значение – 0,492110, а максимальное для строгого лавинного критерия – 0,506650. Из полученных результатов видно, что уже после одного цикла с начальным и конечным преобразованиями (IT и FT) статистика шифра становится удовлетворительной. Для детализации в табл. 4 приводятся результаты тестирования ослабленной версии криптоалгоритма «Торнадо» (без IT и FT преобразований) для 1-4 итераций.

Таблица 4

Тест	1 итерация без IT и FT блоков		2 итерация без IT и FT блоков		3 итерация без IT и FT блоков		4 итерация без IT и FT блоков	
	M	D	M	D	M	D	M	D
AE:	49,023829	250,41669	63,998885	32,001051	63,997372	31,990770	64,000275	31,997621
SAC:	0,500000	0,059994	0,500000	0,000003	0,500000	0,000003	0,500000	0,000003

Как видно из результатов, представленных в табл.4, две итерации шифрования без IT и FT преобразований по приведенным тестам уже обеспечивают статистическую безопасность.

Эквивалентно рассмотренные выше показатели статистической безопасности могут быть выражены через «степени» (degree) «полноты» (d_c), «лавинного» эффекта (d_a), строгого лавинного критерия (d_{sa}) [1]. Алгоритм БСШ может считаться статистически безопасным по этим показателям, если они принимают следующие значения: $d_c = 1$; $d_a \approx 1$; $d_{sa} \approx 1$.

$$d_c = 1 - \frac{\#\{(i,j) \mid a_{ij} = 0\}}{nm},$$

где: i и j – размерности матрицы зависимости (n x m);

a_{ij} – элементы матрицы зависимости, причем, $a_{ij} = \#\left\{x \in \{0,1\}^n \mid \left(f(x^{(i)})\right)_j \neq \left(f(x)\right)_j\right\}$ для $i = \overline{1,n}$, $j = \overline{0,m}$.

$$d_a = 1 - \frac{\sum_{i=1}^{n}\left|\frac{1}{\#X}\sum_{i=1}^{m} 2jb_{ij} - m\right|}{nm},$$

где: $b_{ij} = \#\left\{x \in \{0,1\}^n \mid W_H\left(F(x^{(i)}) - f(x)\right) = j\right\}$ для $i = \overline{1,n}$, $j = \overline{0,m}$.

$$d_{sa} = 1 - \frac{\sum_{i=1}^{n}\sum_{i=1}^{m}\left|\frac{2a_{ij}}{\#X} - 1\right|}{nm}.$$

В табл. 5 приведены результаты, полученные для различных вариантов алгоритма «Торнадо».

Таблица 5

Тип теста	1 итерация без IT и FT	1 итерация + IT и FT	2 итерации без IT и FT	2 итерации + IT и FT	3 итерации + IT и FT	4 цикла + IT и FT
d_c	0,812012	1,000000	1,000000	1,000000	1,000000	1,000000
d_a	0,765328	0,893086	0,999783	0,996816	0,999789	0,999784
d_{sa}	0,755583	0,892801	0,997468	0,996107	0,997478	0,997470

Для сравнения в табл. 6 приведены результаты аналогичного исследования победителей проектов AES и NESSIE. Представлены циклы и соответствующие значения, после которых данные начинают удовлетворять указанным критериям.

Таблица 6

Наименование шифра	Циклы	Число бит, которые изменились	Полнота, d_c	Критерий распространения, d_a	Обнаруженные линейные факторы, d_{sa}
Camellia	4 + 0,5 5 + 0,5	63,555463 64,004316	1,000000 1,000000	0,999271 0,999254	0,991995 0,991951
SAFER++	2 + 0,5	63,996523	1,000000	0,999224	0,991944
RC6	3+0,5 4+0,5 5+0,5	60,301572 63,923282 63,994611	1,000000 1,000000 1,000000	0,942141 0,998479 0,999273	0,937845 0,991661 0,992041
Rijndael	2 3	64,247014 64,001791	1,000000 1,000000	0,996140 0,999350	0,991466 0,992043

Как видно из табл. 5, БСШ «Торнадо» удовлетворяет рассмотренным критериям уже после одного цикла шифрования. Сравнивая эти данные с результатами, представленными в табл. 6, можно отметить, что степень полноты, лавинный эффект и строгий лавинный критерий для БСШ «Торнадо» достигаются так же быстро, как и для БСШ «RIJNDAEL». Степень полноты и лавинного эффекта наступают для БСШ «Торнадо» быстрее, чем для БСШ Camellia и RC6.

6 Статистическое исследование процедуры разворачивания ключа БСШ «Торнадо»

Так же как и процедура шифрования, процедура разворачивания ключа должна удовлетворять требованиям статистической безопасности. Для проверки статистической безопасности процедуры разворачивания ключей мы применили описанную методику тестирования NIST STS, а также воспользовались тестами корреляционного анализа. Тесты корреляционного анализа были расширены, чтобы иметь возможность проанализировать влияние входных полублоков пользовательского ключа (64 бита) на выходные полублоки развернутого «сырого»ключа.

На рис. 1 приводится диаграмма прохождения тестов для ПСП, сгенерированной процедурой разворачивания ключей БСШ «Торнадо». В табл. 7 приведены сводные результаты тестирования.

Рис. 1

Таблица 7

Количество циклов	Процедура разворачивания ключей БСШ «Торнадо»	Генератор BBS
Количество тестов, в которых тестирование прошло 99% последовательностей	142	134
Количество тестов, в которых тестирование прошло более 96% последовательностей	189	189
Количество тестов, в которых значение вероятности P<=0,01	4	0
Количество тестов, в которых значение вероятности P<=0,001	3	0
Количество тестов, в которых значение вероятности P<=0,05	12	—
Непройденный тест	—	—

По результатам проведенных испытаний можно сделать следующие частные выводы:

1. Значение вероятности P_j по всем тестам для всех протестированных выборок удовлетворяет ограничению $P_j > 0{,}0001$;

2. Выборок, не прошедших ограничение $r_j \geq 0{,}96015$ по некоторому тесту, зафиксировано не было;

3. Выборок, не прошедших какой-либо тест в ходе тестирования, зафиксировано не было;

4. Полученные результаты сопоставимы с результатами для эталонной выборки, рекомендованной NIST STS. Кроме того, количество тестов, в которых тестирование прошло 99% последовательностей для предлагаемой процедуры разворачивания ключа, выше, чем для BBS генератора.

Для более адекватной оценки статистической безопасности процедуры разворачивания ключей были проведены исследования ее корреляционных свойств, аналогичные рассмотренным исследованиям процедуры шифрования. Процедура разворачивания ключа может рассматриваться как блочное преобразование: вход – 7 полублоков исходного пользовательского ключа (7 x 64 = 448 бит), выход – 4 x 5 + 4 полублоков «сырого» рабочего (развернутого) ключа ((4 x 5 + 4) x 64 = 1536 бит). Использовались тесты лавинного эффекта (AE) и строгого лавинного критерия (SAC). Результаты представлены в табл. 8 и табл. 9.

Таблица 8

Наименование теста	Зависимость между отдельными битами	
	M	D
AE:	799,993670	400,242937
SAC:	0,500000	0,000027
da	0,998758	
ds	0,991720	
dc	1,000000	

Таблица 9

Наименование теста	Зависимость между входными-выходными полублоками	
	M	D
AE: Min:	31,981905	15,999699
Max:	32,012294	16,026774
SAC: Min:	0,500000	0,000027
Max:	0,500000	0,000027

Выводы

Результаты статистического тестирования алгоритма «Торнадо» в режимах поточного шифрования с использованием методики NIST STS показали, что четырехцикловый вариант алгоритма является «статистически безопасным». Полученные данные сопоставимы с эталонной выборкой, сгенерированной генератором ПСП BBS. Кроме того, в режиме «усиленного» поточного шифрования (счетчик с «плавающим периодом»), начиная с трех циклов шифрования без начального и конечного преобразований, выборок, не прошедших какой-либо тест, не наблюдалось. Для более адекватной оценки «статистической безопасности» БСШ был применен корреляционный анализ, который показал, что одноцикловая версия криптоалгоритма является «статистически безопасной».

Аналогичным исследованиям подверглась процедура разворачивания ключа для выявления статистических зависимостей в развернутом («сыром») ключе. Исследование не выявило каких-либо скрытых аномалий в сгенерированной выборке. Количество тестов, в которых тестирование прошло 99% выборок, выше, чем для предлагаемой эталонной выборки ГПСП BBS.

Проведенное исследование показало, что существенное влияние на показатели статистической безопасности оказывают начальное и конечное преобразования IT и FT.

Обобщая полученные результаты, можно сказать, что статистическая безопасность алгоритма «Торнадо» достигается на таком же числе итераций, что и для алгоритма Rijndael (FIPS-197), но меньшем, чем для алгоритмов, победивших в проекте NESSIE.

Список литературы: 1. *NESSIE* Call for Cryptographic Primitives, Version 2.2, 8th March 2000 // NESSIE home page. http://cryptonessie.org. 2. «*A Statistical* Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications», NIST Special Publication 800-22, Washington, 2000. 3. *Головашич С.А.* Безопасность режимов блочного шифрования // Радиотехника: Всеукр межвед. науч.-техн. сб. 2001. Вып. 119. С. 135. 4. *Потий А.В., Орлова С.Ю. и др.* Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NIST STS // Там же. 2000. Вып. 114. С. 14 – 21. 5. *Гриненко Т.А., Горбенко Ю.И., Орлова С.Ю.* Метод формирования и свойства псевдослучайных последовательностей на эллиптических кривых // Там же. 2001. Вып. 119. С. 119 – 123.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 11.06.2003

УДК 681.3.06 : 519.248.681

И. Д. ГОРБЕНКО, д-р техн. наук, М. С. МИХАЙЛЕНКО, Т. А. ГРИНЕНКО

CAMELLIA – 128-БИТНЫЙ ПЕРСПЕКТИВНЫЙ БЛОЧНЫЙ СИММЕТРИЧНЫЙ ШИФР: МЕТОДЫ ПРЕОБРАЗОВАНИЙ, СВОЙСТВА И ОБЛАСТИ ПРИМЕНЕНИЯ

27 февраля 2003 года в европейском проекте NESSIE был представлен финальный список криптоалгоритмов, [1 – 3]. Алгоритм Camellia был включен в этот список как рекомендованный криптографический примитив.

Под аббревиатурой NESSIE скрывается название международного криптографического проекта New European Schemes for Signatures, Integrity and Encryption («Новые европейские схемы для электронных подписей, обеспечения целостности информации и шифрования»). Этот рассчитанный на три года проект запущен в январе 2000 года под эгидой Европейской комиссии. Его цель – создать «строительные блоки для будущих стандартных протоколов информационного сообщества», главными участниками являются научные и индустриальные институты семи стран: Бельгии, Великобритании, Германии, Израиля, Италии, Норвегии и Франции.

Задачи перед проектом были поставлены весьма широкие: отобрать десяток основных криптографических примитивов. В этот набор, в частности, входят алгоритмы блочного и поточного шифрования, генераторы случайных чисел, схемы быстрой аутентификации пакетов данных, хэш-функции и алгоритмы цифровой подписи. В качестве основных критериев отбора претендентов названы безопасность, производительность, гибкость и требования рынка.

Camellia это следующее поколение 128-битного блочного криптографического алгоритма, разработанного в Японии специалистами Телеграфной и Телефонной Корпорации Nippon и Электрической Корпорации Mitsubishi, поддерживающего три размера ключей: l_k = 128, 192 и 256 бит. Блочный симметричный алгоритм Camellia был разработан не только как высоко защищенный криптографический шифр, но также как алгоритм, легко переносимый на различные аппаратные платформы.

Целью настоящей статьи является ознакомление специалистов в области информационной безопасности с алгоритмом блочного симметричного шифрования, рекомендованного ведущими специалистами в данной области в качестве криптографического примитива.

1 Общая характеристика криптоалгоритма

Алгоритм Camellia является блочным симметричным криптоалгоритмом. Длина блока равна l_u = 128 бит. Криптографические преобразования блоков информации осуществляются с использованием ключевых данных. Ключ, вводимый в средства реализации Camellia, называют исходным ключом K. Разрешенными длинами исходного ключа есть l_k = 128, 192 и 256 бит. Ключи, используемые в циклах преобразования, формируются из исходного K и называются цикловыми ключами. Эти ключи формируются специальным криптографическим алгоритмом.

Криптоалгоритм Camellia может применяться в пяти режимах:

1) блочного шифрования;
2) поточного шифрования;
3) поточного шифрования с обратной связью;
4) выработки ключевой хэш-функции (кода-аутентификации);
5) генератора псевдослучайных последовательностей.

2 Представление преобразуемой информации и ключей

В алгоритме Camellia преобразования выполняются при длине блоков информации l_u, = = 128 бит и длинах исходных ключей l_k. = 128, 192 и 256 бит. Необходимая стойкость в алгоритме обеспечивается за счет многоциклового преобразования, причем в каждом из циклов применяются ключевые преобразования, то есть преобразования по развернутым ключам.

Пусть необходимо зашифровать M_i блок информации длиной 128 бит. Блок представляется в виде одномерного массива, выходные данные C_i и развернутый цикловой ключ K_i также представляются одномерными массивами. В алгоритме Camellia число циклов преобразования зависит от длины исходного ключа l_k . В табл. 1 приведено значение количества циклов как функции l_k .

Таблица 1

Длина ключа l_k, бит	128	192	256
Число циклов	18	24	24

При l_k = 128 после 6-го и 12-го циклов находятся два FL/FL^{-1} функциональных уровня (при l_k = 192, 256 функции FL/FL^{-1} расположены после 6-го, 12-го и 18-го циклов). Кроме того, перед первым и после последнего циклов выполняются криптографические преобразования – сложение по модулю 2 с цикловыми ключами. Сам цикл имеет структуру Фейстеля. Ключ преобразования имеет вид конкатенации двух 128-битных последовательностей $K_{(128)} = K_{L(128)} \| K_{R(128)}$, где $K_{R(128)}$ используется только на длинах l_k = 192, 256 бит.

3 Табличные и криптографические преобразования

На рис. 1 показана процедура зашифрования для 128-битового ключа [5]. Блок зашифрования данных имеет 18-цикловую структуру Фейстеля с двумя FL/FL^{-1}-функциональными уровнями после 6-го и 12-го циклов и 128-битовыми XOR операциями рандомизации перед первым циклом и после последнего цикла. Блок развертывания ключей генерирует подключи $kw_{t(64)}$ (t = 1, 2, 3, 4), $ku(64)$ (u = 1,2, … 18) и $kl_{v(64)}$ (v = 1,2,3,4) из секретного исходного ключа K.

В блоке рандомизации данных сначала выполняется операция XOR (сумма по модулю 2) открытого текста $M_{(128)}$ с $kw_{1(64)} \| kw_{2(64)}$ и выделение $L_{0(64)}$ и $R_{0(64)}$ равной длины, то есть:

$$M_{(128)} \oplus (kw_{1(64)} \| kw_{2(64)}) = L_{0(64)} \| R_{0(64)} ;$$

где $\oplus$ побитовая операция «Исключающее ИЛИ», а $\|$ конкатенация двух операндов.
В циклах для r = от 1 до 18 выполняется операции (кроме r = 6 и 12):

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_t),$$
$$R_r = L_{r-1}.$$

Для r = 6 и 12 выполняется следующее:

$$L_{\bar{r}} = R_{r-1} \oplus F(L_{r-1}, k_r),$$
$$R_{\bar{r}} = L_{r-1},$$
$$L_r = FL(L_{\bar{r}}, kl_{2r/6-1}),$$
$$R_r = FL^{-1}(R_{\bar{r}}, kl_{2r/6}).$$

И, наконец, $R_{18(64)}$ и $L_{18(64)}$ конкатенируются и складываются по модулю 2 с $kw_{3(64)} \| kw_{4(64)}$. В результате в первом режиме формируется блок-криптограмма длиной 128 бит:

$$C_{(128)} = (R_{18(64)} \| L_{18(64)}) \oplus (kw_{3(64)} \| kw_{4(64)}) .$$

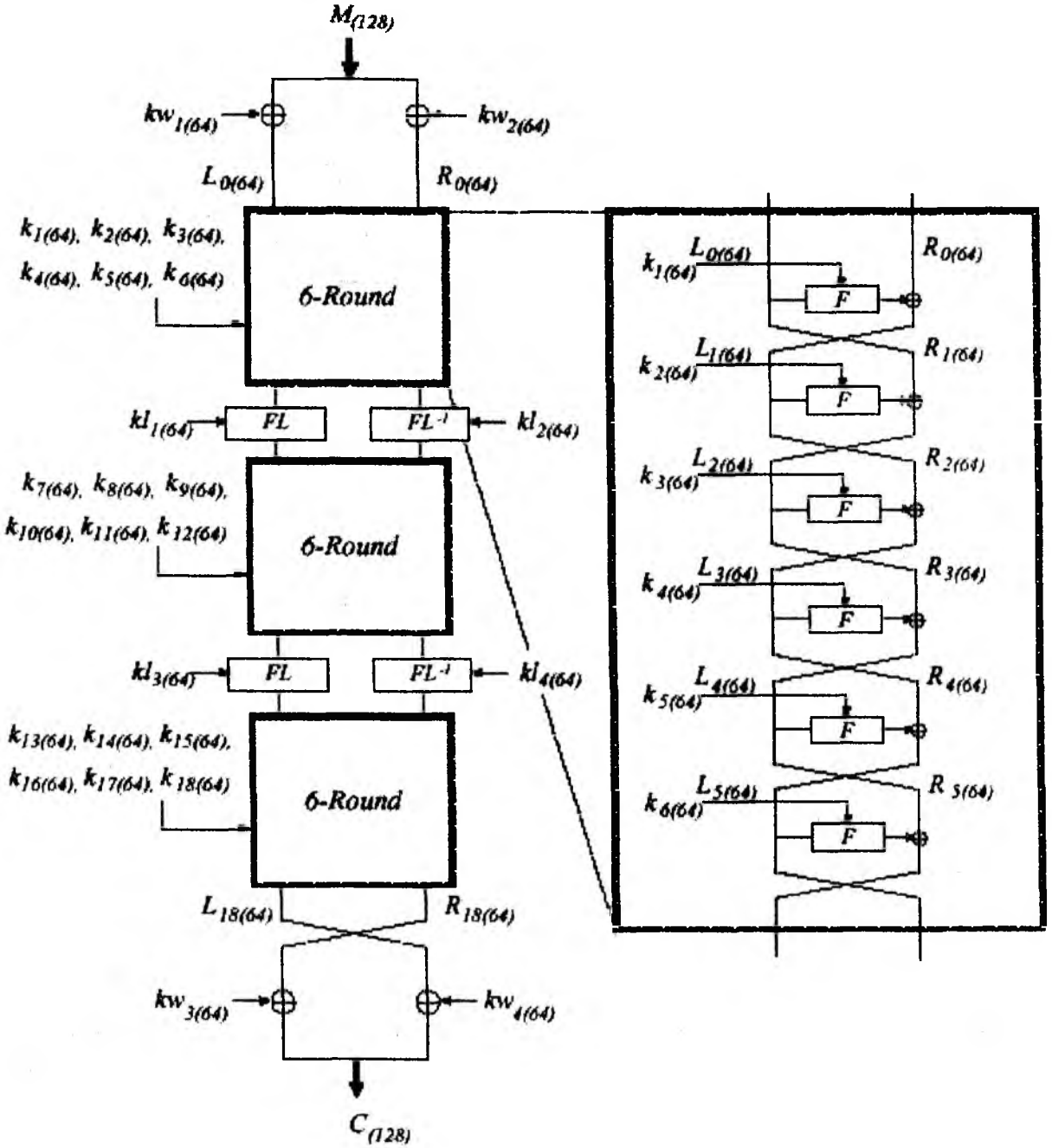


Рис. 1

При осуществлении криптопреобразований в алгоритме Camellia используется некоторый набор функций. Рассмотрим кратко функции F, FL и FL^{-1}, которые используются в алгоритме Camellia.

F-функция

На рис. 2 показана F-функция, которая определяется следующим образом:

$F : L * L \to L$,

L обозначает векторное пространство 64-битовых элементов

$$(X_{(64)}, k_{(64)}) \to Y_{(64)} = P(S(X_{(64)} \oplus k_{(64)})).$$

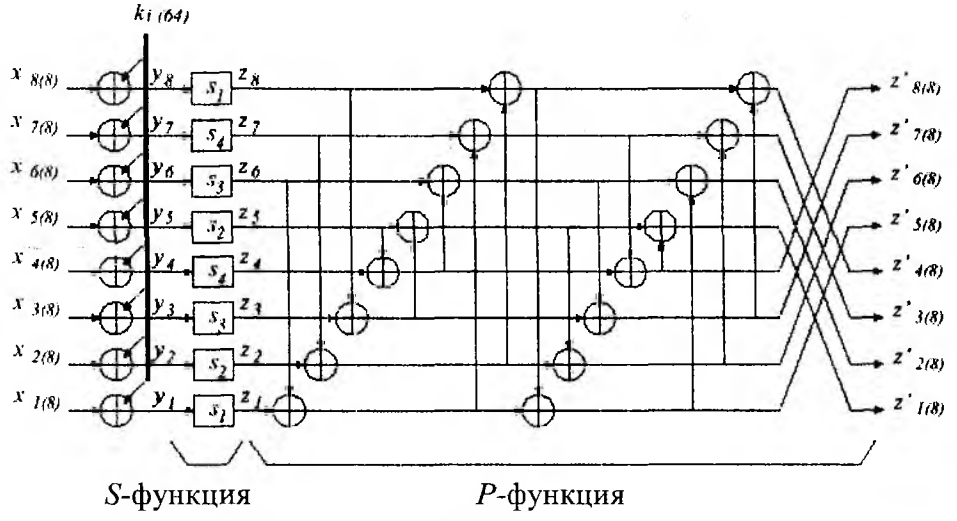


Рис. 2

S-функция

S-функция является частью *F*-функции, которая определяется следующим образом: $S: L \to L$.

$$l_{1(8)} \| l_{2(8)} \| l_{3(8)} \| l_{4(8)} \| l_{5(8)} \| l_{6(8)} \| l_{7(8)} \| l_{8(8)} \to$$
$$\to l_{\bar{1}(8)} \| l_{\bar{2}(8)} \| l_{\bar{3}(8)} \| l_{\bar{4}(8)} \| l_{\bar{5}(8)} \| l_{\bar{6}(8)} \| l_{\bar{7}(8)} \| l_{\bar{8}(8)},$$

где

$l_{\bar{1}(8)} = s_1(l_{1(8)})$; $l_{\bar{2}(8)} = s_2(l_{2(8)})$; $l_{\bar{3}(8)} = s_3(l_{3(8)})$; $l_{\bar{4}(8)} = s_4(l_{4(8)})$; $l_{\bar{5}(8)} = s_2(l_{5(8)})$; $l_{\bar{6}(8)} = s_3(l_{6(8)})$;
$l_{\bar{7}(8)} = s_4(l_{7(8)})$; $l_{\bar{8}(8)} = s_1(l_{8(8)})$.

s-блоки

Четыре *s*-блока шифра Camellia являются аффинным эквивалентом функции инверсии на $GF(2^8)$, что представлено в табл. 2,3,4 и 5. Ниже показано алгебраическое представление *s*-блоков:

$$s_1 : B \to B,$$
$$x_{(8)} \to h(g(f(0xc5 \oplus x_{(8)}))) \oplus 0x6e,$$
$$s_2 : B \to B,$$
$$x_{(8)} \to s_1(x_{(8)}) <<< 1,$$
$$s_3 : B \to B,$$
$$x_{(8)} \to s_1(x_{(8)}) >>> 1,$$
$$s_4 : B \to B,$$
$$x_{(8)} \to s_1(x_{(8)} <<< 1),$$

здесь <<<n (>>>n) – циклический сдвиг влево (вправо) операнда на n бит.
B обозначает векторное пространство 8-битовых элементов (байтов), функции *f*, *g* и *h* задаются следующим образом:
$f : B \to B$

$$a_{1(1)} \| a_{2(1)} \| a_{3(1)} \| a_{4(1)} \| a_{5(1)} \| a_{6(1)} \| a_{7(1)} \| a_{8(1)} \rightarrow$$

$$\rightarrow b_{1(1)} \| b_{2(1)} \| b_{3(1)} \| b_{4(1)} \| b_{5(1)} \| b_{6(1)} \| b_{7(1)} \| b_{8(1)},$$

где

$b_1 = a_6 \oplus a_2$; $b_2 = a_7 \oplus a_1$; $b_3 = a_8 \oplus a_5 \oplus a_3$; $b_4 = a_8 \oplus a_3$; $b_5 = a_7 \oplus a_4$; $b_6 = a_5 \oplus a_2$;
$b_7 = a_8 \oplus a_1$; $b_8 = a_6 \oplus a_4$.

$g : B \rightarrow B$

$$a_{1(1)} \| a_{2(1)} \| a_{3(1)} \| a_{4(1)} \| a_{5(1)} \| a_{6(1)} \| a_{7(1)} \| a_{8(1)} \rightarrow$$

$$\rightarrow b_{1(1)} \| b_{2(1)} \| b_{3(1)} \| b_{4(1)} \| b_{5(1)} \| b_{6(1)} \| b_{7(1)} \| b_{8(1)},$$

где

$(b_8 + b_7 a + b_6 a^2 + b_5 a^3) + (b_4 + b_3 a + b_2 a^2 + b_1 a^3)\beta =$
$= 1/((a_8 + a_7 a + a_6 a^2 + a_5 a^3) + (a_4 + a_3 a + a_2 a^2 + a_1 a^3)\beta)$;

Эта инверсия выполняется в $GF(2^8)$, где β является элементом $GF(2^8)$, который удовлетворяет условию $\beta^8 + \beta^6 + \beta^5 + \beta^3 + 1 = 0$, и $a = \beta^{238} = \beta^6 + \beta^5 + \beta^3 + \beta^2$ является элементом $GF(2^4)$, который удовлетворяет условию: $a^4 + a + {}^1 = 0$.

$h : B \rightarrow B$

$$a_{1(1)} \| a_{2(1)} \| a_{3(1)} \| a_{4(1)} \| a_{5(1)} \| a_{6(1)} \| a_{7(1)} \| a_{8(1)} \rightarrow$$

$$\rightarrow b_{1(1)} \| b_{2(1)} \| b_{3(1)} \| b_{4(1)} \| b_{5(1)} \| b_{6(1)} \| b_{7(1)} \| b_{8(1)},$$

где

$b_1 = a_5 \oplus a_6 \oplus a_2$; $b_2 = a_6 \oplus a_2$; $b_3 = a_7 \oplus a_4$; $b_4 = a_8 \oplus a_2$; $b_5 = a_7 \oplus a_3$; $b_6 = a_8 \oplus a_1$;
$b_7 = a_5 \oplus a_1$; $b_8 = a_6 \oplus a_3$.

В табл. 2, 3, 4, 5 приведены s-блоки подстановок.

Таблица 2

112	130	44	236	179	39	192	229	228	134	87	53	234	12	174	65
35	239	107	147	69	25	165	33	237	14	79	78	29	101	146	189
134	184	175	143	124	235	31	206	62	48	220	95	94	197	11	26
166	225	57	202	213	71	93	61	217	1	90	214	81	86	108	77
139	13	154	102	251	204	176	45	116	18	43	32	240	177	132	153
223	76	203	194	52	126	118	5	109	183	169	49	209	23	4	215
20	88	58	97	222	27	17	28	50	15	156	22	83	24	242	34
254	68	207	178	195	181	122	145	36	8	232	168	96	252	105	80
170	208	160	125	161	137	98	151	84	91	30	149	224	255	100	210
16	196	0	72	163	247	117	219	138	3	230	218	9	63	221	148
135	92	131	2	205	74	144	51	115	103	246	243	157	127	191	226
82	155	216	38	200	55	198	59	129	150	111	75	19	190	99	46
233	121	167	140	159	110	188	142	41	245	249	182	47	253	180	89
120	152	6	106	231	70	113	186	212	37	171	66	136	162	141	250
114	7	185	85	248	238	172	10	54	73	42	104	60	56	241	164
64	40	211	123	187	201	67	193	21	227	173	244	119	199	128	158

Таблица 3

224	5	88	217	103	78	129	203	201	11	174	106	213	24	93	130
70	223	214	39	138	50	75	66	219	28	158	156	58	202	37	123
13	113	95	31	248	215	62	157	124	96	185	190	188	139	22	52
77	195	114	149	171	142	186	122	179	2	180	173	162	172	216	154
23	26	53	204	247	153	97	60	232	33	86	64	225	99	9	51
191	152	151	133	104	252	238	10	218	111	83	98	163	46	8	175
40	176	116	194	189	54	34	56	100	30	57	44	166	48	229	68
253	136	159	101	135	107	244	35	72	16	209	81	192	249	210	160
85	161	65	250	67	19	196	47	168	182	60	43	193	255	200	165
32	137	0	144	71	239	234	183	21	6	205	181	18	126	187	41
15	184	7	4	155	148	33	102	230	206	237	231	59	254	127	197
164	55	177	76	145	110	141	118	3	45	222	150	38	125	198	92
211	242	79	25	63	220	121	29	82	235	243	109	94	251	105	178
240	49	12	212	207	140	226	117	169	74	87	132	17	69	27	245
228	14	115	170	241	221	89	20	108	146	84	208	120	112	227	73
128	80	167	246	119	147	134	131	42	199	91	233	238	143	1	61

Таблица 4

56	65	22	118	217	147	96	242	114	194	171	154	117	6	87	160
145	247	181	201	162	140	210	144	246	7	167	39	142	178	73	222
67	92	215	199	62	245	143	103	31	24	110	175	47	226	133	13
83	240	156	101	234	163	174	158	236	129	45	107	168	43	54	166
197	134	77	51	253	102	88	150	58	9	149	16	120	216	66	204
239	38	229	97	26	63	59	130	182	219	212	152	232	139	2	235
10	44	29	176	111	141	136	14	25	135	78	11	169	12	121	17
127	34	231	89	225	218	61	200	18	4	116	84	48	126	180	40
85	104	80	190	208	196	49	203	42	173	15	202	112	255	50	105
8	98	0	36	209	251	188	237	69	129	115	109	132	159	238	74
195	46	193	1	230	37	72	153	185	179	123	249	206	191	223	113
41	205	108	19	100	155	99	157	192	75	183	165	137	95	177	23
244	187	211	70	207	55	94	71	148	250	252	91	151	254	90	172
60	76	3	53	243	35	184	93	106	146	213	33	68	81	198	125
57	131	220	170	124	119	86	5	27	164	21	52	30	28	208	82
32	30	233	189	221	228	161	224	138	241	214	122	187	227	64	79

Таблица 5

112	44	179	192	228	87	234	174	35	107	69	165	237	79	29	146
134	175	124	31	62	220	94	11	166	57	213	93	217	90	81	108
139	154	251	176	116	43	240	182	223	203	53	118	109	169	209	4
20	58	222	17	50	156	83	242	254	207	195	122	36	232	96	105
170	160	161	98	84	30	224	100	16	0	163	117	138	230	9	221
135	131	205	144	115	246	157	191	82	216	200	198	129	111	19	99
233	167	159	188	61	249	47	180	120	6	251	113	212	171	136	141
114	185	248	172	54	42	60	241	64	211	187	67	21	173	119	128
130	236	39	229	133	53	12	65	239	147	25	33	14	78	101	189
134	143	235	206	48	85	197	26	225	202	71	61	1	214	86	77
13	102	204	45	16	32	177	153	76	194	126	5	183	49	23	215
88	97	27	28	15	22	24	34	68	178	181	145	8	168	252	80
208	125	137	151	91	149	255	210	196	72	247	219	3	218	63	148
92	2	74	51	103	243	127	226	155	38	55	59	150	75	190	46
121	140	110	142	246	132	253	89	152	106	70	136	37	66	162	250
7	85	238	10	73	104	56	164	40	123	201	193	227	244	199	158

P-функция

P-функция является частью *F*-функции, которая определяется следующим образом: $P: L \to L$,

$$z_{1(8)} \| z_{2(8)} \| z_{3(8)} \| z_{4(8)} \| z_{5(8)} \| z_{6(8)} \| z_{7(8)} \| z_{8(8)} \to$$
$$\to z_{\bar{1}(8)} \| z_{\bar{2}(8)} \| z_{\bar{3}(8)} \| z_{\bar{4}(8)} \| z_{\bar{5}(8)} \| z_{\bar{6}(8)} \| z_{\bar{7}(8)} \| z_{\bar{8}(8)},$$

где

$z_{\bar{1}} = z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8$; $z_{\bar{2}} = z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7 \oplus z_8$; $z_{\bar{3}} = z_1 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8$;

$z_{\bar{4}} = z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7$; $z_{\bar{5}} = z_1 \oplus z_2 \oplus z_6 \oplus z_7 \oplus z_8$; $z_{\bar{6}} = z_2 \oplus z_3 \oplus z_5 \oplus z_7 \oplus z_8$;

$z_{\bar{7}} = z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_8$; $z_{\bar{8}} = z_1 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7$.

Это преобразование может быть представлено так же в следующем виде:

$$\begin{pmatrix} z_8 \\ z_7 \\ \vdots \\ z_1 \end{pmatrix} \to \begin{pmatrix} z_{\bar{8}} \\ z_{\bar{7}} \\ \vdots \\ z_{\bar{1}} \end{pmatrix} = P \begin{pmatrix} z_8 \\ z_7 \\ \vdots \\ z_1 \end{pmatrix},$$

где

$$P = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

FL-функция

На рис. 3 показана *FL*-функция, которая определяется так, как показано ниже: $FL: L*L \to L$,

$$(X_{L(32)} \| X_{R(32)}, kl_{L(32)} \| kl_{R(32)}) \to Y_{L(32)} \| Y_{R(32)},$$

где

$Y_{R(32)} = ((X_{L(32)} \cap kl_{L(32)}) <<< 1) \oplus X_{R(32)}$ ($\cap$ – побитовая операция И);

$Y_{L(32)} = (Y_{R(32)} \cup kl_{R(32)}) \oplus X_{L(32)}$ (U – Побитовая операция ИЛИ).

FL^{-1}-функция

На рис. 4 показана FL^{-1}-функция, которая определяется следующим образом: $FL^{-1}: LxL \to L$,

$$(Y_{L(32)} \| Y_{R(32)}, kl_{L(32)} \| kl_{R(32)}) \to X_{L(32)} \| X_{R(32)},$$

где

$$X_{L(32)} = (Y_{R(32)} \cup kl_{R(32)}) \oplus Y_{L(32)};$$
$$X_{R(32)} = ((X_{L(32)} \cap kl_{L(32)}) <<< 1) \oplus Y_{R(32)}.$$

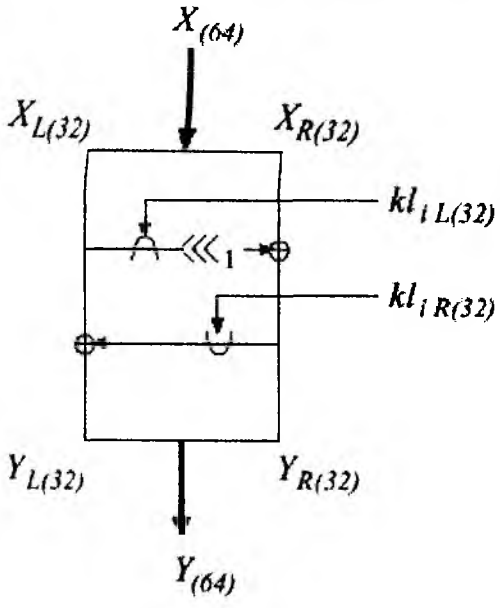


Рис. 3

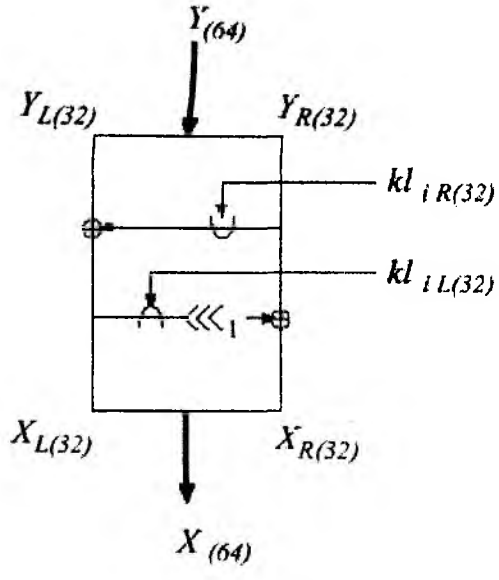


Рис. 4

Мы рассмотрели схему шифрования на длине l_k = 128 бит, шифрование на длинах l_k = 192, 256 бит выполняется аналогично за исключением того, что в схему добавляется блок из шести циклов Фейстеля и после 18-го цикла выполняются FL / FL^{-1} преобразования.

4 Выработка цикловых ключей

В алгоритме Camellia применяется принцип разворачивания цикловых ключей [5]. Сущность его заключается в том, что необходимое число цикловых ключей вырабатывается из исходного ключа K. Схема разворачивания ключей представлена на рис. 5. Как исходные так и цикловые ключи представляются в виде одномерных массивов. В блоке таблицы ключей шифра Camellia вводятся две 128-битовые переменные $K_{L(128)}$, $K_{R(128)}$ и четыре 64-битовые переменные $K_{LL(64)}$, $K_{LR(64)}$, $K_{RL(64)}$ и $K_{RR(64)}$, которые заданы так, чтобы удовлетворялись следующие отношения:

$K_{(128)} = K_{L(128)}$, $K_{R(128)} = 0$ для 128-битового ключа,

$K_{(192)} = K_{L(128)} \| K_{RL(64)}$, $K_{RR(64)} = \overline{K}_{RL(64)}$ для 192-битового ключа,

$K_{(256)} = K_{L(128)} \| K_{R(128)}$ для 256-битового ключа.

С помощью этих переменных генерируются две 128-битовых переменных $K_{A(128)}$ и $K_{B(128)}$, как показано на рис. 5, где $K_{B(128)}$ используется только при длине секретного ключа 192 или 256 бит. Сначала выполняется XOR операция $K = K_{L(128)}$ с $K_{R(128)}$, и результат «шифруется» за два цикла с использованием постоянных значений $\sum_{1(64)}$ и $\sum_{2(64)}$ как «ключей»; результат XOR-ся с $K_{L(128)}$ и зашифровуется еще на двух циклах с использованием $\sum_{3(64)}$ и $\sum_{4(64)}$. В итоге результирующим значением является $K_{A(128)}$. И наконец, выполняется XOR-операция $K_{A(128)}$ с $K_{R(128)}$, результат шифруется за два цикла с использованием постоянных значений $\sum_{5(64)}$ и $\sum_{6(64)}$. Результирующим значением является $K_{B(128)}$. Постоянные значения $\sum_i$ приведены в табл. 6.

Подключи $kw_{t(64)}$, $k_{u(64)}$ и $kl_{v(64)}$ генерируются из (левой или правой половины) циклически сдвинутых значений $K_{L(128)}$, $K_{R(128)}$, $K_{A(128)}$ и $K_{B(128)}$.

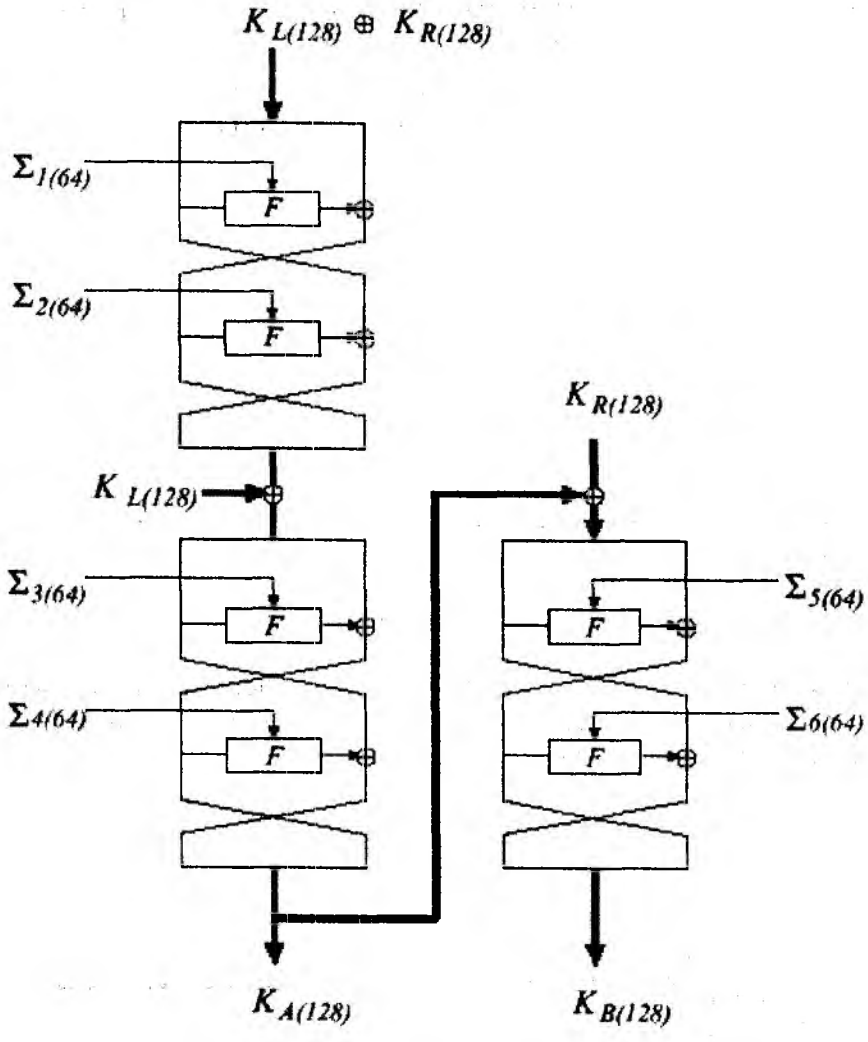


Рис. 5

Таблица 6

$\sum_{1(64)}$	0xA09E667F3BCC908B
$\sum_{2(64)}$	0xB67AE8584CAA73B2
$\sum_{3(64)}$	0xC6EF372FE94F82BE
$\sum_{4(64)}$	0x54FF53A5F1D36F1C
$\sum_{5(64)}$	0x10E527FADE682D1D
$\sum_{6(64)}$	0xB05688C2B3E6C1FD

5 Обратные преобразования (расшифрование)

Процедура расшифрования в алгоритме Camellia может быть выполнена таким же способом, как процедура зашифрования, путем изменения на обратный порядка подключей. На рис. 6 показана процедура расшифрования для 128-битового ключа. Алгоритм расшифрования также имеет 18-цикловую структуру Файстеля с двумя FL/FL^{-1}-функциональными слоями после 6-го и 12-го циклов и 128-битовыми операциями XOR перед первым циклом и после последнего цикла. Блок таблицы ключей генерирует подключи $kw_{t(64)}$ (t = 1, 2, 3, 4), $ku(64)$ (u = 1,2, … 18) и $kl_{v(64)}$ (v = 1,2,3,4) из секретного ключа K.

При расшифровании сначала выполняется операция XOR шифротекста $C_{(128)}$ с $kw_{3(64)}$ $kw_{4(64)}$ и выделение в $R_{18(64)}$ и $L_{18(64)}$ равной длины, то есть

$$C_{(128)} \oplus (kw_{3(64)} \| kw_{4(64)}) = R_{18(64)} \| L_{18(64)} .$$

Для r = от 18 до 1, выполняются следующие операции (кроме r = 13 и 7):

$$R_{r-1} = L_r \oplus F(R_r, k_t),$$
$$L_{r-1} = R_r .$$

Для r = 13 и 7 выполняется следующее:

$$R_{\bar{r}-1} = L_r \oplus F(R_r, k_r),$$
$$L_{\bar{r}-1} = R_r ,$$
$$R_{r-1} = FL(R_{\bar{r}-1}, kl_{2(r-1)/6-1}),$$
$$L_{r-1} = FL^{-1}(L_{\bar{r}-1}, kl_{2(r-1)/6-1}) .$$

И, наконец, $L_{0(64)}$ и $R_{0(64)}$ конкатенируются и XOR-ся с $kw_{1(64)} \| kw_{2(64)}$. Результирующее значение является открытым текстом, то есть

$$M_{(128)} = (L_{0(64)} \| R_{0(64)}) \oplus (kw_{1(64)} \| kw_{2(64)}) .$$

Схема расшифрования на длине l_k = 192, 256 бит выполняется аналогично зашифрованию, за исключением того, что в схему добавляется блок из шести циклов Фейстеля и перед 19-м циклом выполняются FL / FL^{-1} преобразования.

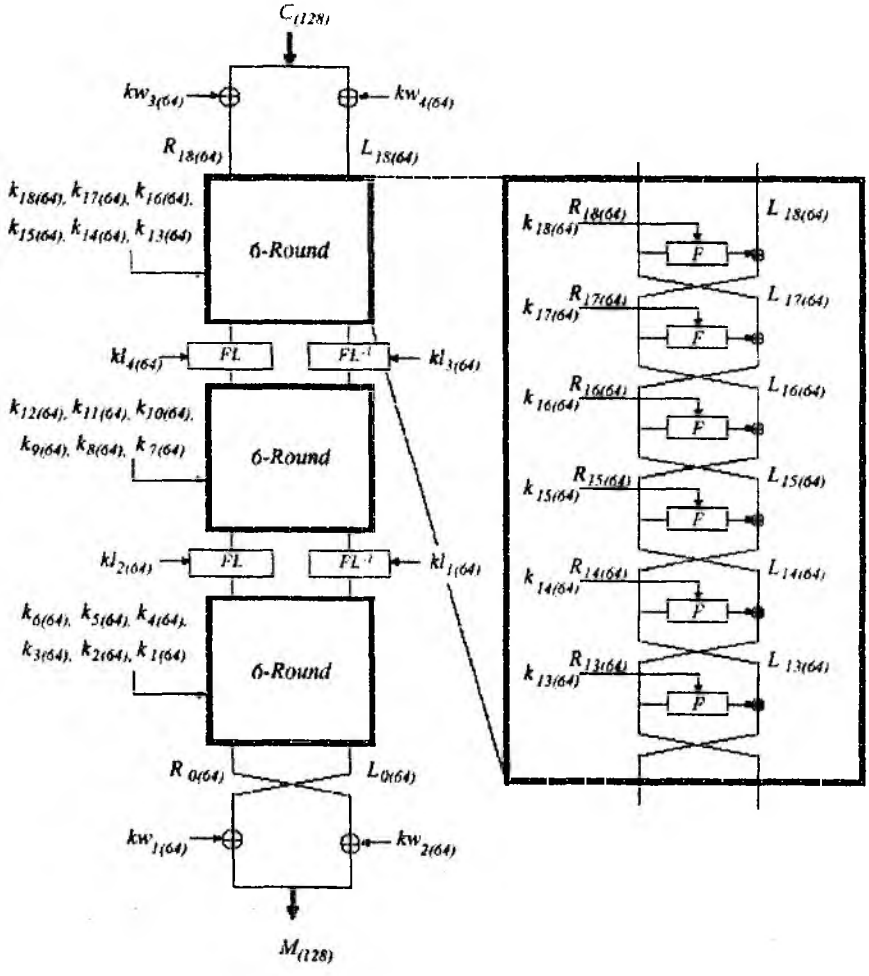


Рис. 6

6 Анализ стойкости криптоалгоритма

В процессе всех этапов «общественного», научного и практического обсуждения и анализа алгоритма Camellia, а также исследования и тестирования его со стороны Европейской комиссии NESSIE особое внимание уделялось анализу и оценке по критериям реальной криптозащищенности, статистической безопасности и надежности математической базы криптоалгоритма. На сегодняшний день относительно алгоритма Camellia не известно, а, возможно, и не существует ни одного метода криптоанализа, сложность реализации которого была бы меньше, чем методы, основанные на «грубой силе» (переборе). Алгоритм оказался защищенным от атак, реализованных на основе: дифференциального криптоанализа; поиска наилучшей дифференциальной характеристики; расширения для дифференциального криптоанализа; линейного криптоанализа; вторжения с использованием связанных ключей; вторжения с частичным угадыванием ключа; вторжения на основе обработки сбоев; интерполяционного (алгебраического) вторжения; поиска лазеек. Основу криптозащищенности в алгоритме составляют криптографические преобразования на множестве циклов.

Ниже будут рассмотрены результаты анализа статистической криптостойкости алгоритма, проведенные авторами. При анализе статистической безопасности алгоритма Camellia оценивалась связанность криптограмм между собой и со входными блоками открытых текстов, а также определялась избыточность криптограмм. Избыточность связана с зависимостью символов сообщений соответствующего алфавита и неодинаковой вероятностью их появления.

Пусть m_1, m_2, …, m_n – последовательность бит, принадлежащих открытому блоку исходного текста, а c_1, c_2, …, c_n – последовательность бит, принадлежащих соответствующей криптограмме. Символы m_i и c_i являются двоичными, то есть принимают значения 1 или 0. Связанность блоков M_i и C_i можно определить по формуле:

$$F = f(m_i, c_i) = \sum_{j=1}^{l_u} (m_i \oplus c_i),$$

а блоков C_i и C_k :

$$F = f(c_i, c_k) = \sum_{j=1}^{l_u} (c_i \oplus c_k),$$

где l_u длина блока шифруемой информации (128 бит).

Математическое ожидание зашифрованного блока можно определить по формуле:

$$M = f(b_j) = \frac{\sum_{j=1}^{l_u} b_j}{l_u}.$$

Расстояние Хемминга для блоков M_i и C_i можно найти как

$$H = f(M_j, C_j) = M_j \oplus C_j.$$

Будем считать блоки независимыми (некоррелированными), если $F = l_u / 2$, а математическое ожидание стремится к ½. Оценка статистической безопасности проведена при следующих параметрах работы криптоалгоритма: случайный ключ, случайные данные, ключ постоянный.

Таблица 7

Режим	Связанность блоков		Математическое ожидание	Расстояние Хемминга M_i, C_i
	M_i, C_i	C_k, C_i	C_i	
Работа алгоритма	64,0015	64,0494	0,499996	63,9985
Схема разворачивания ключей	64,0207	64,9193	0,48711	63,9793

При определении уровня корреляции блоков брался исходный файл и криптограмма, полученная в результате работы криптоалгоритма Camellia над этим исходным файлом. Затем считывались оба файла по 16 байт (128 бит), и находилась связанность каждого из прочитанных блоков. В конце результат делился на число прочитанных блоков. При поиске корреляции между криптограммами исходный текст изменялся, зашифровывался, и связанность искалась уже между этой криптограммой и криптограммой предыдущей. Такая же схема анализа связанности блоков использовалась и при нахождении связанности для схемы разворачивания ключей.

Математическое ожидание, также как и расстояние Хемминга, считалось для криптограмм, полученных в результате работы криптоалгоритма и для схемы разворачивания ключей.

Оценим качество зашифрования на основе анализа избыточности

Существование избыточности в зашифрованных текстах исследовалось с использованием программы – архиватора WinRAR. В роли входных текстов использовались:

DLL – динамическая библиотека WINDOWS;

DOC – документ Microsoft WORD 2000;

EXE – исполняемый файл WINDOWS;

PDF – документ переносимого формата;

TXT – обычный текст ANSI;

ZIP – файл-архив.

В качестве показателей сжатия использовались следующие соотношения:

$$k_1 = \frac{l'_m}{l_m} \text{ и } k_2 = \frac{l'_c}{l_m},$$

где

l'_m – длина сжатого открытого текста; l_m – длина входного открытого текста; l'_c – длина сжатой криптограммы.

В табл. 8 в качестве примеров приведены значения k_1 и k_2.

Таблица 8

Тип файла	l_m	l'_m	l'_c	k_1	k_2
DLL	1096736	524234	1071688	0,477994	0,977161
DOC	1085952	142893	607064	0,131583	0,559015
EXE	1060864	247488	810104	0,233289	0,763626
PDF	1062516	568422	887720	0,534977	0,835488
TXT	1318671	114096	707309	0,086523	0,536380
ZIP	1081634	1078088	1081727	0,996721	1,000085

Из анализа данных табл. 7 и 8 следует, что связанность блоков M_i и C_i стремится к $F = l_u / 2$ (то есть к 64), математическое ожидание стремится к ½, расстояние Хемминга – к 64, k_2 намного больше k_1, а в некоторых случаях приближается к единице. Следовательно, зашифрованный файл практически не имеет избыточности.

Дифференциальный и линейный криптоанализ

В 1990 г. Бихам и Шамир опубликовали статью, в которой описали аналитическую атаку на DES, сложность которой оказалась меньше, чем грубая сила (дифференциальный криптоанализ). Суть метода дифференциального криптоанализа сводится к нахождению разности между специально подбираемыми блоками M_i и M_j и соответствующими им блоками криптограмм C_i и C_j. То есть в начале подбираются пары сообщений M_i и M_j, которые имеют определенное расстояние между собой. Затем эти два блока пропускаются через схему шифрования, где просчитываются разности на каждом из циклов. Находится разность расстояний $\Delta M, \Delta C$. Далее для некоторых разностей, используя стандартные подстановки, можно найти вероятности появления ключей. По мере того, как все больше пар M_i и M_j проходят через алгоритм, производится уточнение вероятности появления ключей. Истинный ключ после большого числа испытаний становится высоко вероятным.

В 1993 году японский математик Мацуки опубликовал линейный метод криптоанализа. В нем ищутся произведения битов информации Mv на C_ξ биты криптограммы, которые равны произведению битов ключа $ПМ_v \cdot ПС_\xi = ПК_\varepsilon$. Решая это уравнение, осуществляем криптоанализ. Сложность линейного криптоанализа приблизительно равна сложности дифференциального криптоанализа.

Что касается криптоалгоритма Camellia, то для него вероятность нахождения s – блока $p_s = q_s = 2^{-6}$. Поскольку коэффициент линейной трансформации (P-функции) равен $B = 5$, то после 18 циклов Фейстеля без учета FL и FL^{-1} функций мы имеем следующие вероятности:

$$p_s^{2(2B+1)} = (2^{-6})^{22} = 2^{-132} \text{ и } q_s^{2(2B+1)} = (2^{-6})^{22} = 2^{-132}.$$

Обе вероятности p_s, q_s ниже порога безопасности для 128-битного блочного шифра: 2^{-128}. С учетом FL и FL^{-1} эти вероятности будут еще меньше.

7 Результаты анализа характеристик алгоритма Camellia

Экспериментально были исследованы следующие характеристики алгоритма Camellia – скорость зашифрования / расшифрования, сложность зашифрования / расшифрования и сложность разворачивания ключей.

В табл. 9 приведены результаты измерения скорости зашифрования.

Таблица 9

Процессор	Язык	Длина ключа	Зашифрование (Mbit/sec)
Pentium III (1)	Assembler	128	241,5
Pentium III (1)	Assembler	192	181
Pentium III (1)	Assembler	256	181
Pentium II (2)	ANSI C	128	66,6

(1) IBM совместимый компьютер с процессором Intel Pentium III (700 Mhz), 256Kb кэш L2, 128Mb оперативной памяти, операционная система FreeBSD 4.0R.
(2) IBM совместимый компьютер с процессором Intel Pentium II (300 Mhz), 512Kb кэш L2, 160Mb оперативной памяти, операционная система Windows 95.

В табл. 10 приведены результаты [6] измерения скоростей шифрования для ряда шифров.

Таблица 10

Алгоритм	Процессор	Скорость (cycles/byte)	
Camellia	Sparc	22,2	22,2
	Alpha	17,6	17,6
4-Way IDEA	PIII	13,75	
	PI/MMX	16,96	
Misty1	PII	26,6	26
	Alpha	25,4	25,8
RC6	P.ProII	16	
	Pentium	44	
	PIII	22,18	
Rijndael	PIII	14,13	14,93
	P.ProII	18	
	Pentium	20	
Mars	P.ProIII	20	
	Pentium	34	
Twofish	P.ProII	16	
	Pentium	19	

В табл. 11 приведены оценки сложности зашифрования / расшифрования / разворачивания ключей для различных шифров в числе циклов на байт. Эти результаты взяты из [6].

Таблица 11

название	$l_{k\ бит}$	Процессоры </операц. сист>		
		PIII/Linux	PIII/MS	Pentium4
Camellia	128	35/35/313	37/37/334	64/63/453
	192	45/45/420	49/48/434	86/86/546
	256	45/45/429	49/49/447	86/87/555
RC6 (20 rounds)	128	17/17/1054	24/25/1040	36/37/2056
	192	18/17/1175	30/25/1258	37/37/2141
	256	18/17/1168	25/25/1262	37/37/2153
Safer++	128	50/63/1333	46/60/1464	47/58/2918
	256	68/88/1805	63/84/1865	65/83/3998
Anubis	128	37/37/3550	37/37/3727	35/35/3750
	160	40/40/4519	39/39/4927	37/37/4845
	192	43/43/5857	42/42/6993	40/39/6054
	224	45/45/7356	44/44/8546	41/41/7338
	256	48/48/8876	47/47/10K	44/44/8902
	288	50/50/10K	49/49/12K	46/46/10K
	320	53/53/12K	51/51/13K	48/48/11K
Hiero-crypt-3	128	51/64/125K	50/56/13 IK	54/63/195K
	192	60/75/147K	59/67/1 54K	63/75/229K
	256	69/86/170K	67/76/1 78K	73/85/266K
Mars	128	31/30/2520	36/35/2354	82/72/3492
	192	31/30/2530	36/35/2337	82/72/3480
	256	31/30/2525	36/35/2336	82/73/3599
Rijndael	128	25/26/504	23/23/497	24/25/689
	192	30/31/601	27/27/552	28/29/820
	256	34/35/949	32/32/775	32/35/1327
Seed	128	45/45/409	51/50/421	63/63/401
Serpent	128	68/80/1577	59/57/1276	154/172/2235
	192	68/80/1578	59/57/1272	155/171/2232
	256	68/80/1566	59/57/1257	154/171/2231
Twofish	128	28/26/10K	28/29/15K	51/49/8871
	192	28/26/12K	30/30/17K	52/49/11K
	256	28/25/16K	28/29/20K	52/49/13K

Из табл. 11 следует, что по сложности алгоритм Camellia уступает RC6 и несколько уступает только Rijndael.

Заключение

Криптоалгоритм Camellia был разработан японскими Телеграфной и Телефонной Корпорацией Nippon и Электрической Корпорацией Mitsubishi [4]. В его разработке принимали участие такие известные специалисты в области практической криптографии как Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, Toshio Tokita.

Криптоалгоритм Camellia работает с блоками данных длиной 128 бит и длиной ключа 128, 192 и 256 бит, то есть он имеет такую же спецификацию интерфейса как и Advanced Encryption Standard (AES). Криптоалгоритм Camellia эффективно работает как в программной, так и в аппаратной реализации. По сравнению с финалистами AES, такими как MARS, RC6, Rijndael, Serpent и Twofish, Camellia показывает приемлемую скорость зашифрования / расшифрования. Оптимальная программная реализация криптоалгоритма Camellia, написанная на языке Assembler, позволяет достичь скорости зашифрования на Pentium III (1,13 GHz) 471 Мбит/сек. [4].

На сегодняшний день криптоалгоритм Camellia оказался устойчивым к следующим атакам:

- Дифференциальный и линейный криптоанализ.
- Усеченный дифференциальный криптоанализ.
- Усеченный линейный криптоанализ.
- Криптоанализ с невозможным дифференциалом.
- Атака типа бумеранг.
- Дифференциальная атака высших порядков.
- Квадратная атака.
- Интерполяционная и линейная суммарная атака.
- Атака на основе эквивалентных ключей.
- Скользящая атака.
- Атака на основе связанных ключей.
- Атаки, основанные на грубой силе.

Таким образом, международный проект NESSIE выполнен.

10 марта 2000 года NTT и Mitsubishi Electric Corporation представили совместно разработанный новый блочный симметричный шифр, названный Camellia. 17 апреля 2001 года NTT анонсировала, что NTT и Mitsubishi Electric Corporation намереваются разрешить бесплатное использование патентов Camellia.

24 сентября 2001 года алгоритм Camellia был представлен как один из криптоалгоритмов, участвующих во втором туре проекта NESSIE.

20 февраля 2003 года алгоритм Camellia был включен в список криптографических примитивов для использования в таких электронных правительственных системах Японии как: Министерство Иностранных Дел, Министерство Внутренних Дел, Министерство Экономики, Министерство Почты и Телекоммуникаций, Министерство Торговли и Индустрии. Криптостойкость алгоритма была оценена Комитетом Криптографических Оценок и Исследований (CRYPTREC) как высокая.

27 февраля 2003 года проект NESSIE [7] анонсировал финальный выбор криптографических алгоритмов, и Camellia была включена в список рекомендованных криптографических примитивов для использования в европейском сообществе.

Криптоалгоритм Camellia [8] эффективно реализуется как программно, так и аппаратно. В нем используются только 8-битовые таблицы подстановок (s-блоки) и логические операции, которые могут эффективно работать на различных аппаратных платформах. Поэтому криптоалгоритм способен работать на большой совокупности платформ, включая 8-, 32-, 64-битовые процессоры. Вместе с тем в криптоалгоритме Camellia не используется 32-битовая целочисленная операция сложения и умножения, которые широко используются некоторыми программно-ориентированными 128-битными блочными шифрами.

Схема разворачивания ключей очень проста и базируется на схеме зашифрования. Подключи, которые генерируются из основного ключа, могут быть вычислены в любом порядке. Для генерации подключей используется 32 байта для 128-битного ключа и 64 байта для ключей длиной 192/256 бит.

Список литературы: 1. *News* Releases № 2300 http://www.Global.MitsubishiElectric.com. 2 *Proposal* of addition of new cipher suites to TLS to support Camellia, EPOC, and PSEC 8/14/00 http://www.ietf.org 3. http: // www. cryptonessie.org 4. *K. Aoki,* T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, \ Camellia: A 128-bit block cipher suitable for multiple platforms. See also http://info.isl.ntt.co.jp/camellia/. 5. *Specification* of Camellia – a 128-bit Block Cipher. 6. *Performance* of Optimized Implementations of the NESSIE Primitives B. Preneel, B. Van Rompay, S.B. Ors, A. Biryukov, L. Granboulan, E. Dottax. 7 *NESSIE* PROJECT ANNOUNCES FINAL SELECTION OF CRYPTO ALGORITHMS. 8 *Specification* of Camellia – 128-bit Block Cipher. Difference between specifications.

Харьковский национальный

университет радиоэлектроники

Поступила в редколлегию 25.06.2003

УДК 681.3.06

А. И. ШУМОВ

МЕТОДЫ ОЦЕНКИ НЕЛИНЕЙНОСТИ БУЛЕВЫХ ФУНКЦИЙ

В современных информационных системах важная роль отводится криптографическим методам защиты информации, среди которых одно из основных мест занимает блочное и поточное шифрование.

В настоящее время в Украине отсутствует стандарт поточного шифрования, а в качестве стандарта блочного шифрования используется ГОСТ 28147-89, разработанный и введенный в действие в конце 80-х годов прошлого века. Очевидно, что в нынешних условиях Украине необходима разработка и ввод в действие национальных стандартов симметричного блочного и поточного шифрования, обеспечивающих высокий уровень безопасности.

Для обеспечения стойкости будущих алгоритмов необходимо применение криптографически стойких нелинейных преобразований (таблиц подстановок), которые, как правило, строятся на основе булевых функций нелинейного типа. Однако следует отметить, что в открытой литературе практически отсутствуют отечественные публикации по методике оценки нелинейности и криптографической стойкости булевых функций. Среди зарубежных публикаций можно отметить работу Уэбстера и Тавареса [1], где впервые вводится концепция строгого лавинного критерия (СЛК); работы [2 – 5], в которых проводится обобщение и сведение к единой концепции критерия распространения; в [6, 7] описывается техника построения криптографически сильных функций, удовлетворяющих критериям сбалансированности, высокой нелинейности и критерию распространения высокой степени, разработанная Себери, Чжанем и Чженем.

Одним из подходов к построению нелинейных преобразований вида $GF(2)^n \to GF(2)^m$ (S-блоков) является использование критериев нелинейности булевых функций $f:GF(2)^n \to GF(2)$, когда использование булевых функций с хорошими криптографическими свойствами необходимо для обеспечения аналогичных свойств всего S-блока. Кроме того широко используется понятие линейной структуры S–блоков: $S:GF(2)^n \to GF(2)^m$ имеет линейную структуру, если существует ненулевой вектор $a \in GF(2)^n$ совместно с нетривиальным отображением $LS:GF(2)^n \to GF(2)$ таким, что $LS(x+a)+LS(x)$ принимает одно и то же значение (0 или 1) для всех $x \in GF(2)^m$.

Структура S может быть описана булевыми функциями в алгебраической нормальной форме

$$f^q(x_1, x_2 \cdots, x_n) = a_0 + \sum a_i x_i + \sum a_{ij} x_i x_j + \ldots + a_{12\ldots n} x_1 x_2 \cdots x_n, \; 0 \le q \le m-1. \qquad (1)$$

Функция f^q является нелинейной (или не аффинной), если ее алгебраическая нормальная форма (многочлен Жегалкина) содержит в себе термы степени выше первой.

В данной статье основное внимание уделяется обзору известных подходов к оценке показателей нелинейности булевых функций, таких как расстояние до аффинных функций, расстояние до линейных структур и порядок нелинейности.

Расстояние до аффинных функций

Расстояние до ближайшей аффинной функции [9] определяется как расстояние от f до множества A(n), в виде

$$\delta(f) = \min_{L \in A(n)} d(f, L), \qquad (2)$$

где $d(f,L)$ есть расстояние Хэмминга между f и L, а $A(n)$ – множество всех аффинных функций $L(x_1,...,x_n) = a_0 + a_1 x_1 + ... + a_n x_n$. Таким образом, $\delta(f)$ есть расстояние от f до множества A(n). Для дальнейшего изложения свойств показателя δ необходимо введение нескольких дополнительных понятий.

Пусть $\Omega(n)$ означает группу всех обратимых преобразований пространства $GF(2)^n$, и пусть $AGL(n)$ означает подгруппу всех аффинных преобразований. Известно, что элементы $AGL(n)$ могут быть описаны как функция $\alpha(x) = A(x) + a$, где A есть невырожденная матрица $n \times n$ и $a \in GF(2)^n$. Для представления пары (α,f) $f \in \Phi(n)$ и $\alpha \in \Omega(n)$ принято использовать обозначение $\alpha \bullet f$. В этих терминах операция группы $\Omega(n)$ над множеством $\Phi(n) = \{f : [GF(2)]^n \to GF(2)\}$ определяется в виде

$$\alpha \bullet f(x) = f(\alpha(x)). \tag{3}$$

Более общим является описание свойств нелинейности с помощью оценочной функции D

$$D : \Phi(n) \to W, \tag{4}$$

которая принимает значения из множества W. Функция f должна быть выбрана из $D(f)$, $D(f) \in W_1$, где множество $W_1 \subset W$, в свою очередь, содержит булевы функции с необходимыми криптографическими свойствами.

Существенно, чтобы значения D оставались инвариантными над преобразованиями из $\Omega(n)$, которые являются криптографически слабыми [9]. Обычно $\Omega(n)$ содержит все аффинные преобразования.

Для определения критерия нелинейности рассматривается максимальная из подгрупп $I(D) \subseteq \Omega(n)$, которая оставляет D инвариантным, т.е.

$$I(D) = \{\alpha \in \Omega(n) | D(\alpha \bullet f) = D(f) \text{ для } \forall f \in \Phi(n)\} \tag{5}$$

В этом случае $I(D)$ называется группой симметрий D. Критерии нелинейности связаны с описанием группы симметрий.

Пусть H будет подмножеством $\Phi(n)$ и для всех $f \in \Phi(n)$ выполняется равенство $d_H(f) = \min_{h \in H} d(f,H)$. Для δ это будет множество $A(n)$ всех аффинных функций. Кроме того, пусть $\Omega(n)^H$ будет подгруппой группы $\Omega(n)$, состоящей из элементов, удовлетворяющих условию

$$\Omega(n)^H = \{\alpha \in \Omega(n) | \alpha \bullet h \in H \text{ для всех } h \in H\}. \tag{6}$$

Эта подгруппа называется группой симметрий множества H. Она обладает следующим свойством [9]:

Для любого подмножества $H \subset \Phi(n)$ группа симметрий d_H совпадает с группой симметрий H, то есть

$$I(d_H) = \Omega(n)^H, \tag{7}$$

откуда следует, что группа симметрий $I(\delta)$ для минимума расстояний δ до множества аффинных функций L есть аффинная группа $AGL(n)$.

Поскольку множество аналитических атак используют аффинные свойства криптографических преобразований, то для обеспечения стойкости S-блоков необходимо использование множества булевых функций W_1, которые имеют максимум минимального расстояния до множества аффинных функций, причём критерий отбора функций в W_1 должен оставаться инвариантным для множества $\Omega(n)$, содержащего все аффинные преобразования.

Расстояние до линейных структур

Линейная структура булевой функции $f:GF(2)^n \to GF(2)$ определяется существованием вектора $a \in GF(2)^n$ такого, что выражение

$$f(x+a)+f(x) \tag{8}$$

принимает одно и то же значение (0 или 1) для всех $x \in GF(2)^n$. Пусть $LS(n)$ - множество булевых функций, имеющих линейную структуру. Заметим, что множество $A(n)$ всех аффинных функций содержит в себе $LS(n)$. Для булевой функции f расстояние до линейных структур определяется как расстояние от f до множества $LS(n)$:

$$\sigma(f) = d(f, LS(n)) = \min_{S \in LS(n)} d(f,s). \tag{9}$$

Расстояние до линейных структур может служить в качестве показателя нелинейности булевой функции, что вытекает из (7). Оттуда же следует, что группа симметрий $I(\sigma)$ содержит в себе аффинную группу $AGL(n)$, то есть имеет место включение $I(\sigma) \subset AGL(n)$ [8, 9].

Так как множество $A(n)$ всех аффинных функций содержит в себе $LS(n)$, то для отбора булевых функций с необходимыми криптографическими свойствами с учётом расстояния до линейных структур достаточно применения критериев, анализирующих аффинные свойства булевых функций.

Порядок нелинейности

Пусть для булевой функции $f \in \Phi(n)$ значение $O(f)$ будет степенью наивысшего порядка в алгебраической нормальной форме. В этом случае величина $O(f)$ называется порядком нелинейности f. Использование порядка нелинейности, определяемого функцией $O:\Phi(n) \to \{0,...,k,...,n\}$, удовлетворяет требованиям критерия нелинейности в силу следующего свойства: группа симметрий $I(O)$ функции O совпадает с аффинной группой $AGL(n)$ [8].

Показано [9], что другой критерий нелинейности, называемый расстоянием σ_k к функции с порядком нелинейности ограниченным k, также остается инвариантным над операцией из $AGL(n)$.

Для булевых функций, используемых в криптографических приложениях, должно выполняться условие $O(f) \geq 2$.

Совершенно нелинейные функции

Булеву функцию $f:GF(2)^n \to GF(2)$ называют совершенно нелинейной, если для каждого ненулевого вектора $a \in GF(2)^n$ значения функций $f(x+a)$ и $f(x)$ совпадают точно для половины аргументов $x \in GF(2)^n$.

Показано [8], что совершенно нелинейные функции достигают оптимума расстояния σ до линейных структур.

Для произвольной функции $f:GF(2)^n \to GF(2)$ расстояние до линейных структур вычисляется следующим образом. Пусть $a \in GF(2)^n$ есть ненулевой вектор. Тогда пространство $GF(2)^n$ может быть разбито на 2^{n-1} пар вида $(x, x+a)$. Через n_0 обозначают число элемен-

тов множества W_0 пар вида $(x, x+a)$, для которых $f(x)=f(x+a)$, а через n_1 – число элементов множества W_1 пар вида $(x, x+a)$, для которых $f(x)\neq f(x+a)$.

Любая булева функция может быть получена из произвольной булевой функции f путем изменения множества выходных значений. Таким образом, f может быть преобразована в функцию с линейной структурой изменением выходного значения либо x, либо $x+a$ для пары $(x, x+a)\in W_0$, или изменением соответствующего значения x или $x+a$ для пары $(x, x+a)\in W_1$. Отсюда для получения функции g с линейной структурой, такой, что $g(x)\neq g(x+a)$ для всех x должно быть изменено n_0 значений, и n_1 значений, чтобы получить функцию g с линейной структурой со свойством $g(x)\neq g(x+a)$ для всех x. Для генерации любой другой функции с такой же линейной структурой необходимо произвести, по крайней мере, $min(n_0,n_1)$ модификаций. Следовательно, $n=min(n_0,n_1)$ есть расстояние от функции f к ближайшим функциям с линейной структурой a. Заметим, что n зависит от вектора a, то есть $n=n_f(a)= min(n_0(a),n_1(a))$. Отсюда, расстояние от f до линейных структур определяется как

$$\sigma(f)=\min_{a\neq 0} n_f(a). \tag{10}$$

Поскольку $n_0(a)+n_1(a)=2^{n-1}$, то из (10) вытекает, что $n_f(a)\le 2^{n-2}$ для всех $a\neq 0$. Максимум расстояния достигается совершенно нелинейными функциями, так как они характеризуются равенством $n_0(a)=n_1(a)=2^{n-2}$ для $a\neq 0$ или, что эквивалентно, $\sigma(f)=2^{n-2}$. Это определяет существование следующего свойства [8].

Класс $\pi(n)$ совершенно нелинейных функций совпадает с классом функций с максимальным расстоянием 2^{n-2} к линейным структурам. Ниже описан класс функций, являющихся совершенно нелинейными.

Бент-функции

Рассмотрим зависимость между совершенно нелинейными функциями и бент-функциями. Поскольку эта зависимость выражается в терминах преобразования Уолша, то все булевы функции рассматриваются со значениями +1 и –1 (то есть значения $f(x)\in\{0,1\}$ заменяются на $(-1)^{f(x)}$).

Преобразование Уолша определено следующим образом:

$$F(W)=\sum_{x\in GF(2)^n} f(x)(-1)^{X\bullet W}, \tag{11}$$

где $W\in GF(2)^n$ и $X\bullet W$ – скалярное произведение в $GF(2)$.

Для булевой функции $f: GF(2)^n\to\{+1,-1\}$ и любого ненулевого вектора $a\in GF(2)^n$ имеет место равенство

$$\sum_{x\in GF(2)^n} f(x)f(x+a)=0. \tag{12}$$

Таким образом, ± 1 – значная функция f является совершенно нелинейной тогда и только тогда, когда для каждого ненулевого вектора $a\in GF(2)^n$, то есть тогда и только тогда, когда $f\bullet f$ является σ - функцией. В [9] отмечается, что функция $f\bullet f$ преобразуется в F^2 и σ -функция преобразуется в константу. Следовательно, ± 1 – значная булевая функция f является совершенно нелинейной, тогда и только тогда, когда $F(W)$ является константой для всех w. Так как $f\bullet f(O)=2^n$, то эта константа есть

$$F(w) = 2^{n/2}. \qquad (13)$$

Это свойство определяет совпадение класса совершенно нелинейных функций $\pi(n)$ и класса бент-функций.

Показано, что бент-функции существуют только для четных чисел аргументов и их порядок нелинейности не превосходит $\frac{n}{2}$ [10].

Для четного числа аргументов $n=2m$ бент-функции могут быть построены следующим образом [8]:

1) пусть $n=2m$. Тогда функция вида $f(x_1,..,x_n)=g(x_1,...,x_m)+x_1x_{m+1}+x_2x_{m+2}+...+x_{mn}$ является бент-функцией, где $g(x_1,...,x_m)$ – произвольная функция m-переменных.

2) пусть $x=(x_1,...,x_n)$ и пусть $a(x),b(x)$ и $c(x)$ будут бент функциями, такими что $a(x)+b(x)+c(x)$ есть также бент-функция. Тогда функция

$$f(x,x_{n+1},x_{n+2})=a(x)b(x)+b(x)c(x)+c(x)a(x)+[a(x)+b(x)]x_{n+1}+[a(x)+c(x)]x_{n+2}+x_{n+1}x_{n+2}$$

есть бент-функция. Требование, чтобы $a(x)+b(x)+c(x)$ являлось бент-функцией, легко удовлетворить, если взять $a(x),b(x)$ и $c(x)$ из п.1 или выбрать $a(x)=b(x)$ или $b(x)=c(x)$.

Правило п.1 описывает точные конструкции бент-функций, тогда как п.2 ведет к генерации новых совершенно нелинейных функций на основе произвольной совершенно нелинейной функции. Эта процедура может быть скомбинирована с линейными операциями на заданной совершенно нелинейной функции. Действительно, класс совершенно нелинейных функций является инвариантным над операцией аффинной группы $AGL(n)$. Кроме того, прибавление произвольной аффинной функции не влияет на совершенную нелинейность. Поэтому назначение к $f \in \Phi(n)$ функции $x \to f(\alpha(x)) + L(x)$ определяет операцию $AGL(n) \times A(n)$ на $\Phi(n)$, которая оставляет $\pi(n)$ инвариантным. В результате может быть построен рекуррентный алгоритм формирования совершенно нелинейных функций:

1. Для $n=2$ берется класс *(C2),* состоящий из всех функций с порядком нелинейности 2.

2. Для $n>2$ берутся любые функции a,b,c в $C(n-2)$ такие, что их сумма есть также в $C(n-2)$ и применяется конструкция п.2. Это определяет класс $C`(n)$ совершенно нелинейных функций. Класс $C`(n)$ является расширением класса $C(n)$, разрешая операции целой группы $G = AGL(n) \times A_n$ на $C(n)$.

Отметим, что п.1 предполагает существование по крайней мере $2^{2^{n/2}}$ совершенно нелинейных функций из всех 2^{2^n} булевых функций. Таким образом, только малая часть всех булевых функций является совершенно нелинейными, что значительно усложняет их нахождение. Уже для $n=6$ (то есть для *S*-блоков DES) поиск совершенно нелинейных функций при помощи полного перебора является фактически невозможным.

Однако совершенно нелинейные функции не являются сбалансированными, то есть $\left|\{f(x_0,...,x_n)=0 \,|\, x_0,...,x_n \in GF(2)\}\right| \neq \left|\{f(x_0,...,x_n)=1 \,|\, x_0,...,x_n \in GF(2)\}\right|$. Это означает, что если распределение элементов входной последовательности согласовано с равномерным законом распределения, для выходной последовательности функции такого согласования не будет. Это определяет невозможность использования совершенно нелинейных булевых функций в криптографических преобразованиях в чистом виде, однако применяемые функции должны быть максимально приближены по своим свойствам к совершенно нелинейным функциям.

Расстояние до аффинных функций и корреляция

Здесь мы приводим основные соображения, приведенные в [9]. Пусть $L_w(x) = w \bullet x$ означает произвольную линейную функцию. Тогда $(-1)^{w \bullet x}$ есть соответствующая ± 1- значная функция, которую также обозначим $L_w(x)$. Из определения Уолша [10] следует

$$F(w)=\left|\{x:f(x)=L_w(x)\}\right|-\left|\{x:f(x)\neq L_w(x)\}\right|=2^n-2d(f,L_w)\Big| \ ,$$

где d означает расстояние Хэмминга. Следовательно,

$$d(f,L_w) = 2^{n-1} - \frac{1}{2}F(w). \tag{14}$$

Известно [8], что для соответствующей аффинной функции $L_w = 1 + L_w$ расстояние d вычисляется как $d(f,L_w) = 2^{n-1} + \frac{1}{2}F(w)$. Формулу (14) предлагается использовать для нахождения лучшей аффинной аппроксимации к заданной функции нахождением w такого, чтобы $|F(w)|$ являлся максимумом, то есть

$$\delta(f) = 2^{n-1} - \frac{1}{2}\max_w |F(w)|. \tag{15}$$

Таким образом, как следует из (13), совершенно нелинейные функции всегда имеют расстояние до ближайших аффинных функций, равное

$$\delta(f) = 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{16}$$

При предположении, что f не является совершенно нелинейной, из теоремы Парсеваля следует [9]:

$$\sum_w F(w)^2 = 2^n \sum_x f(x)^2 = 2^{2n}, \tag{17}$$

где существует w с $|F(w)| > 2^{\frac{n}{2}}$. Показано, что $\delta(f) < 2^{n-1} - 2^{\frac{n}{2}-1}$ и, следовательно, f является более близкой к множеству всех аффинных функций, чем совершенно нелинейные функции. В результате делается вывод, что, совершенно нелинейные функции являются оптимальными не только в отношении к расстоянию до линейных структур, но и в отношении к расстоянию до всех аффинных функций, т.е. класс $\pi(n)$ совершенно нелинейных функций является классом функций с максимальным расстоянием $2^{n-1} - 2^{\frac{n}{2}-1}$ к аффинным функциям [8]. Показано, что этот результат может быть расширен до утверждения, что расстояние от совершенно нелинейной функции f к любой аффинной функции равно $2^{n-1} + 2^{\frac{n}{2}-1}$ или $2^{n-1} - 2^{\frac{n}{2}-1}$. Этот факт может быть точно выражен через корреляцию f к аффинным функциям. В общем случае, расстояние Хэмминга между двумя булевыми функциями $f, g : GF(2)^n \to \{+1,-1\}$ связано со взаимной корреляцией между f и g, которая определяется как

$$c(f,g)=\frac{\left|\{x:f(x)=g(x)\}\right|-\left|\{x:f(x)\neq g(x)\}\right|}{2^n}.$$

Для $g = L_w$ преобразование Уолша определяется следующим образом (смотри также (14)):

$$c(f,g)=\frac{F(w)}{2^n}. \tag{18}$$

На основе этого результата делается вывод, что абсолютная величина взаимной корреляции между совершенно нелинейной функцией и любой аффинной функцией есть константа, равная $2^{\frac{n}{2}}$. Кроме того, для функции g, которая не является совершенно нелинейной, всегда есть аффинная функция L с взаимной корреляцией $c(g,L)$ большей, чем $2^{\frac{n}{2}}$ в абсолютном значении. Совершенно нелинейные функции являются классом функций с минимальной корреляцией ко всем аффинным функциям [8].

Это свойство является противоположным по своей сути к корреляционному иммунитету [8]. Известно, что корреляционный иммунитет m-го порядка функции f удовлетворяет соотношению $F(w)=0$ для всех w с весом Хэмминга меньшим или равным m. Следовательно, для этих векторов взаимная корреляция $c(f, L_w)$ исчезает. С другой стороны, из теоремы Парсеваля следует, что для произвольной булевой функции f

$$\sum_{w \in GF(2)^n} c(f, L_w)^2 = 1. \qquad (19)$$

Это означает, что корреляция ко всем линейным (или аффинным) функциям не зависит от функции f. Таким образом, для функций с корреляционным иммунитетом исчезновение определенной взаимной корреляции неизбежно ведет к увеличению корреляции с другими аффинными функциями.

Известно, что взаимная корреляция $c(f,0)$ с нулевой функцией измеряется отклонением от ± 1-баланса булевой функции f. Как уже отмечено выше, совершенно нелинейная функция никогда не может быть сбалансированной. Вместе с тем, ее отклонение от сбалансированности, равное $2^{\frac{n}{2}}$, быстро стремится к нулю, когда n неограниченно возрастает. То же самое имеет силу для корреляции f с любой другой аффинной функцией.

Требование обеспечения минимальной корреляции входных и выходных значений противоречит требованию к сбалансированности выходных значений булевой функции. Совершенно нелинейные функции имеют минимальную корреляцию с аффинными функциями, однако являются несбалансированными. Как уже отмечено выше, необходим поиск функций, своими свойствами обеспечивающих компромисс между требованиями обеспечения сбалансированности, минимальной корреляции и нелинейности.

Рассмотренные показатели отражают современные подходы к оценке нелинейности булевых функций при построении симметричных криптографических алгоритмов. Для обеспечения стойкости преобразований булевы функции должны быть на максимальном расстоянии от множества аффинных функций и множества линейных структур, иметь высокий порядок нелинейности, обладать корреляционным иммунитетом, удовлетворять лавинному критерию и критерию распространения. Класс совершенно нелинейных функций имеет максимум минимального расстояния до аффинных функций и линейных структур, и совпадает с известным классом бент-функций. Однако совершенная нелинейность не может быть достигнута в сочетании со сбалансированностью или высоким порядком нелинейности.

Следует отметить, что для построения таблиц подстановок, применяемых в блочных и поточных алгоритмах шифрования, использования только критериев нелинейности булевых функций недостаточно. Например, в [13] предлагаются критерии построения узлов замены ГОСТ 28147-89 на основе применения булевых функций с заданными криптографическими свойствами, однако в [14] показано, что алгоритм шифрования с такими подстановками может быть уязвимым для дифференциального криптоанализа. Поэтому при построении блоков нелинейного преобразования для симметричных шифров наряду с критериями нелинейности булевых функций необходимо использовать и дополнительные критерии, анализирующие свойства всей подстановки в целом.

Список литературы: 1. *A.F.Webster and S.E. Tavares.* On the design of S-boxes. In Lecture Notes in Computer Science *218*; Advances in Cryptology: Proc. Crypto'*85*, H.C. Williams, Ed., Santa Barbara, CA, Aug. *18-22*, *1985*, pp. 523 – 534. Springer-Verlag, 1986. 2. *R. Forre.* The strict avalanche criterion: Special properties of boolean functions and an extended definition. In Lecture Notes in Computer Science 403; Advances in Cryptology: Proc. Crypto'88, pp. 450 – 468. Berlin: Springer-Verlag, 1990. 3. *C.M. Adams and S.E. Tavares.* The use of bent sequences to achieve higher-order strict avalanche criterion. Technical Report, TR 90-013, Department of Electrical Engineering, Queen's University, 1990. 4. *B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle.* Propagation characteristics of boolean functions. In Lecture Notes in Computer Science 473; Advances in Cryptology: Proc. Crypto'90, I. Damgard, Ed., Aarhus, Denmark, May 21-24. 1990, pp. 161 – 173. Berlin: Springer-Verlag. 5. *B. Preneel, R. Govaerts, and J. Vandewalle.* Boolean functions satisfying higher order propagation criteria. In Lecture Notes in Computer Science 547; Advances in Cryptology: Proc. Crypto'91, 1991, pp. 141 – 152. Berlin: Springer-Verlag. 6. *J. Seberry, X. M. Zhang, and Y. Zheng.* Nonlinearity balanced Boolean function and their propagation characteristics. In D.R. Stinson, editor, Advances in Cryptology – Crypto '93, pp. 49 – 60, Springler-Verlag, New York, 1994. 7. *J. Seberry, X.-M. Zhang, and Y. Zheng.* Nonlinearity and propagation Characteristics of Balanced Boolean Functions. Information and Computation, Vol. 119, No 1, pp. 1 – 13, 1995. 8. *W. Meier and O. Staffelbach.* Nonlinearity criteria for cryptographic functions. In Lecture Notes in Computer Science 434; Advanced in Cryptology; Proc. Eurocrypt'89, J.-J. Quisquater and J. Vandewalle, Eds., Houthalen, Belgium, April 10-23, 1989, pp. 549 – 562. Berlin: Springer-Verlag, 1990. 9. *А.В. Бабаш, Г.П. Шанкин.* Криптография. М.: СОЛОН-Р, 2002. 10. *O.S. Rothaus.* On bent functions. Journal of Combinatorial Theory (A), Vol. 20, pp. 300 – 305, 1976. 11. *C.E. Shannon.* Communications theory of secrecy systems. Bell Sys. Tech. Journal, Vol. 28, pp. 656 – 715, Oct. 1949. 12. *Донской В.И.* Дискретная математика. Симферополь: Изд. «СОНАТ», 2000. 13. *Холоша А.А.* Об одном подходе к анализу качества блока подстановки битовых векторов // Збірник наукових праць інституту проблем моделювання в енергетиці НАНУ. 1998. Вип. 2. С. 59 – 74. 14. *Р. Олейников, И. Лисицкая, А. Шумов.* Исследование свойств подстановок ГОСТ 28147-89, построенных на основе анализа свойств координатных функций. Правовое, нормативное и метрологическое обеспечение системы защиты информации в Украине. К.: ИВЦ «Политехника», 2002. Вып. 5.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 12.05.2003

УДК. 681.3.06:519.248.681

В. И. ДОЛГОВ, д-р. техн. наук, И. В. РУБАН, канд. техн. наук, С. В. ДУДЕНКО

ПОСТРОЕНИЕ НЕЛИНЕЙНЫХ СИСТЕМ НА ОСНОВЕ УСЕЧЕННОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ В КОНЕЧНЫХ ПОЛЯХ

В настоящий момент нелинейные преобразования в полях являются отдельной областью науки, представляющей не только теоретический, но и практический интерес. Они находят широкое применение в теории помехоустойчивого кодирования, в системах обработки информации и в последнее время широко используются при разработке криптографических протоколов.

Унитарные преобразования в полях относятся к классу теоретико-числовых преобразований, к этому классу можно отнести и преобразование Фурье в конечном поле. Следует заметить, что преобразование Фурье вектора длины n в поле $GF(q)$ существует только тогда, когда поле $GF(q)$ содержит элемент порядка n. Данный класс преобразований обладает свойством линейности. Интересным является тот факт, что при изменении порядка выполнения операций в теоретико-числовых преобразованиях связь между входным и выходным вектором становится нелинейной.

Представляется, что определенный научный и практический интерес может вызвать преобразование векторов, имеющих чётную длину, так как при этом повышается эффективность операций, выполняемых на современных микропроцессорах.

Вводится в рассмотрение усеченное преобразование Фурье (УПФ), которое в отличие от традиционного, существующего только для векторов длины n над полем Галуа, существует для векторов длины $n-1$. Данное преобразование при вычислениях в поле, являющемся расширением двоичного поля, имеет чётную длину всех входных векторов.

1 Преобразование Фурье над полем Галуа *GF (q)*

Для того чтобы ввести математические обозначения и облегчить восприятие материала, изложенного в статье, напомним определение и свойства преобразования Фурье в конечном поле, введенные Р. Блейхутом [1].

Определение. Пусть $v = \{v_i,\ i = 0, \ldots, n-1\}$ – вектор над $GF(q)$, где n делит $q^m - 1$ при некотором m, и пусть w – элемент порядка n в поле $GF(q^m)$. Преобразование Фурье в поле Галуа вектора v определяется как вектор $V = \{V_j,\ j = 0, \ldots, n-1\}$, задаваемый равенствами

$$V_j = \sum_{i=0}^{n-1} w^{ij} \cdot v_i, \qquad j = 0, \ldots, n-1,$$

где v_i – i-я точка входного вектора; V_j – j-я точка выходного вектора.

В качестве длины преобразования Фурье можно выбрать произвольный делитель числа $q^m - 1$, но наиболее важную роль играют примитивные длины $n = q^m - 1$. В последнем случае w является примитивным элементом поля $GF(q^m)$.

В [1] также приведена и доказана следующая теорема.

Над полем GF (q) характеристики p вектор и его спектр связаны соотношениями

$$V_j = \sum_{i=0}^{n-1} w^{ij} \cdot v_i, \qquad v_i = \left(1/n\right) \sum_{j=0}^{n-1} w^{-ij} \cdot V_j, \tag{1}$$

где n интерпретируется как число поля, т. е. по модулю p.

Например, в поле $GF(2^8)$ преобразование Фурье существует для $n = 3,5,15,17,51,85,255$. Для многих целей таких длин векторов оказывается достаточно.

Отметим два свойства преобразования Фурье, которые накладывают ограничения на его использование.

Первое заключается в том, что при проведении вычислений в поле, являющемся расширением двоичного поля, на вход преобразования Фурье мы можем подавать только векторы, имеющие нечётную длину. Любое число в формате ЭВМ может быть представлено в виде суммы степеней двойки. В связи с этим наибольший интерес вызывают именно преобразования в полях, являющихся расширением двоичного поля. Используемые же в современных ЭВМ микропроцессоры имеют разрядность 8, 32 и 64. Поэтому это свойство при программной реализации преобразования Фурье является недостатком, так как мы не можем в некоторых случаях полностью оптимизировать работу программы под аппаратную платформу. Особенно актуально вопрос о длине блоков, которые требуется преобразовать, стоит перед разработчиками криптографических протоколов, поскольку на современном уровне развития вычислительной техники система, стойкая к атакам криптоаналитика, должна поддерживать размер блока базовой функции шифрования, равный 128, 192, 256 бит.

Второе свойство – линейная связь входного вектора с выходным – является недостатком при использовании преобразования Фурье в криптографических целях. Так, например, в [2] Шнорр предлагает структуру хэш-функции, основанную на преобразовании Фурье в поле *GF* (65537), порядок которого есть простое число Ферма. Известно, что если результат сложения и умножения чисел не превышает значения модуля в таких полях, то соответствующие операции оказываются линейными. В данном случае линейность преобразования Фурье дала возможность доказать нестойкость такой конструкции к коллизиям [3].

Цель настоящей работы заключается в устранении указанных недостатков.

2 Способ достижения нелинейности теоретико-числовых преобразований

Идея способа может быть пояснена на примере достижения нелинейности преобразования Фурье. Обязательным условием является вычисления в поле, являющемся расширением двоичного поля.

Пусть имеется входной вектор

$$\boldsymbol{v} = \{v_0, v_1, \ldots, v_{n-1}\},$$

где $v_i \in GF(2^m)$ и w- элемент порядка n в $GF(2^m)$ при условии, что $n|(2^m-1)$.

Тогда первая точка выходного вектора может быть записана как

$$V_0 = OS\,[S(w^0 \otimes v_0) \oplus S(w^0 \otimes v_1) \oplus \ldots \oplus S(w^0 \otimes v_{n-1})], \qquad (2)$$

где $\otimes$ – операция умножения в поле, которая является линейной;

$\oplus$ – операция побитного ИСКЛЮЧАЮЩЕГО ИЛИ, которая также является линейной;

$S(v_i)$ – переход от десятичного представления элемента поля к двоичному представлению;

$OS(v_i)$ – переход от двоичного представления элемента поля к десятичному представлению.

Переходы $S(v_i)$ и $OS(v_i)$ являются нелинейными, однако в связи с тем, что они биективны (т.е. взаимообратны) и в выражении (2) применяются последовательно один за другим, то связь входного и выходного векторов оказывается линейной.

Суть достижения нелинейности состоит в изменении порядка выполнения операций в выражении (1). При этом первой точкой выходного вектора считается точка

$$V_0 = S(w^0 \otimes v_0) \oplus S(w^0 \otimes v_1) \oplus \ldots \oplus S(w^0 \otimes v_{n-1}), \qquad (3)$$

а переход $OS(v_i)$ осуществляется только при обратном преобразовании. В этом случае связь входного и выходного векторов для преобразования Фурье нельзя будет описать линейной

функцией в силу нелинейности перехода $S(\nu_i)$. Приведенные результаты говорят о возможности достижения нелинейности преобразования Фурье, однако ограничение на длины входных векторов остается в силе.

В третьем разделе рассмотрим унитарное преобразование, которое в полях характеристики два позволяет работать с векторами, имеющими чётную длину.

3 Усеченное преобразование Фурье (УПФ) над полем Галуа *GF (q)*

Прежде всего докажем теорему о существовании усеченного унитарного преобразования в конечном поле. Доказательство строится на основе выражения (1) путем удаления из традиционного преобразования Фурье нулевой компоненты (точки) входного вектора. Оно далее названо усеченным преобразованием Фурье. Обозначим символом $\Phi_l(\nu)$ – операцию усеченного преобразования входного вектора ν длины l. Результатом преобразования есть вектор длины l, который будем называть выходным вектором.

Теорема. *Над полем GF (q) характеристики* p *существует унитарное преобразование вида:*

$$V_j = \sum_{i=1}^{n-1} w^{ij} \cdot \nu_i, \tag{4}$$

$$\nu_i = \left(\frac{1}{n \bmod p}\right) \sum_{j=1}^{n-1} \left(w^{-ij} \oplus L\right) \cdot V_j, \tag{5}$$

где w *– элемент порядка* n *в поле GF (q);*
$\oplus$ – означает операцию сложения в поле; $L = -1$.

Доказательство. В поле *GF (q)* справедливо $\sum_{i=0}^{n-1} w^{ir} = 1 \oplus \sum_{i=1}^{n-1} w^{ir} = \begin{cases} n \bmod p, & \text{если } r = 0, \\ 0, & \text{если } r \neq 0 \end{cases}$

и следовательно $\sum_{i=1}^{n-1} w^{ir} = \begin{cases} n \bmod p \oplus L, & \text{если } r = 0, \\ L, & \text{если } r \neq 0. \end{cases}$ (6)

Подставив выражение (4) в (5), получим

$$\nu_i = \left(\frac{1}{n \bmod p}\right) \sum_{j=1}^{n-1} \left[\left(w^{-ij} \oplus L\right) \cdot \sum_{k=1}^{n-1} w^{jk} \cdot \nu_k \right] =$$

$$= \left(\frac{1}{n \bmod p}\right) \cdot \sum_{j,k=1}^{n-1} \left[\nu_k \cdot w^{jk} \cdot \left(w^{-ij} \oplus L\right) \right] = \left(\frac{1}{n \bmod p}\right) \cdot \sum_{j,k=1}^{n-1} \nu_k \cdot \left(w^{(k-i)j} \oplus w^{jk} \cdot L\right) =$$

$$= \left(\frac{1}{n \bmod p}\right) \cdot \left[\sum_{\substack{j=1 \\ k=i}}^{n-1} \nu_i \cdot \left(w^{(k-i)j} \oplus w^{jk} \cdot L\right) \oplus \sum_{\substack{j=1 \\ k \neq i}}^{n-1} \sum_{k=1}^{n-1} \nu_k \cdot \left(w^{(k-i)j} \oplus w^{jk} \cdot L\right) \right].$$

Воспользовавшись теперь соотношением (6), можем прийти к результату

$$\left(\frac{1}{n \bmod p}\right)\cdot\left[\sum_{\substack{j=1\\k=i}}^{n-1} v_i\cdot\left(w^{(k-i)j}\oplus w^{jk}\cdot L\right)\oplus\sum_{\substack{j=1\\k\neq i}}^{n-1}\sum_{k=1}^{n-1} v_k\cdot\left(w^{(k-i)j}\oplus w^{jk}\cdot L\right)\right]=$$

$$=\left(\frac{1}{n \bmod p}\right)\cdot\left[v_i\cdot\left[\sum_{j=1}^{n-1} w^{(k-i)j}\oplus L\cdot\sum_{j=1}^{n-1} w^{jk}\right]\oplus\sum_{\substack{k=1\\k\neq i}}^{n-1} v_k\cdot\left(\sum_{j=1}^{n-1} w^{(k-i)j}\oplus\sum_{j=1}^{n-1} w^{jk}\cdot L\right)\right]=$$

$$=\left(\frac{1}{n \bmod p}\right)\cdot\left[v_i\cdot\left[n \bmod p\oplus L\oplus L^2\right]\oplus\sum_{\substack{k=1\\k\neq i}}^{n-1} v_k\cdot\left(L\oplus L^2\right)\right]=v_i\cdot$$

Последнее следует из того, что $L \oplus L^2 = 0$. **Теорема доказана.**

Легко убедиться теперь, что использование данного преобразования в полях, являющихся расширением двоичного поля, в отличие от преобразования Фурье позволяет подавать на вход УПФ векторы, имеющие четную длину. Например, в поле $GF(2^8)$ преобразование (4) существует для векторов длины = 2,4,14,16,50,84,254, что соответствует двоичным блокам входных данных 16, 32, 112, 128, 400, 672 и 2032 бит. Блок длиной в 32 бита может быть обработан на 32-разрядном микропроцессоре за один такт, т.е. такие длины уже позволяют оптимально использовать существующую элементную базу вычислительной техники для программной и аппаратной реализации УПФ.

Интересным на наш взгляд является также то, что существует два способа нахождения выходного вектора для $\boldsymbol{\Phi_l(v)}$ в $GF(2^m)$:

1. Выполнение преобразования согласно выражению (4).
2. Табличный способ.

Первый способ заключается в выполнении всех математических операций, определенных выражением (4). В этом случае часто пользуются матричным представлением выполняемых преобразований. Так, например преобразование $\boldsymbol{\Phi_4(v)}$ в $GF(2^8)$ может быть записано выражением вида

$$\begin{vmatrix} V_1\\ V_2\\ V_3\\ V_4\end{vmatrix} = \begin{vmatrix} 52 & 103 & 154 & 205\\ 103 & 205 & 52 & 154\\ 154 & 52 & 205 & 103\\ 205 & 154 & 103 & 52\end{vmatrix}\cdot\begin{vmatrix} v_1\\ v_2\\ v_3\\ v_4\end{vmatrix}.$$

В соответствии с предлагаемым способом для входного вектора $v = \{2, 45, 178, 236\}$ выходной вектор будет иметь вид $V = \{53, 217, 187, 73\}$.

Прежде всего заметим, что операция сложения в поле, являющемся расширением двоичного поля – есть операция побитного ИСКЛЮЧАЮЩЕГО ИЛИ, а каждая точка входного вектора, умноженная на константу, влияет на все точки выходного вектора. Суть табличного способа нахождения выходного вектора для $\boldsymbol{\Phi_l(v)}$ в $GF(2^m)$ заключается в том, что строится $\boldsymbol{l}$ таблиц, состоящих из $\boldsymbol{2^m}$ элементов размерностью $\boldsymbol{m \cdot l}$ бит, а выходной вектор получается путем сложения элементов таблиц, соответствующих точкам входного вектора.

Например для $\Phi_4(v)$ в $GF(2^8)$ таблицы будут иметь вид

$$T_1[v_1]=\begin{bmatrix} S(52\cdot v_1) \\ S(103\cdot v_1) \\ S(154\cdot v_1) \\ S(205\cdot v_1) \end{bmatrix}, \quad T_2[v_2]=\begin{bmatrix} S(103\cdot v_2) \\ S(205\cdot v_2) \\ S(52\cdot v_2) \\ S(154\cdot v_2) \end{bmatrix},$$
$$T_3[v_3]=\begin{bmatrix} S(154\cdot v_3) \\ S(52\cdot v_3) \\ S(205\cdot v_3) \\ S(103\cdot v_3) \end{bmatrix}, \quad T_4[v_4]=\begin{bmatrix} S(205\cdot v_4) \\ S(154\cdot v_4) \\ S(103\cdot v_4) \\ S(52\cdot v_4) \end{bmatrix}, \tag{6}$$

где $S(v)$ означает нелинейный переход от десятичного представления элемента поля к двоичному представлению. Выходной вектор V может быть получен как

$$V = T_1[v_1] \oplus T_2[v_2] \oplus T_3[v_3] \oplus T_4[v_4] \tag{7}$$

Заметим, что для хранения таблиц (6) требуется 4 кБайта памяти, а выражение (7) можно рассматривать как преобразование входного блока размера 32 бита в выходной блок такой же размерности. Для сравнения в случае, когда необходимо построить полную таблицу замены 32-битного блока, потребуется $2^{32} \cdot 32$ бита памяти, что соответственно равно 16 Гбайтам.

Способ достижения нелинейности теоретико-числовых преобразований (описанный выше) может быть использован и для достижения нелинейности УПФ. Учитывая, что программная и аппаратная реализации УПФ могут быть оптимизированы под использование их на современных микропроцессорах и в тоже время УПФ позволяет производить нелинейную замену блоков при ограниченном размере памяти, на наш взгляд, УПФ может быть полезно при построении криптографических протоколов.

4 Использование усеченного преобразования Фурье над полем Галуа *GF (q)* в системах криптографической защиты информации

В разных источниках приводятся различные подходы к классификации блочных шифров, но чаще всего выделяют два основных класса: шифры, построенные с использованием цепи Фестеля (DES, DEAL, E2, LOKI97, RC6, Twofish, MARS, Гост 28147- 89), и шифры, которые построены на основе чередования процедур перестановок и подстановок (SPN – substitution-permutation network) (Square, Rijndael, SAFER+, Serpent, CRYPTON). При этом особенностью цепи Фестеля является то, что она дает возможность использовать на одном цикле необратимые операции в отличие от SPN – структуры, где биективность преобразований основное требование, в связи с тем, что процесс дешифрования состоит из операций, которые являются обратными к используемым операциям при шифровании. УПФ является биективным преобразованием, поэтому наиболее целесообразно будет его использование при проектировании симметричных блочных шифров на основе SPN – структуры.

Один цикл криптографического преобразования в шифре, построенном на основе SPN – структуры, может быть представлен тремя последовательно выполняемыми шагами:

1. Нелинейная замена входного блока.
2. Транспозиция входного блока.
3. Сложение с ключом.

Следует отметить, что стойкость такой конструкции к атакам криптоаналитика полностью зависит от вводимой в цикле нелинейности, так как основой построения и реализации таких атак является использование характеристик, описывающих вероятность прохождения через циклы шифрования специфических пар открытых текстов. И если удается найти характеристику с высокой вероятностью, то можно ставить и решать задачи криптоанализа со сложностью меньшей, чем прямой перебор ключей.

Построению S-блоков (таблиц нелинейной замены) посвящено очень много работ как отечественных, так и мировых ученых. Наиболее прогрессивным подходом в этом направлении можно выделить использование для построения S-блоков бент-функций, которые являются максимально нелинейными в криптографическом смысле. Однако такой подход имеет одно ограничение, заключающееся в том, что для хранения S-боксов больших размеров (осуществляющих замену блока размера 32 и более бит) требуется недостижимый объем памяти. Как показывает анализ симметричных блочных шифров, которые были предложены в качестве кандидатов на новый стандарт в конкурсах AES и NESSIE, размер блока данных, обрабатываемый одним оператором, функции шифрования в одном цикле равен 4 (Rijndael), либо 16 байт (RC6). При этом такой размер блока для функции шифрования достигается за счет использования на шаге транспозиции входного блока линейных преобразований вместо битовых перестановок (DES) или битовых сдвигов (ГОСТ 28147-89). Сами S-блоки имеют небольшой размер, как правило, 8×8 бит. Среди линейных преобразований широкое применение получили преобразования на основе МДР-кодов. Подобные преобразования используются в шифрах Shark, Square, Rijndael, Khazad, Anubis. Главное преимущество линейных преобразований этого вида заключается в том, что число задействованных S-блоков в контексте дифференциального или линейного криптоанализа ("количество ветвлений") до и после такого линейного преобразования будет максимально возможным, т.е. равным $M+1$, где M – число S-блоков, покрываемых МДР-преобразованием.

Например, в симметричном блочном шифре AES-Rijndael [5] нелинейность обеспечивается за счет последовательного выполнения двух операций: замены элемента на его мультипликативно обратный элемент в поле $GF(2^8)$ и аффинного преобразования. Но при этом активизация одного S-блока на входе цикла за счет использования МДР-преобразования приводит к тому, что на следующем цикле задействованными являются уже четыре S-блока, что не позволяет криптоаналитику работать с одним активным S-блоком на каждом цикле и соответственно затрудняет задачу проведения криптоанализа.

На последнем этапе конкурса AES все кандидаты отмечаются как устойчивые к известным методам криптоанализа, но, согласно результатам исследований, опубликованных в [8], Rijndael имеет не лучший, в сравнении с остальными, показатель статистической безопасности. На наш взгляд выбор шифра AES-Rijndael в качестве нового стандарта во многом определила возможность его реализации на любой аппаратно-программной платформе вычислительной техники. Наилучший временной показатель обеспечивается при использовании 32-битных микропроцессоров.

Рассмотрим, каким образом удается достичь оптимизации при использовании 32-битного микропроцессора для шифрования одного блока в AES-Rijndael. В этом случае разработчики стандарта используют цикловое преобразование, описываемое выражением

$$V_j = T_1[v_1] \oplus T_2[v_2] \oplus T_3[v_3] \oplus T_4[v_4] \oplus K_j,$$

где K_j – ключ цикла;

$T_i(v_i)$ – таблицы вида: $T_1[v_1]=\begin{bmatrix} 2\cdot S(v_1) \\ 1\cdot S(v_1) \\ 1\cdot S(v_1) \\ 3\cdot S(v_1) \end{bmatrix}$, $T_2[v_2]=\begin{bmatrix} 3\cdot S(v_2) \\ 2\cdot S(v_2) \\ 1\cdot S(v_2) \\ 1\cdot S(v_2) \end{bmatrix}$,

$$T_3[v_3]=\begin{bmatrix} 1\cdot S(v_3) \\ 3\cdot S(v_3) \\ 2\cdot S(v_3) \\ 1\cdot S(v_3) \end{bmatrix}, \quad T_4[v_4]=\begin{bmatrix} 1\cdot S(v_4) \\ 1\cdot S(v_4) \\ 3\cdot S(v_4) \\ 2\cdot S(v_4) \end{bmatrix},$$

где $S(v)$ означает нелинейную замену байта, определенную для данного стандарта. При этом время выполнения такого преобразования будет зависеть от времени осуществления четырех поисков по таблице и времени выполнения трех операций сложения. Заметим, что складываются 32-битные числа, а их сложение на 32-битном микропроцессоре выполняется за один такт.

Обратим внимание на то, что такая же схема реализуется и для нахождения выходного вектора в УПФ: выражения (6) – таблицы УПФ и выражение (7) – вычисление $\Phi_4(v)$ в $GF(2^8)$. Следовательно, мы можем использовать УПФ для построения симметричного блочного шифра по той же схеме, что использовалась разработчиками шифра Rijndael. Стойкость такого шифра к известным видам криптоанализа будет зависеть от введенной в УПФ нелинейности.

Нелинейность в УПФ определяется переходом от десятичного представления элемента поля к двоичному представлению. Так как поле строится на основе регистра сдвига с обратной связью и при этом допускается использование любого примитивного многочлена над данным полем, то вариантов построения, а, следовательно, и переходов будет несколько. Например, поле $GF(2^8)$ может быть построено на основе восьми примитивных многочленов, полный список которых из [7] представлен в табл. 1.

Таблица 1

№ п/п	Коэффициенты								
	X^8	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
1.	1	1	1	1	0	0	1	1	1
2.	1	0	1	1	0	0	0	1	1
3.	1	0	0	0	1	1	1	0	1
4.	1	0	0	1	0	1	0	1	1
5.	1	0	1	0	1	1	1	1	1
6.	1	0	1	1	0	1	0	0	1
7.	1	0	1	1	0	0	1	0	1
8.	1	1	1	0	0	0	0	1	1

Учитывая, что величина сдвига в регистре может варьироваться, количество вариантов построения поля будет еще больше. Поле $GF(2^{16})$ мы можем построить с использованием уже порядка 1000 примитивных многочленов [7].

При исследовании стойкости симметричного блочного шифра используют максимальные значения таблиц распределения дифференциалов и линейных аппроксимаций. Для примитивных многочленов из табл. 1 и величине сдвига, равной одному биту, соответствующие

им максимальные значения дифференциалов ($NS_D(\alpha,\beta)$) и линейных аппроксимаций ($NS_{LA}(\alpha,\beta)$) представлены в табл. 2. В табл. 2 также показаны соответствующие значения для нелинейных преобразований, используемых в Rijndael и Safer++. При нахождении значений *шифра Rijndael использовались таблицы нелинейной замены [5].*

Таблица 2

№ п/п	Вид используемого нелинейного преобразования	$NS_D(\alpha,\beta)$	$NS_{LA}(\alpha,\beta)$
1.	В поле, построенном на многочлене 1	[210,50] = 8	[222,90] = 30
2.	В поле, построенном на многочлене 2	[171,5] = 8	[187,251] =38
3.	В поле, построенном на многочлене 3	[217,49] = 8	[102,13] = 28
4.	В поле, построенном на многочлене 4	[85,149] =10	[110,14] = 28
5.	В поле, построенном на многочлене 5	[107,66] = 8	[255,16] = -32
6.	В поле, построенном на многочлене 6	[170,63] = 14	[102,255] = 30
7.	В поле, построенном на многочлене 7	[204,9] = 8	[238,207] = 30
8.	В поле, построенном на многочлене 8	[153,14] = 8	[255,155] = 30
9.	Rijndael	[40,9] = 6	[207,82] = 23
10.	Safer++ ($(45^X \bmod 257) \bmod 256$)	[128,253] = 128	[58,80] = -46
11.	Мультипликативно обратный элемент в поле GF (2^8)	[1,1] = 256	[2,2] = 128
12.	Мультипликативно обратный элемент в поле GF (2^8+1)	[253,253] = 72	[126,1] = -34

Из табл. 2 видно, что нелинейное преобразование на основе регистра сдвига с обратной связью имеет хорошие линейные и дифференциальные характеристики и допускает использование УПФ для нелинейной замены блоков в произвольном блочном шифре. Наилучшие характеристики достигаются при использовании примитивного многочлена вида

$$X^8 + X^4 + X^3 + X^2 + 1.$$

Экспериментальные исследования преобразования $\boldsymbol{\Phi_4(v)}$ в $GF(2^8)$ показали также, что число задействованных нелинейных переходов применительно к дифференциальному или линейному криптоанализу до и после такого преобразования равно 5 , т.е. максимально возможное. Это говорит о том, что УПФ в данном применении обеспечивает те же свойства, что и МДР-преобразование.

При исследовании статистической безопасности для $\boldsymbol{\Phi_4(M)}$ и $\boldsymbol{\Phi_{16}(M)}$ в $GF(2^8)$ использовался следующий алгоритм:

1. Генерируется исходный блок $\boldsymbol{M_1}$ (нулевой, единичный или случайный).
2. Находится $\boldsymbol{C_1 = \Phi_1(M_1)}$.
3. Определяется блок $\boldsymbol{M_2 = M_1}$ **xor** $\boldsymbol{G}$, где $\boldsymbol{G}$ в цикле изменяет $\boldsymbol{M_1}$ в $\boldsymbol{g}$ битовых позициях.
4. Находится $\boldsymbol{C_2 = \Phi_1(M_2)}$.
5. Вычисляется расстояние Хэмминга $\boldsymbol{W(C_2, C_1)}$. Переходим на шаг 3.

Группированный статистический ряд для $W(C_i, C_j)$ как случайной величины при использовании $\Phi_4(M_l)$ в $GF(2^8)$, где M_i (случайный) и g в диапазоне [1÷5 бит], представлен в табл. 3.

Таблица 3

Интервал	Частота измеренная	Частота совокупная	Ожидаемая частота	Ожид. частота совокуп.	Разность
0	0	0	0	0	0
2	0	0	0	0	0
4	0	0	0	0	0
6	0	0	0	0	0
8	3	3	2	2	1
10	9	12	12	14	-3
12	37	49	46	61	-9
14	116	165	104	164	12
16	130	295	144	308	-14
18	129	424	125	433	4
20	73	497	67	500	6
22	27	524	22	522	5
24	2	526	4	527	-2
26	1	527	0	527	1
28	0	527	0	527	0
30	0	527	0	527	0

Проверка гипотезы о виде распределения $W(C_i, C_j)$ (на основе критерия Пирсона [6]) для результатов, представленных в табл. 2, показала, что распределение расстояния Хэмминга $W(C_i, C_j)$ как случайной величины подчинено биномиальному закону. Описательная статистика для $W(C_i, C_j)$ при использовании $\Phi_4(M_l)$ в $GF(2^8)$ представлена в табл. 4.

Таблица 4

Параметр	M_l (случайный) и g в диапазоне [1÷5 бит]	M_l (случайный) и g в диапазоне [7÷24 бит]	M_l (нулевой) и g в диапазоне [1÷5 бит]	M_l (единичный) и g в диапазоне [7÷23 бит]
Среднее	15,89184061	15,944	15,96875	16,032
Стандартная ошибка	0,122839257	0,12706699	0,04157825	0,123598159
Медиана	16	16	16	16
Стандартное отклонение	2,819957027	2,841304281	2,35202087	2,763738863
Дисперсия выборки	7,952157632	8,07301002	5,53200219	7,638252505
Интервал	17	16	11	16
Минимум	8	8	11	8
Максимум	25	24	22	24
Сумма	8375	7972	8012	8016
Количество опытов	527	500	500	500

Из результатов, представленных в табл. 4, видно, что при условии отличия двух входных векторов в g битовых позициях для $\Phi_4(M_l)$ в поле $GF(2^8)$ получаются выходные векторы, отличающиеся в среднем в половине битовых позиций, что говорит о хороших свойствах

статистической безопасности для данного преобразования. При увеличении количества циклов преобразования, т.е. когда выходной вектор подается на вход УПФ несколько раз, полученные характеристики улучшаются.

Группированный статистический ряд расстояний Хэмминга двух выходных векторов как случайной величины для $\Phi_{16}(M_i)$ в $GF(2^8)$, где M_i (случайный) и g в диапазоне [1÷5 бит], представлен в табл. 5.

Таблица 5

Интервал	Частота измеренная	Частота совокупная	Ожидаемая частота	Ожид. частота совокуп.	Разность
50	100	100	1	1	99
52	0	100	6	7	-6
54	600	700	33	40	567
56	800	1500	134	174	666
58	500	2000	431	606	69
60	1000	3000	1072	1678	-72
62	1700	4700	2027	3705	-327
64	2000	6700	2849	6555	-849
66	1800	8500	2898	9453	-1098
68	1200	9700	2055	11508	-855
70	1100	10800	965	12473	135
72	1200	12000	279	12752	921
74	400	12400	44	12797	356
76	100	12500	3	12800	97
78	300	12800	0	12800	300

В ходе проведения экспериментов наблюдалось группирование значений $W(C_i, C_j)$ в интервале [50, 80] бит. В данном применении УПФ этот результат является положительным, так как одно из требований к базовой функции шифрования заключается в том, чтобы минимальные изменения входного блока вызывали максимальные изменения в выходном. Описательная статистика для $W(C_i, C_j)$ при использовании $\Phi_{16}(M_1)$ в $GF(2^8)$ представлена в табл. 6.

Таблица 6

Параметр	M_1 (случайный) и g в диапазоне [1÷5 бит]	M_1 (нулевой) и g в диапазоне [1÷5 бит]	M_1 (единичный) и g в диапазоне [1÷5 бит]
Среднее	64,2890625	63,8875	64,78125
Стандартная ошибка	0,050622545	0,04757158	0,045541853
Медиана	64	64	64,5
Стандартное отклонение	5,727287143	5,382109884	5,152472465
Дисперсия выборки	32,80181801	28,96710681	26,5479725
Интервал	28	25	17
Минимум	50	54	56
Максимум	78	79	73
Сумма	822900	815200	829200
Количество опытов	12800	12800	12800

Из табл. 6 можно увидеть, что $\Phi_{16}(M_1)$, как и $\Phi_4(M_1)$ в поле $GF(2^8)$, представленное в табл. 4, имеет хорошую статистическую безопасность.

Выводы

1. Предложен способ достижения нелинейности теоретико-числовых преобразований в полях характеристики 2.
2. Установлено, что нелинейность усеченного преобразования Фурье зависит от используемого для построения поля примитивного многочлена и величины сдвига в формирующем регистре.
3. Показано, что усеченное преобразование Фурье позволяет производить нелинейную замену блоков большой длины при ограниченном размере памяти. Так для организации нелинейной замены блока размера 128 бит требуется 2^{19} бит памяти, для сравнения стандартный подход требует $2^{128} \cdot 128 = 2^{135}$ бит памяти.
4. Подтверждено, что использование усеченного преобразования Фурье в поле $GF(2^8)$ для построения симметричного блочного шифра на основе SPN-структуры позволяет получить показатель статистической безопасности и цикловые дифференциальные и линейные характеристики, соизмеримые с соответствующими показателями кандидатов на новый стандарт в конкурсах AES и NESSIE.

Список литературы: 1. *Блейхут Р.* Теория и практика кодов, контролирующих ошибки. М.: Мир, 1986. 576 с. 2. *Schnorr C.P.* FFT-Hash 2, Efficient Cryptographic Hashing // Advances in cryptology – Eurocrypt'92. 1992. P. 45 – 54. 3. *Vaudenay S.* FFT-Hash 2 is not yet Collision-free // Proc. Of Crypto'92. 1992. P. 587 – 593. 4. *Ященко В.В.* О критерии распространения для булевых функций и бент-функций // Проблемы передачи информации. 1997. Вып. 1. С. 52 – 63. 5. *Горбенко И.Д., Скрыпник Л.В., и др.* Стандарт симметричного шифрования 21 века: свойства, режимы работы, реализация // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 119. С. 22 – 35. 6. *Гмурман В.Е.* Теория вероятностей и математическая статистика. М.: Высш. шк., 1977. 479 с. 7. *Питерсон У., Уэлдон Э.* Коды, исправляющие ошибки: Пер. с англ. М.: Мир, 1976. 600 с. 8. *Бондаренко М.Ф., Горбенко И.Д. и др.* Улучшенный стандарт симметричного шифрования XXI века: концепция создания и свойства кандидатов // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2000. Вып. 114. С. 5 – 14.

Харьковский национальный
университет радиоэлектроники
Харьковский военный университет

Поступила в редколлегию 15.01.2003

МЕТОДЫ, ПРОТОКОЛЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

УДК 681.3.06

И. Д. ГОРБЕНКО, д-р техн. наук, Д. С. БАЛАГУРА

ИССЛЕДОВАНИЕ СВОЙСТВ И ВЫБОР ПАРАМЕТРОВ СХЕМ ШИФРОВАНИЯ, РЕАЛИЗОВАННЫХ НА ОСНОВЕ ПРОТОКОЛОВ ДИФФИ-ХЕЛЛМАНА

В различных коммерческих и банковских технологиях значительное распространение получают криптографические преобразования, получившие название направленных шифров. Такие шифры используются для реализации состоятельных протоколов управления ключами, аутентификации, разделения секрета и др. Криптографические преобразования могут строиться на основе таких схем: схема направленного шифрования RSA, которая представлена на рис. 1 (на рис. 1: M – сообщение; O_O – открытый ключ получателя; $Д_П$ – личный ключ получателя), схема направленного шифрования Эль-Гамаля [1], комбинированные схемы и схемы направленного шифрования, базирующиеся на примитиве Диффи-Хеллмана [2]. К последним относятся схемы с преобразованиями в полях, кольцах и группе точек эллиптических кривых [3]. Наибольшее распространение получают схемы направленного шифрования на основе примитива Диффи-Хеллмана с преобразованиями в группе точек эллиптических кривых. Это объясняется меньшей сложностью таких преобразований и, как следствие, более высокой скоростью при сохранении стойкости. Сохраняя же скорость преобразований, можно существенно повысить стойкость. Однако такие схемы являются схемами направленного шифрования с оговоркой, так как для выполнения операций зашифрования и расшифрования используются гамма-последовательности на базе общих секретов. Общие секреты могут формироваться на долговременных и/или сеансовых ключах. Общая схема реализации шифрования на основе примитивов Диффи-Хеллмана представлена на рис. 2 (на рис. 2: KDF – алгоритм выработки гаммы шифрующей; *д_д* – дополнительные данные; Q_O – открытый ключ отправителя; $Q_П$ – открытый ключ получателя; D_O – личный ключ отправителя; $D_П$ – личный ключ получателя).

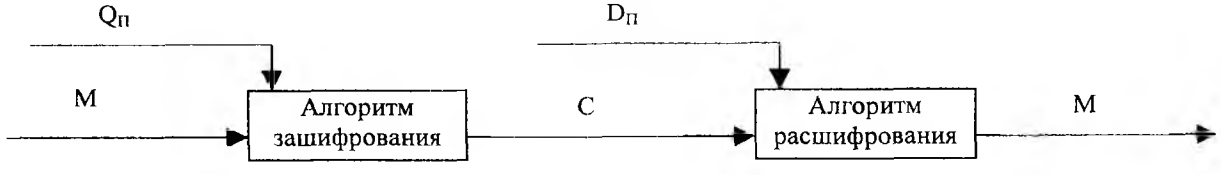


Рис. 1

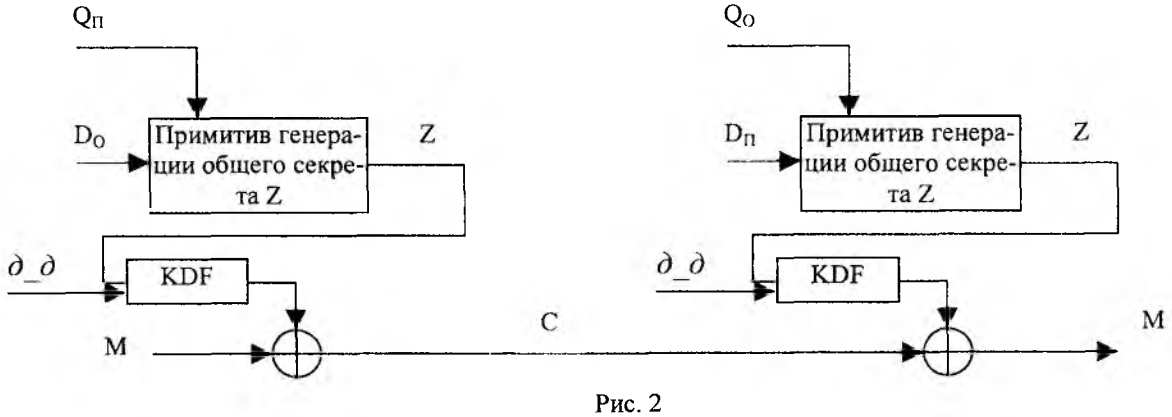


Рис. 2

Проведенный анализ показал, что стойкость шифрования схемы, представленной на рис. 2, в определяющей мере зависит от свойства гаммы шифрующей, формируемой KDF. Однако в известных источниках нет данных обоснований выбора схем KDF, методик исследований свойств гамм шифрования и выбора наиболее предпочтительных схем.

Целью настоящей статьи является проведение теоретического обоснования и практических исследований схем направленного шифрования в группе точек эллиптических кривых на основе примитивов Диффи-Хеллмана, а также разработка рекомендаций по их практическому применению.

1 Методы построения алгоритмов формирования гамм шифрующих

К настоящему времени известно несколько методов построения функций формирования гамм шифрующих. Логика преобразований этих функций может базироваться на однонаправленных хэш-функциях (ГОСТ 34.311-95, SHA-1, SHA-2, MD-5), на симметричных блочных алгоритмах, работающих в режиме выработки гаммы шифрующей, а также на основе поточных шифров.

При выработке гаммы шифрующей с использованием симметричных алгоритмов используется соответствующий алгоритм блочного шифрования в режиме выработки гаммы. При этом вектор инициализации и ключ могут формироваться из общего секрета либо вектор инициализации может быть заранее определён или выработан абонентами. При выработке гаммы шифрующей с использованием поточных шифров общий секрет используется в качестве вектора инициализации.

При выработке гаммы шифрующей с использованием однонаправленных хэш-функций используют префиксно-суффиксный вариант подстановки ключей, причём в качестве ключей применяют общий секрет, а также при необходимости дополнительные данные. При использовании для генерации гаммы шифрующей хэш-функции SHA-1 выполняются следующие этапы:

1. Выполняются шаги 2 и 3 до тех пор, пока гамма не достигнет размера, который на 4 и менее 32-битных блоков будет меньше размера шифруемой информации.
2. Вычисляется Γ_i = SHA-1(Дополнительные данные||Общий секрет||i).
3. Увеличивается значение счётчика (i++).
4. Если размер гаммы меньше размера шифруемой информации, выполнить генерацию неполного блока по формуле аналогично шагу 2. Уменьшить последний блок до необходимой длины.

2 Оценка безопасности функций развёртывания гаммы шифрующей (KDF)

При использовании функций формирования гаммы шифрующей на основе хэш-функций основными атаками на системы направленного шифрования могут быть атаки, основанные на коллизиях. Следовательно, для определения опасности таких атак необходимо дать оценку вероятностей возникновения коллизии. Как было показано в [4], вероятность возникновения коллизии подчиняется «парадоксу» о дне рождения. Существуют два варианта возникновения коллизий: возникновение коллизии в одной выходной последовательности и возникновение коллизии в разных выходных последовательностях. Дадим оценку вероятности появления коллизий в одной выходной последовательности.

Постановка задачи. Пусть имеется некоторая функция преобразования H

$$h=H(M), \qquad (1)$$

где M есть данные (информация) произвольной длины l_M, причем h может принимать $n=2^m$ значений независимо от длины l_M. Необходимо определить число k случайных сообщений M_i, которые необходимо подать на вход преобразователя H, чтобы с вероятностью $P_э$ состоялось хотя бы одно совпадение вида (1), т.е. состоялась коллизия.

Проведенный анализ показал [4], что задача 1 может быть решена с использованием «парадокса» о дне рождения, но при подробном рассмотрении выясняется, что она носит более общий характер. В нашем случае имеется выборка из k значений целочисленной случайной величины с равновероятным законом распределения, причем она может принимать значения от 1 до $n=2^m$, а $k \le n$. При этих условиях необходимо найти вероятность $P(n,k)$ того, что среди значений $H(M)$ выборки по крайней мере две совпадают, т.е.

$$H(M_i)=H(M_j).$$

Для решения задачи 1 найдем вероятность того, что в группе из k событий не состоится коллизия, т.е. соотношение (1) не выполнится ни разу. Обозначим эту вероятность как $R(n,k)$. Ясно, что $P(n,k)$ и $R(n,k)$ составляют полную группу событий, т.е.

$$P(n,k)+R(n,k)=1$$

и

$$P(n,k)=1-R(n,k). \tag{3}$$

Далее найдем общее число N различных способов, которыми можно получить k значений без повторений. Для первого элемента имеем n значений без повторений, для второго n-1, третьего – n-2 и т.д., для k-го (n-k+1). Поэтому общее число способов, при которых совпадений вида (1) нет, равно

$$N = n\cdot(n-1)(n-2)..(n-k+1) = \frac{n!}{(n-k)!}. \tag{4}$$

Проведя преобразования, аналогичные [4], мы получим общие формулы для определения количества сообщений, которые нужно подать на вход, чтобы с вероятностью P_k состоялась коллизия (5), а также вероятность осуществления коллизии при заданных значениях k и n (6).

$$k^2 = -2n\ln(1-P_k) \tag{5}$$

или

$$k = \sqrt{2\ln\left(\frac{1}{1-P_k}\right)\cdot n} = 1{,}41\sqrt{\ln\left(\frac{1}{1-P_k}\right)\cdot n}, \quad P_k = 1-e^{-\frac{k^2}{2n}}. \tag{6}$$

Здесь P_k – вероятность, с которой может возникнуть коллизия, а k – количество сформированных блоков гаммы.

Важной также является задача оценки вероятности появления коллизий в нескольких выходных последовательностях, т.е. появление одинаковых блоков гамм шифрующих длиной m битов [4].

Постановка задачи. Пусть на выходе преобразователя H из полного множества значений $n=2^m$ формируются k случайных значений функции преобразования (2), причем $k \le n$ и k подчиняются равновероятному закону распределения. Пусть выполнено z экспериментов, в каждом из которых получено k значений h. Обозначим реализации двух экспериментов соответственно как X и Y, причем $X=(x_1,x_2,\ldots,x_k)$ и $Y=(y_1,y_2,\ldots,y_k)$. Необходимо найти вероятность $P(n,k)$ того, что эти множества содержат в себе хотя бы по одному элементу x_i и y_j таких, что $x_i=y_j$.

Рассмотрим решение этой задачи, опираясь на результаты задачи 1. Вторая задача возникает при рассмотрении и выборе способов создания коллизий. Например, в нашей постановке эта задача имеет смысл, если рассмотрение сразу двух или большего числа множеств

из k реализаций позволяет ускорить процесс создания коллизий или извлечь полезную для криптоаналитика информацию для выполнения последующих атак.

В нашем случае событие $x_i=y_j$ может состояться с вероятностью $1/n$. Поэтому вероятность того, что $x_i \neq y_j$

$$Q(x_i \neq y_j) = 1 - \frac{1}{n}. \tag{21}$$

Если Y включает в себя k событий, то вероятность того, что все значения $y_1, y_2, \ldots, y_k$ не совпадут с x_i, может быть вычислена как

$$Q(x_i \neq Y) = \left(1 - \frac{1}{n}\right)^k. \tag{22}$$

Вероятность того, что хотя бы одно значение Y совпадет с x_i, есть

$$R(x_i \in Y) = 1 - \left(1 - \frac{1}{n}\right)^k. \tag{23}$$

Проведя необходимые преобразования, получим общую формулу расчёта количества элементов в последовательностях для получения совпадения с вероятностью P_k:

$$k = \sqrt{n \ln\left(\frac{1}{1 - P_k}\right)}. \tag{29}$$

Как видно из формулы, для совпадения с определённой вероятностью двух блоков из разных последовательностей необходимо сгенерировать последовательности в корень из двух раз короче, чем одиночная последовательность для задачи типа 1. То есть общее количество блоков при равной вероятности коллизии в задаче типа 1 меньше, чем в задаче типа 2.

В табл. 1 приведены оценки значений величины k, которые приводят к коллизиям с вероятностью коллизии P_k для случая использования в качестве KDF функций хэширования SHA-1, SHA-2(256), SHA-2(384), SHA-2(512), ГОСТ 34.311-95, ГОСТ 28.147-89 и Rijndael (длины ключей 128, 192 и 256 битов).

Таблица 1

P_k	10^{-16}	10^{-12}	10^{-6}	0,1	0,5	0,99	$1\text{-}10^{-5}$	$1\text{-}10^{-10}$
SHA-1	$1{,}9 \cdot 2^{53}$	$1{,}5 \cdot 2^{59}$	$1{,}47 \cdot 2^{70}$	$0{,}45 \cdot 2^{80}$	$1{,}18 \cdot 2^{80}$	$3{,}03 \cdot 2^{80}$	$4{,}78 \cdot 2^{80}$	$6{,}77 \cdot 2^{80}$
SHA-2 (256)	$1{,}9 \cdot 2^{101}$	$1{,}5 \cdot 2^{107}$	$1{,}47 \cdot 2^{118}$	$0{,}45 \cdot 2^{128}$	$1{,}18 \cdot 2^{128}$	$3{,}03 \cdot 2^{128}$	$4{,}78 \cdot 2^{128}$	$6{,}77 \cdot 2^{128}$
SHA-2 (384)	$1{,}9 \cdot 2^{165}$	$1{,}5 \cdot 2^{171}$	$1{,}47 \cdot 2^{182}$	$0{,}45 \cdot 2^{192}$	$1{,}18 \cdot 2^{192}$	$3{,}03 \cdot 2^{192}$	$4{,}78 \cdot 2^{192}$	$6{,}77 \cdot 2^{192}$
SHA-2 (512)	$1{,}9 \cdot 2^{229}$	$1{,}5 \cdot 2^{235}$	$1{,}47 \cdot 2^{246}$	$0{,}45 \cdot 2^{256}$	$1{,}18 \cdot 2^{256}$	$3{,}03 \cdot 2^{256}$	$4{,}78 \cdot 2^{256}$	$6{,}77 \cdot 2^{256}$
ГОСТ 34.311-95	$1{,}9 \cdot 2^{101}$	$1{,}5 \cdot 2^{107}$	$1{,}47 \cdot 2^{118}$	$0{,}45 \cdot 2^{128}$	$1{,}18 \cdot 2^{128}$	$3{,}03 \cdot 2^{128}$	$4{,}78 \cdot 2^{128}$	$6{,}77 \cdot 2^{128}$
ГОСТ 28.147-89	$1{,}9 \cdot 2^{5}$	$1{,}5 \cdot 2^{11}$	$1{,}47 \cdot 2^{22}$	$0{,}45 \cdot 2^{32}$	$1{,}18 \cdot 2^{32}$	$3{,}03 \cdot 2^{32}$	$4{,}78 \cdot 2^{32}$	$6{,}77 \cdot 2^{32}$

Продолжение табл. 1

P_k	10^{-16}	10^{-12}	10^{-6}	0,1	0,5	0,99	$1-10^{-5}$	$1-10^{-10}$
Rijndael (128 бит)	$1,9 \cdot 2^{37}$	$1,5 \cdot 2^{43}$	$1,47 \cdot 2^{54}$	$0,45 \cdot 2^{64}$	$1,18 \cdot 2^{64}$	$3,03 \cdot 2^{64}$	$4,78 \cdot 2^{64}$	$6,77 \cdot 2^{64}$
Rijndael (192 бит)	$1,9 \cdot 2^{69}$	$1,5 \cdot 2^{75}$	$1,47 \cdot 2^{86}$	$0,45 \cdot 2^{96}$	$1,18 \cdot 2^{96}$	$3,03 \cdot 2^{96}$	$4,78 \cdot 2^{96}$	$6,77 \cdot 2^{96}$
Rijndael (256 бит)	$1,9 \cdot 2^{101}$	$1,5 \cdot 2^{107}$	$1,47 \cdot 2^{118}$	$0,45 \cdot 2^{128}$	$1,18 \cdot 2^{128}$	$3,03 \cdot 2^{128}$	$4,78 \cdot 2^{128}$	$6,77 \cdot 2^{128}$

В табл. 2-8 приведены оценки возникновения вероятностей появления коллизий в зависимости от числа k, т.е. числа блоков гаммы шифрующей, при которых наступает коллизия с вероятностью P_k. Оценки приведены для алгоритмов ГОСТ 28147-89; Rijndael с длиной блока 128бит; SHA-1; Rijndael с длиной блока 192бит; Rijndael 256бит, SHA-2 256бит, ГОСТ-34.11-95; SHA-2 384бит; SHA-2 512бит. Так, для ГОСТ 28147-89 коллизия наступает с $P_k \approx 1$ уже при $k \geq 10^{12}$, а для Rijndael(128) при $k \geq 10^{20}$. Таким образом, Rijndael по сравнению с ГОСТ 28147-89 имеет существенно большую стойкость против коллизий, а также существенно меньшую вероятность перекрытия шифра. Это объясняется большей длиной блока у Rijndael. При использовании для реализации KDF функции хэширования SHA-1 с длиной хэш-функции 160 битов реальные коллизии возможны при $k \geq 10^{23}$, для SHA-2 с длиной хэш-функции 256 битов при $k \geq 10^{37}$, при длине хэш-функции 384 бита при $k \geq 10^{56}$, при длине хэш-функции 512 бита при $k \geq 10^{75}$. Алгоритм шифрования ГОСТ 34.311-95 обеспечивает такую же степень защиты от коллизий, как и SHA-2 при длине хэш-функции 256 битов.

Таблица 2

K	10	10^2	10^3	10^4	10^5	10^6
P_k	$2,71 \cdot 10^{-18}$	$2,71 \cdot 10^{-16}$	$2,71 \cdot 10^{-14}$	$2,71 \cdot 10^{-12}$	$2,71 \cdot 10^{-10}$	$2,71 \cdot 10^{-8}$

K	10^7	10^8	10^9	10^{11}	10^{12}	10^{13} и более
P_k	$2,71 \cdot 10^{-6}$	$2,71 \cdot 10^{-4}$	$2,71 \cdot 10^{-2}$	0,933	≈ 1	≈ 1

Таблица 3

K	$10-10^9$	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}
P_k	≈ 0	10^{-19}	$1,47 \cdot 10^{-17}$	$1,47 \cdot 10^{-15}$	$1,47 \cdot 10^{-13}$	$1,47 \cdot 10^{-11}$

K	10^{15}	10^{16}	10^{17}	10^{18}	10^{19}	10^{20} и более
P_k	$1,47 \cdot 10^{-9}$	$1,47 \cdot 10^{-7}$	$1,47 \cdot 10^{-5}$	$1,47 \cdot 10^{-3}$	$1,47 \cdot 10^{-1}$	≈ 1

Таблица 4

K	$10-10^{14}$	10^{15}	10^{16}	10^{17}	10^{18}	10^{19}
P_k	≈ 0	$3,25 \cdot 10^{-19}$	$3,42 \cdot 10^{-17}$	$3,42 \cdot 10^{-15}$	$3,42 \cdot 10^{-13}$	$3,42 \cdot 10^{-11}$

Таблица 4 – продолжение

K	10^{20}	10^{21}	10^{22}	10^{23}	10^{24}	10^{25} и более
P_k	$3,42 \cdot 10^{-9}$	$3,42 \cdot 10^{-7}$	$3,42 \cdot 10^{-5}$	$3,42 \cdot 10^{-3}$	0,29	≈ 1

Таблица 5

K	10-10^{19}	10^{20}	10^{21}	10^{22}	10^{23}	10^{24}
P_k	≈ 0	$7{,}59 \cdot 10^{-19}$	$7{,}59 \cdot 10^{-17}$	$7{,}59 \cdot 10^{-15}$	$7{,}59 \cdot 10^{-13}$	$7{,}59 \cdot 10^{-11}$

K	10^{25}	10^{26}	10^{27}	10^{28}	10^{29}	10^{30} и более
P_k	$7{,}59 \cdot 10^{-9}$	$7{,}59 \cdot 10^{-7}$	$7{,}59 \cdot 10^{-5}$	$7{,}59 \cdot 10^{-3}$	0,549	≈ 1

Таблица 6

K	10-10^{29}	10^{30}	10^{31}	10^{32}	10^{33}	10^{34}
P_k	≈ 0	$4{,}31 \cdot 10^{-18}$	$4{,}31 \cdot 10^{-16}$	$4{,}31 \cdot 10^{-14}$	$4{,}31 \cdot 10^{-12}$	$4{,}31 \cdot 10^{-10}$

K	10^{35}	10^{36}	10^{37}	10^{38}	10^{39}	10^{40} и более
P_k	$4{,}31 \cdot 10^{-8}$	$4{,}31 \cdot 10^{-6}$	$4{,}31 \cdot 10^{-4}$	$4{,}31 \cdot 10^{-2}$	0,99	≈ 1

Таблица 7

K	10-10^{48}	10^{49}	10^{50}	10^{51}	10^{52}	10^{53}
P_k	≈ 0	$1{,}3 \cdot 10^{-18}$	$1{,}3 \cdot 10^{-16}$	$1{,}3 \cdot 10^{-14}$	$1{,}3 \cdot 10^{-12}$	$1{,}3 \cdot 10^{-10}$

K	10^{54}	10^{55}	10^{56}	10^{57}	10^{58}	10^{59} и более
P_k	$1{,}3 \cdot 10^{-8}$	$1{,}3 \cdot 10^{-6}$	$1{,}3 \cdot 10^{-4}$	$1{,}3 \cdot 10^{-2}$	0,72	≈ 1

Таблица 8

K	10-10^{67}	10^{68}	10^{69}	10^{70}	10^{71}	10^{72}
P_k	≈ 0	$3{,}2 \cdot 10^{-19}$	$3{,}7 \cdot 10^{-17}$	$3{,}7 \cdot 10^{-15}$	$3{,}7 \cdot 10^{-13}$	$3{,}7 \cdot 10^{-11}$

K	10^{73}	10^{74}	10^{75}	10^{76}	10^{77}	10^{78} и более
P_k	$3{,}7 \cdot 10^{-9}$	$3{,}7 \cdot 10^{-7}$	$3{,}7 \cdot 10^{-5}$	$3{,}7 \cdot 10^{-3}$	0,31	≈ 1

3 Статистическая безопасность гамм шифрующих

Исследование статистической безопасности гаммы проведено с использованием пакета NIST STS [5].

В 1999 году специалистами NIST в рамках проекта AES (Advanced Encryption Standard) был разработан набор статистических тестов NIST STS (NIST Statistical Test Suite) и предложена методика проведения статистического тестирования ГСЧ (ГПСЧ) [6], которые на настоящий момент наилучшим образом отвечают потребностям всех заинтересованных сторон.

Для принятия решения о прохождении последовательностью случайных (псевдослучайных) чисел статистического теста в пакете NIST STS заложен следующий критерий принятия решения[5].

Пусть дана двоичная последовательность $S = \{s_1, s_2, \ldots, s_n\}$, $s_i \in \{0,1\}$ длиной n бит. Необходимо принять решение, проходит данная последовательность статистический тест на случайность, равновероятность и независимость или нет .

В NIST STS построение критерия принятия решения опирается на вычисление для статистики теста $c(S)$ соответствующего значения вероятности P. Здесь статистика теста рассматривается как реализация случайной величины, которая подчиняется известному закону распределения. Статистка теста строится таким образом, чтобы её большие значения указы-

вали на какой-либо дефект случайности последовательности. Значение вероятности P есть вероятность того, что статистика теста примет значение большее, чем наблюдаемое при опыте в предположении случайности последовательности. Следовательно малые значения P ($P < 0{,}05$ или $P < 0{,}01$) интерпретируются как доказательство того, что последовательность не случайна. Решающее правило формулируется так: *для фиксированного уровня значимости* α *двоичная последовательность* S *не проходит статистический тест, если значение вероятности* $P < \alpha$. Значения α рекомендуется выбирать из интервала [0,001, 0,01].

Пакет NIST STS включает в себя 16 статистических тестов, которые разработаны для проверки гипотезы о случайности двоичных последовательностей произвольной длины, порождаемых генераторами случайных последовательностей или генераторами псевдослучайных последовательностей. Все тесты направлены на выявление различных дефектов случайности. Основным принципом тестирования является проверка нулевой гипотезы H_0, заключающейся в том, что тестируемая последовательность является случайной. Альтернативной гипотезой H_a является гипотеза о том, что тестируемая последовательность не случайна. По результатам применения каждого теста нулевая гипотеза либо принимается, либо отвергается. Решение о том, будет ли заданная последовательность нулей и единиц случайной или нет, принимается по совокупности результатов всех тестов.

Тестирование отдельной двоичной последовательности S с использованием NIST STS выполнялось следующим образом.

Выдвигается нулевая гипотеза H_0– предположение о том, что данная двоичная последовательность S случайна.

По последовательности S вычисляется статистика теста $c(S)$.

С использованием специальной функции и статистики теста вычисляется значение вероятности $P=f(c(S))$, $P \in [0,1]$.

Значение вероятности P сравнивается с уровнем значимости α, $\alpha \in [0{,}001, 0{,}01]$. Если $P \geq \alpha$, то гипотеза H_0 принимается. В противном случае принимается альтернативная гипотеза.

Как уже было сказано, пакет включает в себя 16 статистических тестов. Но фактически в зависимости от входных параметров вычисляется 189 значений вероятности P, которые можно рассматривать как результат работы отдельных тестов.

В результате тестирования двоичной последовательности формируется вектор значений вероятности $\mathbf{P} = \{P_1, P_2, \ldots, P_{189}\}$. Анализ составляющих P_i данного вектора позволяет указать на конкретные дефекты случайности тестируемой последовательности.

Рассмотрим более подробно каждый тест и дефекты последовательности, которые может выявить данный тест.

1. Частотный (монобитный) тест (одно значение вероятности) – выявляет дефект слишком большого количества нулей или единиц в последовательности.
2. Частотный тест для блока (одно значение вероятности) – локализованные отклонения частоты появления единиц в блоке от идеального значения ½.
3. Проверка серий (одно значение вероятности) – слишком быстрая или слишком медленная перемена знака в ходе генерации последовательности.
4. Проверка максимальной длины серии в блоке (одно значение вероятности) – отклонение от теоретического закона распределения максимальных длин серий единиц.
5. Проверка ранга двоичной матрицы (одно значение вероятности) – отклонение эмпирического закона распределения значений рангов матрицы от теоретического, что указывает на зависимость символов в последовательности.
6. Спектральный тест на основе дискретного преобразования Фурье (одно значение вероятности) – отклонение эмпирического закона распределения значений рангов матрицы от теоретического, что указывает на зависимость символов в последовательности.

7. Проверка перекрывающихся шаблонов (одно значение вероятности) – большое количество m- битных серий из единиц в последовательности.
8. Проверка перекрывающихся шаблонов (148 значений вероятности (при длине шаблона 9 бит)) – большое количество заданных непериодических шаблонов в последовательности.
9. Универсальный тест Маурера (одно значение вероятности) – сжимаемость последовательности.
10. Проверка сжатия по алгоритму Лемпеля-Зива (одно значение вероятности) – большая степень сжатия тестируемой последовательности по сравнению с ожидаемой степенью сжатия для случайной последовательности.
11. Последовательный тест (2 значения вероятности) – неравномерность распределения m-битных слов в последовательности.
12. Энтропийный тест (одно значение вероятности) – неравномерность распределения m-битных слов в последовательности (регулярность свойств источника).
13. Проверка накопленных сумм (2 значения вероятности) – большое значение единиц или нулей вначале или в конце двоичной последовательности.
14. Проверка случайных отклонений (8 значений вероятности) – отклонение от теоретического закона распределения попаданий в конкретное состояние при случайном блуждании.
15. Проверка случайных отклонений (18 значений вероятности) – отклонение от теоретического ожидаемого общего количества попаданий при случайном блуждании в заданное состояние.
16. Проверка линейной сложности (одно значение вероятности) – отклонение эмпирического распределения длин эквивалентных ЛРР для последовательности фиксированной длины от теоретического закона распределения для случайной последовательности, что указывает на недостаточную сложность тестируемой последовательности.

4 Результаты экспериментальных исследований

С использованием пакета NIST STS проведено тестирование 9 последовательностей, генерация которых выполнена при помощи функций KDF, внутренняя структура которых основывается на различных алгоритмах (SHA-1, SHA-2, Rijndael, ГОСТ-28.147-89, ГОСТ-34.11-95) .

Для осуществления тестирования были выбраны следующие параметры:

1. Длина тестируемой последовательности $n = 10^6$ бит.

2. Количество тестируемых последовательностей $m = 100$. Таким образом, объем тестируемой выборки составил $N=10^6\times100=10^8$ бит.

3. Уровень значимости $\alpha=0,01$.

4. Количество тестов $q=189$. Таким образом, статистический портрет генератора содержит 18900 значений вероятности P.

В идеальном случае при $m=100$ и $\alpha=0,01$ может быть отвергнута только одна последовательность из ста, т.е. коэффициент прохождения каждого теста должен составлять 99%. Но это слишком жесткое правило. Поэтому и применяется правило на основе доверительного интервала для r_j. Нижняя граница в этом случае составит значение $r_{min} = 0,96015$.

Проведенное тестирование позволяет сделать следующие выводы. Гарантировано полное прохождение всех тестов обеспечивается только KDF на основе функци SHA-2 с длиной блока 512 битов (рис. 3). Последовательности, сформированные с использованием функции хэширования SHA-2 с длиной блока 256 или 384 бита и функции SHA-1 имеют по одному из тестов долю прохождения 0,95 (необходимо 0,96), т.е. значения, близкие к требуемым. При использовании ГОСТ 28147-89 и ГОСТ 34.311-95 по одному из тестов вероятность прохож-

дения составляет 0,94. На рис. 4 приведен статистический портрет последовательностей, сформированных KDF на основе алгоритма Rijndael с длиной блока 256 бит. Как видно из рис. 4, по двум тестам вероятность прохождения составляет 0,95, для длины блока 128 бит ситуация ещё хуже: один тест – меньше 0,94, другой чуть больше. Среди функций KDF на основе Rijndael наиболее приемлемым по статистической безопасности является вариант с 192-битной длиной блока: в этом случае лишь один тест даёт вероятность меньше необходимой (0,95).

Рис. 3

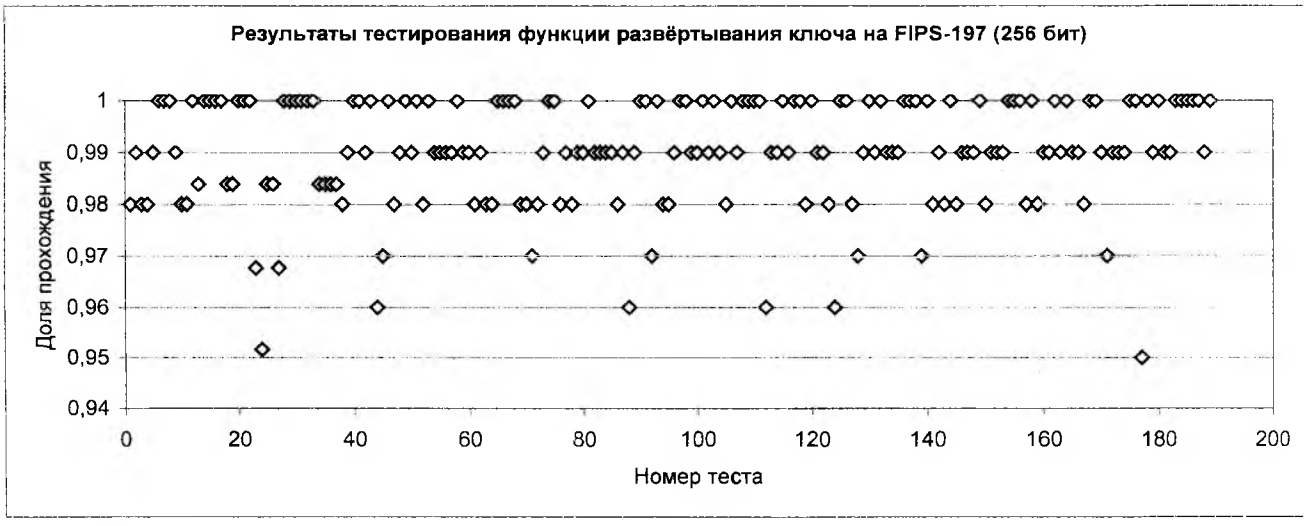


Рис. 4

Как видно из представленных выше рисунков, полное прохождение всех тестов (с коэффициентом прохождения 96%) обеспечивает только функция развёртывания ключа на основе хэш-функции SHA-2 в режиме 512 бит. Функции на основе остальных хэш-функций и симметричных алгоритмов шифрования не проходят по заданному коэффициенту как минимум в одном тесте, хотя все они имеют вероятностные портреты.

Заключение

Схемы шифрования, базирующиеся на примитивах Диффи-Хеллмана, обеспечивают высокую скорость формирования гамм шифрования и поточное шифрование. Наиболее сложным является этап формирования общего секрета, который требует одного возведения в степень по модулю при преобразовании в простом поле или одного скалярного умножения в группе точек эллиптической кривой. Если эти вычисления выполнять предварительно или на сопроцессоре, то скорость поточного шифрования (зашифрование/расшифрование) определяется скоростью хэширования или блочного шифрования.

Вероятности появления коллизий на выходе KDF зависят в определяющей мере от длины блока хэш-функции или блочного шифрования. С увеличением длины блока вероятность коллизии уменьшается соответственно. Приемлемые значения P_k достигаются с использованием функции хэширования SHA-2, в том числе и в перспективе на десятки лет. Приведенные результаты могут быть достигнуты вычислением хеш-функции или блочного шифрования из общего секрета, дополнительных данных и значений счётчиков числа сформированных блоков.

Наилучшую статистическую безопасность обеспечивают KDF, реализованные на основе функции хэширования SHA-2 и использовании дополнительных данных и/или значений счётчиков блоков. Так, последовательности, сформированные с использованием SHA-2, имеют долю прохождения по всем тестам не менее 96%, что позволяет считать их случайными, равновероятными и независимыми.

Список литературы: 1. *О.В. Вербицький.* Вступ до криптології. Львів: ВНТЛ, 1998. 2. X9.42 – 1998, Public Key Cryptography for The Financial Service Industry : Agreement of Symmetric Keys on Using Diffie-Hellman and MQV Algorithms. 3. X9.63 Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, 1999. 207 с. 4. *А.Д. Тевяшев, Ю.И. Горбенко.* Оценка опасности криптоаналитических атак методом создания коллизий // Радиотехника: Всеукр.межвед. науч. техн. сб. 2002. Вып. 126. С. 125 – 131. 5. *Потий А.В., Орлова С.Ю.* Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NIST STS // Там же. С. 144 – 150. 6. *J. Soto* Randomness Testing of the Advanced Encryption Candidate Algorithms // NIST, 1999.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 16.04.2003

УДК 681.3.06:519.248.681

О. В. ПОТІЙ, канд. техн. наук, Ю. І. ГОРБЕНКО, Є. В. ПОПОВИЧ

МЕТОД ОЦІНКИ ІМОВІРНОСТЕЙ КОЛІЗІЙ У БЕЗУМОВНО СТІЙКИХ ТА ОБЧИСЛЮВАЛЬНО СТІЙКИХ КРИПТОСИСТЕМАХ

В останні роки знаходять застосування системи криптографічного захисту інформації, що забезпечують безумовну або обчислювальну стійкість. В безумовно стійких системах для шифрування (зашифрування/розшифрування) використовуються ключові послідовності K, що будуються на основі випадкових процесів [1,2]. При цьому довжина ключової послідовності $l_{кл}$ повинна бути не менше довжини l_M шифруємої інформації. Зашифрування здійснюється згідно правила

$$C_i = (M_i + K_i) \bmod m, \tag{1}$$

де C_i – i-й зашифрований символ; M_i – i-й – символ відкритої інформації; K_i – i-й – символ ключової послідовності; m – модуль криптографічного перетворення.

Розшифрування здійснюється за правилом

$$M_i = (C_i - K_i) \bmod m. \tag{2}$$

Одним із слабких місць такої криптосистеми є можливість перекриття ключових послідовностей деякої критичної довжини $l_{кр}$ в просторі чи часі. Тут та далі за текстом таке перекриття ключових послідовностей будемо називати колізією гами шифру. Дійсно, якщо $C_i' = (M_i' + K_i) \bmod m$, тобто відбулося перекриття K_i з довжиною $l_{кр}$, то

$$C_i + C_i' = (M_i + K_i + M_i' + K_i) \bmod m = (M_i + M_i') \bmod m.$$

Далі розклад суми $(M_i + M_i') \bmod m$ є практично поліноміальною задачею [3].

Тому, на наш погляд, дуже важливою є задача дослідження можливостей виникнення таких колізій та визначення обмежень на величину $l_{кр}$ в залежності від допустимої імовірності виникнення колізії P_k та числа k сформованих відрізків ключової послідовності з довжиною $l_i \le l_{kp}$. Слід зазначити, що, не враховуючи методи формування гами шифру, дані показники будуть справедливі для будь-якого шифру гамування.

Розглядаючи клас блокових шифрів, необхідно зазначити, що блокові симетричні криптоперетворення (шифри) можуть використовуватись принаймі в п'ятьох режимах [4]. Запропоновано ще декілька режимів криптографічних перетворень [4], що здійснюються на основі блокового симетричного шифрування. Основними з них є слідуючи режими:

- блокового шифрування,
- потокового шифрування зі зворотнім зв'язком по шифротексту,
- потокового шифрування зі зв'язком по гамі шифруючій,
- потокового шифрування з лічильником,
- потокового шифрування з лічильником та автентифікацією,
- блокового шифрування зі зв'язком блоків (виробки імітоприкладки).

Проведемо аналіз можливостей виникнення колізій в режимі блокового шифрування. В цьому режимі результат зашифрування є деякий блок C_i, причому

$$C_i = F(M_i, K_j, R_{i-v}) \tag{3}$$

де M_i – блок відкритого тексту; K_j – ключ зашифрування; R_{i-v} – сінхромаркер (інформація зворотного зв'язку або стан лічильника).

Метою даної статті є обґрунтування та розробка методу оцінки колізій та розробка рекомендацій на допустимі мінімальні значення $l_{кр}$ для безумовно стійких та обчислювально стійких симетричних шифрів.

1 Оцінка імовірностей колізій для безумовно стійких шифрів

Нехай генератор $Г$ формує випадкові ключові послідовності K_i, еквівалентна двійкова довжина яких є l_i. На виході такого генератора може з'явитись рівноімовірно та незалежно кожна із $n = 2^{l_i}$ ключових послідовностей. Нехай також необхідно сформувати k ключових послідовностей довжини l_i. Необхідно знайти математичне співвідношення, що зв'язує між собою такі величини, як $n(l_i)$, k та допустиму імовірність колізій P_k, а також визначити допустиму довжину l_k ключових послідовностей з урахуванням можливих застосувань безумовно стійких криптосистем та допустимих імовірностей колізії.

Розв'язок виконаємо на основі узагальненого «парадоксу дня народження» [5].

В даному генераторі усього може бути сформовано $\bar{n} = 2^{l_i}$ ключових послідовностей K_i, і усі вони є рівноімовірними та незалежними. Нехай усього сформовано k ключових послідовностей довільної довжини l_i. Визначимо імовірність події, що при цьому відбудеться колізія, тобто два ключі із k співпадуть.

Визначимо загальну множину подій N_Σ, які можливі при формуванні послідовності $K_1, K_2, ..., K_k$. Оскільки усі n значень є незалежними та рівноімовірними, то

$$N_\Sigma = \underbrace{n \cdot n \cdot n \cdot ... \cdot n}_{k} = n^k. \tag{4}$$

Далі визначимо множину подій, її величину, при якій не буде жодної колізії.

В перший раз без колізії може відбутися n подій, в другий n-1, третій n-2 і т.д., і при k-й події $n-(k-1)$. Оскільки усі ці події рівноімовірні та незалежні, то загальне число подій при k експериментах, при яких колізій не буде, можна визначити як

$$N_k = n \cdot (n-1)(n-2)...(n-(k-1)). \tag{5}$$

Знаючи N_k та N_Σ, імовірність того, що при експериментах формування ключових послідовностей довжини l_i колізії не буде, визначимо імовірність відсутності колізії $P_b(k,n)$ як

$$P_b(k,n) = \frac{N_k}{N_\Sigma} = \frac{n(n-1)(n-2)...(n-(k-1))}{n^k} = \frac{n!}{n^k (n-k)!}. \tag{6}$$

Оскільки імовірність колізії $P_k(k,n)$ та відсутність колізії складають повну групу подій, то

$$P_k(k,n) = 1 - P_b(k,n). \tag{7}$$

Враховуючи складність розрахунків під час роботи з великими числами, підставимо (6) в (7):

$$P_k(k,n) = 1 - \frac{n(n-1)(n-2)...(n-(k-1))}{n \quad n \quad n \qquad\qquad n} =$$
$$= 1 - 1\left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right)...\left(\frac{n-(k-1)}{n}\right) = \qquad (8)$$
$$= 1 - \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)...\left(1-\frac{k-1}{n}\right).$$

Оскільки в реальних випадках $k < 0{,}1n$, то для спрощення (8) можна зробити заміну $(1-x) \le e^{-x}$, в результаті маємо

$$P_k(k,n) = 1 - e^{-\frac{1}{n}} \cdot e^{-\frac{2}{n}} ... e^{-\frac{k-1}{n}} = 1 - e^{-\frac{1}{n(1+2+3+...+k-1)}} = 1 - e^{-\frac{k(k-1)}{2n}} = 1 - e^{-\frac{k(k-1)}{2 \cdot 2^{l_i}}}. \qquad (9)$$

Таким чином при вказаних обмеженнях одержано аналітичне співвідношення, що зв'язує між собою імовірність колізії $P_k(k,n)=P_k$, число сформованих ключових послідовностей k довжиною $l_i \ge l_{kp}$ та загальне число імовірних послідовностей $n = 2^{l_i}$.

Наявність співвідношення (9) дозволяє:

1. Оцінити імовірності колізій, змінюючи значення k та l_i.
2. Визначити критичне значення l_{kp} в залежності від допустимого значення імовірності колізії P_k для різних величин k (вони визначаються практичними додатками).
3. Визначити обмеження на число сформованих в системі (просторі та часі) послідовностей k_o, при яких імовірність колізії не перевищує P_k, якщо довжина ключової послідовності є $l_i \ge l_{kp}$.

Попередній аналіз допустимих джерел показав, що можливість здійснення на безумовно стійкі криптосистеми атаки методом колізій у них не враховано.

Розглянемо порядок розв'язку перелічених вище задач та проведемо їх дослідження.

При оцінці величин імовірності колізії можна використовувати вираз (9). При цьому для випадку, коли $k^2 >> k$, його можна спростити до виду

$$P_k(k,n) \approx 1 - e^{-\frac{k^2}{2^{(l_i+1)}}}. \qquad (10)$$

В табл. 1 наведені значення імовірностей колізії P_k в залежності від k та l_i.

Таблиця 1

l \ k	2	16	32	64	128	256	1024	65536	10^9	10^{12}
8	0,004	0,38	0,867	0,999	~1	~1	---	---	---	---
16	$3.05 \cdot 10^{-5}$	$1.9 \cdot 10^{-3}$	$7.7 \cdot 10^{-3}$	0,031	0,118	0,393	0,999	~1	---	---
32	$4{,}6 \cdot 10^{-10}$	$2{,}9 \cdot 10^{-8}$	$1{,}1 \cdot 10^{-7}$	$4{,}7 \cdot 10^{-7}$	$1{,}9 \cdot 10^{-6}$	$7{,}6 \cdot 10^{-6}$	$1{,}2 \cdot 10^{-4}$	0.393	~1	---
64	$9{,}7 \cdot 10^{-20}$	$6{,}2 \cdot 10^{-18}$	$2{,}5 \cdot 10^{-17}$	$9{,}9 \cdot 10^{-17}$	$3{,}9 \cdot 10^{-16}$	$1{,}7 \cdot 10^{-15}$	$2.8 \cdot 10^{-14}$	$1.1 \cdot 10^{-10}$	0,02	~1
128	~0	~0	~0	~0	~0	~0	~0	$1{,}7 \cdot 10^{-29}$	$1{,}2 \cdot 10^{-21}$	$1{,}2 \cdot 10^{-15}$
256	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
512	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0

Аналіз даних табл. 1 дозволяє зробити такі висновки. Незалежно від довжини одноразової гами завжди існує ймовірність виникнення колізії і, як наслідок, здійснення атаки на безумовно стійку систему. Тому, якщо довжина шифруємого повідомлення не перевищує 64 бітів, то існують реальні імовірності колізій і, як наслідок, розкриття безумовно стійкої системи. Так при довжині гами l=64 бітів n=10^{13}, при k=10^{12} імовірність колізії дорівнює 1. Тому довжини повідомлень, що зашифровуються одноразовою гамою, повинні складати не менше 128 бітів.

Критичне значення l_{kp} можна знайти із співвідношень (9) або (10), подавши його у виді

$$1-P_k = e^{-\frac{k(k-1)}{2^{l_i+1}}} = e^{-\frac{k(k-1)}{2n}}. \qquad (11)$$

Прологарифмувавши даний вираз, маємо

$$\ln(1-P_k) = -\frac{(k^2+k)}{2^{l_i+1}} = -\frac{(k^2+k)}{2n}. \qquad (12)$$

Далі із (12) спочатку знаходимо n_{kp} як

$$n_{kp} = -\frac{(k^2+k)}{(2\ln(1-P_k))}. \qquad (13)$$

Критичне значення l_{kp} знаходимо із виразу $n_{kp} = 2^{l_{kp}}$, отже

$$l_{kp} = \log_2 n_{kp}. \qquad (14)$$

В табл. 2 наведено значення l_{kp} мінімально допустимих довжин ключових послідовностей (бітів) для безумовно стійких криптосистем в залежності від величин k та P_k.

Таблиця 2

P_k \ k	2	8	16	32	64	128	256	1024	32768	65536	10^6	10^9	10^{12}
10^{-3}	11	15	17	19	20	22	24	28	38	40	48	68	88
10^{-6}	21	25	27	28	30	32	34	38	48	50	58	78	98
10^{-9}	31	35	36	38	40	42	44	48	58	60	68	88	108
10^{-12}	41	45	46	48	50	52	54	58	68	70	78	98	118
10^{-16}	54	58	60	62	64	66	68	72	82	84	91	111	131

Таким чином ми отримали значення мінімальних довжин блоків, на які повинна буті поділена інформація з огляду на її загальний обсяг та імовірність виникнення колізій. Добуток значень k – кількості блоків та l – довжини блоку, отримані з таблиці, дадуть загальну довжину відкритого тексту, який може бути зашифрований, з відповідною імовірністю виникнення колізій. Слід зазначити, що під блоком інформації розуміється

таким чином скомпонована інформація, що її подальша обробка або аналіз можливі лише цілим блоком. Такі властивості інформація може отримати завдяки рандомізації.

Розв'язок третьої задачі, тобто визначення величини k_o, можна також здійснити, використавши (12). В результаті маємо

$$-\frac{(k^2+k)}{2n} = \ln(1-P_k)$$

або

$$k^2 + k + 2n\ln(1-P_k) = 0. \tag{15}$$

При значенні $k^2>>k$ можна використовувати співвідношення

$$k^2 + 2n\ln(1-P_k) = 0.$$

Тоді оцінкою величини k є значення

$$k = \sqrt{-2n\ln(1-P_k)} = \sqrt{-2^{l+1}\ln(1-P_k)}. \tag{16}$$

При відомих P_k та n або l можна обчислити величину k та загальну довжину відкритого тексту як

$$L = l \cdot k = l\sqrt{-2^{l+1}\ln(1-P_k)}. \tag{17}$$

Таким чином ми отримали показник загальної довжини тексту, який можливо зашифрувати в залежності від довжини блоків відкритого тексту та ймовірності колізій.

2 Оцінка імовірностей колізій для обчислювально стійких шифрів

Аналіз показує, що під час використання блокових шифрів може виникнути ситуація, коли два блоки криптограми дорівнюють один одному, тобто $C_i=C_j$. Виходячи з властивостей блокових шифрів, такий збіг може статися лише за наступних умов:

1. $M_i \neq M_j$, $K_i \neq K_j$, $R_i \neq R_j$.
2. $M_i = M_j$, $K_i \neq K_j$, $R_i \neq R_j$.
3. $M_i \neq M_j$, $K_i = K_j$, $R_i \neq R_j$.
4. $M_i \neq M_j$, $K_i \neq K_j$, $R_i = R_j$.
5. $M_i = M_j$, $K_i = K_j$, $R_i = R_j$.

Розглядаючи дані умови, слід зазначити, що по-перше інтерес викликає імовірність колізії ключів, тобто $C_i=C_j$ за умов $M_i \neq M_j$, $K_i = K_j$, $R_i \neq R_j$, та імовірність колізії текстів, тобто $C_i=C_j$ за умов $M_i = M_j$, $K_i \neq K_j$, $R_i \neq R_j$. Якщо M_i рандомізовано, то це дозволяє розглядати M_i як рівноімовірні та незалежні реалізації. Крім того, оскільки K_j є випадковим, то і C_i можна вважати випадковими, рівноімовірними, незалежними (однорідними) на повній множині подій. Тому задачу можливо звести до розрахунку імовірності появи на виході шифратора двох однакових криптограм в залежності від випадкового параметра K, який може бути блоком відкритого тексту або ключем. Використовуючи співвідношення (11), імовірність колізії можна визначити як

$$P_k(k,n) = 1 - e^{\frac{-k(k-1)}{2n}} = P_k, \tag{18}$$

де k – кількість оброблених або використаних блоків (кількість експериментів); n – розмір повної множини виходу перетворювача F, причому $n = 2^{l_б}$, де $l_б$ – довжина блоку.

Після простих перетворень (18) може бути подано у вигляді аналогічно (15), тобто

$$k^2 + k + 2n\ln(1 - P_k) = 0.$$

Оскільки для блокового криптоперетворення значення $l_б$ фіксоване, то $n = 2^{l_б}$ і також є величиною сталою. Тому для блокових криптоперетворень можна розв'язувати два типи задач. Визначення імовірностей колізії для різних значень K, а також визначення величини k, при якій імовірність колізії не перевищує допустимого значення P_g.

Для розв'язку першої задачі використовуємо співвідношення (10). В табл. 3 наведено значення імовірностей колізії в залежності від величини k для $l_б$=64, 128, 256 бітів.

Таблиця 3

$l_б$ \ k	2	2^8	2^{16}	2^{32}	2^{64}	2^{96}	2^{128}	2^{192}	2^{256}
64	$9{,}5\cdot10^{-20}$	$1{,}55\cdot10^{-15}$	$1{,}02\cdot10^{-10}$	0,393	~1	---	---	---	---
128	0	0	$2{,}2\cdot10^{-29}$	$9{,}49\cdot10^{-20}$	0,393	~1	~1	---	---
256	0	0	0	0	0	$2{,}37\cdot10^{-20}$	0,393	~1	~1

Аналіз даних табл. 3 показує, що імовірність колізії суттєво залежить від довжини блоку $l_б$, і довжини блоків повинні складати не менше 128 бітів. Цей висновок дозволяє зрозуміти, чому в європейському проекті Nessie для стійких шифрів довжина блоку відкритого тексту повинна складати не менше 128 бітів.

Для розв'язку другої задачі краще використовувати (15). Причому, якщо $k^2 >> k$, то його можна спростити до виду (16).

В табл. 4 наведені значення величин k в залежності від довжин l блоку та допустимих імовірностей колізій P_k.

Дані, наведені в табл. 4, дозволяють зрозуміти причини та необхідність збільшення в перспективних блокових симетричних шифрах довжин блоку $l_б$.

Таблиця 4

$l_б$ \ P_k	10^{-16}	10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{-1}	0,5
64	64	$6{,}07\cdot10^{3}$	$1{,}92\cdot10^{5}$	$6{,}07\cdot10^{6}$	$1{,}92\cdot10^{8}$	$1{,}97\cdot10^{9}$	$5{,}05\cdot10^{9}$
128	$8{,}2\cdot10^{11}$	$8{,}2\cdot10^{13}$	$2{,}6\cdot10^{15}$	$8{,}2\cdot10^{16}$	$8{,}2\cdot10^{17}$	$8{,}4\cdot10^{18}$	$2{,}1\cdot10^{19}$
256	$5{,}07\cdot10^{30}$	$4{,}8\cdot10^{32}$	$1{,}5\cdot10^{34}$	$4{,}8\cdot10^{35}$	$1{,}5\cdot10^{37}$	$1{,}5\cdot10^{38}$	$4\cdot10^{38}$

Використовуючи допустимі значення величини k, можна визначити також допустимі строки дії ключів, у тому числі в залежності від обсягу трафіка, швидкості шифрування, а також параметрів блокового симетричного шифру. Так, при умові, що допустима імовірність колізії $P_k \le 10^{-16}$, при $n = 2^{64}(l_б = 64)$ допустиме значення $k \le 64$, при $n = 2^{128}(l_б = 128)$ допустиме значення $k \le 8{,}2\cdot10^{11}$, при $n = 2^{256}(l_б = 256)$ допустиме значення $k \le 5\cdot10^{30}$. А знаючи розміри ключів, можна розрахувати максимально допустимий обсяг мережі.

Розглядаючи умови виникнення колізій та враховуючи незалежність потоку відкритих повідомлень та ключів, можна стверджувати, що імовірність співбігання криптограм за рахунок виникнення умови дорівнює добутку імовірності колізії ключів та імовірності колізії відкритих текстів.

Проведемо також дослідження можливостей появи колізії при застосуванні блокових симетричних шифрів в потоковому режимі, а також для потокових шифрів. Будемо вважати, що початковий стан шифруючого пристрою задається початковим ключем з довжиною l_n. При двійковій основі всього можна задати (ввести) $N_{кл} = 2^{l_n}$ ключів. Оскільки для початкових шифрів входом шифроутворюючого алгоритму є ключ, то число різних значень входів співпадає з повним простором ключів $N_{кл}$.

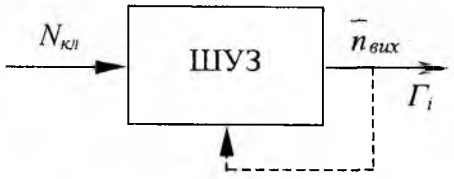


Рис. 1

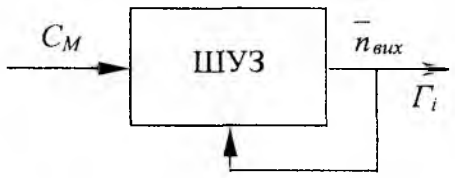


Рис. 2

На рис. 1 розглядається випадок, коли заміна початкового стану здійснюється з деяким інтервалом.

На рис. 2 розглядається випадок, коли здійснюється режим зі зворотним зв'язком по шифртексту чи лічильнику.

Виходом ШУЗ є гама зашифрування/розшифрування. В обох випадках настання колізії по суті буде призводити до перекриття шифру. Дійсно, якщо відрізки M_i^1 та M_i^2 зашифровано Γ_i, то

$$M_i^1 \oplus \Gamma_i \oplus M_i^2 \oplus \Gamma_i = M_i^1 \oplus M_i^2.$$

Далі розклад $M_i^1 \oplus M_i^2$ на складові може бути здійснений з поліноміальною складністю. Тому перекриття Γ_i у просторі або часі при одному і тому ж ключі є дуже загрозливим фактором.

Висновок

При розробці безумовно стійких шифрів необхідно враховувати можливість здійснення криптоаналізу на основі створення колізій. При цьому для забезпечення допустимої імовірності колізії мінімальна довжина блока інформації, а відповідно і одноразової гами, повинна бути обмежена. Для обчитслювально стійких шифрів імовірність перекриття шифру може бути визначена з використанням парадоксу «дні народження» і розрахунку імовірностей колізій, при цьому довжина шифруючої гами також обмежується з низу і залежить від допустимої імовірності колізії.

Список літератури: 1. *Шеннон К.* Теория связи в секретных системах // Работы по теории информации и кибернетике. М.: Изд-во. иностр. лит., 1963. С. 333 – 402. 2. *Замула А.А., Попович Е.В., Горбенко Ю.И.* Условия и возможности создания безусловно стойких криптосистем // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 119. С. 95 – 100. 3. *Шнайер Б.* Прикладная криптография. М.: Изд-во. «Триумф», 2002. 797 с. 4. *Стандарт* симетричного шифрування XXI століття: властивості, режими роботи, реалізація / І.Д. Горбенко, Л.В. Скрипник, С.О. Головашич, Т.О. Гріненко // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 119. С. 22 – 35. 5. *В. Столлингс.* Криптография и защита сетей: Принципы и практика. 2-е изд. Киев: Изд. дом «Вильямс», 2001. 669 с.

Харківський національний
університет радіоелектроніки
Департамент спеціальних телекомунікаційних
систем та захисту інформації СБ України

Надійшла до редколегії 15.05.2003

УДК 681.3.06:519.248.681

А. А. ПОЛЯКОВ

МЕТОД НАХОЖДЕНИЯ СЛУЧАЙНОЙ КРИПТОГРАФИЧЕСКИ СТОЙКОЙ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ НА РАСШИРЕННЫХ ПОЛЯХ ХАРАКТЕРИСТИКИ 2

Безопасность криптосистем, использующих группу точек эллиптической кривой, по сегодняшним взглядам, существенно зависит от сложности решения задачи эллиптического дискретного логарифма. Основными алгоритмами решения задачи дискретного логарифма в абелевых группах являются алгоритм Шанкса «больших и маленьких шагов», ρ-метод Полларда [1], алгоритм Полинга-Хелмана [2]. Для защиты от этой атаки необходимо выбирать группу, порядок которой кратен большому простому числу.

Известно, что существует множество слабых кривых, имеющих особые свойства, прежде всего это – суперсингулярные и аномальные кривые. На такие кривые известна атака Menezes-Okamoto-Vanstone (MOV),позволяющая перевести задачу эллиптического дискретного логарифма в задачу дискретного логарифма в мультипликативном поле F_{q^B}.

Учитывая требования по уровню прочности, предложенные в стандартах, рекомендуется, чтобы эллиптическая кривая была случайной, т.е. ее параметры были сгенерированы случайным образом. Случайная кривая, порядок которой делится на простое число как минимум длиной 160 бит, обеспечивает безопасность, сопоставимую с симметричными системами с длиной ключа 80 бит или системой RSA с длиной ключа 1024 бита. Эллиптические кривые, соответствующие данному требованию, а также требованию MOV ($B \geq 25$) и аномальности ($\#E \neq q$), будем называть криптографически стойкими. Для случайной кривой вероятность того, что будет сгенерирована слабая кривая, относительно мала. В приложениях, требующих более высокий уровень безопасности, необходимо увеличивать длину модуля преобразования.

В данной работе приведены и проанализированы основные методы вычисления порядка случайной эллиптической кривой (алгоритмы Сато и SEA). Ставится задача нахождения случайных криптографически стойких кривых, имеющих порядок эллиптической кривой, кратный 2 или 4.

Напомним основные сведения об эллиптических кривых на расширенном поле.

Эллиптическая кривая E с ненулевыми j-инвариантами на расширенном поле F_q, где $q = 2^m$, задается в следующем виде [6-7]:

$$E: y^2 + xy = x^3 + ax^2 + b, \text{ где } b \in F_q^*,$$

где a – некоторый фиксированный элемент из поля F_q следа 1.

j-инвариант, определяющий изоморфные кривые, вычисляется $j(E) = 1/b$. Следует заметить, что данные кривые несуперсингулярные и также необходимо выполнение следующего дополнительного условия: $j(E) \notin F_{2^2}$.

Множество точек кривой $E(F_q)$, описывающее абелевую группу, определяется как:

$$E(F_{2^m}) = \{(x, y) \in F_q \mid E\} \cup (O_E),$$

где O_E – точка бесконечности.

Эндоморфизм Фробениуса ϕ и мультипликативный эндоморфизм на кривой E определяются следующими отображениями точек:

$$\phi : E(\mathrm{F}_q) \to E(\mathrm{F}_q), \qquad [m] : E(\mathrm{F}_q) \to E(\mathrm{F}_q),$$
$$(x, y) \mapsto (x^q, y^q), \quad \text{и} \quad (x, y) \mapsto m(x, y).$$

Характеристическое уравнение эндоморфизма Фробениуса на эллиптической кривой имеет вид:

$$\phi^2 - t\phi + q = 0.$$

По теореме Хассе число точек кривой E ограничено и равно:

$$u = q + 1 - t, \text{ где } |t| \le 2\sqrt{q},$$

где t – след эндоморфизма Фробениуса на кривой E.

Заметим, что на кривой E всегда существует точка $(0, \sqrt{b})$ порядка 2 и, следовательно, порядок кривой кратен 2, $2 \mid u$. Если на кривой существует точка $(\sqrt[4]{b}, \sqrt{b})$ порядка 4, то соответствующий порядок кривой будет кратен 4.

Маленький автоморфизм Фробениуса σ задает отображение $x \mapsto x^2$, которое может быть расширено для изогении кривой E, сопряженной кривой $E^\sigma : y^2 + xy = x^3 + b^2$, следовательно:

$$\sigma : E(\mathrm{F}_q) \to E^\sigma(\mathrm{F}_q),$$
$$(x, y) \mapsto (x^2, y^2).$$

1 Основные методы вычисления порядка случайной эллиптической кривой на поле F_q

К основным методам вычисления порядка случайной эллиптической кривой необходимо отнести полиномиальные методы Скуфа, SEA, Сато.

В 1985 году Скуф [8] предложил первый полиномиальный алгоритм, основанный на эндоморфизме Фробениуса ϕ в подгруппах точек ℓ-кручения. Данный алгоритм позволяет вычислить количество точек на эллиптической кривой E на конечном поле F_q.

Аткин и Елкис [9] предложили некоторые уточнения по использованию сомножителей *полиномов деления,* полученных из вычисления некоторых *изогений* на поле характеристики $char(\mathrm{F}_q) > 3$. Лерсье и Ковергенц [10] построили эллиптические кривые на расширенных полях малой характеристики и получили изогении для характеристики $char(\mathrm{F}_q) = 2$ и 3.

Совокупность данных исследований позволила вычислить порядок эллиптической кривой на простом поле F_p, где $p = 10^{499} + 15$, и на расширенном поле $\mathrm{F}_{2^{1999}}$, используя алгоритм Schoof-Atkin-Elkies (SEA)[11].

Основной идеей алгоритма Скуфа является нахождение следа кривой по модулю маленьких простых чисел ℓ, используя свойства эндоморфизма Фробениуса ϕ в точках ℓ-кручения на кривой E.

Обозначим $\bar{\phi}$ сокращение ϕ в некоторой подгруппе $E[\ell]$ точек ℓ-кручения на кривой E. В результате получаем уравнение:

$$\bar{\phi}^2 - [t \bmod \ell^n]\phi + [q \bmod \ell^n] = 0.$$

Находим значение τ. Для этого выполняем поиск по всем значениям τ из интервала $[0, \ell-1]$, удовлетворяющее тождеству $[\tau]\bar{\phi} \equiv \bar{\phi}^2 + [q \bmod \ell]$.

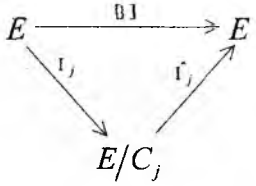
Согласно следствию из теоремы Хассе, что порядок кривой ограничен $|t| \le 2\sqrt{q}$, достаточно подсчитать значения t по модулю ℓ_i, где количество маленьких простых чисел ℓ_i ограничено следующим уравнением:

$$\prod \ell_i > 4\sqrt{q}.$$

Используя остатки τ_i, можно вычислить значение следа Фробениуса t, воспользовавшись китайской теоремой об остатках, и тем самым получить порядок кривой.

Для вычисления точки $[q](X,Y)$ необходимо найти многочлены деления $f_\ell(X)$ степени $O(\ell^2)$. Решение этой задачи, а также вычисление значений X^q и Y^q составляют основную вычислительную сложность метода Скуфа. Все вычисления выполняются в кольце $F[X,Y]/(f_\ell(X), E)$. При этом размер элементов поля равняется $\ell^2 \log q$. Вычислительная сложность метода Скуфа с использованием быстрой арифметики равна $O(\log^{5+\tau} q)$ и с обычной арифметикой – $O(\log^8 q)$. Использование данного алгоритма при значениях простого ℓ, большего 61, является неэффективным, потому что требует больших вычислительных затрат.

В основе алгоритма SEA [9, 11, 12] лежит базовый алгоритм Скуфа. Отличие алгоритма SEA заключается в том, что Аткин и Елкиес попытались уменьшить степени многочленов деления $f_\ell(X)$, используя только некоторый сомножитель степени $\frac{\ell-1}{2}$. Они рассмотрели подгруппу $E[\ell]$, образованную двумя точками P и Q, содержащую $\ell+1$ подгрупп. В этом случае изогении I описывают мультипликативный эндоморфизм $[\ell]$, который представлен следующей диаграммой:



Следовательно,

$$\mathrm{Ker}(\mathrm{I}_j) \subset \mathrm{Ker}([\ell]),$$

где $\#\mathrm{Ker}(\mathrm{I}_j) = \ell$.

Построив изогении I_j и $\hat{\mathrm{I}}_j$, можно вычислить сомножитель $h(X)$ многочлена деления $f_\ell(X)$. Использование модулярного уравнения $\Phi_\ell(j(E_\ell), j(E)) = 0$ позволяет эффективным образом определять существование изогении I.

Как и в алгоритме Скуфа, вычисления производятся в кольце $F[X,Y]/(h_\ell(X), E)$ при использовании полинома $h_\ell(X)$ степени $\frac{\ell-1}{2}$. Поэтому вычислительная сложность алгоритма SEA меньше за счет использования полиномов степени $O(\ell)$ и равна $\log^5 p$.

Например: используя алгоритм SEA, Морейн смог вычислить порядок кривой, определенной на простом поле размера $10^{499}+153$ [10].

Алгоритм Сато [13,14] коренным образом отличается от алгоритмов SEA и Скуфа и реализуется двумя этапами. Первый этап – поднятие цикла изогений и коэффициентов кривых на $\mathbf{Z}_q$ с требуемой точностью, второй – вычисление следа эндоморфизма Фробениуса. Алгоритм Сато имеет следующий вид:

Задано: Эллиптическая кривая E, определенная на поле F_q, j-инвариант, который удовлетворяет условию $j \notin \mathrm{F}_{p^2}$.

Получаем: След эндоморфизма Фробениуса на эллиптической кривой E. При этом выполняется:

1. Поднятие j-инвариантов, коэффициентов кривых и их подгрупп p-кручения, при этом предварительно вичислив цикл d кривых E_i и их j-инвариантов j_i;
2. Подсчет следа на $\mathbf{Z}_q$.

Рассматривая маленький эндоморфизм Фробениуса σ в поле F_q, строятся циклы j-инвариантов.

При использовании многомерной итерации Ньютона производится их одновременное поднятие. Таким образом, не вычисляя эндоморфизма Фробениуса Σ в $\mathbf{Z}_q$, можно получить сопряженные j-инварианты J в $\mathbf{Z}_q$. Поднимаются коэффициенты кривых, используя одномерную итерацию Ньютона, последовательно поднимаются сами кривые.

Изогении $\hat{\sigma}_i$ и $\hat{\Sigma}_i$ сепарабельны в степени p, они определяются своим ядром порядка p. Следовательно, поднятие $\hat{\sigma}_i$ к $\hat{\Sigma}_i$ осуществляется поднятием их ядра, которое является подгруппой p-кручения.

Можно поднимать каждую подгруппу p-кручения или поднимать единственную точку, используя итерацию Ньютона, когда $p=2$ и $p=3$. В случае характеристики $p \geq 5$ необходимо поднимать коэффициент многочлена p-деления с помощью поднятия Хэнселя. Схематично первый этап можно представить следующим образом:

Задано: Цикл эллиптических кривых E_i, определенных на поле F_q.

Получаем: Цикл эллиптических кривых $\mathbf{Z}_q : \mathrm{E} \to \mathrm{E}_1 \to \ldots \to \mathrm{E}_{d-1} \to \mathrm{E}_0$ и их подгруппы p-кручения. При этом производится:

1. Подсчет цикла d кривых E_i и их j-инварианта j_i;
2. Поднятие всех j_i, которые нам дает J_i;
3. Поднятие каждой кривой при поднятии ее коэффициентов;
4. Поднятие подгруппы p-кручения для каждой кривой.

Задачей второго этапа алгоритма Сато является вычисление следа эндоморфизма Фробениуса $F : E \to E$. Предположим, что получено каноническое поднятие E кривой E, которая имеет свойство, $\mathrm{Tr}\, F = \mathrm{Tr}\, \mathrm{F}$, где F – эндоморфизм Фробениуса кривой E на $\mathbf{Z}_q$. След эндоморфизма равен следу дуального эндоморфизма:

$$\mathrm{Tr}\, F = \mathrm{Tr}\, \mathrm{F} = \mathrm{Tr}\, \hat{\mathrm{F}}.$$

Эндоморфизм Фробениуса F можно разложить на произведение малых эндоморфизмов Фробениуса, которые проходят через цикл d кривых E. Такое разложение применимо и для сопряженных кривых E. Получаем:

$$\mathrm{Tr}\, F = \mathrm{Tr}\left(\hat{\Sigma}_{d-1} \circ \hat{\Sigma}_{d-2} \circ \ldots \circ \hat{\Sigma}_0\right).$$

Следующий шаг описывает переход в формальные группы кривой E и ее сопряжений. Обозначим через τ локальный параметр $-X/Y$ кривой E и τ_i – локальный параметр кривых E_i. Выражая действие $\hat{F}$ через этот параметр, получаем:

$$\hat{F}(\tau) = \sum_{k \geq 1} c_k \tau^k .$$

Далее, используя утверждение Сато, что уравнение $\phi^2 - \mathrm{Tr}\left(\hat{F}\right)\phi + q = 0$, след эндоморфизма Фробениуса равен:

$$\mathrm{Tr}\, F = \mathrm{Tr}\left(\hat{F}(\tau)\right) = c_1 + \frac{q}{c_1}.$$

Таким образом, подсчета первого коэффициента достаточно для нахождения следа Фробениуса. Так как $\hat{F}$ является композицией морфизмов, то можно вычислить c_1 как произведение первых коэффициентов в разложении дуального эндоморфизма Фробениуса $\hat{F}$. Представим эндоморфизм $\hat{\Sigma}_i : E_i \to E_{i+1}$ в формальных группах как:

$$\hat{\Sigma}_i(\tau_i) = g_i \tau_i + O\left(\tau_i^2\right).$$

Тогда имеем:

$$c_1 = \prod_{0 \leq i \leq d} g_i .$$

Произведение g_i есть норма элемента g_0 из $\mathbf{Z}_q$ на $\mathbf{Z}_p$, а также это след F с точностью $O\left(p^d\right)$. Для увеличения точности необходимо добавить слагаемое $\frac{q}{c_1}$.

На втором этапе алгоритма Сато вычисляем g_i из кривых E_i и E_{i+1}, и ядра $\hat{\Sigma}_i$. Используя формулы Велу, получаем след эндоморфизма Фробениуса.

Рассмотрим алгебраическое расширение степени d поля F_q с p-адической точностью $O\left(p^{O(d)}\right)$. Каждый элемент $\mathbf{Z}_q$ требует для своего представления $O\left(d^2 \log p\right)$ памяти. При вычислении для каждой сопряженной кривой необходимо хранить его в памяти и при этом хранить d сопряженных кривых. Значит, общий объем памяти будет не менее $O\left(d^3 \log p\right)$. Сложность вычислений с использованием быстрой арифметики можно оценить как $O\left(d^3 \log d \log\log d\right)$ или с использованием обычной – $O\left(d^5\right)$, для фиксированной характеристики p.

Применение алгоритма Сато позволило вычислить эллиптическую кривую на поле 2^{8001} за 313 часов, при этом использовался объем памяти около 16 GB.

2 Сравнение алгоритмов Сато и SEA

При вычислении порядка кривой необходимо выбрать, какой из алгоритмов SEA или Сато использовать для вычисления порядка кривой. Поэтому необходимо рассмотреть ограничения алгоритма.

Применение алгоритма Сато ограничено характеристикой базового поля. Вычислительная сложность алгоритма Сато сильно зависимост от характеристики p базового поля, что вытекает из вычисления модулярного уравнения Φ_p для поднятия кривых, имеющего

$O(p^2)$ коэффициентов. При больших значениях характеристики p базового поля применение алгоритма Сато становится неэффективным, в то время как алгоритм SEA полиномиально зависит от p.

Алгоритм Сато более эффективен, по сравнению с алгоритмом SEA, для небольшой характеристики базового поля, особенно для характеристики 2. В таблице 1 приведены численные характеристики сложности алгоритмов SEA и Сато для расширенного поля F_{2^m}.

Таблица 1

Размер поля	155	196	300
SEA (сек)	86,5	308	2434
Сато(сек)	41	78	445,4
SEA/Сато	2	4	5,4

Сравнительный анализ результатов вычисления алгоритмов Сато и SEA показывает, что при увеличении размера поля применения алгоритма Сато предпочтительнее по сравнению с алгоритмом SEA. Этот связано с зависимостью степеней изогении от маленького простого числа ℓ в алгоритме SEA.

Основной сложностью подсчета кривой для криптографических приложений является подсчет числа точек многочисленных кривых до момента нахождения требуемой, криптографически стойкой, кривой. Данную задачу оптимально решает алгоритм SEA. Например, для анализа 1000 случайных кривых на поле F_{2^m} алгоритмом SEA при поиске «правильных» кривых потребовалось 14112 сек. В то время, как алгоритм Сато вычисляет одну кривую за 41 сек., на поиск правильных кривых потребуется около 41000 сек.

Для поиска криптографически стойкой кривой в связи с эффективностью вычисления порядка одной эллиптической кривой алгоритмом Сато предлагается использовать выявленную особенность по вычислению большого количества кривых алгоритмом SEA.

3 Нахождение криптографически стойких эллиптических кривых

Пусть кривая E определена над полем F_q. Будем называть кривую E «хорошей», если ее порядок при маленьких простых числах ℓ_i удовлетворяет условию $НОД(\#E, \ell_i) = 1$, и соответственно «плохой» кривой, если $НОД(\#E, \ell_i) \neq 1$.

Таким образом, попытаемся отбросить кривые, чей порядок кратен маленьким простым числам ℓ, данное действие назовем «решетом». Отсюда вытекает задача, как, не вычисляя порядка кривой, определить ее кратность числу ℓ.

Теорема *Если для некоторого простого числа ℓ, модулярное уравнение $\Phi_\ell(X, j(E))$ в своем разложении по модулю q не содержит простых корней, то порядок эллиптической кривой $\#E$ не кратен простому числу ℓ, $НОД(\#E, \ell_i) = 1$.*

Из теоремы получаем алгоритм «решета».

Алгоритм «решета»:

1. Для каждого маленького простого числа ℓ и эллиптической кривой строим модулярные уравнения Φ_ℓ;
2. Вычисляем корни модулярного уравнения $\Phi_\ell(X, j(E))$;
3. Если корней не существует, то кривая E – «хорошая»;

4. Если корни Φ_ℓ существуют, то для каждого из них необходимо построить соответствующий сомножитель многочлена деления $h_\ell(x)$ и найти его корень. В случае существования корня полинома $h_\ell(x)$ кривая E считается «плохой»;
5. Если ни один из корней Φ_ℓ не принадлежит кривой, то кривая E является «хорошей».

Использование данного алгоритма позволяет проанализировать порядок эллиптической кривой на кратность маленькому простому числу, не вычисляя порядка кривой. Таким образом, попробуем скомбинировать алгоритм «решета» с быстрым алгоритмом Сато:

1. Вначале исключим значительное число кривых алгоритмом «решета».
2. Затем, используя алгоритм Сато, вычислим порядки оставшихся кривых и тем самым найдем стойкие кривые.

Наиболее вычислительно трудоемкой операцией в алгоритме «решета» является вычисление X^q по модулю модулярного уравнения $\Phi_\ell(X, j(E))$, имеющего степень $\ell+1$. Для ускорения вычислений можно заменить Φ_ℓ на канонический модулярный полином Φ_ℓ^c.

Также необходимо найти такое максимальное значение ℓ, которое бы позволило сбалансировать вычислительные затраты между ℓ – проверками кривой и вычислением порядка кривой. Экспериментально было получено, что уже для $\ell \le 19$ эффективность применения этого комбинированного метода превосходит алгоритм SEA. В таблице 2 приведены вычисления модулярных полиномов для различных ℓ.

Таблица 2

	ℓ	3	5	7	11	13	17	19
Вычисление корня Φ_ℓ (мс)	$q = 2^{161}$	0,25	0,33	1,37	4,38	6,65	12,7	15,9
	$q = 2^{257}$	0,41	1,08	2,18	13,1	15,36	27,9	33,5

Таблица 2 может быть расширена вследствие увеличения размера базового поля. Поэтому для каждого значения базового поля будет соответствовать свое максимальное значение маленького простого числа ℓ.

Объединение алгоритмов «решета» и Сато позволило получить более эффективное решение, чем с алгоритм SEA. Нахождение криптографических кривых из 1000 случайных кривых в расширенном поле $F_{2^{155}}$, используя алгоритм Сато с алгоритмом «решета», потребовало 5218 сек., что в 8 раз быстрее, чем просто использовать алгоритм Сато.

В табл. 3 показан анализ 10000 кривых и приведено количество «хороших» кривых, оставшихся после выполнения алгоритма «решета», а также время вычисления алгоритма Сато для поиска криптографических кривых из оставшихся «хороших» кривых.

Таблица 3

Поле	Хорошие кривые	Крипт. кривые	Время работы (мин)
163	457	56	23,3
193	434	38	31,5
239	487	34	58,3
283	398	22	180
512	507	11	1460

Из табл. 3 видно, что использование алгоритма «решета» позволяет с ростом размера поля существенно уменьшить вычислительные затраты. Так, при использовании алгоритма Сато потребовалось 713 час, что в 27 раз больше, чем при использовании алгоритма Сато с алгоритмом «решета».

Выводы

Применение алгоритма Сато для вычисления порядка одной эллиптической кривой, определенной на расширенном поле характеристики 2, эффективно. Но, несмотря на это, алгоритм Сато не позволяет вычислить порядок кривой на простом поле большой характеристики, и в этом случае лучшим алгоритмом остается алгоритм SEA. Для увеличения эффективности нахождения криптографически стойкой кривой алгоритмом Сато был применен алгоритм «решета».

При совместном применении двух данных алгоритмов появилась возможность отказаться от использования предварительно вычисленных кривых, так как построение новой кривой будет проходить за несколько секунд. Таким образом, в криптографии получен эффективный метод для нахождения криптографически стойких кривых, используя при вычислении порядка эллиптической кривой комбинированный алгоритм Сато с алгоритмом «решета» для полей характеристики 2.

Список литературы: 1. *И.Д. Горбенко, С.И. Збитнев, А.А. Поляков* Криптоанализ криптографических преобразований в группах точек эллиптических кривых методом полларда // Радиотехника: Всеукр. межвед. науч-тех. сб 2001. Вып. 119. С. 43 – 50. 2. *S. Pohlig, M. Hellman* An improved algorithm for computing logarithm over *GF (p)* and its cryptographic significance, IEEE Translation on Information Theory, vol. 24, pp. 106 – 110, 1978. 3. *FIPS 186-2,* Digital signature standard, National Institute of Standards and Technology, 2000. 4. *ANSI X9.62-1999*, Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999. 5. *IEEE P1363* / D9 (Draft Version 9). Standard Specifications for Public Key Cryptography. Number-Theoretic Background. 1999. 6. *J. Silverman*, Advanced Topic in the Arithmetic of Elliptic Curves, Graduate Text in Math., Vol. 151, Springer-Verlag, Berlin and New York, 1994. 8. *R. Schoof* Elliptic curves over finite fields and the computation o square roots mod p, Mathematics of Computation, 55 (1990), 745 – 763. 7. *J.H. Silverman*. The arithmetic of elliptic curve, volume 106 of Graduate Texts in Mathematics. Springer, 1986. 9. *R. Lercier*. Finding good random elliptic curves for cryptosystems defined over F_{2^n}. In W. Fumy, editor, Advanced in Cryptology – EUROCRYPT '97, Vol. 1233 of Lecture Notes in Comput. Sci., pages 379 – 392. Springer-Verlag, 1997. International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 1997, Proceeding. 10. *Schoof R.* Counting points on elliptic curve over finite fields, Journal de Theories des Nombres de Bordeaux, Vol. 7, pp. 219 – 254, 1995. 11. *R. Lercier and A. Morain*. Counting the number of points on elliptic curves over finite fields: Strategies end performances. In L. C. Guillou and J.-J. Quisquater, Editors, Advanced in Cryptology – EUROCRYPT '95, Vol. 921 of Lecture Notes in Comput. Sci., pp. 79 – 94. Springer-Verlag, 1995. International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 1995, Proceeding. 12. *L. Dewaghe.* Remarks on the Schoof-Elkies-Atkin algorithm. Math. Comp., 67(233): 1247 – 1252, July 1998. 13. *B. Skiernaa.* Satoh's algorithm in characteristic 2. 14. *T. Satoh.* The canonical lift of an ordinary elliptic curve over a finite field and its point counting. J. Ramanujan Math. Soc., 15: 247 – 270, 2000.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 12.05.2003

УДК 681.3.06:519.248.681

Е. Г. КАЧКО, канд. техн. наук, А. Е. ТАРАКАНОВ

ОПТИМИЗАЦИЯ ОПЕРАЦИЙ МНОГОКРАТНОЙ ТОЧНОСТИ

В настоящее время широко используются криптографические методы для организации защиты информации. Особенностью современных криптографических методов является то, что алгоритмы преобразований известны. Секретными являются ключи. В связи с этим для исключения атаки, связанной с полным перебором ключевых данных, их длины выбираются большими. На сегодняшний день стандартными размерами ключевых данных являются 256, 512 и 1024 бит (в некоторых системах используются более длинные ключи). Поэтому при построении систем защиты проблема работы с длинными числами является актуальной. Безусловно, крайне важным требованием к подобным операциям является их максимальная скорость выполнения. В настоящее время существует большое количество различных библиотек многократной точности, доступных, например, в Интернете. Но было бы интересно узнать, так ли они быстры, как говорится в их документации. Ведь скорость работы программы зависит не только от выбранного алгоритма, но и от его реализации на конкретной машине. Что же касается оптимизации кода, то в этой сфере современные процессоры предоставляют огромное поле для деятельности. В первую очередь, это новые технологии, реализуемые на аппаратном уровне, такие, как расширения SSE2 процессора Intel Pentium 4.

Данная статья посвящена проблеме уменьшения вычислительной сложности арифметических операций над числами многократной точности. В частности, рассматриваются алгоритмы, наиболее оптимальные для длин чисел, применяющихся в большинстве криптографических методов, а также описание их эффективной реализации на современных процессорах.

Обзор существующих алгоритмов

Рассмотрим последовательно арифметические операции и методы, применяемые в настоящее время для их реализации при работе с длинными числами [2].

Сложение и вычитание

Повсеместно использующимся способом для выполнения данных операций является хорошо известный школьный метод сложения или вычитания «в столбик» с попарным сложением или вычитанием соответствующих разрядов чисел и последовательным учетом переносов. Время выполнения данного алгоритма линейно зависит от количества цифр числа, т.е. $T(n) = O(n)$. Трудно предложить какой-то иной, более эффективный способ выполнения сложения или вычитания длинных чисел.

Умножение

Классическим методом для выполнения данной операции также является школьное умножение в столбик. Все цифры одного числа последовательно умножаются на каждую цифру другого числа, выполняется учет переносов и сложение с результатами умножений на предыдущие разряды числа. Очевидно, что между временем выполнения алгоритма и длиной числа существует квадратичная зависимость: $T(n) = O(n^2)$. Несмотря на такую асимптотическую оценку, данный метод является одним из самых эффективных алгоритмов умножения чисел, применяющихся в большинстве криптографических протоколов. Во все библиотеки многократной точности непременно входит процедура вычисления произведения длинных чисел, основанная именно на умножении «в столбик».

Для выполнения данной операции можно также предложить следующую модификацию классического метода. Вместо обычного умножения, учета переноса и сохранения очередной цифры результата, можно выполнять накопление результата по столбцам и только после

суммирования всего столбца сохранять очередную цифру ответа [5]. При использовании чисел многократной точности с длиной цифры, меньшей максимального размера слова компьютера, данный подход дает реальный выигрыш в скорости за счет уменьшения количества обращений к памяти и команд внутри главного цикла процедуры. Однако в случае равенства длины цифры и максимального слова процессора ситуация изменяется. Даже при условии, что ведется умножение чисел ограниченной длины, для хранения суммы столбца возникает потребность в некоторой ячейке тройной точности. А это связано с дополнительными обращениями к памяти, что негативно сказывается на производительности алгоритма.

Еще одним методом умножения является алгоритм, основанный на использовании следующей формулы:

$$U \cdot V = (2^{2n} + 2^n) \cdot U_1 \cdot V_1 + 2^n \cdot (U_1 - U_0) \cdot (V_0 - V_1) + (2^n + 1) \cdot U_0 \cdot V_0 .$$

Т.е. умножение двух $2n$-разрядных чисел сводится к трем операциям умножения n-разрядных чисел $U_1 \cdot V_1$, $(U_1 - U_0) \cdot (V_0 - V_1)$ и $U_0 \cdot V_0$, а также к нескольким операциям сложения и вычитания. Поскольку операция умножения является одной из самых медленных арифметических операций в компьютере, то уже замена четырех умножений на три теоретически должна давать преимущества. Важно то, что на основе данной формулы можно построить рекурсивный алгоритм, дающий время выполнения $T(n) = O(n^{\log 3}) = O(n^{1,585})$ вместо традиционного n^2 [1]. Однако при реализации данного метода возникают дополнительные расходы на рекурсию и взаимодействие с временной памятью, сводящие на-нет его преимущества на многих процессорах. Несмотря на это, на отдельных компьютерах (при определенной оптимизации кода процедуры) данный алгоритм дает выигрыш.

Что касается таких методов умножения, как алгоритм Тоома-Кука, быстрое преобразование Фурье [1], то, хотя их асимптотические оценки, несомненно, лучше, чем у алгоритмов, рассмотренных выше, фактически же их применение для чисел длиной не более 1024 бит не дает никаких преимуществ. Напротив, традиционные алгоритмы являются более эффективными.

Умножение на число однократной точности

Эта арифметическая операция является частным случаем операции умножения длинных чисел. Наиболее эффективным алгоритмом является все то же умножение «в столбик», дающее в данном случае линейную асимптотическую характеристику.

Деление

Основным алгоритмом, применяющимся для деления длинных чисел, является традиционный школьный метод, на каждом шаге которого на основании старших цифр делимого и делителя находится очередная цифра частного, которая умножается на все цифры делителя и вычитается из делимого. Время выполнения данного алгоритма дает квадратичную зависимость.

Для выполнения операции деления используется также метод, основанный на том, что $\frac{U}{V} = U \cdot \frac{1}{V}$, т.е. для того, чтобы разделить U на V, можно найти приближение к $\frac{1}{V}$ и умножить данную величину на U [1]. Для нахождения обратной величины используется метод Ньютона. Вычислительная сложность получения обратного элемента достаточно велика, и данный алгоритм может иметь смысл, если необходимо многократно выполнять деление на одно и тоже число. Недостатком данного способа является и то, что вычисление остатка от деления (данная операция необходима в криптографии) также затруднительно.

Приведенный анализ показывает, что в настоящее время не известны алгоритмы, позволяющие существенно увеличить скорость вычислений для арифметических операций по

сравнению с традиционными методами. В связи с этим, для уменьшения их вычислительной сложности необходимо максимально учитывать аппаратные особенности современных процессоров. При оптимизации необходимо исходить из того, что разработанный код должен быть наиболее эффективным для данного типа процессора и процессоров других типов.

Характеристика разработанной библиотеки

В ходе исследований была разработана библиотека функций, реализующих основные арифметические операции над числами многократной точности, а именно:

- сложение;
- вычитание;
- умножение;
- умножение на число однократной точности;
- деление с получением частного и остатка.

Код библиотеки написан исключительно на языке ассемблера для обеспечения максимальной эффективности. С целью достижения переносимости кода при разработке использовался компилятор NASM, позволяющий получать объектные файлы как для Windows, так и для Linux.

В библиотеке имеются две процедуры, реализующие операцию умножения. Во-первых, это функция, выполняющая умножение «в столбик». Во-вторых, это функция, использующая рекурсивный алгоритм, требующий трех операций умножения и нескольких операций сложения на каждом уровне рекурсии вместо обычных четырех. Как уже отмечалось, непосредственная реализация рекурсии и ряда циклов, имеющихся в алгоритме, приводит к тому, что скорость выполнения такой программы в несколько раз меньше скорости работы классического метода.

Однако существует иной подход. Используя, например, макросредства, поддерживаемые большинством существующих ассемблеров, и ориентируясь на некоторую определенную длину числа, можно заменить физические рекурсивные вызовы функций и циклические участки программы вызовами макросов. В итоге, после препроцессорной обработки может быть получена программа с полностью развернутыми циклами и рекурсивными обращениями. Несмотря на значительный размер кода такой функции (для чисел длиной 1024 бита – более 40 Кб), на современных процессорах (AMD Athlon XP, Intel Pentium 4) в ряде случаев был получен выигрыш в скорости примерно на 15-25% по сравнению с умножением «в столбик». Необходимо заметить, что применение чисел фиксированной длины оправдано тем, что современные криптографические стандарты предполагают работу именно с такими числами. Таковым, например, является ГОСТ 34.310-95.

Библиотеки, участвующие в тестировании

Анализ скорости выполнения реализованных операций проводился в сравнении с функциями существующих библиотек. В тестировании участвовали следующие библиотеки:

- Библиотека MIRACL фирмы Shamus Software. В ней реализованы практически все операции, используемые в современных криптографических системах. Библиотека написана на C/C++. Благодаря наличию ряда мер, направленных на оптимизацию ее функций (использование ассемблерных вставок, макросредств), считается одной из наиболее быстрых и эффективных библиотек.
- Библиотека многократной арифметики ММАТН АО «ИИТ». Представляет собой набор основных функций, применяемых в криптографии. Будучи написанной полностью на языке ассемблера, она является неплохим образцом для сравнения с процедурами разработанной библиотеки.

Результаты тестов

С целью анализа производительности разработанной библиотеки было выполнено сравнение скорости выполнения ее функций с аналогичными функциями библиотек MMATH и MIRACL. Конфигурации компьютеров, принимавших участие в тестировании, приведены в табл. 1. Далее подробно рассмотрены результаты тестирования каждой арифметической операции.

Таблица 1

Обозначение	Конфигурация
[1]	Intel Pentium MMX, 166 МГц, 16 Мб RAM, ОС Windows 98
[2]	AMD Athlon XP 1800+, 1533 МГц, 256 Мб RAM, ОС Windows 2000
[3]	Intel Pentium 4, 2 ГГц, 512 Мб RAM, ОС Windows XP

Сложение и вычитание

Проверка скорости выполнения данных операций выполнялась на случайным образом сгенерированных числах длинами 32*4 байт (первый операнд) и 8*4, 16*4, 24*4, 32*4 байт (второй операнд). Количество итераций цикла – 1000000 раз. Полученные результаты приведены в таблицах 2 (для операции сложения) и 3 (для операции вычитания).

Таблица 2

CPU	Длины операндов	Время выполнения (мс)		
		Разработанная библиотека	MMATH	MIRACL
[1]	32*4 × 8*4	1606	3219	5279
	32*4 × 16*4	1605	2829	7236
	32*4 × 24*4	1606	2926	9183
	32*4 × 32*4	1605	2926	11197
[2]	32*4 × 8*4	78	190	292
	32*4 × 16*4	88	201	389
	32*4 × 24*4	86	205	477
	32*4 × 32*4	89	205	567

Таблица 3

CPU	Длины операндов	Время выполнения (мс)		
		Разработанная библиотека	MMATH	MIRACL
[1]	32*4 × 8*4	1602	3354	4830
	32*4 × 16*4	1603	2944	6531
	32*4 × 24*4	1640	3033	8137
	32*4 × 32*4	1692	3063	9633
[2]	32*4 × 8*4	84	181	297
	32*4 × 16*4	97	191	421
	32*4 × 24*4	97	189	516
	32*4 × 32*4	105	194	592

Как следует из экспериментально полученных результатов, быстродействие функции сложения разработанной библиотеки в среднем в 1,9 – 5,1 раз (на CPU Intel Pentium MMX) и 2,4 – 5 раз (на CPU AMD Athlon XP) превышает быстродействие аналогичных функций рассмотренных библиотек.

Что касается операции вычитания разработанной библиотеки, то ее выигрыш в скорости составляет в среднем 1,9 – 4,4 раз (на CPU Intel Pentium MMX) и 2 – 4,7 раз (на CPU AMD Athlon XP).

Следует отметить, что такое отставание функций сложения и вычитания библиотеки MIRACL от процедур разработанной библиотеки (да и библиотеки MMATH) связано с тем, что их код написан полностью на C/C++. Значительное же преимущество разработанных процедур объясняется их тщательной низкоуровневой оптимизацией [3,4].

Умножение на число однократной точности

Анализ скорости выполнения данной операции осуществлялся на числах длиной 32*4 байт (первый операнд) и 4 байта (второй операнд). Число итераций цикла – 1000000 раз. В тесте принимала участие функция разработанной библиотеки, а также библиотеки MIRACL. Полученные результаты приведены в табл. 4.

Таблица 4

CPU	Длины операндов	Время выполнения (мс)	
		Разработанная библиотека	MIRACL
[1]	32*4 × 4*4	2943	5382
[2]	32*4 × 4*4	157	298

Из экспериментальных результатов можно заключить, что быстродействие разработанной процедуры в 1,8 раз (на CPU Intel Pentium MMX) и в 1,9 раз (на CPU AMD Athlon XP) превышает быстродействие функции библиотеки MIRACL, написание которой на C/C++ не лучшим образом сказалось на ее скорости выполнения.

Умножение

Производительность данной операции оценивалась на случайных числах длиной 32*4 байт (оба операнда). Число итераций цикла – 100000 раз. Таблица 5 демонстрирует полученные результаты.

Таблица 5

CPU	Время выполнения (мс)			
	Разработанная библиотека		MMATH	MIRACL
	«В столбик»	Макро		
[1]	9638	23600	10206	10358
[2]	473	423	553	497

Как следует из экспериментально полученных результатов, быстродействие функции умножения, реализованной на основе алгоритма умножения «в столбик», разработанной библиотеки на 6–7% (на CPU Intel Pentium MMX) и 5–15% (на CPU AMD Athlon XP) превышает быстродействие аналогичных функций MMATH и MIRACL.

Что касается процедуры умножения, основанной на полной раскрутке циклов и рекурсии, то, хотя ее производительность в 2,2 раза меньше на CPU Intel Pentium MMX, на процессоре AMD Athlon XP получен реальный выигрыш в скорости на 15-24%.

Необходимо отметить довольно тщательную проработку функции умножения библиотеки MIRACL – за счет ассемблерных вставок в ее главном цикле она продемонстрировала достойные скоростные характеристики (обогнать ее даже на 6-7% было не так легко).

Деление

Проверка скорости выполнения данной операции (также как и процедур сложения и вычитания) выполнялась на случайным образом сгенерированных числах длинами 32*4 байт (первый операнд) и 8*4, 16*4, 24*4, 32*4 байт (второй операнд). Количество итераций цикла – 100000 раз. Результаты тестирования приведены в табл. 6.

Таблица 6

CPU	Длины операндов	Время выполнения (мс)		
		Разработанная библиотека	MMATH	MIRACL
[1]	32*4 × 8*4	4185	9021	5515
	32*4 × 16*4	4705	8863	5687
	32*4 × 24*4	3481	6652	4988
	32*4 × 32*4	517	1812	448
[2]	32*4 × 8*4	295	459	335
	32*4 × 16*4	330	460	347
	32*4 × 24*4	241	347	325
	32*4 × 32*4	34	108	77

Из результатов эксперимента видно, что быстродействие функции деления разработанной библиотеки в среднем в 1,2–2,4 раз (на CPU Intel Pentium MMX) и 1,5–1,9 раз (на CPU AMD Athlon XP) превышает быстродействие аналогичных функций библиотек MMATH и MIRACL.

Все та же низкоуровневая оптимизация и тщательная проработка кода разработанной процедуры позволила добиться такого выигрыша.

Использование расширений SSE2

С выходом в свет процессора Intel Pentium 4, а точнее, его расширений SSE2 [3,4] появилась уникальная возможность увеличения скорости выполнения операций над длинными числами. В отличие от SIMD-расширений предыдущих процессоров (MMX, SSE), блок команд SSE2 позволяет выполнять одновременные операции над целыми числами длиной до 64 бит. Следует подчеркнуть, что большую ценность из себя представляют именно целочисленные команды SSE2. Благодаря их наличию, цифра числа многократной точности может рассматриваться как целое число длиной 32 бита для любых арифметических операций. Что касается расширений MMX, то их использование ограничивает размер цифры 16 битами. В расширения SSE же вообще не входят команды, оперирующие над целочисленными данными. Наиболее примечательными командами блока SSE2 являются инструкции, расширяющие размер операндов целочисленного блока MMX до 128 бит. Например, такие команды, как paddq, psubq выполняют одновременное сложение или вычитание двух 64-разрядных беззнаковых целых чисел, а команда pmuludq – умножение двух 32-разрядных беззнаковых целых чисел с получением двух 64-битных результатов.

Важной особенностью расширений SSE2 является наличие в их составе команд управления кэшированием. Использование этих команд совместно с потоковыми инструкциями пересылки данных позволяет достичь максимальной скорости выполнения программы.

В ходе исследований проверка производительности команд блока SSE2 осуществлялась на примере операции копирования памяти. Было разработано две программы: одна – на основе базовых инструкций процессора, другая – с использованием SSE2. В первом случае для копирования блока памяти применялась обычная строковая команда rep movsd, во втором имел место следующий алгоритм: страница памяти (4096 байт) предварительно помещалась в кэш второго уровня 32-мя командами prefetchnta (размер линейки кэша второго уровня – 128 байт) и далее последовательно перемещалась командами movdqa и movntdq с целью минимального загрязнения кэша. Проверка программы осуществлялась на блоке данных длиной 100 страниц (100*4096 байт) с числом итераций цикла 10000 раз. Следует заметить, что для более полного анализа скорости выполнения программы тестирование проводилось не только под управлением операционной системы, но и в «голом» защищенном режиме, а именно, с запрещенными аппаратными и программными прерываниями, flat-памятью и отключенной страничной адресацией. Результаты тестирования приведены в табл. 7.

Таблица 7

CPU	Среда	Время выполнения (с)	
		Базовые команды	SSE2
[2]	ОС	15,44	——
	РМ	16,26	——
[3]	ОС	9,9	3,83
	РМ	8,55	2,49

Как следует из результатов тестирования, операция копирования памяти, использующая команды SSE2, в среднем в 2,9 раз быстрее аналогичной операции, выполняемой на процессоре Intel Pentium 4, и в 5 раз быстрее операции, выполняемой на процессоре AMD Athlon XP. Интересно отметить тот факт, что время выполнения операции в непосредственном защищенном режиме на процессоре AMD Athlon XP больше времени выполнения программы под управлением ОС. Что касается Intel Pentium 4, то в этом случае наблюдается совершенно противоположная картина.

Результаты тестирования показали, что использование блока SSE2 (а точнее, команд управления кэшированием), несомненно, дает реальные преимущества при работе с большими объемами памяти. Но что можно сказать о применении расширений SSE2 собственно в программировании арифметических операций над числами многократной точности?

В разработанной библиотеке была реализована процедура умножения длинного числа на 32-битное число однократной точности, использующая целочисленные команды блока SSE2. В основу программы положен стандартный алгоритм умножения «в столбик». Следует отметить следующие ключевые моменты. Длина цифры числа многократной точности принимается равной 128 бит. Это означает, что в программе используются команды пересылки 128-битных данных (movdqa, movntdq), т.е. в четыре раза сокращается число обращений к памяти по сравнению с аналогичной процедурой, использующей базовые 32-битные команды. Умножение 128-битного числа (или четырех 32-битных чисел) на 32-битное с получением четырех 64-битных результатов выполняется двумя командами pmuludq вместо обычных четырех инструкций mul. Главной проблемой, возникшей при программировании процедуры, является учет переносов. Из-за отсутствия в блоке SSE2 встроенных средств контроля переполнения при выполнении арифметических операций, а также за неимением инструкции беззнакового сравнения по больше/меньше среди его команд, реализация учета переносов оказалась довольно громоздкой. Команды, осуществляющие данную функцию, составляют примерно 60% от общего числа команд главного цикла процедуры.

Программа была реализована в двух вариантах: с использованием команд управления кэшированием и без их использования. Первый вариант ориентирован на работу с длинными числами, размеры которых кратны странице (4096 байт): на каждой итерации цикла осуществляется перемещение текущей страницы в кэш второго уровня командами prefetchnta, после чего выполняется собственно умножение с сохранением каждой цифры результата командой movntdq.

Было произведено сравнение производительности данных программ и процедуры, реализованной на основе базовых инструкций. Первый вариант программы тестировался на числах длиной 4096*10 байт с количеством итераций цикла 100000 раз, второй вариант – на числах длиной 32*4 байт и числом итераций 1000000 раз. Анализ скорости выполнения осуществлялся как под управлением операционной системы, так и непосредственно в защищенном режиме. Полученные результаты приведены в табл. 8.

Таблица 8

CPU	Среда	Время выполнения (мс)		Время выполнения (с)		
		32*4 × 4*4		10*4096 × 4*4		
		Базовые команды	SSE2	Базовые команды	SSE2	SSE2 и управление кэшированием
[2]	ОС	157	——	4,62	——	——
	РМ	193	——	5,4	——	——
[3]	ОС	414	203	12,35	8,16	6,2
	РМ	249	202	7,17	7,91	6,15

Исходя из экспериментально полученных результатов, можно сделать следующие выводы. Во-первых, для размеров чисел 32*4 байт преимущества программы, основанной на SSE2, перед процедурой, выполненной на базе основных команд, очевидны. Что касается длин чисел 10*4096 байт, то здесь наблюдается та же картина: программа, использующая расширения SSE2 вместе с командами управления кэшированием, оказалась быстрее процедуры, реализованной только на базе арифметических команд SSE2, а та, в свою очередь, – обычной функции, включающей в себя основные инструкции процессора. Однако это справедливо лишь для программ, тестируемых под управлением ОС. В непосредственном защищенном режиме при выполнении тестов на числах длиной 32*4 байт традиционная процедура вплотную «подбирается» к программе, использующей SSE2, а на числах с размерами 10*4096 байт – даже ее обгоняет. Во-вторых, следует констатировать тот факт, что процессору Intel Pentium 4 так и не удалось обогнать AMD Athlon XP. Это касается как программы, использующей базовые команды, так и процедуры, выполненной на основе SSE2. Похоже, что давно высказываемое мнение о низкой производительности ядра процессора Intel Pentium 4, отвечающего за выполнение основных инструкций, является небезосновательным.

Таким образом, использование низкоуровневой оптимизации позволяет существенно уменьшить вычислительную сложность всех арифметических операций над длинными числами для наиболее распространенных типов современных процессоров.

Список литературы: 1. *Кнут Д.* Искусство программирования. Т. 2. М: Мир, 1977. 728 с. 2. *Качко Е.Г., Свинарев А..В., Горбенко ИД., Мельникова О.А.* Программирование операций многократной точности // Безопасность информации. К.: 1995. .№ 1. С. 18 – 22. 3. *Intel Corporation.* IA-32 Intel Architecture Software Developer's Manual. Vol. 1 – 3. 4. *Intel Corporation.* Intel Pentium 4 and Intel Xeon Processor Optimization. 5. *Gourdon X.* Arbitrary precision computation. http://numbers.computation.free.fr/Constants/constants.html.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 29.04.2003

УДК 681.3.06

Г. З. ХАЛИМОВ, канд. техн. наук

БЕЗУСЛОВНАЯ АУТЕНТИФИКАЦИЯ С ИСПОЛЬЗОВАНИЕМ СЛАБО СМЕЩЕННЫХ МАССИВОВ

Безусловная аутентификация, предложенная Г.Ж. Симмонсом [1, 2], основывается на использовании семейства почти строго универсальных хэш-функций. Аутентификация Симмонса снимает ограничение на большую размерность ключевых данных в конструкции кодов аутентификации Картера-Вергмана [3, 4]. Семейство почти строго универсальных хэш-функций можно определить в рамках теории мало смещённых, почти независимых массивов (weakly biased arrays, almost independent arrays). Определение смещённых массивов введено в работах [5, 6] для массивов дискретных значений большой размерности с распределением незначительно отличающимся от равномерного. Хеллесет и Джохансон впервые установили взаимосвязь теории кодирования и смещённых массивов [7]. Применение в схемах аутентификации незначительно смещённых, почти независимых массивов с использованием алгеброгеометрических кодов, как будет показано в статье, позволяет получить при заданном объёме ключа и данных источника сообщений вероятность коллизии не больше, чем в случае использования строго универсальных хэш-функций.

Задачей данной статьи является изложение общетеоретических вопросов построения аутентификации с применением почти строго универсального хэширования на основе слабо смещенных, почти независимых массивов в конструкции с алгебраическими кодами. С этой целью в разделе 1 приводятся определения смещённых и зависимых массивов. В разделе 2 рассматривается применение теории кодирования для построения мало смещённых массивов в схемах аутентификации.

1 Определение смещённых и зависимых массивов

Понятия ограниченной зависимости, слабого смещения используются в различных приложениях криптографии и теории сложности: при аутентификации, универсальном хэшировании, оценке устойчивости против корреляционных атак, псевдорандомизации, тестировании комбинаторных схем и др. Определения ε -смещенных массивов, t -связных и ε – зависимых массивов приведём в изложении Биербрауэра [8].

Определение 1. $(n,k)_p$-массив, содержащий n строк, k столбцов и записи из набора p элементов.

Определение 2. Пусть p - простое число, $u=(u_1,u_2,...,u_n)\in F_p^n$. Для $\forall i\in F_p$, $v_i(u)$ есть частота появления элемента i в последовательности u $v_i(u)=\frac{n}{p}+\delta_i(u)$, где $\delta_i(u)$ – есть отклонение частоты $v_i(u)$ от среднего значения и $\sum_{i\in F_p}\delta_i(u)=0$. Пусть ξ комплексный корень p - степени из единицы, тогда смещение вектора u определяется как

$$bias(u)=\frac{1}{n}\left|\sum_{i\in F_p}\delta_i(u)\xi^i\right|=\frac{1}{n}\left|\sum_{i\in F_p}v_i(u)\xi^i\right|.$$

Докажем следующее полезное утверждение.

Утверждение 1. Для произвольного вектора u $0\le bias(u)\le 1$ и $bias(u)=1$ только тогда, когда $u=const$.

Действительно, по определению нормы $bias(u)$ не может быть меньше нуля. Так как $\sum_{i\in F_p} \delta_i(u) = 0$, имеем

$$\frac{1}{n}\left|\sum_{i\in F_p} v_i(u)\xi^i\right| = \frac{1}{n}\left|\sum_{i\in F_p}\left(\frac{n}{p}+\delta_i(u)\right)\xi^i\right| = \frac{1}{n}\left|\sum_{i\in F_p}\delta_i(u)\xi^i\right|.$$

Для произвольного вектора в силу того, что $|\delta_i(u)| \le \frac{n}{p}$, получим

$$bias(u) = \frac{1}{n}\left|\sum_{i\in F_p}\delta_i(u)\xi^i\right| \le \frac{1}{n}\sum_{i\in F_p}\left|v_i(u)\xi^i\right| \le \frac{1}{n}\sum_{i\in F_p}|\delta_i(u)|\left|\xi^i\right| \le 1.$$

Если $u = const$ тогда $bias(u) = \frac{1}{n}\left|\sum_{i\in F_p}\delta_i(u)\xi^i\right| = \frac{1}{n}\left|n\xi^i\right| = 1.$

Определение 3. Пусть $0 \le \varepsilon \le 1$. Массив $(n,k)_p$ является ε -смещённым (-biased), если любая нетривиальная линейная комбинация столбцов имеет смещение $bias \le \varepsilon$.

Смещение массива является свойством F_p – линейного кода, построенного с помощью столбцов порождающей матрицы. Следующие определения являются обобщением понятия ортогональных массивов силы t. В отечественной литературе с ортогональными массивами отождествляются t -схемы, ортогональные таблицы [9], t-универсальные семейства хэш-функций [10].

Определение 4. Пусть $0 \le \varepsilon \le 1$. Массив $(n,k)_p$ является t – связным (t – wise), ε - смещённым, если любая нетривиальная линейная комбинация не более чем t столбцов имеет смещение $bias \le \varepsilon$.

Определение 5. Пусть $0 \le \varepsilon \le 1$. Массив $(n,k)_p$ является t – связным, ε - зависимым (ε - dependent), если для любого набора U из $s \le t$ столбцов и каждого вектора $a \in F_p^s$ частота $v_U(a)$ появления в столбцах значения a удовлетворяет условию

$$\left|\frac{v_U(a)}{n} - \frac{1}{p^s}\right| \le \varepsilon.$$

Пример 1. Рассмотрим двоичный (n,k) код, ненулевые слова которого имеют вес, удовлетворяющий условию

$$\frac{1-\varepsilon}{2} \le \frac{\omega_i}{n} \le \frac{1+\varepsilon}{2}, \qquad \varepsilon < 1.$$

Транспонированная порождающая матрица кода определяет массив $(n,k)_2$ со смещением

$$bias(u) = \frac{1}{n}\left|\sum_{i\in F_2}\delta_i(u)\xi^i\right| = \frac{1}{n}\left|\delta_0\xi^0 + \delta_1\xi^1\right| = \frac{1}{n}\left|-\frac{\varepsilon n}{2} - \frac{\varepsilon n}{2}\right| = \varepsilon.$$

Пример 2. Рассмотрим t – связный, независимый (0- зависимый) массив $(n,k)_p$. По определению 5 имеем $\frac{v_U(a)}{n} = \frac{1}{p^t}$. В этом случае $(n,k)_p$ является ортогональным массивом силы t и образует t – строго универсальное семейство хэш-функций.

Пример 3. Рассмотрим семейство ε – почти строго универсальных $_2$ (ASU $_2$) хэш – функций $h \in H$. По определению (см. [10]) имеем:

1. Для любых значений $x, y \in A$, $x \neq y$ число хэш-функций h таких, что $h(x) = h(y)$ не больше чем $\varepsilon|H|$;
2. Для произвольных $x \in A, y \in B$, число функций таких, что $h(x) = y$, строго равно $|H|/|B|$.

Как следует из второго условия, массив хэш-значений определяет $(n,k)_p$ массив со смешением равным нулю.

Применение слабо смещённых и почти независимых массивов для целей аутентификации определяется тем, что можно построить большие наборы случайных переменных, которые являются почти статистически независимыми. Ниже приводятся конструкции таких кодов аутентификации.

2 Коды аутентификации со слабо смещёнными массивами

Методы построения массивов со смещением достаточно полно представлены в [5, 6, 12]. Хеллесет и Джохансон впервые установили взаимосвязь теории кодирования и смещённых массивов[6]. Наиболее полно эта взаимосвязь была исследована Биербрауэром [8]. Рассмотрим основные кодово-теоретические результаты.

Теорема 1. Пусть (n,k) двоичный код, минимальное расстояние кода d и максимальное D удовлетворяет условию

$$Min\left\{\frac{d}{n}, \frac{n-D}{n}\right\} \geq \frac{1-\varepsilon}{2},$$

где $0 \leq \varepsilon \leq 1$. Тогда транспонированная порождающая матрица кода является ε – смещённым $(n,k)_2$ массивом.

Очевидно следующее утверждение.

Утверждение 2. Пусть $(n,k+1)$ двоичный код C' образован (n,k,d) кодом C и единичным кодовым словом 1, тогда минимальное кодовое расстояние точно равно $Min\{d, n-D\}$.

Отсюда сразу следует практическая конструкция.

Теорема 2. Пусть $(n,k+1,d)$ двоичный код C', содержащий единицу 1 и $\frac{d}{n} \geq \frac{1-\varepsilon}{2}$, $0 \leq \varepsilon \leq 1$, тогда транспонированная порождающая матрица кода является ε – смещённым $(n,k)_2$ массивом.

Построение асимптотически хороших ε- смещённых $(n,k)_2$ массивов (в смысле достижения как можно большего отношения $\frac{k}{n}$ при фиксированном значении смещения ε) эквивалентно решению задачи нахождения асимптотически хороших кодов, содержащих единицу 1.

Для случая p -ичных кодов основной результат получен в [8].

Теорема 3. Пусть C' $(n,k+1,d)_p$ линейный код, содержащий подкод C и m слов веса n. Транспонированная порождающая матрица кода C является $(n,k)_p$- массивом со смещением $\varepsilon \leq p-1-p\frac{d}{n}$.

Пример 4. Пусть $(p,k,p-k+1)_p$ код Рида-Соломона. По предыдущей теореме получим $(n,k-1)_p$ со смещением $\varepsilon \le p-1-p\dfrac{p-k+1}{p} = p-1-p+k-1 = k-2$. Для РС кода размерности $k=2$ имеем 0 - смещенный $(n,1)_p$ массив.

В работе [5] предложено усиление для кодовых конструкций в ε - смещенных $(n,k)_p$ массивах.

Теорема 4. Пусть p - простое число и C – (n,k,d) код. Тогда существует массив $(pn,k)_p$, который имеет смещение $\varepsilon \le 1-\dfrac{d}{n}$.

Пример 5. Рассмотрим $(p,k)_p$ код РС. Применяя теорему 4, получим $\dfrac{k-1}{p}$ – смещенный $\left(p^2,k\right)_p$ массив.

Дальнейшее обобщение теоремы 4 получено в [8].

Теорема 5. Пусть C линейный $(n,k,d)_{q=p^m}$ код и B внутренний $(n_0,m)_p$ массив со смещением ε_0. Тогда существует $(nn_0,km)_p$ массив со смещением $\varepsilon = 1-\delta+\delta\varepsilon_0 \le 1-\delta+\varepsilon_0$, где $\delta = \dfrac{d}{n}$.

Для построения внутренних массивов $(n_0,m)_p$ можно использовать несколько относительно простых конструкций [5].Несколько лучшие результаты имеет метод сумм экспонент Вейля- Карлитца- Ушиямы (ВКУ) [13]. Метод ВКУ состоит в построении массива A с записями вида $Tr\left(a_j\alpha^i\right)$, где a_j - базис поля $F_{p^f} \mid F_p$, $i \le n$ и i не кратно p, $Tr: F_{p^f} \to F_p$ – след элемента $a_j\alpha^i$. Строки массива A индексируются элементами $\alpha \in F_{p^f}$, а столбцы-функциями a_jX^i. Результирующий массив имеет параметры $\left(p^f, f*(n-n/p)\right)_p$ и смещение $bias \le (n-1)p^{-f/2}$, где $(n-n/p)$ определяет возможное число экспонент a_jX^i при $i \le n$ и i не кратно p.

Пример 6. Построим массив ВКУ $\left(p^f, f*(n-n/p)\right)_p$ со смещением $bias \le (n-1)p^{-f/2}$ при $p=2, f=4, n=1$. Базисные элементы поля имеют вид $a_j: 1,\alpha,\alpha^2,\alpha^3$. Так как $n=1$, следует взять только одну экспоненту $\varphi: X$. Строки массива индексируются элементами $\alpha \in F_{2^4}$, столбцы – функциями: $X, \alpha X, \alpha^2 X, \alpha^3 X$, а записи – $Tr(\beta) = \beta+\beta^2+\beta^4+\beta^8$. Получим $\left(2^4,4\right)$ массив со смещением $bias = (1-1)2^{-2} = 0$.

Пусть $p=3, f=2, n=2$. Тогда $a_j: 1,\alpha; \varphi: X, X^2; Tr(\beta) = \beta+\beta^3$. Строки массива индексируются элементами $\alpha \in F_{3^2}$ (порождающий многочлен поля z^2+z+2), столбцы – функциями : $X, \alpha X, \alpha X^2, X^2 = \alpha^4 X+1 \pmod{X^2+X+2}$. Массив $\left(3^2,4\right)_3$ имеет вид (см. табл.).

Таблица

α^i	X	αX	αX^2	$\alpha^4 X$
0	0	0	0	0
α^0	0	α^4	α^4	0
α^1	α^4	0	α^4	0
α^2	0	α^4	α^0	α^4
α^3	α^4	0	α^0	α^4
α^4	0	α^4	α^4	0
α^5	α^0	0	α^4	α^4
α^6	0	α^4	α^0	α^4
α^7	α^0	0	α^0	α^0

Зададим произвольную линейную комбинацию столбцов $Y = \sum_{j=1}^{4} \gamma^j Y_j$, $\gamma_j \in F_3, j = 1,4$, например, $Y = Y_1 + \alpha^4 Y_2 + \alpha^4 Y_4$. Получим результирующий вектор $Y_p = (0, \alpha^0, \alpha^4, \alpha^4, 0, \alpha^4, \alpha^4, 0, 0)$. Значения частот элементов $0, \alpha^0, \alpha^4$ равны: $\nu_0 = 4, \delta_0 = +1; \nu_{\alpha^0} = 1, \delta_{\alpha^0} = -2; \nu_{\alpha^4} = 4, \delta_{\alpha^4} = +1$, а смещение

$$bias(\nu_Y) = \frac{1}{9}\left|1 * e^{j\frac{2\pi}{3}*0} + (-2)e^{j\frac{2\pi}{3}*1} + 1 * e^{j\frac{2\pi}{3}*2}\right| = \frac{1}{9}\left|1 + 1 - \sqrt{3}j - \frac{1}{2} - \frac{\sqrt{3}}{2}j\right| = \frac{1}{3}.$$

Для всех нетривиальных линейных комбинаций столбцов значение $bias \le \frac{1}{3}$ удовлетворяет соотношению $bias \le p^{-1}$, при $n = 2, f = 2$.

Обобщая полученные результаты, отметим, что существует массив ВКУ с параметрами $(p^2, 4)_p$ и смещением $bias = \frac{1}{p}$, где p - простое.

Пример 7. Зададим в поле $F_{q=p^4}$ РС код с параметрами $(p^4, p^3, p^4 - p^3 + 1)$ и внутренний массив $(p^2, 4)_p$ (см. предыдущий пример). Тогда по теореме 5 получим массив $(p^6, 4p^3)_p$ со смещением $bias = 1 - \frac{p^4 - p^3 + 1}{p^4} + \frac{p^4 - p^3 + 1}{p^4} * \frac{1}{p} \le \frac{2}{p}$.

Метод ВКУ при тех же параметрах массива гарантирует смещение $bias = \left(\frac{4}{6}p^3 - 1\right)p^{-\frac{6}{2}} < \frac{4}{6}$ и это уступает каскадной схеме.

Рассмотрим t – связные массивы. Основная кодово-теоретическая конструкция описывается в [6].

Теорема 6. Пусть $(n,k)_p$ массив B со смещением ε и $(N, N-k, t+1)$ -линейный код. Тогда существует $(n,N)_p$ – массив, который является t – связным и ε – смещенным.

Результирующий массив можно построить путем умножения матрицы B и проверочной матрицы кода H. Важное соотношение между смещением и зависимостью установлено в [12].

Теорема 7. Если массив является t – связным и ε – смещенным, он является также и t – связным и ε' – зависимым, причём, $\varepsilon' < \varepsilon$.

Фундаментальное значение этой теоремы заключается в том, что она определяет возможность применения слабо смещённых массивов в схемах аутентификации.

Приведём основные определения для универсальных кодов аутентификации в терминологии слабо смещенных и почти независимых массивов.

Определение 6. $(n,k)_q$ – массив является ε – почти строго универсальным $_2$ (ASU_2), если его столбцы имеют смещение 0 и для различных столбцов C, C' и любых записей e, e', при равновероятном выборе i строки, условная вероятность $\Pr(c_i = e \mid c_i' = e') \le \varepsilon$.

Выбор строки массива определяется значением ключа, столбец – состоянием источника данных и значение записи является кодом аутентификации.

Определение 7. $(N,m)_p$ – массив является (δ,t) – почти строго универсальным $_t$ $(\delta - ASU_t)$, если для любого набора $U = U_0 \bigcup \{u\}$ из t столбцов и каждой записи $a' \in F_p{}^{t-1}, x \in F_p$ отношение частот $\nu_{U_0}(a')$ и $\nu_U(a', x)$ удовлетворяет условию

$$\left|\nu_U(a', x)/\nu_{U_0}(a')\right| \le \delta .$$

Связь между почти независимыми массивами и $\delta - ASU_t$ кодами достаточно очевидна и была установлена в [12].

Теорема 8. $\delta - ASU_t$ является t – связным ε – зависимым массивом, где $\delta = \left(p^{-t} + \varepsilon\right) / \left(p^{-(t-1)} - \varepsilon\right)$.

Пример 8. Построим массив аутентификаторов ASU_2. Зададим C – $(n,k)_p$ линейный код с ε_0 - смещёнными кодовыми словами. Строки массива ASU_2 индексируем набором $(i, \alpha_1, \alpha_2, ..., \alpha_t)$, где i - номер элемента C кода, $\alpha_r \in F_p$. Зададим линейные отображения $M_r : C \to C, r = 1,2,...,t$ так, что любая нетривиальная F_p - линейная комбинация из отображений M_r является несингулярной. Столбцы массива ASU_2 являются словами кода. Запись в массиве на пересечении строки $(i, \alpha_1, \alpha_2, ..., \alpha_t)$ и столбца f определяется выражением $(M_1(f)(i) + \alpha_1, M_2(f)(i) + \alpha_2, ..., M_t(f)(i) + \alpha_t)$. Смещение каждого столбца равно нулю, так как $(\alpha_1, \alpha_2, ..., \alpha_t)$ пробегают p^t значений. По определению 6 имеем $\delta - ASU_t$ $(np^t, p^k)_{p'}$ массив. Используя результат теоремы 7 и определение 5, для t - связных и ε - зависимых массивов получим границу $\varepsilon \le p^{-t} + \varepsilon_0$.

Пример 9. Рассмотрим $\left(p^3, p^2/2, p^3 - p^2\right)_{p^2}$ код Эрмита (см. [11]). Построим методом ВКУ несмещённый $\left(p^2, 2\right)_p$ массив. Каскадная конструкция кода Эрмита и несмещённого массива по теореме 5 дает $\left(p^5, p^2\right)_p$ - массив с $\varepsilon = \frac{1}{p}$. Используя результаты примера 8, получим дальнейшее улучшение $\rightarrow$ $\frac{2}{p}$ – $\left(p^6, p^{p^2}\right)$ массив ASU_2, что лучше, чем дает использование кодов РС (см. пример 7).

Применение алгеброгеометрических кодов высокого порядка в каскадных конструкциях со слабо смещёнными массивами дает лучшие результаты, что согласуется с выводами, сделанными в работе [11].

Существующее широкое многообразие кодово-теоретических схем построения смещённых и связных массивов позволяет исследовать коды аутентификации с новыми дополнительными свойствами, например, t – связные универсальные схемы.

Список литературы: 1. *Simmons G.J.* A game theory model of digital message authentication // Congressus Numerantium 34 (1992). С. 413 – 424. 2. *Simmons G.J.* Authentication theory/coding theory, in Advances in Cryptology // Proceedings of Crypto 84, Lecture Notes in Computer Science 196 (1985). С. 411 – 431. 3. *Wegman. M.N., Carter J.L.* New hash functions and their use in authentication and set equality // J. Computer and System Sci. 22 (1981).С. 265 – 279. 4. *Carter J. L., Wegman M. N.* Universal classes of hash functions // J. Computer and System Sci. 18 (1979). С. 143 – 154. 5. *Alon N., Goldreich O., Hastad J., Peralta R.* Simple constructions of almost k-wise independent random variables // Random Structures and Algorithms 3 (1992). С. 289 -- 304. 6. *Naor J., Naor M.* Small-bias probability spaces: efficient constructions and applications // SIAM Journal on Computing 22 (1993). С. 838 – 856. 7. *Helleseth T., Johansson T.* Universal hash functions from exponential sums over finite fields and Galois rings // Lecture Notes in Computer Science 1109 (1996). С. 31 – 44 (CRYPTO 96). 8. *Bierbrauer J., Schellwat H.* Weakly biased arrays, almost independent arrays and error-correcting codes // Publication in Proceedings of AMS-DIMACS (2000). 9. *Мак-Вильямс Ф.Дж., Слоэн Н.Дж.А.* Теория кодов, исправляющих ошибки. М.: Связь, 1979. 744 с. 10. *Халимов Г.З., Кузнецов А.А.* Аутентификация и универсальное хеширование // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 120. С. 100 – 110. 11. *Халимов Г.З., Кузнецов А.А.* Аутентификация с применением алгеброгеометрических кодов // Там же С. 103 – 109. 12. *Kurosawa K., Johansson T., Stinson D.* Almost k-wise independent sample spaces and their cryptologic applications // Lecture Notes in Computer Science 1233 (1997). С. 409 – 421. 13. *Carlitz L., Uchiyama S.* Bounds for exponential sums // Duke Mathematical Journal 24 (1957). С. 37 – 41.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 20.04.2003

УДК 681.3+681.5:007

Н. В. АЛИПОВ, д-р техн. наук, И. Н. АЛИПОВ, канд. техн. наук,
М. И. ХИЛЬ, канд. техн. наук, Л. Н. РЕБЕЗЮК, канд. техн. наук

ПОСЛЕДОВАТЕЛЬНЫЕ АЛГОРИТМЫ ПОИСКА ТОЧКИ С ХАРАКТЕРНЫМ ПРИЗНАКОМ, ПОМЕХОУСТОЙЧИВЫЕ К СИММЕТРИЧНЫМ НЕРЕГУЛЯРНЫМ ВИРТУАЛЬНЫМ ПОСЛЕДОВАТЕЛЬНОСТЯМ

Поиск в нашей жизни занимает одно из важнейших мест, а тема «Поиск» исключительно широка и сложна [1, 2]. В исследовании рассматривается проблема помехоустойчивого одномерного поиска точки с характерным признаком, исходным интервалом неопределенности для которой является интервал $(0,1)$, характерным признаком – ее координата [1, 3]. Решение этой проблемы важно потому, что процесс поиска на отрезке $[0,1]$ точки с характерным признаком аналогичен процессам аналого-цифрового преобразования, поиска данных, поиска неисправного элемента, к нему сводится ряд задач помехоустойчивого преобразования информации, защиты информации, избыточных представлений десятичных чисел и теории вопросников [1, 3, 4]. Структура алгоритмов помехоустойчивого поиска точки с характерным признаком определяется датчиками виртуальных последовательностей [5].

К настоящему моменту разработаны алгоритмы поиска точки с характерным признаком, помехоустойчивые к регулярным последовательностям [6] либо к нерегулярным последовательностям, у которых случайной величиной является длительность выброса [7]. Известны также последовательные алгоритмы [8], помехоустойчивые к несимметричным нерегулярным последовательностям, у которых интервал времени между соседними выбросами последовательности является случайной величиной.

Целью исследования является разработка последовательных алгоритмов, помехоустойчивых к симметричным нерегулярным виртуальным последовательностям, для которых интервал времени между соседними выбросами последовательности является случайной величиной.

Заметим [4], что эти последовательности описываются такими параметрами: амплитудой выброса (a), длительностью выброса (ℓ) и интервалом времени между соседними выбросами (h). Они бывают несимметричными (однополярными) и симметричным (двуполярными) [4]. В работе рассматриваются двуполярные виртуальные последовательности, у которых только параметр h является случайной величиной. Алгоритм поиска характеризуется количеством шагов i (длиной поиска) и точек (k), в которых одновременно выполняется эксперимент на j-м шаге алгоритма $(j \le i)$ [4]. Для последовательных алгоритмов поиска характерно то, что $k = 1$. Формулировка задачи синтеза помехоустойчивых алгоритмов поиска приведена в работе [4]. Решение этой задачи направлено на то, чтобы синтезировать правила формирования новых интервалов неопределенности и стратегий поиска (правил размещения точки эксперимента во вновь выделенном интервале неопределенности).

В дальнейшем будем полагать, что $h \in [h_1, h_2]$; $\ell, a = const$; $k = 1$; виртуальная последовательность – двуполярная, посредством которой точка с характерным признаком x смещается в направлении $0 \to 1$ и в направлении $1 \to 0$; h_1 – минимальное значение параметра h; h_2 – максимальное значение параметра h; h – целое положительное число; $x \in [0,1]$.

Первоначально сформулируем правила выделения нового интервала неопределенности на первом шаге алгоритма, а затем на любом другом его шаге. Пусть некоторым образом выбрана точка первого эксперимента в исходном интервале неопределенности $(0,1)$. Тогда по итогам выполнения этого шага алгоритма может быть сформирован один из исходов:

$$\text{а) } x(t_1) < x_1^1; \quad \text{б) } x(t_1) > x_1^1, \tag{1}$$

где $x(t_1)$ – аддитивная смесь координаты точки x и $\xi(t_1)$; $\xi(t_1)$ – значение амплитуды виртуальной последовательности:

$$\xi(t_1)=\begin{cases}0, \textit{виртуальная последовательность на первом шаге не проявилась};\\ a \ \textit{в противном случае.}\end{cases}$$

Для указанных исходов на основании принципа «пересечения» [4] устанавливаем:

$$x\in[0,x_1^{1,2}),\quad x\in[x_1^{1,1},1), \tag{2}$$

где $x_1^{1,1}=\begin{cases}x_1^1-a\delta, & (x_1^1-a\delta)>0;\\ 0 \ \textit{в противном случае;}\end{cases}$ $x_1^{1,2}=\begin{cases}x_1^1+a\delta, & (x_1^1+a\delta)\le 1;\\ 1 \ \textit{в противном случае;}\end{cases}$

δ – дискретность квантования исходного интервала неопределенности.

Пусть $\ell>1$ и на первом шаге алгоритма был сформирован исход типа а), а на втором шаге алгоритма выбрана точка x_1^2 второго эксперимента, для которой $x_1^1\in(0,x_1^1)$. Тогда по итогам выполнения второго шага алгоритма может быть сформирован один из исходов типа а) либо – типа б). В этом случае на основании принципа «пересечения» соответственно устанавливаем:

$$x\in[0,x_1^{2,2}),\quad x\in[x_1^{2,1},x^{1,2}), \tag{3}$$

где $x_1^{2,2}=\begin{cases}x_1^2+a\delta, & (x_1^2+a\delta)<x_1^{1,2};\\ x_1^{1,2} \ \textit{в противном случае;}\end{cases}$ $x_1^{2,1}=\begin{cases}x_1^2-a\delta, & (x_1^2-a\delta)>0;\\ 0 \ \textit{в противном случае.}\end{cases}$

Если же на первом шаге алгоритма был сформирован исход типа б), а на втором шаге алгоритма выбрана точка эксперимента $x_1^2\in[x_1^1,1)$, то по итогам выполнения второго шага алгоритма также может сформироваться один из исходов типа а) либо – типа б).

В этом случае на основании принципа «пересечения» соответственно устанавливаем:

$$x\in[x_1^{1,1},x_1^{2,2}),\quad x\in[x_1^{2,1},1), \tag{4}$$

где $x_1^{2,2}=\begin{cases}x_1^2+a\delta, & (x_1^2+a\delta)<1;\\ 1 \ \textit{в противном случае;}\end{cases}$ $x_1^{2,1}=\begin{cases}x_1^2-a\delta, & (x_1^2-a\delta)<x_1^{1,1};\\ x_1^{1,1} \ \textit{в противном случае.}\end{cases}$

Пусть по описанной выше схеме выполняются последующие эксперименты: третий, четвертый, ..., $(j-1)$-й и на $(j-1)$-м шаге алгоритма сформирован новый полуоткрытый интервал неопределенности относительно точки с характерным признаком $x\in[x_1^{j_1,1},x_1^{j_2,2})$.

При выполнении j-го шага алгоритма может сформироваться один из исходов а) или б), для которых соответственно выделяются новые полуоткрытые интервалы неопределенности:

$$x\in[x_1^{j_1,1},x_1^{j,2}),\quad x\in[x_1^{j,1},x_1^{j_2,2}), \tag{5}$$

где $x_1^{j,2}=\begin{cases}x_1^j+a\delta, & (x_1^j+a\delta)\le x_1^{j_2,2};\\ x_1^{j_2,2} \ \textit{в противном случае;}\end{cases}$ $x_1^{j,1}=\begin{cases}x_1^j-a\delta, & (x_1^j-a\delta)>x_1^{j_1,1};\\ x_1^{j_1,1} \ \textit{в противном случае.}\end{cases}$

При продолжении поиска для исхода а) может возникнуть такая ситуация, для которой характерно следующее: на $(j+1)$, $(j+2)$,..., $(j+z-1)$-м шагах алгоритма формировался

исход типа б), а по итогам выполнения $(j+z)$-го шага $(z \geq \ell)$ может появиться один из исходов а) или б). Для них формируются такие полуоткрытые интервалы неопределенности:

$$x \in [x_1^{(j+z-1),1}, x_1^{(j+z),2}), \quad x \in [x_1^{(j+z),1}, x_1^{(j+z-1),2}), \tag{6}$$

где $x_1^{(j+z),2} = \begin{cases} x_1^{j+z} + a\delta, & (x_1^{j+z} + a\delta) < x_1^j; \\ x_1^j & \textit{в противном случае;} \end{cases}$ $x_1^{(j+z),1} = \begin{cases} x_1^{j+z} - a\delta, & (x_1^{j+z} - a\delta) > x_1^{(j+z-1),1}; \\ x_1^{(j+z-1),1} & \textit{в противном случае.} \end{cases}$

Аналогично при продолжении поиска для исхода б) может возникнуть ситуация, когда на $(j+1)$, $(j+2)$,..., $(j+z-1)$-м шагах алгоритма формировался исход типа а), а на $(j+z)$-м шаге алгоритма $(z \geq \ell)$ появляется один из исходов а) или б). Для этих исходов в этом случае формируют такие полуоткрытые интервалы неопределенности:

$$x \in [x_1^{(j+z-1),1}, x_1^{(j+z),2}), \quad x \in [x_1^{(j+z),1}, x_1^{(j+z-1),2}), \tag{7}$$

где $x_1^{(j+z),2} = \begin{cases} x_1^{j+z} + a\delta, & (x_1^{j+z} + a\delta) < x_1^{(j+z-1),2}; \\ x_1^{(j+z-1),2} & \textit{в противном случае;} \end{cases}$ $x_1^{(j+z),1} = \begin{cases} x_1^{j+z} - a\delta, & (x_1^{j+z} - a\delta) > x_1^j; \\ x_1^j & \textit{в противном случае.} \end{cases}$

При организации поиска точки с характерным признаком может возникнуть и такая ситуация: на j-м шаге алгоритма сформировался исход типа а), на последующих $(j+1)$, $(j+2)$,..., $(j+z-1)$-м шагах алгоритма формировался исход типа б), а на $(j+z)$-м шаге алгоритма $(z \geq \ell)$ была использована пессимистическая стратегия

$$x_1^{j+z} = x_1^j. \tag{8}$$

В результате применения такой стратегии может сформироваться один из исходов а) или б). Для исхода а) характерно то, что результаты экспериментов, разнесенных во времени на ℓ шагов алгоритма, совпадают. Это означает, что на j-м шаге алгоритма виртуальная последовательность не проявлялась. На этом основании устанавливаем:

$$x \in [x_1^{j,1}, x_1^j) \tag{9}$$

и продолжаем процесс поиска в выделенном на $(j+z-1)$-м шаге алгоритма интервале неопределенности.

Для исхода б) возникшее противоречие свидетельствует о действии виртуальной последовательности на $[j \div (j+z-1)]$-х шагах алгоритма либо на последующих $[(j+z) \div (j+z+\ell-1)]$-х шагах. Для этого исхода формируем такой интервал неопределенности:

$$x \in [x_1^j, x_1^{j,2}). \tag{10}$$

В этой неопределенности, которая будет продолжаться на последующих $(\ell-1)$-х шагах, размещаем на этих шагах точки экспериментов в полуоткрытом интервале неопределенности (10) согласно стратегии классического алгоритма поиска. Затем на $(j+z+\ell)$-м шаге алгоритма повторяем эксперимент:

$$x_1^{j+z+\ell} = x_1^j. \tag{11}$$

При этом если возникает исход типа а), то возникшее противоречие свидетельствует о том, что виртуальная последовательность проявилась на $[(j+z)\div(j+z+\ell-1)]$-х шагах алгоритма, смещая точку с характерным признаком вправо (виртуальная последовательность первоначально проявилась выбросом положительной полярности). На этом основании выделяем относительно точки x такой полуоткрытый интервал неопределенности:

$$x\in[x_1^{j,1},x_1^{j}) \tag{12}$$

и в интервале, выделенном на $(j+z-1)$-м шаге, организуем процесс поиска.

Поскольку проявление виртуальной последовательности обнаружено, то на полуоткрытом интервале неопределенности, выделенном на $(j+z-1)$-м шаге алгоритма, применяем комбинированный алгоритм поиска: на последующих (h_1-1)-х шагах применяем классический алгоритм поиска; затем пропускаем $(h_2-h_1+\ell)$ шагов алгоритма; потом снова на последующих h_1-х шагах алгоритма используем классический алгоритм поиска и так до конца поиска. Посредством такой стратегии поиска этот полуоткрытый интервал неопределенности разобьем за оставшиеся шаги алгоритма на $\psi_2^{h_1,h_2,\ell,a}(i-j-z-\ell,1)$ равных частей. Для этой функции справедливо соотношение

$$\psi_2^{h_1,h_2,\ell,a}(i-j-z-\ell,1)=2^{h_1-1}\cdot 2^{h_1\left\lfloor\frac{i-j-z-\ell-h_1+1}{h_2+\ell}\right\rfloor}\cdot 2^{\alpha}, \tag{13}$$

где

$$\alpha=\begin{cases}0, & \left(i-j-\ell-z-h_1+1\right)-\left(\ell+h_2\right)\left\lfloor\dfrac{i-j-\ell-z-h_1+1}{\ell+h_2}\right\rfloor\le\left(\ell+h_2-h_1\right);\\ \left(i-j-\ell-z-h_1+1\right)-\left(\ell+h_2\right)\left\lfloor\dfrac{i-j-z-\ell-h_1+1}{\ell+h_2}\right\rfloor, & \text{если 1);}\\ \left(i-j-\ell-z-h_1+1\right)-\left(\ell+h_2\right)\left\lfloor\dfrac{i-j-z-\ell-h_1+1}{\ell+h_2}\right\rfloor>\left(\ell+h_2-h_1\right). & \text{1)}\end{cases}$$

Если же при выполнении эксперимента (11) возникает исход типа б), то это будет свидетельством того, что виртуальная последовательность имела место на $[j\div(j+z-1)]$-х шагах алгоритма, смещая точку с характерным признаком влево (проявился выброс отрицательной полярности). Поскольку последние $(\ell-1)$ шаги алгоритма выполнялись по стратегии классического алгоритма, то интервал неопределенности (10) за эти шаги будет разбит на $2^{\ell-1}$ равные части. Затем на выделенном на $(j+z+\ell-1)$-м шаге алгоритма в полуоткрытом интервале неопределенности применяется комбинированный алгоритм поиска.

Как нетрудно заметить, для обнаружения виртуальной последовательности было использовано $(z-\ell+1)$ шагов паузы между двумя соседними выбросами. Затем после обнаружения противоречия в полуоткрытом интервале (10) было совершено $(\ell-1)$ шагов алгоритма; для окончательного принятия решения о действии виртуальной последовательности был еще выполнен эксперимент (11). Поэтому общее количество использованных шагов алгоритма, расположенных в интервале между двумя соседними выбросами, составит:

$$N=(z-\ell+1)+\ell=z+1.$$

Количество неиспользованных шагов, расположенных между двумя соседними выбросами последовательности, определяется соотношением

$$N_1 = (h_1 - z - 1).$$

Поэтому комбинированный алгоритм поиска на последующих $(h_1 - z - 1)$ шагах использует стратегию классического алгоритма; затем пропускает $(h_2 - h_1 + \ell)$ шагов алгоритма; потом на последующих h_1 шагах алгоритма использует снова стратегию классического алгоритма поиска и так до конца поиска.

Посредством такой комбинации алгоритмов полуоткрытый интервал неопределенности (10) разобьем на $\psi_2^{h_1,h_2,\ell,a}(i-j-z-1,1)$ равных частей. Для этой функции справедливо соотношение:

$$\psi_2^{h_1,h_2,\ell,a}(i-j-z-1,1) = 2^{\ell-1} \cdot 2^{h_1-z-1} \cdot 2^{h_1 \left\lfloor \frac{i-j-\ell-h_1+1}{h_2+\ell} \right\rfloor} \cdot 2^{\alpha_1}, \qquad (14)$$

где

$$\alpha_1 = \begin{cases} 0, (i-j-\ell-h_1+1) - (\ell+h_2) \left\lfloor \dfrac{i-j-\ell-h_1+1}{\ell+h_2} \right\rfloor \le (\ell+h_2-h_1); \\ (i-j-\ell-h_1+1) - (\ell+h_2) \left\lfloor \dfrac{i-j-\ell-h_1+1}{\ell+h_2} \right\rfloor - (\ell+h_2-h_1), \text{ если 2);} \end{cases}$$

$$2)\ (i-j-\ell-h_1+1) - (\ell+h_2) \left\lfloor \frac{i-j-\ell-h_1+1}{\ell+h_2} \right\rfloor > (\ell+h_2-h_1).$$

Из анализа выражения (14) следует истинность неравенства

$$h_1 \ge (z+1), \quad (z-\ell+1) < h_1.$$

В процесс поиска возможна и такая ситуация: на j-м шаге алгоритма был сформирован исход типа б), на последующих $(j+1)$, $(j+2)$,..., $(j+z-1)$ шагах формировался исход типа а), а на $(j+z)$-м шаге $(z \ge \ell)$ была использована пессимистическая стратегия (8). По итогам ее выполнения может быть сформирован исход типа а) либо – типа б).

Для исхода б) характерно то, что результаты экспериментов, разнесенных во времени на ℓ шагов алгоритма, совпали. На этом основании устанавливаем

$$x \in [x_1^j, x_1^{j,2}) \qquad (15)$$

и продолжаем процесс поиска в выделенном на $(j+z-1)$-м шаге алгоритма интервале неопределенности.

Для исхода а) возникшее противоречие свидетельствует о проявлении виртуальной последовательности на $[j \div (j+z-1)]$-х шагах алгоритма либо на последующих $[(j+z) \div (j+z+\ell-1)]$-х шагах. Для этого исхода формируют следующий интервал неопределенности:

$$x \in [x_1^{j,1}, x_1^j). \qquad (16)$$

В этой неопределенности на последующих шагах алгоритма, как было уже показано, размещаем точки экспериментов на последующих $(\ell-1)$ шагах алгоритма в полуоткрытом интервале неопределенности (16) согласно стратегии классического алгоритма поиска. Затем на $(j+z+\ell)$-м шаге применяем пессимистическую стратегию (11). При этом если возникает исход типа б), то возникшее противоречие свидетельствует о том, что виртуальная последовательность проявилась на $[(j+z) \div (j+z+\ell-1)]$-х шагах алгоритма, смещая точку с харак-

терным признаком влево. На этом основании выделяем относительно точки x такой полуоткрытый интервал неопределенности:

$$x \in [x_1^j, x_1^{j,2}), \tag{17}$$

– и в интервале неопределенности, выделенном на $(j+z-1)$-м шаге, организуем процесс поиска, применяя для этих целей ранее описанный комбинированный алгоритм: на последующих (h_1-1) шагах применяем классический алгоритм поиска; затем пропускаем $(h_2-h_1+\ell)$ шагов алгоритма; потом снова на последующих h_1 шагах алгоритма используем классический алгоритм поиска и так до конца поиска.

Как было показано, такой алгоритм позволяет разбить полуоткрытый интервал неопределенности, выделенный на $(j+z-1)$-м шаге алгоритма, разбить на $\psi_2^{h_1,h_2,\ell,a}(i-j-z-\ell,1)$ равных частей. Для этой функции справедливо соотношение (13).

Если же при выполнении эксперимента (11) возникает исход типа а), то это будет свидетельствовать о проявлении виртуальной последовательности на $[j \div (j+z_1-1)]$-х шагах алгоритма, смещая точку с характерным признаком вправо (проявился выброс положительной полярности).

Поскольку последние $(\ell-1)$-е шаги алгоритма выполнялись по схеме классического алгоритма, то полуоткрытый интервал неопределенности (16) за эти шаги будет разбит на $2^{\ell-1}$ равные части. На последующих шагах алгоритма в выделенном на $(j+z+\ell-1)$-м шаге алгоритма применяется уже описанный ранее комбинированный алгоритм, который позволит полуоткрытый интервал неопределенности (16) разбить на $\psi_2^{h_1,h_2,\ell,a}(i-j-z-\ell,1)$ равных частей. Для данной функции справедливо соотношение (14).

Как известно [1, 2], для решения задачи синтеза помехоустойчивых алгоритмов поиска точки с характерным признаком необходимо разработать правила формирования нового интервала неопределенности (это мы уже проделали) и стратегию поиска (правила распределения точек экспериментов во вновь сформированном интервале неопределенности).

Для последовательных алгоритмов поиска применяют две стратегии [2]: оптимистическую и пессимистическую.

Оптимистическую стратегию всегда применяют на первом шаге алгоритма

$$x_1^1 \in (0,1). \tag{18}$$

Ее применяют и в таком случае, когда на j-м шаге, к примеру, был сформирован интервал неопределенности

$$[x_1^{j,1}, x_1^{j,2}], \quad x(t) \gtrless x_1^j,$$

а при выполнении $(j+z)$-го шага алгоритма имеет место неравенство

$$z < \ell. \tag{19}$$

Для всех других вариантов, когда истинно соотношение

$$z \geq \ell, \tag{20}$$

применяют оптимистическую либо пессимистическую стратегию.

Выбор стратегии осуществляют на основании соотношения (14). Как следует из этого выражения, функция $\psi_2^{h_1,h_2,\ell,a}(i-j-z-1,1)$ определяет количество равных частей, на которое могут быть разбиты интервалы $(x_1^{j,1}, x_1^{j})$, $(x_1^{j}, x_1^{j,2})$.

Поскольку на последнем шаге алгоритма исходный интервал неопределенности (0,1) разбивается на $\varphi_2^{h_1,h_2,\ell,a}(i,1)$ частей, каждая из которых имеет длину δ, то на основании соотношения (14) можно установить длину полуоткрытого интервала неопределенности, который может быть проквантован с той же дискретностью δ.

Следовательно, если на $(j+z)$-м шаге алгоритма будет применена пессимистическая стратегия, то в случае проявления виртуальной последовательности на $[j \div (j+z-1)]$-х шагах алгоритма с дискретностью δ могут быть проквантованы только те интервалы неопределенности, для которых выполняется неравенство

$$\ell([a,b]) \le \delta \cdot \psi_2^{h_1,h_2,\ell,a}(i-j-z-1,1), \tag{21}$$

где $\ell([a,b])$ – длина полуоткрытого интервала $[a,b]$.

Если же на z-м шаге применяется оптимистическая, а на $(z+1)$-м шаге – пессимистическая стратегия и при этом возникает исход, противоположный исходу, образованному на j-м шаге алгоритма, то в случае проявления виртуальной последовательности на $[j \div (j+z-1)]$-х шагах алгоритма с дискретностью δ могут быть проквантованы те интервалы неопределенности, для которых справедливо соотношение:

$$\ell([a_1,b_1]) \le \delta \cdot \psi_2^{h_1,h_2,\ell,a}(i-j-z,1). \tag{22}$$

Неравенства (21), (22) позволяют утверждать, что оптимистическая стратегия на $(j+z)$-м шаге алгоритма $(z \ge \ell)$ применяется тогда, когда истинными являются соотношения:
на j-м шаге алгоритма был сформирован исход типа б)

$$\begin{gathered}\ell([x_1^{j,1}, x_1^{1})) \le \delta \cdot \psi_2^{h_1,h_2,\ell,a}(i-j-z-1,1);\\ \ell([x_1^{j,1}, x_1^{1})) \le \delta \cdot \psi_2^{h_1,h_2,\ell,a}(i-j-z-2,1);\\ h_1 \ge (z+1),\ h_1 \ge (z+2),\end{gathered} \tag{23}$$

на j-м шаге алгоритма был сформирован исход типа а)

$$\begin{gathered}\ell([x_1^{j}, x_1^{j,2})) \le \delta \cdot \psi_2^{h_1,h_2,\ell,a}(i-j-z-1,1);\\ \ell([x_1^{j}, x_1^{j,2})) \le \delta \cdot \psi_2^{h_1,h_2,\ell,a}(i-j-z-2,1);\\ h_1 \ge (z+1),\ h_1 \ge (z+2).\end{gathered} \tag{24}$$

Пессимистическая стратегия применяется, как нетрудно заметить, в том случае, когда не выполняется четвертое либо третье, либо второе неравенства соотношений (23), (24).

Пессимистическая стратегия всегда используется на последующих $(2\ell+1)$-м шагах алгоритма. При этом точки эксперимента выбираются таким образом, чтобы разбить выделенный интервал неопределенности на две равные части. На этом основании записываем очевидные соотношения:

$$\varphi_2^{h_1,h_2,\ell,a}(1,1) = \varphi_2^{h_1,h_2,\ell,a}(2,1) = \ldots = \varphi_2^{h_1,h_2,\ell,a}(2\ell,1) = 1;$$

$$\varphi_2^{h_1,h_2,\ell,a}(2\ell+1,1)=2. \qquad (25)$$

Построение таких алгоритмов осуществляют методом индукции. Первый алгоритм строят для $i=(2\ell+1)$, используя только пессимистическую стратегию. Для других значений параметра алгоритма i используя такую схему [2]:

1.Построить $(i-1)$-шаговый помехоустойчивый алгоритм поиска точки с характерным признаком. Расположить точку первого эксперимента в исходном интервале. Переменной j присвоить значение, равное единице.

2.Сформировать возможный исход. Если все исходы сформированы, то перейти на п.7, иначе – п.3.

3.Используя одно из соотношений (2) – (7), (9), (10), (12), (15) – (17), выделить новый полуоткрытый интервал неопределенности.

4.Положить $j=j+1$ и, если $j\le i$, то на основании (17), (23) – (25) выбрать точку следующего эксперимента (оптимистическую либо пессимистическую стратегию) и перейти на п.5, иначе – п.6.

5.Сформировать возможный исход на j-м шаге алгоритма. Если все исходы проанализированы, то перейти на п.6, иначе – на п.3.

6.Положить $j=j-1$. Если $j=1$, то перейти на п.2, иначе перейти на п.5.

7.Местоположения для всех исходов определены (алгоритм построен).

В процессе поиска точки с характерным признаком возможна такая ситуация, когда один и тот же интервал неопределенности в зависимости от исхода разбивается на различное количество частей. Поступают в таких ситуациях следующим образом.

Пусть $\ell=1$, $k=1$, $h_1=3$, $h_2=4$, параметр виртуальной последовательности «a» превосходит длину исходного интервала неопределенности и на первом шаге алгоритма некоторым образом выбрана точка первого эксперимента. Тогда независимо от исхода, сформированного на первом шаге алгоритма, на втором принимаем пессимистическую стратегию $(x_1^2=x_1^1)$.

Если исход, сформированный на втором шаге алгоритма, подтверждает исход первого шага, то устанавливаем, что для исхода а) полуоткрытые интервалы $[0,x_1^1)$, $[x_1^1,1)$ будут соответственно разбиты на $\varphi_2^{3,4,1,a}(i-2,1)$ равных частей.

Если исход, сформированный на втором шаге алгоритма, не подтверждает исход первого шага, то на третьем шаге применяем снова пессимистическую стратегию $(x_1^3=x_1^1)$.

Если исход, сформированный на третьем шаге, подтверждает исход второго шага, то в этом случае на полуоткрытых интервалах неопределенности $[0,x_1^1)$, $[x_1^1,1)$ действует комбинированный алгоритм поиска, который разобьет указанные интервалы неопределенности на $\psi_2^{3,4,1,a}(i-j-z-1,1)$ равных частей (см. соотношение (14)).

Исходя из минимаксного критерия, устанавливаем, что полуоткрытые интервалы $[0,x_1^1)$, $[x_1^1,1)$ будут в зависимости от исходов разбиты на N_2 равных частей:

$$N_2=\min\left\{\varphi_2^{3,4,1,a}(i-2,1),\psi_2^{3,4,1,a}(i-j-z-1,1)=\psi_2^{3,4,1,a}(i-2,1)\right\}, \qquad (26)$$

где $j=1$; $z=1$.

Приведенные соотношения для формирования нового интервала неопределенности относительно точки с характерным признаком, соотношения, устанавливающие закономер-

ность распределения точек j-го эксперимента во вновь выделенном интервале неопределенности и схема построения помехоустойчивого алгоритма позволяют синтезировать последовательный помехоустойчивый алгоритм поиска для любых параметров виртуальной последовательности и параметров алгоритма и тем самым определить функционирование конечного автомата с псевдослучайными переходами системы защиты информации.

Список литературы: 1. *Алипов Н.В.* Помехоустойчивый поиск точки с характерным признаком и кодирование информации // Радиоэлектроника и информатика. 2000. №. 4. С. 82 – 86. 2. *Альведе Р., Вагенер И.* Задачи поиска. М.: Мир, 1982. 365 с. 3. *Алипов Н.В.* Дискретные автоматы с псевдослучайными переходами и подстановочные методы защиты информации на их основе // Радиоэлектроника и информатика. 2001. № 4. С. 95 – 98. 4. *Алипов Н.В.* Синтез оптимальных полихотомичных вопросников для угадывания числа с ложными ответами // Проблемы бионики. Вып. 38. 1987. С. 108 – 117. 5. *Алипов Н.В., Алипов И.Н., Ребезюк Л.Н. и др.* Датчики виртуальных помех, используемые для организации функционирования дискретных автоматов в системах защиты информации // Радиотехника. 1999. Вып. 111. С. 33 – 39. 6. *Алипов Н. В., Ребезюк Л.Н., Охапкин А.А.* Защита информации в дискретном канале на основе устойчивых к периодическим помехам алгоритмов поиска точки с характерным признаком // АСУ и приборы автоматики. 1999. Вып.109. С. 108 – 115. 7. *Алипов Н.В., Ребезюк Л.Н.и др.* Методы защиты информации в дискретном канале на основе помехоустойчивых к несимметричным нерегулярным виртуальным помехам алгоритмов поиска точки с характерным признаком // Радиоэлектроника и информатика. 2000. №. 2. С. 104 – 111. 8. *Алипов Н.В., Алипов И.Н., Ребезюк Л.Н.* Последовательные алгоритмы поиска точки с характерным признаком, помехоустойчивые к несимметричным нерегулярным виртуальным последовательностям // Радиоэлектроника и информатика. 2003. № 2.

Харьковский национальный

университет радиоэлектроники

Поступила в редколлегию 28.04.2003

УДК 681.324.067

В. П. ШИРОЧИН, д-р техн. наук, І. В. ВАСИЛЬЦОВ, канд. техн. наук,
Б. З. КАРПІНСЬКИЙ, Л. О. ВАСИЛЬКІВ

ПІДВИЩЕННЯ ЛІНІЙНОЇ СКЛАДНОСТІ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ ПОБУДОВАНИХ НА ОСНОВІ РЕГІСТРІВ ЗСУВУ

Генерування псевдовипадкових чисел (ПВЧ) є однією з актуальних та важливих задач захисту інформації [1-3]. Отримані псевдовипадкові послідовності використовуються як для формування ключової інформації, так і для реалізації потокових шифрів. На сьогодні існує багато різних способів отримання псевдовипадкових послідовностей як програмними засобами, так і апаратно реалізованими, проте найбільш широкого застосування набули апаратно реалізовані генератори на базі регістрів зсуву з оберненими зв'язками [1-2].

Однією з найважливіших характеристик, що характеризує стійкість такого генератора проти спеціалізованих криптоатак, є лінійна складність (*linear complexity*) [2]. Тому проблема підвищення лінійної складності генераторів ПВЧ, побудованих на основі регістрів зсуву, є важливою та актуальною.

1 Способи підвищення лінійної складності LFSR

Існують різні способи комбінування LFSR – генераторів для підвищення лінійної складності кінцевого генератора. У [4] описані деякі з них.

Сума виходів двох LFSR – генераторів. Якщо s і t є виходи двох LFSR – генераторів довжиною L_1 і L_2 відповідно, то загальна лінійна складність буде визначатися за такою формулою:

$$L(s \otimes t) \le L(s) + L(t). \tag{1}$$

Використання такого методу комбінації дозволяє підвищити загальну складність атаки на $O(1)$.

Добуток виходів двох LFSR – генераторів. Якщо s і t є виходи двох LFSR – генераторів довжиною L_1 і L_2 відповідно, то загальна лінійна складність буде визначатися за такою формулою:

$$0 \le L(s \times t) \le L(s) \times L(t). \tag{2}$$

Використання такого методу комбінації дозволяє підвищити загальну складність атаки на $O(n^2)$.

Розділення виходу LFSR – генератора. Якщо s є виходом LFSR – генератора довжиною L, то для отриманих послідовностей u та v загальна лінійна складність буде визначатися за такими співвідношеннями:

$$\begin{aligned} 0 \le L(v) \le L \times \beta \\ 0 \le L(u) \le L \times \alpha \end{aligned}, \tag{3}$$

де α та β – коефіцієнти.

Використання такого методу дозволяє підвищити загальну складність атаки на $O(n)$.

Чергування виходів двох LFSR – генераторів. Якщо s і t є виходи двох LFSR – генераторів довжиною L_1 і L_2 відповідно, то загальна лінійна складність буде визначатися за такою формулою:

$$0 \le L(u,v) \le (L_1 + L_2) \times (m+n), \tag{4}$$

де m та n – довжини зчитаних блоків.

Використання такого методу комбінації дозволяє підвищити загальну складність атаки на $O(n)$.

Перемішування виходу LFSR – генератора. Якщо s є виходом LFSR – генератора довжиною L, то загальна лінійна складність буде визначатися за такою формулою:

$$L \le L(s) \le L \times n, \tag{5}$$

де n – довжина блоку перемішування.

Використання такого методу комбінації дозволяє підвищити загальну складність атаки на $O(n)$.

Авторами розроблено структуру генератора ПВЧ з використанням трьох базових LFSR у каскаді Голлмана. У даному випадку були використані LFSR з реконфігуративною структурою, тобто реалізовано можливість зміни початкового стану регістру зсуву та поліному зворотних зв'язків.

Це дало можливість використовувати генератор ПВЧ на базі каскаду Голлмана у динамічному режимі, тобто почергово змінювати ініціалізуючі значення базових LFSR у каскаді, що є еквівалентом роботи декількох генераторів Голлмана із різними внутрішніми станами.

З точки зору ефективного застосування розробленого авторами генератора в динамічному режимі найоптимальнішим є застосування режимів чергування та перемішування.

2 Дослідження комбінації використання способів чергування та перемішування

У загальному випадку можна застосовувати комбінацію режимів чергування та перемішування двома способами: а) спочатку проводиться чергування виходів n генераторів Голлмана, а потім над отриманою послідовністю здійснюється перемішування; б) спочатку проводиться перемішування кожного з виходів n генераторів Голлмана, а потім здійснюється їх чергування. Розглянемо детальніше ці два випадки.

Спосіб а)

Позначимо через u_i довжину блоку вихідної послідовності, отриманої за допомогою i-го стану генератора Голлмана. Під станом генератора розуміємо генератор Голлмана із множиною наборів ініціалізуючих векторів та поліномів зворотних зв'язків. Тоді лінійна складність отриманої послідовності s після застосування процедури чергування буде визначатися за формулою:

$$0 \le L(s) \le (L_1 + L_2 + ... + L_n) \times (m_1 + m_2 + ... + m_n), \tag{6}$$

де L_i – лінійні складності кожного i-го стану генератора Голлмана; $1 \le m_i \le u_i$ – розміри блоків чергування [4].

Приріст загальної складності криптоатаки залишається таким же, тобто $O(n)$.

При перемішуванні вхідними параметрами є послідовність s, параметр $k \in N^*$ – довжина блоку перемішування, причому

$$k < 2^{L(s)}. \tag{7}$$

Тоді, маємо:

$$L(s) \le L(u) \le L(s) \times k. \tag{8}$$

Використовуючи (6), отримаємо нерівність:

$$0 \le L(s) \le L(u) \le L(s) \times k \le (L_1 + L_2 + ... + L_n) \times (m_1 + m_2 + ... + m_n) \times k. \tag{9}$$

Враховуючи (7) і (6), можна отримати остаточну формулу

$$0 \le L(u) < (L_1 + ... + L_n) \times (m_1 + ... + m_n) \times 2^{(L_1 + ... + L_n) \times (m_1 + ... + m_n)}, \tag{10}$$

що буде еквівалентно:

$$0 \le L(u) < \sum_{i=1}^{n} L_i \times \sum_{i=1}^{n} m_i \times 2^{\sum_{i=1}^{n} L_i \times \sum_{i=1}^{n} m_i} \tag{10.1}$$

Спосіб б)

Позначимо через u_i вихід i-го стану генератора Голлмана довжиною L_i, а $k_i < 2^{L_i}$ – довжина блоку перемішування для u_i відповідно. Тоді лінійна складність для кожного перемішаного u_i буде визначатися за формулою:

$$\begin{aligned} L_1 \le L(u_1) \le L_1 \times k_1 < L_1 \times 2^{L_1} \\ L_2 \le L(u_2) \le L_2 \times k_2 < L_2 \times 2^{L_2} \\ \vdots \\ L_n \le L(u_n) \le L_n \times k_n < L_n \times 2^{L_n} \end{aligned} \tag{11}$$

Якщо $1 \le m_i \le u_i$ – розміри блоків чергування, тоді лінійна складність отриманої послідовності після чергування буде визначатися за формулою:

$$\begin{aligned} 0 \le L(s) \le (L(u_1) + L(u_2) + \ldots + L(u_n)) \times (m_1 + m_2 + \ldots + m_n) < \\ < (L_1 \times 2^{L_1} + L_2 \times 2^{L_2} + \ldots + L_n \times 2^{L_n}) \times (m_1 + m_2 + \ldots + m_n). \end{aligned} \tag{12}$$

Дану формулу ще можна записати у вигляді:

$$0 \le L(s) < \sum_{i=1}^{n} \left(L_i \times 2^{L_i}\right) \times \sum_{i=1}^{n} m_i \tag{12.1}$$

Скористаємося методом математичної індукції для доведення, що використання способу а) дозволяє досягти більшого теоретично можливого значення лінійної складності, ніж способу б).

Доведення.

Крок 1. При $n = 1$ з 10.1 та 12.1 випливає: $L_1 \times m_1 \times 2^{L_1 \times m_1} > L_1 \times 2^{L_1} \times m_1$. Очевидно, що ця нерівність правильна.

Крок 2. Припустимо, що при $n = p$ нерівність

$$\begin{aligned} (L_1 + L_2 + \ldots + L_p) \times (m_1 + m_2 + \ldots + m_p) \times 2^{(L_1 + L_2 + \ldots + L_p) \times (m_1 + m_2 + \ldots + m_p)} > \\ > (L_1 \times 2^{L_1} + L_2 \times 2^{L_2} + \ldots + L_p \times 2^{L_p}) \times (m_1 + m_2 + \ldots + m_p) \end{aligned}$$

є правильною. Її можна записати в іншому вигляді:

$$\sum_{i=1}^{p} L_i \times 2^{\sum_{i=1}^{p} L_i} > \sum_{i=1}^{p} \left(L_i \times 2^{L_i}\right).$$

Крок 3. Доведемо правильність нерівності при $n = p + 1$:

$$\begin{aligned} (L_1 + L_2 + \ldots + L_{p+1}) \times (m_1 + m_2 + \ldots + m_{p+1}) \times 2^{(L_1 + L_2 + \ldots + L_{p+1}) \times (m_1 + m_2 + \ldots + m_{p+1})} > \\ > (L_1 \times 2^{L_1} + L_2 \times 2^{L_2} + \ldots + L_{p+1} \times 2^{L_{p+1}}) \times (m_1 + m_2 + \ldots + m_{p+1}) \end{aligned}$$

При скороченні на $(m_1 + m_2 + \ldots + m_{p+1})$ дана нерівність матиме вигляд:

$$\sum_{i=1}^{p+1} L_i \times 2^{\sum_{i=1}^{p+1} L_i \times \sum_{i=1}^{p+1} m_i} > \sum_{i=1}^{p+1} \left(L_i \times 2^{L_i}\right).$$

Використовуючи припущення Кроку 2, бачимо, що дана нерівність правильна.

3 Частковий випадок для практичної реалізації

Для практичної реалізації складно реалізувати регістр даних потрібної довжини для виконання функції перемішування, тому автори пропонують розглянути частковий випадок з використанням лише двох станів Голлмана для операції чергування.

Особливістю є те, що проводиться чергування двох сусідніх блоків даних u_i та u_{i+1}. Тоді лінійна складність кожного з отриманих після чергування блоків $L(s_i)$ буде визначатися:

$$\begin{aligned} 0 \le L(s_1) \le (L_1 + L_2) \times (m_1 + m_2) \\ 0 \le L(s_2) \le (L_2 + L_3) \times (m_2 + m_3) \\ \vdots \\ 0 \le L(s_n) \le (L_{n-1} + L_n) \times (m_{n-1} + m_n). \end{aligned} \tag{13}$$

Позначення L_i, $1 \le m_i \le u_i$ ті самі, що й у (6).

Перемішування в одержаних блоках із (13) проводиться з k_i. Кінцева формула визначення лінійної складності $L(f_i)$ буде мати вигляд:

$$\begin{aligned} 0 \le L(f_1) < (L_1 + L_2) \times (m_1 + m_2) \times 2^{(L_1+L_2)\times(m_1+m_2)} \\ 0 \le L(f_2) < (L_2 + L_3) \times (m_2 + m_3) \times 2^{(L_2+L_3)\times(m_2+m_3)} \\ \vdots \\ 0 \le L(f_n) < (L_{n-1} + L_n) \times (m_{n-1} + m_n) \times 2^{(L_{n-1}+L_n)\times(m_{n-1}+m_n)} \end{aligned} \tag{14}$$

або, що те саме:

$$0 \le L(f_i) < (L_{i-1} + L_i) \times (m_{i-1} + m_i) \times 2^{(L_{i-1}+L_i)\times(m_{i-1}+m_i)}. \tag{14.1}$$

Висновок

У даній статті автори розглянули підходи, щодо підвищення лінійної складності генераторів ПВЧ, побудованих на базі каскаду Голлмана. Досліджено два способи комбінації операцій чергування та перемішування. Доведено, що застосування до послідовності спочатку операції чергування, а потім перемішування, дає можливість теоретичного досягнення більшого максимального значення лінійної складності.

Для практичної реалізації запропоновано частковий випадок.

Список літератури: 1. *A. Menezes, P. van Oorschot, S. Vanstone,* Handbook of Applied Cryptography, CRC Press, 1996. *2. http://*www.unix.kg/cgi-bin/document.pl?lang=rus&id=17. 3. *Молдовян А.А., Молдовян В.А., Советов В.Я.* Криптография. Серия «Учебники для вузов. Специальная литература». СПб.: Изд-во «Лань», 2000. 224 с., ил. 4. *http: //* rotranslators. free. fr/pdfdoc /REGPAPER/TU02/100.PDF.

Тернопільська академія
народного господарства

Надійшла до редколегії 14.04.2003

СИСТЕМЫ И СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

УДК 681.3.06

А. В. ПОТИЙ, канд. техн. наук, С. С. ЛАВРИНЕНКО

ЗАЩИЩЕННЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

По данным годового отчета «2001 Computer Crime and Security Survey» [1] Института компьютерной безопасности в Сан-Франциско и ФБР, финансовые потери от компьютерных преступлений в США за минувший год выросли на 43% с 265,6 млн. долл. до 377,8 млн. При этом 85% респондентов из 538, в основном из промышленных и государственных структур, заявили о фактах нарушения компьютерной безопасности, причем не только из-за атак злоумышленников. Почти 64% были озабочены понесенными убытками, но лишь 35% смогли оценить их в денежном выражении. Около 70% респондентов заявили, что чаще всего атакам подвергались Internet-каналы, а 31% показали, что атакам подвергались внутрикорпоративные системы. Случаи вторжения извне подтверждали 40% респондентов (в 2000 г. – 25%), а 38% фиксировали отказ в обслуживании (27% в 2000 г.). На нарушение привилегий из-за злоупотребления сотрудниками работой в Сети жаловались 91% респондентов, а 94% обнаружили в своих системах вирусы (в 2000 г. это отмечали 85%).

Таким образом, одними из ведущих проблем в области информационной безопасности в минувшем году стали атаки на платежные системы, дискредитация компаний (отказ в обслуживании), производственный саботаж, вскрытие корпоративных секретов, нарушение прав интеллектуальной собственности. По оценкам отдела по науке и технологиям при Президенте США, ежегодный урон, наносимый американскому бизнесу компьютерными злоумышленниками в последние годы, достигал 100 млрд. долл. Потери от несанкционированного доступа к информации, связанной с деятельностью финансовых институтов США, составляли не менее 1 млрд. долл. в год. Таким образом, американский бизнес вплотную подошел к тому рубежу, когда своевременное и адекватное решение вопросов безопасности для него становится экономически целесообразным.

1 Критерии и ориентиры в области безопасности

Работы над критериями безопасности систем начались еще в 1967 г., и в 1970 г. появился первый отчет под названием «Security Controls for Computer Systems». В 1983 г. Министерство обороны США выпустило «Orange Book» – книгу в оранжевой обложке под названием «Критерии оценки достоверных компьютерных систем» (Trusted Computer Systems Evaluation Criteria). В «Оранжевой книге» достоверная система определяется как «система, использующая достаточные аппаратные и программные средства для обеспечения одновременной обработки информации разной степени секретности группой пользователей без нарушения прав доступа».

Выделяются два основных критерия оценивания достоверных систем:

- политика безопасности (набор правил и норм, определяющих дисциплину обработки, защиты и распространения информации, а также выбор конкретных механизмов обеспечения безопасности; это активный компонент защиты);
- гарантированность (степень доверия, которая может быть оказана конкретной реализации ОС; отражает уровень корректности механизмов безопасности; является пассивным компонентом защиты).

В соответствии с «Оранжевой книгой» выделяются три роли: системный администратор, системный оператор и администратор безопасности.

За пределами США также появились аналоги «Оранжевой книги»: это руководящие документы Гостехкомиссии (1992 г.), а также «Критерий оценки безопасности информационных технологий» (ITSEC – Information Technology Security Evaluation Criteria, 1991), действующий в Великобритании, Германии, Франции и Нидерландах.

Конечно же, в силу необходимости унификации подходов к информационной безопасности в конце концов возникла потребность снять двойственность регулирования, которая отдельно велась в США (TCSEC) и Европе (ITSEC). Поэтому был принят новый международный стандарт, получивший название «Единые критерии для оценки безопасности в области информационных технологий» [2], являющийся международным стандартом ISO/IEC 15408.

Описание Common Criteria V2.1 содержится в трех книгах:

1. Введение и общая модель (CCIMB-99-031).
2. Функциональные требования к безопасности (CCIMB-99-032).
3. Требования к гарантиям безопасности (CCIMB-99-033).

В «Единых критериях» выделяются 11 функциональных классов:

- аудит;
- криптографическая поддержка;
- передача данных;
- защита данных пользователя;
- идентификация и аутентификация;
- управление безопасностью;
- конфиденциальность;
- защита функций безопасности целевой системы;
- утилизация ресурсов;
- доступ к целевой системе;
- достоверные пути/каналы.

Внутри каждого их этих классов содержится несколько семейств, а в каждом семействе – от одного до нескольких компонентов.

Критерии, сформулированные в TCSEC, ITSEC и CCITSE, определяют разбиение компьютерных систем на 4 уровня безопасности (A, B, C, D) в зависимости от степени достоверности. Уровень A самый высокий. Далее идет уровень B (в порядке понижения безопасности здесь идут классы B3, B2, B1). Затем наиболее распространенный уровень C (классы C2 и C1). Самый нижний уровень – D (системы, которые не смогли получить аттестацию по заявленным выше классам).

2 Защищенность операционных систем

Любую стандартную операционную систему можно сделать более защищенной или укрепить ее с помощью простых процедур – например, отказаться от очевидных значений пароля администратора или отключить соединения с Web, если они не используются. Но эти отвечающие здравому смыслу шаги могут требовать больших затрат времени и, увы, далеко не всегда способны защитить критически важный сервер от целеустремленного хакера.

Истинная защищенная операционная система изначально строится с учетом требований безопасности. Защищенную ОС отличает наличие трех следующих «рычагов»:

- Политика принудительного контроля доступа (mandatory access control). MAC обеспечивает принудительное управление доступом на основе классической модели Белла-Лападулы, цель которой – не допустить перетекания информации из более секретных объектов в менее секретные.
- Администрирование привилегий, которые можно использовать для управления и ограничений возможностей пользователя или приложения, управляющих системой или ее частью. В защищенной ОС можно устанавливать программу, которая никогда не сможет изменить назначение привилегий, даже в том случае, если каким-то образом эта программа попадет в полную власть хакера. Такое решение не позволит хакеру проникнуть в систему через какое-то приложение, если, скажем, ему удастся отключить пароль, защищающий другие приложения.
- Независимая экспертиза, проводимая, например, в Национальном институте стандартов и технологии или Агентством национальной безопасности США.

Если исходить из этих критериев, то самые распространенные ОС – Microsoft Windows NT и Windows 2000, а также различные версии Unix – не являются защищенными системами, хотя Windows 2000 – это заметный шаг вперед благодаря предусмотренной в ней «защите системных файлов», позволяющей обеспечить безопасность некоторых критически важных компонентов.

Защищенные операционные системы крупнейших производителей Unix-систем, таких как Sun Microsystems и Hewlett-Packard, существуют уже давно, но они не пользовались особой популярностью из-за сложности управления и отсутствия ряда важных функций, которые есть у их коммерческих аналогов. Кроме того, они были не полностью совместимы с приложениями, работавшими с их менее защищенными вариантами.

Как правило, эти защищенные ОС использовались только в условиях с высоким уровнем риска – в банках и госучреждениях, которые могли позволить себе уделять достаточно времени и средств управлению подобными системами.

3 Использование Unix-систем

Сегодня многие индивидуумы и организации остановили свой выбор на операционной системе Linux, взяв ее на вооружение в качестве основной платформы. Причем, если раньше эта ОС использовалась преимущественно в качестве основы для построения недорогих маршрутизаторов, Web-серверов и почтовых серверов, то сегодня имеется целый ряд более серьезных задач, которые удобно решать на ее базе: серверы баз данных и доступа, серверы приложений и полнофункциональные межсетевые экраны. Благодаря высокой устойчивости ОС Linux ей можно смело поручать длительные вычислительные задачи.

Как известно, Linux – это клон Unix, семейства операционных систем, объединенных общей идеологией. Первые Unix-системы создавались исключительно как средство совместного ведения проектов научными коллективами университетов, и ее разработчики не придавали особого значения вопросам безопасности и защиты информации. Во всяком случае, эти вопросы точно стояли не на первом месте. Это привело к тому, что ни одну из классических Unix-систем нельзя сегодня назвать по-настоящему безопасной.

Относительно защищенный вариант Unix-системы можно построить на базе ее штатных средств, и для большинства применений это будет вполне адекватный уровень безопасности. Однако в самой идеологии построения Unix имеется ряд фундаментальных изъянов, которые невозможно преодолеть только грамотным администрированием.

Фактически, в Unix предполагается всего два варианта статуса пользователя: обычный и суперпользователь (root), что сразу порождает две проблемы. Первая проблема заключается в том, что root никому не подконтролен: он может изменить любую настройку, имеет доступ абсолютно ко всем объектам в файловой системе, может стереть или модифицировать любые

данные и записи в журналах регистрации. Такая «концентрация власти», конечно же, неприемлема для защищенной системы. Вторая проблема связана с необходимостью изменения текущего уровня привилегий пользователя для выполнения некоторых действий (например, для смены пароля пользователю требуется временное разрешение на запись в файл /etc/shadow или аналогичный). Традиционно это достигалось путем установки флага SUID (или SGID) на файлы исполняемых модулей программ, в результате чего порождаемый при запуске такого файла процесс приобретает не права запустившего его пользователя, а права владельца файла. Нет необходимости говорить, к чему может привести даже небольшая ошибка в программе с установленным флагом SUID и владельцем root.

Реализованная в Unix модель разграничения доступа дискретна – права доступа субъектов (пользователей, процессов) к объектам (файлам, каталогам и т.п.) задаются явно, в матричной форме. На практике же часто возникает необходимость в реализации принудительной модели, при которой права доступа субъекта к объекту определяются уровнем субъекта и классом объекта, либо комбинированной модели, включающей в себя принудительную и доступ на основе списков управления доступом. Причем, если в других ОС, использующих дискретную модель (например, Novell NetWare), необходимого разграничения достичь можно, то в Unix реализация некоторых требований по разграничению может оказаться попросту невозможной. Это связано с тем, что в Unix права доступа задаются всего для трех категорий субъектов: владельца файла, ассоциированной с файлом группы и всех прочих.

Ниже приводятся две системы, которые могут быть применены в качестве решения приведенных проблем с организацией защиты: RSBAC (Rule Set Based Access Control) и Security-Enhanced Linux, позволяющие построить защищенную систему на базе ядра Linux.

3.1 RSBAC

RSBAC – это надстройка над ядром Linux и комплект утилит управления, позволяющие создать на базе любого дистрибутива Linux защищенную систему. Одной из целей создания RSBAC, по заявлению авторов, является выполнение всех требований B1 по классификации так называемой «Оранжевой Книги» (хотя набор этих требований уже несколько устарел).

Реализация механизмов обеспечения безопасности выполнена на уровне ядра системы и позволяет эффективно контролировать все процессы. Системные вызовы, затрагивающие безопасность, дополняются специальным кодом, выполняющим обращение к центральному компоненту RSBAC. Этот компонент принимает решение о допустимости данного системного вызова на основе многих параметров:

- типа запрашиваемого доступа (чтение, запись, исполнение);
- субъекта доступа;
- атрибутов субъекта доступа;
- объекта доступа;
- атрибутов объекта доступа.

Функционально RSBAC [3] состоит из нескольких модулей, а центральный компонент принимает комплексное решение, основываясь на результатах, возвращаемых каждым из активных в данный момент модулей (какие модули задействовать и в каком объеме определяется на этапе настройки системы). Начиная с версии 1.1.0, в RSBAC включены следующие модули, реализующие различные функции и модели управления доступом:

- **MAC (Mandatory Access Control).** Модуль MAC обеспечивает принудительное управление доступом на основе классической модели Белла-Лападулы [4], цель которой – не допустить перетекания информации из более секретных объектов в менее секретные.
- **FC (Functional Control).** Данный модуль реализует простую ролевую модель, в которой доступ к системной информации разрешен только администраторам системы, а

доступ к информации, связанной с безопасностью, разрешен только офицерам безопасности.

- **SIM (Security Information Modification).** Модуль SIM обеспечивает возможность модификации данных, помеченных как «security information», только администраторами безопасности.
- **PM (Privacy Model).** Данный модуль реализует модель безопасности, направленную на обеспечение приватности личных данных. Основная идея [5] состоит в том, чтобы пользователь мог получить доступ к персональным данным только, если они ему необходимы для выполнения текущей задачи и если он авторизован на ее выполнение. Кроме того, цели выполнения текущей задачи должны совпадать с целями, для которых эти данные собраны, либо должно быть получено согласие субъекта этих данных.
- **MS (Malware Scan).** Этот модуль обеспечивает сканирование всех файлов на наличие вредоносного кода. Дополнительно данный модуль может контролировать все запросы на чтение файлов и соединений TCP/UDP. В текущей версии модуль умеет обнаруживать вирусы Bliss.A, Bliss.B, VHP-648, Israeli, Eddie2, Dark Avenger и 1704C. Конечно, это немного и вирусы не самые новые и опасные, но это только начало.
- **FF (File Flags).** Модуль FF предоставляет механизм установки и проверки флагов на файлы и каталоги. Причем модифицировать флаги разрешено только офицерам безопасности системы. Пока поддерживаются флаги execute_only (для файлов), read_only (для файлов и каталогов), search_only (для каталогов), secure_delete (для файлов), no_execute (для файлов) и add_inherited (для файлов и каталогов). Смысл флагов ясен из их названия и, конечно, знаком тем, кто пользовался механизмами атрибутов и прав в файловой системе Novell NetWare.
- **RC (Role Compatibility).** Данный модуль определяет 64 роли и 64 типа для каждого вида объекта. Виды объектов могут быть следующими: file, dir, dev, ipc (interprocess communication), scd (system control data), process. Для каждой роли отношение к различным типам и другим ролям настраивается индивидуально, в зависимости от вида запроса. Используя данный модуль, можно настроить разделение обязанностей между администраторами, избежав при этом назначения избыточных прав.
- **AUTH (Authorization Enforcement).** У этого модуля задача очень конкретная – контролировать запросы процессов на смену текущего идентификатора пользователя. Под контролем данного модуля программе недостаточно просто иметь установленный бит suid, ей необходим специальный атрибут, разрешающий такое действие. Причем он может быть как глобальным (uid может быть сменен на любой), так и списочным (процесс может сменить свой uid только на определенные).
- **ACL (Access Control List).** ACL – самый «прикладной» модуль из данного списка. Он определяет, какие субъекты могут получать доступ к данному объекту и какие типы запросов им разрешены. Субъектом доступа может быть как простой пользователь, так и роль RC и/или группа ACL. Объекты группируются по видам, но каждый имеет собственный список ACL. Если права доступа к объекту не заданы явно, они наследуются от родительского объекта с учетом маски наследования прав. Эффективные права доступа субъекта к объекту складываются из прав, полученных непосредственно, и прав, полученных через назначение на роль или членство в группе ACL. Такой подход к назначению прав покажется очень знакомым специалистам, работавшим с NetWare. Более того, утилита управления acl_grant имеет режим совместимости по названиям прав с аналогичной утилитой из NetWare.

Вся дополнительная информация, используемая RSBAC, хранится в дополнительном каталоге, который доступен только ядру системы. В зависимости от описанного уровня абстракции исполняемых задач и необходимой степени защиты выбираются различные сочетания модулей.

RSBAC дает возможность избавить систему от общих недостатков Unix. Появление должности «офицер безопасности» (security officer, security administrator) решает проблему неподконтрольности основного администратора системы, а категории «офицер по защите данных» (data protection officer) децентрализует администрирование, позволяя практически реализовать принцип «четырех глаз», в соответствии с которым все критичные операции не должны производиться в одиночку. Действительно, возможно построение системы, в которой нового пользователя по-прежнему заводит root, однако его уровень безопасности и права доступа к ресурсам задаются офицером безопасности и офицером по защите данных.

Стоит, однако, помнить об одной простой вещи: любая защита накладывает ограничения, а степень защищенности всегда обратно пропорциональна удобству работы с системой. Погоня за секретностью и безопасностью может привести к временной блокировке доступа к данным, а в отдельных случаях – и к их потере.

RSBAC можно интегрировать с любым дистрибутивом Linux, необходимо лишь внести соответствующие исправления в исходные тексты ядра системы. Эта процедура осуществляется при помощи стандартной утилиты patch и прилагаемого файла-»заплатки», который предлагается для разных версий ядра. После этого происходит его обычная настройка (make config, make menuconfig и т.п.), при этом в настройке появляется несколько новых пунктов. После загрузки ядра настройка системы производится при помощи прилагаемых утилит администрирования. Система снабжена документацией и подборкой теоретических материалов, где описаны все используемые модели защиты.

3.2 Security-Enhanced Linux

Вторая система – Security-Enhanced Linux имеет такое же назначение, как RSBAC и также представляет собой дополнения к ядру и набор утилит. Обе системы доступны под лицензией GPL, однако разработка Security Enhanced Linux продвигается Агентством национальной безопасности США. Security Enhanced Linux «моложе» RSBAC – ее релиз впервые был представлен в декабре 2000 года [6]. Security-Enhanced Linux обеспечивает гибкую архитектуру принудительного контроля доступа (flexible mandatory access control architecture – FLASK) [7], использующую развитый язык описания конфигураций политики безопасности. С использованием этого языка описаний разработана конфигурация системы, реализующая идеологию Type Enforcement [8].

В терминологии данной системы политика безопасности определяет набор доменов и типов. Каждый субъект (процесс) в каждый момент времени ассоциирован с определенный доменом, а каждый объект – с определенным типом. Как и в классической матрице, в политике определяются возможные виды доступа доменов к типам и допустимые способы взаимодействия между доменами. В частности, в зависимости от используемых типов может происходить автоматическое переключение домена.

В политике безопасности также определен набор ролей. Для пользователей первичная установка роли происходит в процедуре регистрации (login), а переключение роли – при помощи команды newrole (в некотором роде аналог su). Системные процессы работают с ролью system_r, обычные пользователи – с ролью usr_r, а для системных администраторов зарезервирована роль sysadm_r.

Для каждой роли в политике безопасности задается набор доменов, в которых допускается работа с этой ролью. Всем пользовательским ролям назначается стартовый домен: user_t для роли user_r и sysadm_t для роли sysadm_r. По мере выполнения программ, запускаемых из стартовой оболочки shell, может происходить автоматическое перемещение в другие домены для обеспечения изменения привилегий. Выбор домена, в который должно быть произведено перемещение, осуществляется не только исходя из типа запускаемой программы, но и с учетом текущего домена. В частности, при запуске браузера Netscape обычным пользовате-

лем (текущий домен user_t) произойдет перемещение в домен user_netscape_t, а при запуске этой же программы администратором (текущий домен sysadm_t) перемещение произойдет в другой домен – sysadm_netscape_t. Такой подход не позволит программе выполнить потенциально опасные действия – соответствующий домен серьезно ограничит права администратора. В обычных дистрибутивах Linux в случае с браузером Netscape проблема решалась проще – в настройке по умолчанию программа просто не запускалась с правами root.

Для администраторов в Security-Enhanced Linux заданы достаточно жесткие ограничения. В частности, нельзя зайти в систему удаленно – при необходимости такой процедуры необходимо сначала произвести вход обычным пользователем, а потом переключиться на административную роль при помощи команды newrole, производящей дополнительную аутентификацию. Впрочем, и в большинстве современных Unix-подобных систем администратору также запрещен удаленный вход и для этой цели используется команда su.

Пример с браузером не случаен – ему уделено особое внимание в перечне целей политики безопасности. Во избежание исполнения браузером вредоносного динамического кода (Java-апплеты, сценарии JavaScript) для него определен специальный домен (точнее, два домена – для пользователей и администраторов), ограничивающий полномочия. Причем определяются два подтипа: один ограничивает доступ браузера к локальным файлам только чтением, другой допускает запись в них.

Политика безопасности должна контролировать различные формы прямого доступа к данным, поэтому в ней определяются различные типы для устройств памяти ядра (kernel memory device), дисковых устройств, специальных файлов в каталоге /proc, а также различные домены для процессов, которым необходим доступ к этим ресурсам. Обеспечение целостности ядра достигается определением различных типов для загрузочных файлов, объектных файлов подключаемых модулей, утилит для работы с модулями и файлов конфигурации модулей. Соответствующим процессам, которым требуется право записи в эти файлы, назначаются специальные домены.

Системное ПО, файлы конфигурации и журналы также нуждаются в защите. Для этой цели также определяются отдельные типы для системных библиотек, исполняемых файлов, файлов конфигурации и журналов, а работающим с ними программам и ролям назначаются специальные домены. В результате запись журналов может вести только система регистрации (syslog), а модификация системного ПО производится только администраторами. Есть решение и для уже упомянутой проблемы, присущей программам с повышенными привилегиями (с установленными флагами suid или sgid). Для таких программ также назначаются отдельные домены, не позволяющие им превысить необходимые полномочия.

Политика безопасности уделяет особое внимание тщательному разделению процессов по привилегиям и защиту привилегированных процессов от выполнения ошибочного или опасного кода. Путем установки атрибута executable только на необходимые для исполнения привилегированным процессом программы политика безопасности может достичь того, что при входе в привилегированный домен процессу будет позволено выполнение только стартовой программы для данного домена, динамического компоновщика и системных разделяемых библиотек. Администратор ограничен выполнением системных программ и программ, созданных им самим – выполнение программ, созданных другими пользователями запрещено. Более того, система минимизирует взаимодействие между обычными пользовательскими процессами и системными. Только системным процессам и администраторам разрешен доступ к записям в procfs, относящимся к другим доменам; ограничено использование вызова ptrace по отношению к другим процессам и доставка сигналов между разными доменами. При использовании файловой системы также приняты дополнительные меры предосторожности: домашние каталоги администраторов и обычных пользователей отнесены к разным типам; создаваемым в каталогах общего пользования (/tmp и др.) файлам также присваиваются разные типы, в зависимости от создавшего их домена.

По сравнению с RSBAC система Security-Enhanced Linux менее гибкая, зато имеет очень хорошую предопределенную политику безопасности. Ее сложно применить для «небольшого усиления» стандартного дистрибутива Linux – настройка достаточно сложна и невозможна без изучения специального языка конфигурации. Но это не недостатки, а скорее следствия целей создания системы. В [8] отмечается, что не стоит пытаться создать на базе идеологии Type Enforcement операционную систему общего назначения, которая автоматически обеспечит надежность, безопасность и другие свойства любым запускаемым в ней приложениям. Type Enforcement – это механизм, позволяющий произвести интеграцию приложений и менеджера ресурсов, сведя при этом к минимуму присущие им недостатки. Это средство построения специализированных защищенных систем, в которых все лишние функции убраны, а поведение оставленных жестко определено матрицей Type Enforcement.

4 Новый стандарт

NIST-ом (Национальный Институт Сертификации и Технологий) ведутся разработки защищенного профиля (ЗП) для операционных систем, который называется «CSPP-OS – COTS Security Protection Profile – Operating Systems» [9]. На данное время существует черновая версия v.0.4 от 5 февраля 2001г. Этот ЗП разработан на основе руководства CSPP [10]. Ниже приводится краткий обзор данного документа.

4.1 Назначение

Назначение CSPP-OS состоит в том, чтобы определить и объяснить требования, необходимые для решения проблемы защиты операционных систем COTS (коммерческого назначения).

4.2 Возможности

Тип системы. CSPP-OS предусматривает требования, необходимые для определения потребностей операционных систем и в автономных, и в распределенных, и в многопользовательских информационных системах.

Тип доступа. CSPP-OS признает две формы легитимного доступа, а именно: общественный (публичный) и «аутентифицированные пользователи». При общественном доступе пользователь не имеет уникальный идентификатор и не аутентифицируется до получения доступа. Пример – доступ к информации относительно публично доступной web-страницы. Такие пользователи имеют законный доступ, но отличаются от «аутентифицированных пользователей», которые:

1) уникально распознаются системой;
2) имеют законный доступ к информации, отличной от общедоступной;
3) аутентифицируются до предоставления такого доступа (2).

Характер использования. CSPP-OS-операционные системы применяются для защиты информации в реальных мировых средах: и коммерческих, и правительственных.

- В контексте правительственной среды CSPP-OS рассматриваются как подходящие для определения базовых требований защиты для «чувствительной-но-несекретной» или одноуровневой секретной информации в среде, где все заверенные пользователи имеют доступ к уровню секретности обрабатываемой информации. Для секретной среды общественный доступ в CSPP-OS запрещен. Для «чувствительной-но-несекретной» среды общественный доступ может использоваться с дополнительным средством управления, вне механизмов ОС, предоставленных операционной средой.

- В контексте коммерческой среды CSPP-OS рассматриваются как подходящие для определения базовых требований защиты для информации в средах, где все аутентифицированные пользователи:
    1. Доверенные, т.е. злонамеренно не пытаются ни войти в систему, ни обходить средство управления доступа.
    2. Не проявляют наклонностей или способностей к сложным попыткам проникновения.

Общественный доступ позволяется со средством управления окружающей средой и находящимися вне ОС механизмами защиты.

Ключевые предположения. Ключевые предположения, которые применяются к CSPP-OS:

- Цель оценивания (TOE – ОС, для которой определяются требования) представляет собой информационную технологию (ИТ) COTS (коммерческого назначения).
- Заверенные пользователи признают потребность в защищенной ИТ.
- Заверенным пользователям можно разумно доверять, чтобы правильно применять политику безопасности организации в их контролируемых действиях.
- Администрирование защиты выполнено компетентно.
- Автоматизация процесса бизнеса (цели) осуществлена с должным отношением к тому, что не может быть предусмотрено в CSPP-OS.

4.3 Обзор требований CSPP-OS

Системы, входящие в число операционных систем COTS, достигают таких преимуществ, как высокий спрос на продукт – например, сочетанием высоких функциональных возможностей с низкой стоимостью. Однако эти преимущества не могут быть достигнуты без некоторых потерь; например – способность защиты. CSPP-OS отождествляют собой рентабельность, базовый уровень безопасности для систем, построенных на основе COTS ОС-м, гарантируя, что был достигнут разумный уровень защиты.

CSPP-OS также затрагивают те области, где нереально ожидать, что типичная операционная система COTS обеспечит достаточную защиту. Эти области – прямой результат того факта, что управляющие факторы COTS (функциональные возможности, стоимость, и время нахождения на рынке) имеют тенденцию работать против увеличения способностей защиты, оговариваемые в CSPP-OS.

Гарантия. CSPP-OS гарантии были выбраны, чтобы обеспечить уровень доверия, следующий из:

1) существующих лучших методик разработки COTS;
2) не всеобщей (а следовательно дорогостоящей) оценки третьей стороной.

Это приводит, в результате, к техническим контрмерам ОС-м, которые:

- Являются достаточными, чтобы управлять сообществом доверенных (то есть непреднамеренно злоумышленных) заверенных пользователей.
- Могут предусматривать защиту против несложных (простых) технических нападений.
- Не предусматривают достаточную защиту против сложных (искушенных), технических нападений (включая отказ в обслуживании).

Функциональные возможности. CSPP-OS-операционная система содержит следующие потребности пользователей:

- Принудительная политика управления доступом между активными сущностями (субъектами) и пассивными объектами, основанная на идентификации субъекта и позволенных для него действий.

- Обеспечение поддержки для управления доступом, основанном на свойствах среды типа времени-дня и точки входа.
- Сопротивление истощению ресурса путем внедрения контроля выделения ресурсов.
- Обеспечение механизмов обнаружения некоторых ненадежностей.
- Обеспечение механизмов для доверенного восстановления в случае некоторых системных отказов или обнаруженной ненадежности.
- Поддержка этих способностей в распределенной системе, связанной через неавторизованную сеть

CSPP-OS не предусматривают следующего:

- Обеспечения средств управления на основе меток, свойственных защите управляемой информации (правительственная секретная информация, собственность компаний или экспортируемые ограниченные данные) в средах, содержащих заверенных пользователей, которым не позволяют доступ к такой информации.
- Защиты против злоупотребления полномочиями.
- Адекватной защиты против сложных нападений (включая отказ в обслуживании).
- Обеспечения достаточной защиты против инсталляции, эксплуатации или ошибок администрирования.

4.4 Требуемые функциональные возможности защиты

CSPP-OS определяет требования для операционной системы к функциональным возможностям защиты, перечисленные ниже:

- Выполнение политики управления доступом, продиктованные политикой безопасности ИТ.
- Назначение уникального идентификатора каждому заверенному пользователю.
- Назначение уникального идентификатора каждому системному процессу, включая те, которые выполнены не от имени пользователя (например, процессы, запущенные в системе bootup подобно Unix «inetd»).
- Подтверждение требуемых полномочий перед разрешением любому пользователю действий, отличных от общеизвестного набора операций (например, чтение с общественного сайта).
- Аудит для поддержки индивидуальной ответственности и обнаружения причин ненадежностей и неполадок.
- Предоставление управления разрешениями доступа – то есть инициализация, назначение и модификация прав доступа (например чтение, запись, выполнение) к объектам данных относительно:
    1) названия (имени) объекта или принадлежности к группе;
    2) свойств среды, как-то системного времени и точки входа.
- Особенности распределения ресурсов, обеспечивающие меру сопротивления к их истощению.
- Механизмы для обнаружения некоторой ненадежности.
- Системные особенности восстановления, обеспечивающие меру живучести в контексте системных отказов и ненадежности.
- Автоматизированная поддержка для помощи в проверке безопасной доставки, инсталляции, эксплуатации и администрирования.

5 Заключение

Исследование всего вышесказанного позволило сделать следующие выводы о недостатках Secure Patches (SP):

- **Новизна** – при создании SP, и RSBAC, и Security-Enhanced Linux разработчики руководствовались уже устаревшими на сегодняшний день требованиями к защищенным ОС (например, RSBAC основывается на «Оранжевой книге»). Таким образом, эти SP не удовлетворяют всем требованиям к защищенности, которые существуют сегодня.
- **Сложность** – SP сильно усложнили архитектуру ОС, а также работу с ней. Например, защищенная ОС может заключить приложения в непроницаемые «скафандры», и в этом случае у системного администратора может возникнуть впечатление, что в приложении возник сбой, когда всего-навсего ему не предоставлено право данное приложение контролировать.
- **Совместимость версий** – неизвестно, будет ли работать SP при выходе следующей версии ядра ОС. Таким образом, для каждой версии ядра необходимо будет и обновлять SP.
- **Стоимость** – SP должны использовать лишь тогда, когда даваемые ими преимущества оправдывают затраты на обучение и время, которое приходится тратить на их сопровождение. Например, SP семейства PitBull, которые улучшают защиту Sun Solaris, IBM AIX и Linux, – стоят от 5 тыс. долл. за ОС, работающую на однопроцессорном Web-сервере, до 50 тыс. долл. за развертывание в рамках всей корпоративной информационной системы. При обновлении каждой версии ядра ОС, которые выходят примерно раз в полгода – это выльется в огромную сумму.
- **Проверка временем** – SP сложны в реализации и должно пройти довольно много времени, чтобы оно послужило доказательством работоспособности данного ПО.

Вышеуказанные недостатки SP позволяют сформулировать следующее: необходимо отказаться от практики «латания дыр» и начать строить новую ОС, изначально удовлетворяющую требованиям безопасности. Предлагается использовать руководящий документ, рассмотренный в (4) для создания изначально защищенной ОС. В качестве фундамента можно использовать ядро Linux – linux-2.4.18. Это перекликается с наметившимся сегодня интересом к достоверным (trusted) и защищенным (secure) операционным системам. Требования к безопасности должны быть определяющими в проектировании ОС, а не вводиться как вспомогательные службы.

Список литературы: 1. *2001* Computer Crime and Security Survey // Computer Security Institute, San Francisco, March 12, 2001; www.gocsi.com/prelea_000321.htm. 2. *Common* Criteria for Information Technology Security Evaluation (CCITSE) V2.1 // 1998. 3. *Станислав Иевлев.* Начала RSBAC. http://linux.ru.net/~inger/RSBAC-DOC-ru.html 4. Rule Set Based Access Control (RSBAC) для Linux – Модели // http://www.asmodeus.com.ua/library/os/linux/rsbac/models.html 5. *Simone Fischer-Hubner, Amon Ott.* From a Formal Privacy Model to its Implementation, http://www.rsbac.org/niss98.htm 6. *Security-Enhanced* Linux. http://www.nsa.gov/selinux/index.html 7. *Peter Loscocco, Stephen Smalley.* Integrating Flexible Support for Security Policies into the Linux Operating System. http://www.nsa.gov/selinux/slinux-abs.html 8. *Earl Boebert,* Some thoughts on the occasion of the NSA Linux release. http://www2.linuxjournal.com/articles/buzz/0043.html 9. *CSPP-OS – COTS* Security Protection Profile – Operating Systems // National Institutes of Standards and Technology, 2001. 10. *CSPP – Guidance* for COTS Security Protection Profiles // National Institutes of Standards and Technology, 1999.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 10.05.2003

УДК 681.3.06:519.248.681

Е. Г. КАЧКО, канд. техн. наук, С. Ю. МАРЧЕНКО, Ф. Г. ДЯГИЛЕВА

ИСПОЛЬЗОВАНИЕ ЯЗЫКА ASN.1 ПРИ ОПРЕДЕЛЕНИИ СТАНДАРТОВ ЦИФРОВЫХ ПОДПИСЕЙ ГОСТ Р 34.10-2001 И ДСТУ 4145-2002

В последние годы, наряду с описанием данных в стандартах на естественном языке, используется для этих целей язык ASN.1. Язык ASN.1 позволяет в формальной форме определить все данные стандарта, их взаимосвязи, что позволяет упростить процедуру перевода стандарта на алгоритмический язык и автоматизировать процедуру проверки соответствия стандарту. Примерами стандартов, для задания которых используется язык ASN.1, являются стандарты X500, X509, X9.62 и т.д. Описание стандартов цифровой подписи на эллиптических кривых, принятые в России (ГОСТ Р 34.10-2001)[2] и на Украине (ДСТУ 4145-2002)[1] не содержат определения данных на языке ASN.1. Данная статья посвящена описанию данных для указанных выше стандартов на языке ASN.1.

При построении описания Р 34.10-2001- и ДСТУ 4145-2002- стандартов будем использовать ASN.1 описание аналогичного стандарта X9.62 [3]. Сначала рассмотрим основные конструкции языка ASN.1[4], а потом использование конструкций языка для определения данных для указанных выше стандартов.

1 Стандартные обозначения и типы в ASN.1[1]

BIT STRING – строка битов. Если необходимо определить длину этой строки, она задается отдельно.

OCTET STRING – строка байтов. Если необходимо определить длину этой строки, она задается отдельно. При установке соответствия между строкой битов и строкой байтов соответствие устанавливается таким образом, чтобы старший бит первой строки соответствовал старшему биту старшего байта второй строки.

INTEGER – целое. В отличие от целых чисел для алгоритмических языков **INTEGER** в ASN.1 не ограничено длиной машинного слова и может иметь произвольную длину. Если эта длина должна быть фиксированной, она задается отдельно.

SEQUENCE – упорядоченный набор из одного или более типов.

SEQUENCE OF – упорядоченный набор из нуля или более типов.

OBJECT IDENTIFIER – последовательность целых компонентов, идентифицирующих объект.

CHOICE – объединение одной или более альтернатив.

2 Определение типа поля

В настоящее время наиболее широко используются два вида полей для эллиптических кривых – простое и расширенное. Стандарт Р 34.10-2001 использует эллиптические кривые для простых полей, ДСТУ 4145-2002 – для расширенных, в стандарте X9.62 разрешено использование обоих полей. В этом случае определение типа поля FieldID:

```
FieldID { FIELD-ID:IOSet } ::= SEQUENCE {
        fieldType FIELD-ID.&id({IOSet}),
        parameters      FIELD-ID.&Type({IOSet}{@fieldType})
}
FieldTypes FIELD-ID ::= {
        {Prime-p IDENTIFIED BY prime-field } |
        { Characteristic-two    IDENTIFIED BY characteristic-two-field },
...
}
```

[1] Здесь приведены только те стандартные типы, которые используются в данной работе

FIELD-ID ::= TYPE-IDENTIFIER

Запись означает, что идентификатор поля задается двумя компонентами: типом поля и его параметрами. Тип поля (**FieldTypes**) задает два возможных варианта – простое (**Prime-p**) и расширенное поле, для обозначения которого используется идентификатор **Characteristic-two.** Слово **two** в поле идентификатора обозначает, что в случае расширенного поля можно использовать полиномиальный и нормальный базисы. Символ **...** означает, что типы могут быть расширены. Знак | означает, что должен быть выбран один из типов.

В стандарте Р 34.10-2001 используется только простое поле, поэтому определение типа для этого стандарта имеет вид:

```
FieldTypes FIELD-ID ::= {
        { Prime-p IDENTIFIED BY  prime-field }
}
```

Для стандарта ДСТУ 4145-2002 имеем:

```
FieldTypes FIELD-ID ::= {
        { Characteristic-two      IDENTIFIED BY  characteristic-two-field },
}
```

Идентификатор объекта **id-fieldType** является корнем дерева, содержащим идентификаторы объекта каждого типа поля. Определим значения этого поля для Р 34.10-2001 и ДСТУ 4145-2002 соответственно:

id-fieldType OBJECT IDENTIFIER ::= { P34.10-2001 fieldType(1) }
id-fieldType OBJECT IDENTIFIER ::= { ДСТУ4145-2002 fieldType(1) }

и соответствующие типы полей:

```
prime-field OBJECT IDENTIFIER ::= { id-fieldType 1 }
characteristic-two-field OBJECT IDENTIFIER ::= { id-fieldType 1 }
```

Определим типы для **prime-field** и **characteristic-two-field:**

```
Prime-p ::= INTEGER
Characteristic-two ::= SEQUENCE {
        m       INTEGER,      -- Размер поля 2**m
        basis   CHARACTERISTIC-TWO.&id({BasisTypes}),
        parameters    CHARACTERISTIC-TWO.&Type({BasisTypes}{@basis})
}
```

Первый тип означает, что простое число должно быть целым числом, второй означает, что для задания расширенного поля необходимо определить размер поля m, который является целым числом, тип базиса (basis) и параметры неприводимых многочленов (parameters).

Для задания типа базиса в **X9.62** имеем:

```
BasisTypes CHARACTERISTIC-TWO::= {
        { NULL        IDENTIFIED BY     gnBasis } |
        { Trinomial   IDENTIFIED BY     tpBasis } |
        { Pentanomial IDENTIFIED BY     ppBasis },
...
}
```

Первое значение **NULL** означает, что параметры неприводимых многочленов не задаются для оптимального нормального базиса. В стандарте Р 34.10-2001 расширенное поле не используется, поэтому данного определения не будет. В стандарте ДСТУ 4145-2002 используются оба базиса, но не предусмотрено применение других базисов, поэтому символ ... в определении должен быть опущен и определение типа базиса для ДСТУ 4145-2002 имеет вид:

```
BasisTypes CHARACTERISTIC-TWO::= {
        { NULL        IDENTIFIED BY    gnBasis } |
        { Trinomial   IDENTIFIED BY    tpBasis } |
        { Pentanomial IDENTIFIED BY    ppBasis },
}
```

Для более точного задания неприводимых многочленов используются определения:

```
Trinomial ::= INTEGER      -- для трехчлена
Pentanomial ::= SEQUENCE {
        k1      INTEGER,
        k2      INTEGER,
        k3      INTEGER
}                                  -- для пятичлена
CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER
```

Идентификатор объекта *id-characteristic-two-basis* является корнем дерева, содержащего идентификаторы объекта для каждого типа базиса, и имеет значение:

```
id-characteristic-two-basis OBJECT IDENTIFIER ::= {
   characteristic-two-field basisType(2)
}
```

Идентификаторы объекта **gnBasis**, **tpBasis** и **ppBasis** – имена трех видов базисов для характеристики полей этого стандарта. Присвоим им значения:

gnBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 1 }
tpBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 2 }
ppBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 3 }

3 Задание элементов поля и точек эллиптической кривой

Элементом поля является строка байтов:

FieldElement ::= OCTET STRING

Для задания точки эллиптической кривой, по аналогии с типом POINT в алгоритмических языках, введен тип ECPoint, координаты точки задаются как **OCTET STRING,** т.е.

ECPoint ::= OCTET STRING.

4 Параметры эллиптической кривой

В рассматриваемых стандартах используются параметры, зависимые и не зависимые от типа поля. Зависимые параметры определяются идентификатором поля, который определен выше, независимыми являются эллиптическая кривая, которая задается своими коэффициентами *a, в*, базовая точка (*base*) и порядок (*order)* эллиптической кривой. Для задания параметров используем следующий синтаксис на языке ASN.1 для обох стандартов:

```
ECParameters ::= SEQUENCE {
        fieldID FieldID {{FieldTypes}},
        curve   Curve,
        base    ECPoint,
        order   INTEGER,
}
```

Тип FieldID определен выше, тип Curve определяется как:

```
  Curve ::= SEQUENCE {
        a       FieldElement,
        b       FieldElement,
  }
```

Для определения параметров можно использовать два способа:

1. Выбрать параметры из списка готовых эллиптических кривых.
2. Вычислить параметры эллиптической кривой, удовлетворяющей требованиям стандарта.

Возможность такого выбора задана следующим определением:

```
Parameters ::= CHOICE {
        ecParameters ECParameters,
        namedCurve   CURVES.&id({CurveNames}),
        implicitlyCA NULL
}
```

1–й вариант (ECParameters) предусматривает использование данной эллиптической кривой с конкретными параметрами. В стандарте Р 34.10-2001 приведена такая эллиптическая кривая в разделе «Контрольный пример». Использование этой кривой допустимо только для тестирования функций и строго не рекомендуется для практических целей. Стандарт не определяет, как выбираются параметры эллиптической кривой.

2-й вариант (namedCurve) предусматривает задание группы эллиптических кривых, определенных в самих стандартах. Такие группы приведены для ДСТУ 4145-2002 (приложение Г). Для этого варианта каждой кривой присваивается идентификатор. Примеры определения идентификаторов эллиптических кривых из ДСТУ 4145-2002:

```
CurveNames CURVES ::= {
    { ID c2pnb163 } |  -- Приложение Г. Кривая № 1. Полиномиальный базис. --
    { ID c2pnb167 } |  -- Приложение Г. Кривая № 2. Полиномиальный базис. --
    { ID c2pnb173 } |  -- Приложение Г. Кривая № 3. Полиномиальный базис. --
    { ID c2onb173 } |  --- Приложение Г. Кривая № 1. Нормальный базис. --
    { ID c2pnb179 } |  -- Приложение Г. Кривая № 4. Полиномиальный базис. --
    { ID c2onb179 } |  -- Приложение Г. Кривая № 2. Полиномиальный базис. --
    { ID c2tnb191 } |  --- Приложение Г. Кривая № 5. Полиномиальный базис. --

    { ID c2onb191 } |  -- Приложение Г. Кривая № 3. Полиномиальный базис. --
    { ID c2pnb233 } |  --- Приложение Г. Кривая № 6. Полиномиальный базис. --
    { ID c2onb233 } |  -- Приложение Г. Кривая № 4. Полиномиальный базис. --
    { ID c2tnb257 } |  --- Приложение Г. Кривая № 7. Полиномиальный базис. --
    { ID c2pnb307 } |  --- Приложение Г. Кривая № 8. Полиномиальный базис. --
    { ID c2tnb367 } |  --- Приложение Г. Кривая № 9. Полиномиальный базис. --
    { ID c2pnb431 } |  --- Приложение Г. Кривая № 10. Полиномиальный базис. --
    { ID c2onb431 } |  -- Приложение Г. Кривая № 5. Полиномиальный базис. --
}
```

При задании идентификатора учтены:
тип используемого поля (c2 соответствует расширенному полю);
размер поля в битах (числа 163, 167,...);
используемый полиномиальный базис (*pnb* для пятичлена и *tnb* для трехчлена);
используемый оптимальный нормальный базис (onb);

3-й вариант (implicitlyCA) предполагает, что параметры эллиптической кривой будут генерироваться какими-то функциями и не приводятся в самом стандарте. В ДСТУ 4145-2002 предусмотрена возможность получения параметров эллиптической кривой у уполномоченного исполнительного органа.

5 Ключи

В стандартах используется личный и открытый ключ.

Личный ключ – это случайное число d, удовлетворяющее неравенству 0<d<order для Р 34.10-2001; длина этого числа должна быть меньше длины порядка эллиптической кривой для ДСТУ 4145-2002.

Для определения личного ключа используем:

PrivateKey *::= INTEGER*

Открытый ключ является точкой эллиптической кривой. Задание открытого ключа с помощью синтаксиса **ASN.1**::

```
ECPublicKey ::= SEQUENCE {
        PublicKey      ECPoint
}
```

6 Синтаксис для ЦП

Для вычисления и проверки цифровой подписи используются параметры эллиптической кривой, личный и открытый ключи и значение функции хэширования. Рассмотрим особенности использования функций хэширования в стандартах.

В Р 34.10-2001 используется фиксированный алгоритм формирования функции хэширования ГОСТ Р 34.11. Для стандарта ДСТУ 4145-2002 используется полностью аналогичный стандарт ГОСТ 34.311 по умолчанию. Предусмотрена возможность задания другого алгоритма для хэширования, рекомендованного уполномоченным исполнительным органом государственной власти. Идентификатор функции хэширования относится к параметрам цифровой подписи.

Цифровая подпись – это конкатенация двух двоичных векторов $r||s$.

Таким образом, цифровая подпись может быть задана для Р 34.10-2001 так:

```
P34-10-Sig-Value ::= SEQUENCE {
        r       INTEGER,
        s       INTEGER
}
```

и для ДСТУ 4145-2002:

```
ДСТУ4145-Sig-Value ::= SEQUENCE {
        r       INTEGER,
        s       INTEGER
}
```

Что дает абстрактное определение данных для Стандарта?

1. Определение данных становится более наглядным.
2. Используя ASN.1 компиляцию, можно получить все структуры, определенные на языке ASN.1 на языке программирования C++, что существенно уменьшает время, требуемое для разработки реализации стандарта.
3. При проверке соответствия реализации стандарту можно выполнить формальное сравнение используемых в реализации структур и структур, определенных с помощью компилятора для ASN.1, а при несовпадении выполнить более детальный анализ различий, что существенно может уменьшить время, требуемое для проверки реализаций стандарта.
4. Сравнение языка ASN.1 с языками программирования, например C++, показывает, что ASN.1 соответствует начальной фазе языков программирования, в которой обеспечивалась возможность задания данных с различными сложными структурами (struct, union). В дальнейшем в языках программирования появилась возможность задания функций (функций-членов), наследования В этом случае язык определяют операции, которые можно выполнять для данных структур. С помощью наследования можно учесть необходимость использования внутренних стандартов, например, стандарта ГОСТ 28147-89 в качестве внутреннего для ГОСТ 34.311-95. Расширение языка ASN.1 в этом направлении позволит автоматизировать этап реализации стандартов

Список литературы: 1. *ДСТУ4145-2002.* Державний стандарт україни. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. Київ: Держстандарт України, 2003. 2. *ГОСТ34.10 – 2001.* Государственный стандарт российской федерации. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. М.: Госстандарт России. 3. *X9.62-1998.* American national standard. Public Key Cryptography For The Financial Services Industry. The Elliptic Curve Digital Signature Algorithm (ECDSA). 4. *A Layman's* Guide to a Subset of ASN.1, BER, and DER. Burton S. Kaliski Jr., RSA Data Security, Inc. Redwood City, CA, 1991.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 29.04.2003

УДК 681.3.06:519.248.681

Е. Г. КАЧКО, канд. техн. наук, С. С. БАТЮШКО

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ «ЦИФРОВЫХ ВОДЯНЫХ ЗНАКОВ» ДЛЯ ЗАЩИТЫ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

С развитием всемирной компьютерной сети Интернет и всеобщей доступностью вычислительных ресурсов все больше и больше информации, в том числе защищенной авторскими правами, хранится, воспроизводится и распространяется в электронном виде. Несмотря на такие преимущества как возможность неограниченного тиражирования, простой и быстрой доставки потребителю, данный метод имеет один, но весьма существенный недостаток – неэффективность традиционных средств защиты интеллектуальной собственности как технических, так и правовых.

По этой причине электронное пиратство и присвоение авторских прав приобрело просто угрожающие размеры. Особенно большие потери, связанные с подобного вида нарушениями, несут аудио- и видеозаписывающие студии, т. е. организации, занимающиеся распространением графической, аудио- и видеопродукции, записанной в цифровом формате. Несанкционированное тиражирование с последующей продажей нелицензионных аудио- и видеокомпакт-дисков стало во многих странах одним из наиболее прибыльных видов бизнеса, приносящего многомиллиардные доходы пиратам в сфере электронных технологий. Большое распространение такой незаконной деятельности характерно для стран, в которых цивилизованные законы рынка еще не занимают господствующего положения.

Сегодня на первый план выходит проблема разработки программных систем, способных обеспечивать надежную защиту авторских прав. Одним из перспективных путей решения этой проблемы может послужить использование «цифровых водяных знаков» (Digital watermarking), которые позволяют автоматически обнаруживать авторство произведенного программного продукта и достаточно эффективно осуществлять защиту авторских прав.

С помощью «цифровых водяных знаков» осуществляется встраивание специфического электронного знака в исходное изображение или аудиозапись для идентификации владельца авторских прав на данную информацию. Вместе с данными об авторе в качестве знака могут быть встроены также данные о конкретном экземпляре, такие как серийный номер, имя владельца и т. п.

В целом сама технология «цифровых водяных знаков» делится на две большие области. Первая – это собственно «цифровые водяные знаки», которые используются для скрытия информации об авторских правах. Вторая – это встраивание в производимый аудио-, видеопродукт «цифровых отпечатков пальцев», которые используются для скрытия серийных номеров или иной информации, позволяющей отличить копии носителя одну от другой.

В основу принципа действия «цифровых водяных знаков» положен тот факт, что в поток исходных данных, записанных в цифровом формате (изображение или звук), можно внести некоторые искажения (изменения), несущие дополнительную информацию, практически неразличимые человеческими органами зрения или слуха и в силу этого не снижающие потребительские качества исходного сигнала.

Несмотря на то, что технология «цифровых водяных знаков» имеет сравнительно короткую историю, на сегодняшний день можно утверждать, что уже теоретически разработаны и проходят соответствующую апробацию десятки, если не сотни, алгоритмов создания соответствующих компьютерных программ данного типа [1 – 2]. Их анализ позволяет рассмотреть некоторые принципы и свойства, которые являются общими для всех. Рассмотрим эти свойства.

- Невидимость: добавление электронного «водяного знака» не должно ухудшить исходный сигнал. Такой знак не должен быть заметен человеческим глазом. Это свойство «водяных знаков» конфликтует со следующими двумя.
- Устойчивость: «водяной знак» должен сопротивляться манипуляциям, которые могут возникнуть при использовании продукта, таких как фильтрация, сканирование и печать, преобразование в другие форматы.
- Защита: «водяной знак» должен сопротивляться попыткам его удаления. Это не абсолютное требование, скорее оно привязано к уровню ухудшения несущего сигнала. Грубая атака, разрушающая исходный сигнал, также может разрушить и «водяной знак».
- Публичность: используемый алгоритм «водяных знаков» должен являться открытым. Как и в криптографии, «защищенность через скрытность» не является верной концепцией. Сохранение метода «водяных знаков» в секрете приводит к тому, что он лишается постороннего анализа, тем самым становясь потенциально менее защищенным.
- Многократность водяных знаков: должна существовать возможность внесения в исходный сигнал нескольких водяных знаков одновременно.
- Масштабируемость: должна существовать возможность использования более новых, улучшенных версий той же технологии при доступности более мощных вычислительных средств. Это означает, например, использование больших по длине криптографических ключей. И в то же время система должна быть устойчивой при более мощных вычислительных ресурсах.
- Самовосстановление: если только некоторый фрагмент несущего сигнала доступен, например, после усечения или поворота изображения, должна существовать возможность восстановления «водяного знака».
- Возможность использования совместно со сжатым битовым потоком: эта возможность особенно полезна для программ реального масштаба времени.
- Сопротивление «столкновению» (усреднению): если несколько изображений, помеченных различными «водяными знаками», усредняются, результат тоже должен быть помеченным. Эта возможность нужна в двух ситуациях: а) при подписи, где одно и то же изображение помечается по-разному для различных заказчиков и б) для пометки видео, где несколько похожих кадров могут быть усреднены.

Следует отметить, что для использования электронных «водяных цифровых знаков» в видеопродукции существуют также и общие дополнительные требования. Среди них наиболее важными являются:

- Возможность добавления «водяных знаков» в реальном режиме.
- Не увеличивать поток передаваемых данных (полосу пропускания).

Поскольку электронный «цифровой водяной знак» должен быть невидимым, устойчивым к различным типичным операциям над изображениями, таким как применение алгоритмов сжатия, фильтрациии изображения (сглаживание краев, изменение контраста и т. п.) и геометрических трансформаций (масштабирование и т. п.), то «цифровой водяной знак» должен храниться не в формате файла, а непосредственно в самом изображении.

Еще одной из областей применения «цифровых водяных знаков» является проверка целостности изображений, то есть выяснение того, что в исходное изображение не внесены какие-либо изменения. Это стало особенно актуально сейчас, когда уровень программного обеспечения достиг такого рубежа, что уже практически невозможно отличить оригинал от подделки.

Использование «цифровых водяных знаков» находит применение и в стеганографии (от греческого – тайнопись). С помощью стеганографических методов появляется возможность невидимого встраивания некоторого количества данных в исходный сигнал. Это дает возможность обмениваться шифрованными сообщениями без привлечения внимания третьей стороны [3]. Как правило, в применяемых в стеганографии алгоритмах большее внимание уделяется объему скрываемых данных и их незаметности, чем устойчивости к повреждениям. В этом своем качестве стеганография весьма близка к криптографии. Вместе с тем, между стеганографией и криптографией имеются как сходства, так и существенные различия.

Общеизвестно, что одной из областей применения криптографии является шифрование данных для передачи через коммуникационные каналы с целью скрытия их содержимого. Сам факт передачи зашифрованного сообщения не скрывается. Стеганография же предполагает имплантацию секретного сообщения в какую-либо форму несущих сигналов, обычно в изображение или видеопоток, объективно определяемую как наличие шума или помех. Без правильного ключа практически невозможно не только извлечь скрытое сообщение, но даже определить его присутствие. Стеганографические сообщения, как правило, шифруются для увеличения безопасности, надежности в передаче данных. При этом возможно комплексное сочетание принципов криптографии и стеганографии. В этом случае информация вначале надежно шифруется, а затем еще и дополнительно прячется. В имеющихся на эту тему единичных открытых публикациях отмечается, что для того, чтобы спрятать подобные секретные сообщения в графических файлах, в большинстве случаев используются торальные автоморфизмы [4 – 5] или потоки Колмогорова [6].

Важным сходством между криптографией и «цифровыми водяными знаками» является использование симметричных и несимметричных шифровальных систем.

Технология встраивания «водяных знаков» может быть основана на использовании так называемых пространств преобразования. В этом случае к исходному изображению применяют некоторые обратимые операции перед встраиванием самого знака. После этого изображение встраивается (изменяются некоторые коэффициенты преобразования), и эти действия проделываются вновь для получения «меченого» изображения. Трансформации, которые обычно используются для этого, могут быть следующими: дискретное косинусное преобразование, дискретное преобразование Фурье, фрактальное преобразование, дискретное wavelet-преобразование. Реже применяются преобразования Френеля, комплексное wavelet-преобразование, преобразование Фурье-Меллина.

Использование различных пространств преобразования дает определенные преимущества: поскольку «водяной знак», встроенный в пространстве преобразования, распределен в изображении иррационально, то его, следовательно, сложно выделить или изменить. Более того, частотное изменение изображения позволяет выбрать только определенные (наилучшие) участки исходного изображения для внесения «водяных знаков».

Рассматриваемая технология встраивания электронных «цифровых водяных знаков», как представляется, является весьма эффективной и том смысле, что не только надежно прячет информацию, но и защищает ее от несанкционированного взлома и деформаций.

Во-первых, обеспечивается необходимая в таких случаях противозашумленность конфиденциальной информации. Это свойство вытекает из того факта, что атакующий не знает привилегированной информации, которой обладают отправитель и получатель. Как результат, атакующий должен зашумить весь спектр широкополосного сигнала. Но зашумление имеет ограниченную возможность, поэтому атакующий может зашумить каждую частоту с малой силой, в то время как отправитель и получатель имеют значительное преимущество по соотношению сигнал-шум. В применении к встраиванию «водяных знаков» это означает, что для того, чтобы разрушить «водяной знак», нужно в изображение внести такие помехи, что оно становится непригодным для дальнейшего использования.

Во-вторых, это весьма малая вероятность перехвата. Это свойство основано на том, что большой сигнал распределяется по всему частотному спектру, поэтому только ничтожно малые изменения добавляются к каждой частоте. Часто такое приращение меньше уровня помех, поэтому атакующий не сможет даже определить наличие сигнала. Это позволяет «водяным знакам» быть более защищенными [7].

Одним из перспективных направлений развития стеганографических технологий является использование в процессе встраивания «цифровых водяных знаков» так называемых псевдошумов. В этом случае в качестве несущего берется такой сигнал, для которого статистические свойства как можно более близки к свойствам истинно случайного сигнала, объективно воспринимаемого как шум. При этом истинный сигнал может быть в точности воспроизведен, если известны некоторые привилегированные (секретные) параметры. К примеру, в качестве несущей может быть использован выход генератора случайных чисел, который был инициализирован с помощью некоторого начального числа, известного только владельцу. В этом смысле использование псевдошумов оказывается очень полезным, поскольку дополнительно затрудняет для атакующего выделение «цифрового водяного знака», а следовательно, и конфиденциальной информации из исходного изображения.

Использование стеганографических методов предполагает глубокое знание свойств человеческих органов зрения и слуха и, прежде всего, порога их чувствительности к изменениям изображения или звука. Если говорить коротко, то «цифровые водяные знаки» в этом смысле должны обеспечивать:

- контрастную маскировку – т.е. невозможность обнаружения человеческими органами зрения одного сигнала в присутствии другого сигнала;
- частотную маскировку – т.е нечувствительность человеческого глаза к изменившимся волновым решеткам на разных частотах;
- световую маскировку – т.е. не преодолевать порога чувствительности к световым изменениям на контрастном фоне;
- непреодоление порога ощущения различий – т.е порога, за которым любые изменения соответствующего коэффициента кажутся практически неразличимыми [8].

Таковы лишь некоторые особенности использования стеганографических методов, которые могут быть применены для защиты интеллектуальной собственности, а также найти свое применение при передаче конфиденциальной информации в электронном виде. Стремление ведущих компаний, работающих в сфере аудио- и видеобизнеса, обезопасить себя от электронного пиратства делает весьма актуальным разработки программных продуктов, решающих эту важную задачу.

Список литературы: 1. *Walter Bender, Daniel Gruhl, Norishige Morimoto, and A. Lu.* Techniques for data hiding. IBM Systems Journal, 1996. 35 (3-4). Pp. 313 – 336. 2. *Ingemar J. Cox and Matt L. Miller.* A review of watermarking and the importance of perceptual modeling. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, SPIE Human Vision and Electronic Imaging II. February 1997. Vol. 3016, Pp. 92 – 99. 3. *Zenon Hrytskiv, Sviatoslav Voloshynovskiy, and Y. B. Rytsar.* Cryptography and steganography of video information in modern communications. In Proceedings of the 3rd International Conference on Telecommunications in Modern Satelite, Cable and Broadcasting Services TELSIKS '97, Vol. 1, Pp 164 – 167. 4. *George Voyatzis and Ioannis Pitas.* Application of toral automorphisms in image watermarking. In Proceedings of the IEEE International Conference on Image Processing, ICIP '96. Vol. 2, Pp. 237 – 240. 5. *George Voyatzis and Ioannis Pitas.* Digital image watermarking using mixing systems. Computer & Graphics, August. 1998. 22(4). Pp. 405 – 416 6. *Josef Scharinger.* Robust watermark generation for multimedia copyright protection. In Markus Vincze, editor, Robust Vision for Industrial Applications. 1999. Pp. 127 – 136. 7. *Зюко А.Г., Кловский Д.Д., Назаров М.В., Финк Л.М.* Теория передачи сигналов. М.: Радио и связь, 1988. 8. *Raymond B. Wolfgang, Christine I. Podilchuk, and Edward J. Delp.* Perceptual watermarks. for digital images and video. Proceedings of the IEEE, Special Issue on Identication and Protection of Multimedia Information, July 1999. 87(7). Pp. 1108 – 1126.

Харьковский национальный

университет радиоэлектроники

Поступила в редколлегию 29.04.2003

УДК 681.3.06: 519.248.681

В. А. ГОРБАЧЕВ, канд. техн. наук, В. В. СТЕПАНЕНКО

СЕРТИФИКАЦИЯ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ КОМПЬЮТЕРНЫХ СИСТЕМ

Рассматривается методика сертификации периферийных устройств современного компьютера с целью выявления неспецифицированных функций.

Анализ структуры современного ПК показывает, что наиболее уязвимыми, с точки зрения внедрения аппаратных закладок (АЗ), являются контроллеры ввода/вывода. Это связано с тем, что именно через них проходит большое количество информации, имеющей конфиденциальный характер. Кроме того, объёмы и скорость обмена информации в данных устройствах сравнительно невелики, а ущерб, наносимый при несанкционированном доступе к ней, может оказаться весьма ощутимым. Структура современного ПК такова, что практически все контроллеры ввода/вывода сосредоточены в одной интегральной схеме (ИС), имеющей название «SUPER I/O». Данная ИС включает в себя: контроллер клавиатуры, контроллер FDD, Secure Digital (SD) Memory card Interface, Extended Hardware Monitor, два контроллера UART с поддержкой IRDA и Smart Card Reader protocols, IEEE 1284 Parallel Port. Таким образом, данная ИС имеет доступ практически ко всей информации, вводимой и выводимой на ПК (исключение составляют контроллеры дисплея, LAN и USB) и, следовательно, необходимо уделять особое внимание её функционированию.

Таким образом, из приведённых выше рассуждений следует, что угроза безопасности информации со стороны АЗ – реальна. На современном уровне технологий возможно создание АЗ интегрированных в структуру штатных ИС, способных реализовать все виды угроз безопасности информации. Для эффективного противодействия аппаратным закладкам необходимо осуществлять сертификацию аппаратных средств, используемых в системах с высоким уровнем безопасности. Кроме того, поскольку уровень интеграции и технологий, используемых в современных вычислительных системах, очень высок, а технические возможности сертификационного оборудования на сегодняшний день не всегда соответствуют такому уровню технологии, то необходимо разрабатывать методики построения вычислительных систем, архитектура которых не позволит эффективно функционировать АЗ и сведёт к минимуму угрозу безопасности информации от данного класса устройств.

Рассмотрим методику сертификации клавиатуры персонального компьютера. Клавиатура выбрана не случайно, и связано это с тем, что через неё вводится большое количество информации, в том числе и различные пароли. Сертификация любого электронного устройства должна осуществляться путём прохождения следующих этапов сертификации:

- Построение поведенческой модели устройства.
- Синтез тестов для проверки специфицированных функций.
- Выполнение функционального тестирования для проверки специфицированных функций.
- Выбор модели АЗ.
- Синтез тестов для проверки наличия неспецифицированных функций.
- Выполнение функционального тестирования для проверки неспецифицированных функций.
- Выводы о соответствии устройства его спецификации.

Таким образом, на первом этапе нам необходимо построить поведенческую модель сертифицируемого устройства, а именно клавиатуры. Поведенческая модель в нашем случае представляет собой алгоритм функционирования двух интерфейсов клавиатуры:

1. Интерфейс опроса матрицы клавиш.
2. Интерфейс передачи данных.

Укрупнённый алгоритм функционирования клавиатуры показан на рис. 1.

Рис.1

Для осуществления сертификации нам необходим некоторый аппаратно-программный комплекс, который позволит осуществлять генерацию управляющих векторов для обоих интерфейсов клавиатуры с постоянным контролем текущих состояний. Обобщённая схема сертификационного комплекса показана на рис. 2.

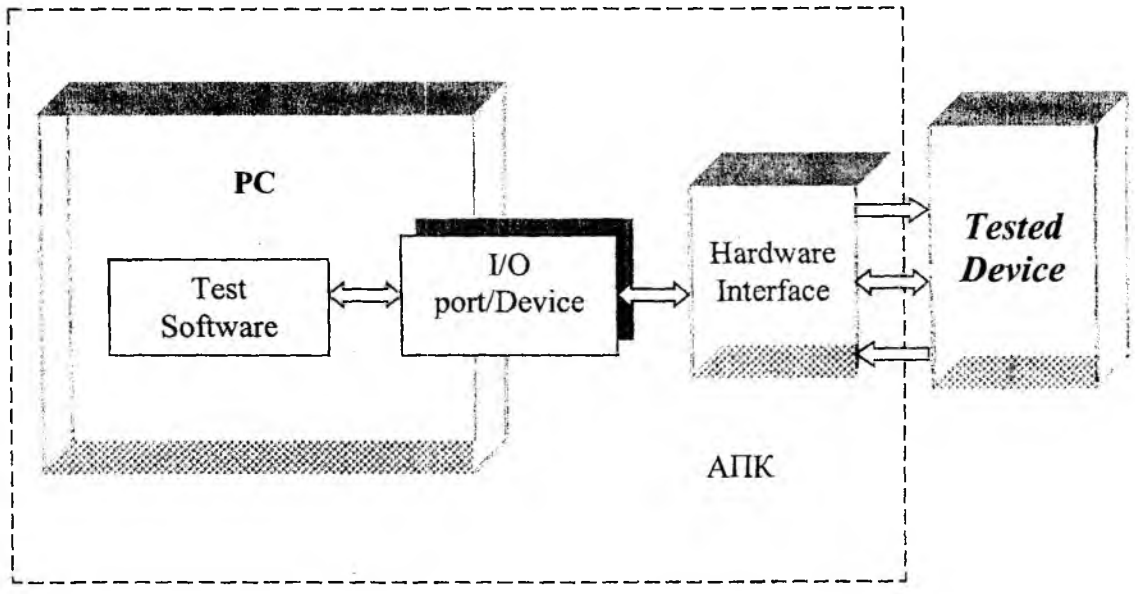


Рис. 2

Физически аппаратная часть сертификационного комплекса для данного случая выполнена в виде двух микроконтроллеров. Первый осуществляет эмуляцию интерфейса обмена данными с персональным компьютером, второй представляет собой управляемую матрицу клавиш.

Переходя ко второму этапу (генерация тестов для проверки специфицированных функций), следует отметить, что поскольку осуществляется сертификация двух интерфейсов, мы будем иметь две группы тестов (по числу интерфейсов). Первая группа тестов сканирует интерфейс обмена данными и проверяет на полном наборе специфицированных функций правильность работы устройства путём сравнения получаемых ответов с ответами, специфицированными в технической документации. Вторая группа тестов проверяет правильность генерации кодов нажатия и отжатия для всех клавиш клавиатуры по отдельности, а также для всех сочетаний в количестве до трёх. Верификация так же, как и в первом случае, осуществляется путём сравнения получаемых значений со значениями из таблицы истинности.

Далее необходимо выполнить функциональное тестирование клавиатуры. Данная операция выполняется при помощи управляющей программы, написанной с использованием знаний о функционировании клавиатуры и рассуждений, приведённых выше. Работа сертификационной программы в обоих режимах показан а на рис. 3.

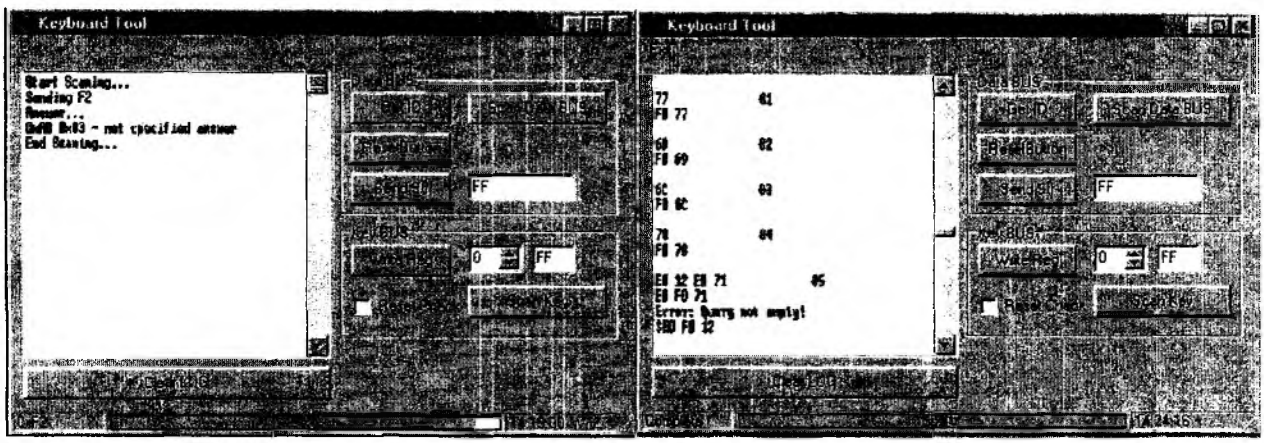


Рис. 3

Переходя к поиску неспецифицированных функций, прежде всего нам необходимо определить модель АЗ, поиск которой мы будем осуществлять. Прежде всего, это необходимо для определения реакции сертифицируемого устройства в нештатном режиме. В данном случае мы выбираем АЗ накопительного типа. Выбор связан с тем, что этот класс АЗ, располагаясь в данном устройстве, может нанести максимальный ущерб безопасности информации.

Генерация тестовых последовательностей для проверки наличия неспецифицированных функций осуществляется аналогично предыдущему случаю, за исключением двух моментов:

1. При сертификации интерфейса обмена данными используются неспецифицированные команды.
2. При сертификации интерфейса опроса матрицы клавиш рассматриваются различные сочетания нажатия и отжатия клавиш в количестве от 4-х до 10-ти.

Искомой реакцией в данном случае является несанкционированная передача данных через интерфейс обмена данными. Этап функционального тестирования в данном случае осуществляется при помощи тех же программных и аппаратных средств, что и в случае проверки специфицированных функций.

Оценка эффективности алгоритма и времени проведения сертификации на примере поиска АЗ накопительного типа [1] в контроллере клавиатуры показывает, что данный подход

является приемлемым. Предлагаемый подход позволяет выполнить сертификацию устройства, как на выполнение им специфицированных, так и неспецифицированных функций. При этом в обоих случаях сертификация проводится с использованием одинаковых алгоритмических, программных и аппаратных средств, что позволяет снизить стоимость сертификационного оборудования. Время поиска неспецифицированных функций зависит от быстродействия специфицируемого устройства и может быть уменьшено путём оптимизации алгоритма построения тестовых последовательностей.

Список литературы: 1. *Горбачев В.А., Степаненко В.В, Саранча С.Н.* Сертификация сложных электронных систем с использованием функциональной модели объекта на полном наборе входных слов // Наук.-техн. зб. Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. Київ: КНУ-КПІ, 2002. Вип. 5. С. 139 – 144.

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 20.05.2003

УДК 621.39

В. І. ЗАБОЛОТНИЙ, канд. техн. наук

КЛАСИФІКАЦІЯ ТЕХНІЧНИХ КАНАЛІВ ВИТОКУ ІНФОРМАЦІЇ

Організація системи захисту об'єкта від витоку інформації, як передбачено державними стандартами України у галузі технічного захисту інформації (ТЗІ) [1 – 3], потребує складати окрему модель загроз. Формалізований опис технічних каналів витоку інформації (ТКВІ) є суттєвою складовою окремої моделі загроз і основою для розробки необхідних заходів захисту. Класифікація ТКВІ являє собою якісну модель сукупності матеріального об'єкту, що містить інформацію з обмеженим доступом (ІзОД), середовища її розповсюдження та засобів технічних розвідок [3]. Декомпозиція ТКВІ дає змогу зробити адекватний кількісний опис кожної з окремих складових каналу, провести їх дослідження як без заходів ТЗІ, так і з певним набором останніх. Наведене далі відкриває шлях до оцінки ефективності захисту від витоку інформації технічними каналами.

1 Форми існування даних, що підлягають захисту

Аналіз змісту [1], а також першої редакції Положення про ТЗІ в Україні [4], дозволяє зробити висновки щодо існування двох взаємопов'язаних форм проявлення даних, що підлягають захисту від технічних розвідок: знакову та предметну. Знакова форма являє сукупність символів, літер, цифр, звуків, які відображають предмети та явища реального світу у віртуальному світі. Носіями ІзОД є документи на папері, магнітна, кіно-, відео-, фотоплівка, інші носії [5]. Також ІзОД може зберігатися, відображатися або передаватися у вигляді інформаційних сигналів [3] у формі фізичних полів (електромагнітних, оптичних, акустичних), електричних сигналів, вібраційних коливань у твердих предметах. Предметна форма існування даних проявляється самими матеріальними об'єктами реального світу у процесі виробництва й застосування продукції різного призначення [1]. Це електромагнітні, оптичні, гравітаційні, акустичні та інші поля й випромінювання, хімічні речовини.

Співвідношення між згаданими формами існування даних може бути проілюстровано (рис. 1) наступними міркуваннями.

Предмети та явища реального світу: продукція підприємств, вихідні комплектуючі та речовини, виробничі технології їх створення, способи застосування продукції у сучасному суспільстві, породжуються завдяки розробці й використанню комплектів документів, обговоренню цього безпосередньо вголос та у засобах зв'язку. При цьому віртуальний світ відображає реальний завдяки природним мовам (різних народів): вголос та письмово і штучним – кресленнями, кодами та символами, електричними сигналами та полями.

У визначенні первинності знакової і предметної форми перша частіше передує, бо у сучасному виробництві спочатку розробляють документацію, а вже потім створюють продукцію чи технологію. Фактичні співвідношення між об'єктами віртуального та реального світів наступні. Віртуальний світ (світ моделей) бідний за властивостями предметів, їх ознаками. Навпаки у реальному світі кількість властивостей предметів якщо не безмежна, то значно більша ніж у віртуальному світі. Ознаки, властивості предметів, явищ можуть проявлятися несподівано, і у небезпечному співвідношенні, не завжди можуть бути досліджені експериментально. У віртуальному світі з моделями можна проводити любі, досить ризиковані досліди.

Конкурентна боротьба вимагає добувати відомості як віртуального світу, так і реального: розвідувати ІзОД і дані предметної форми існування. Надалі не розглядаються такі шляхи здобуття даних конкурентами, як несанкціоноване придбання або викрадення документів, зразків продукції тощо. Навпаки, приділяється увага аналізу механізмів добування відомостей за рахунок використання різноманітних засобів технічних розвідок.

І останнє зауваження. Знакова форма існування ІзОД являє собою достатньо однорідні масиви даних про реальні предмети та явища світу. Структура знаків, з яких складається інформація, практично не залежить від їх змістовного навантаження. Алфавіт окремих знаків, що несуть інформацію, як правило, обмежений. Технічна апаратура розвідки може бути лише одного принципу дії. Добування інформації можливо з однієї точки. Дані щодо об'єкту розвідки можуть бути одержані задовго до виготовлення першого зразка продукції.

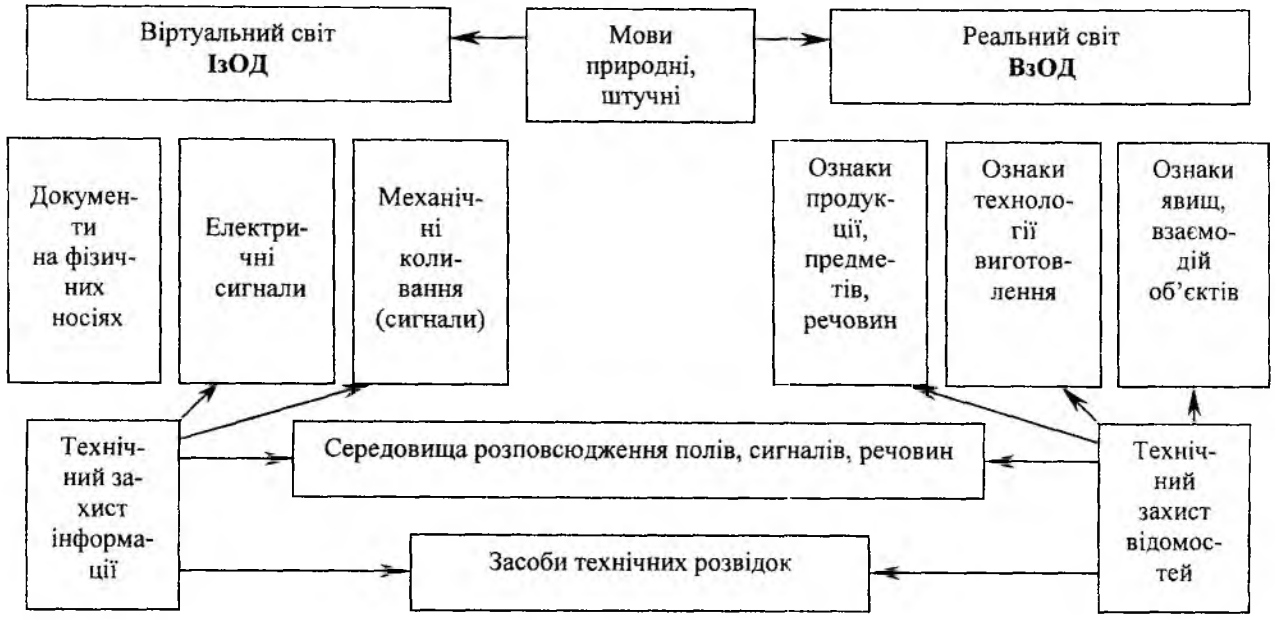


Рис. 1

Предметна форма проявлення та існування даних у наборі їх ознак потребує, як правило, різноманітної за фізикою дії апаратури розвідки, розташування такої апаратури може вимагати декількох пунктів навколо об'єктів. Дані про об'єкт можуть бути одержані лише після виготовлення або застосування першого зразка продукції речовини, нової технології.

Є потреба (хоча б у рамках даної роботи) ввести термін, що дозволить відрізняти предметну форму виразу даних від знакової. Це може бути слово «відомості». Відповідно будуть мати місце терміни «відомості з обмеженим доступом», «технічний захист відомостей» та словоскорочення: ВзОД, ТЗВ.

2 Класифікація ТКВІ знакової форми проявлення

Класифікація ТКВІ дозволяє аналізувати та детально вивчати можливі канали витоку інформації для розробки заходів захисту інформації. На сьогодні дослідження щодо створення повної класифікації ТКВІ ще не завершені. Тому у якості подальшої розробки класифікації можна запропонувати систематизувати ТКВІ за наступними критеріями (рис.2).

За *структурою* сигнали ІзОД, що циркулюють у технічних засобах передачі, обробки, зберігання, відображення інформації (ТЗПІ), поділяються на аналогові та дискретні (цифрові). Аналогові сигнали характерні аудіо- та відео- апаратурі, іншим системам і засобам, які встановлені у приміщеннях, де обговорюються питання обмеженого доступу. Дискретні сигнали характерні засобам обчислювальної техніки, цифровим факсимільним апаратам.

Наявність повторення сигналів, що циркулюють у ТЗПІ, грає суттєву роль у можливості їх розвідки. До сигналів одноразового існування можна віднести розмову, а до тих, що багато разів повторюються – електромагнітні випромінювання моніторів ПЕОМ.

За співвідношенням спектрів вихідного сигналу джерела ІзОД і того, що присутній у ТКВІ, їх можна поділити на співпадаючі, позасмугові, на гармоніках, паразитні та задані штучно. У ТКВІ зі співпадаючим спектром спектр частот, в основному, співпадає зі спектром сигналів, які циркулюють в ТЗПІ. У ТКВІ з позасмуговим спектром спектр частот такого ка-

налу наближений до спектру частот сигналу ІзОД і створюється за рахунок розширених понад заданих значень смуг частот підсилювачів, високої крутизни фронтів імпульсів.

Як правило, частоти позасмугових спектрів лежать вище частот основних спектрів сигналів. Природа ТКВІ на гармоніках обумовлена нелінійностями амплітудних характеристик

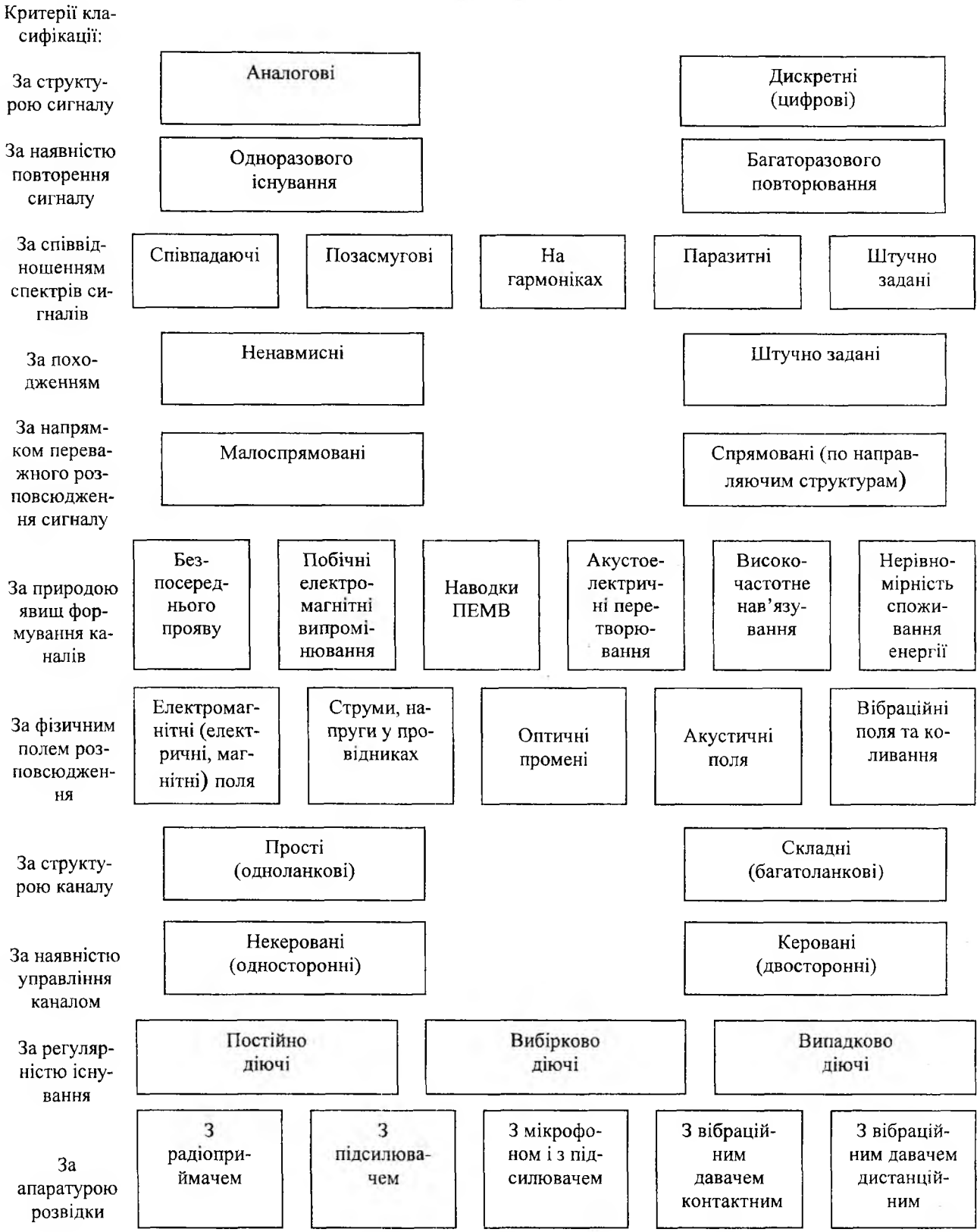


Рис. 2

підсилювачів. ТКВІ на паразитних частотах породжуються за рахунок самозбудження підсилювачів із-за паразитних зворотних зв'язків між каскадами підсилювачів. Значення їх частот обумовлені випадковими причинами виконання умов балансу фаз і амплітуд. Реактивні елементи схем, ємність p-n переходів, розподілені значення індуктивностей, ємностей монтажних проводів, тощо приводять до паразитних збуджень. Задані штучно спектри сигналів ТКВІ відносяться у цій класифікації до підкладних пристроїв і випадків зовнішнього впливу високочастотним сигналом на ТЗПІ. Наведені категорії за змістом подібні до однойменних назв введених в галузі електромагнітної сумісності [6].

За походженням ТКВІ можуть поділятися на ненавмисні та штучні. Ненавмисні ТКВІ породжуються конструктивними особливостями або недоліками апаратури. А навмисні – спеціально створюються зацікавленими в ІзОД особами.

За напрямком переважного розповсюдження ТКВІ підрозділяються на малоспрямовані та ті, що мають певну спрямованість. У малоспрямованих ТКВІ енергія електромагнітних та акустичних полів розповсюджується переважно вільно у всі сторони без особливих переваг. Направляючі структури і проводи, хвильоводи, лазерні випромінювання концентрують суттєву частину енергії ТКВІ у відповідному напрямку.

За природою явищ формування каналів ТКВІ підрозділяються на такі класи. ТКВІ безпосереднього прояву формуються, наприклад, акустичним полем розмови, яка проходить через звукоізолюючі перепони і потрапляє на спрямований мікрофон. Побічні електромагнітні випромінювання (ПЕМВ) формуються змінним електричним струмом сигналів ІзОД. Навід на провідники, що проходять навколо ТЗПІ, дають ПЕМВ. Ефектом акустоелектричних перетворювань володіють ряд приладів та апаратура: дзвоники, гучномовці тощо. Останнім часом все більше застосовується зовнішній вплив на ТЗПІ високочастотними випромінюваннями або напругами, які породжують коливання, модульовані сигналами ІзОД. Рівень споживання підсилювачем енергії залежить від сигналу, що підсилюється, а це може бути шляхом витоку ІзОД.

За фізичним полем ТКВІ поділяються на електромагнітні (електричні, магнітні), акустичні поля, вібрації, оптичні випромінювання, струми та напруги.

За структурою ТКВІ підрозділяються на прості (одноланкові) та складні (багатоланкові). Простий канал являє собою «джерело – середовище – приймач сигналу», а у складному каналі сигнал, перш ніж попаде у приймач, два чи більше разів перетворюється у фізичних носіях. Наприклад, ПЕМВ-джерела перетворюються у провідниках, розташованих поблизу ТЗПІ, у електричний струм, який цими провідниками поширюється за межі контрольованої території, де потрапляє до апаратури розвідки.

За наявністю управління ТКВІ можуть бути некерованими (односторонніми) або керованими (двосторонніми). До останніх належать керовані підкладні пристрої, лазерна апаратура зйому мовної інформації з шибок вікна, що коливаються під змінним звуковим полем.

За регулярністю існування ТКВІ підрозділяються на постійні, вибірково діючі та випадкові. Перші діють досить тривалий час, практично не змінюючи свої характеристики. Вибірково діючі ТКВІ функціонують на «замовлення» оператора. Випадкові ТКВІ суттєво змінюють свої характеристики передачі інформації у залежності від випадкових факторів, які можуть впливати на джерело або середовище розповсюдження сигналу. До випадкових ТКВІ можна віднести підсилювач звукової частоти, який самозбуджується та у якому сила самозбудження залежить від рівня сигналу на вході, напруги джерела живлення, переміщення проводів у просторі тощо.

За апаратурою розвідки ТКВІ підрозділяються на ті, що використовують апаратуру радіоприйому, підсилювачі, мікрофони, вібродавачи контактної та неконтактної дії.

Наведена класифікація ТКВІ може і надалі розширюватися в залежності від потрібного ступеню деталізації для розробки заходів ТЗІ.

3 Класифікація даних предметної форми виразу та їх ознак

На відміну від прояви знакової форми даних ВзОД та ознаки відомостей (ОВ) мають більш широкий спектр фізичного прояву, що потребує відповідної аналітичної роботи щодо їх вивчення з метою проведення подальшої класифікації. Дану роботу доцільно виконувати експертною групою із залученням фахівців, які володіють усіма сторонами розробки та застосування об'єктів дослідження.

Отже, предметні дані *за визначеністю шкал виміру* можуть бути кількісними та якісними, що ілюстровано на рис. 3. Кількісні відомості відображаються дійсними числами, які належать до відповідних шкал виміру. Це можуть бути шкали швидкості, далекості, кількості продукції, величини пропускної спроможності підприємства та ін. Для подальшого аналізу треба мати уяву про діапазон можливих значень показників і точності їх задання. Наприклад, діапазон робочих частот радіостанції – 20...60 МГц, точність настройки – 1 кГц. Якісні відомості виражають відношення об'єкту або явища до визначеного класу за сукупністю характеристик. Наприклад, тип легкового автомобіля, що розробляє фірма: масового застосування, представницький, спортивний. Формально розрізнити якісні відомості інколи буває досить важко. В такому випадку потрібно скласти таблицю за всіма можливими показниками:

потужність двигуна,
тип кузова,
типові палітри кольорів,
кількість посадочних місць,
максимальна швидкість,
час набору певної швидкості,
економічність,
обладнання салону,
розміри багажника,
габарити кузова тощо.

Далі за сукупністю співпадаючих параметрів віднести оцінюваний зразок до потрібної групи.

За проявою у повсякденній ситуації ВзОД підрозділяють на відомості безпосереднього проявлення і на відомості такі, що не проявляються. До першої групи відноситься більшість відомостей, а до другої – такі, наприклад, як стійкість автомобіля при лобовому зіткненні.

За співвідношенням часу існування і терміну часу захисту ВзОД можуть бути довготривалого або термінового часу захисту. ВзОД довготривалого часу захисту – це найчастіше за все нові технології, на розробку яких підприємство затратило великі кошти, і такі технології дають суттєву перевагу перед конкурентами. Довготривалі ВзОД потрібно захищати весь час їх існування. До термінових відомостей відносять такі, що треба захищати певний час, до якогось строку, наприклад, до початку широкого продажу продукції.

Перед проведенням аналізу ВзОД слід провести декомпозицію їх описів, привести до сукупності елементарних висловів, які б за змістом повністю перекривали зміст загального вислову. Простіше за все перевести вислів у групу висловів простих речень. Для кожного з елементарних ВзОД встановити шкали виміру, місце та час захисту.

Технічні розвідки можуть встановлювати значення ВзОД через ознаки, що супроводжують відомості. Поширений у військовій сфері термін «демаскуючі ознаки» не у повній мірі відповідає суті у напрямку захисту ВзОД, оскільки у виробничій сфері йде мова не про маскування матеріального об'єкта, а про захист його характеристик. Таким чином, надалі можна прийняти визначення, що ОВ – це фізичні поля, явища, характеристики різного роду, які піддаються виявленню та аналізу за допомогою розвідувальної апаратури і які можуть бути джерелом інформації про ВзОД. В основу цього визначення покладені підходи та визначення, прийняті у роботах, наприклад [7], А.Л. Горєліка, Г.І. Кутіна. Звідти також використані ідеї щодо критеріїв класифікації ОВ.

Ознаки, що характеризують відомості, можна класифікувати за такими критеріями.

За наближеністю до ВзОД ОВ бувають первинними та вторинними. Первинні ознаки – це фізичні характеристики об`єктів і хімічних речовин, які безпосередньо реєструються апаратурою розвідки та містять дані про ВзОД. До первинних ОВ можна віднести характеристики полів випромінювання, концентрацію хімічних середовищ, тощо. Вторинні ОВ є продуктом накопичення, обробки та аналізу первинних ОВ і дозволяють вирішувати завдання про

розпізнавання ВзОД складних об`єктів, технологій, взаємодій, явищ, хімічного складу речовин.

Критерії класифікації:

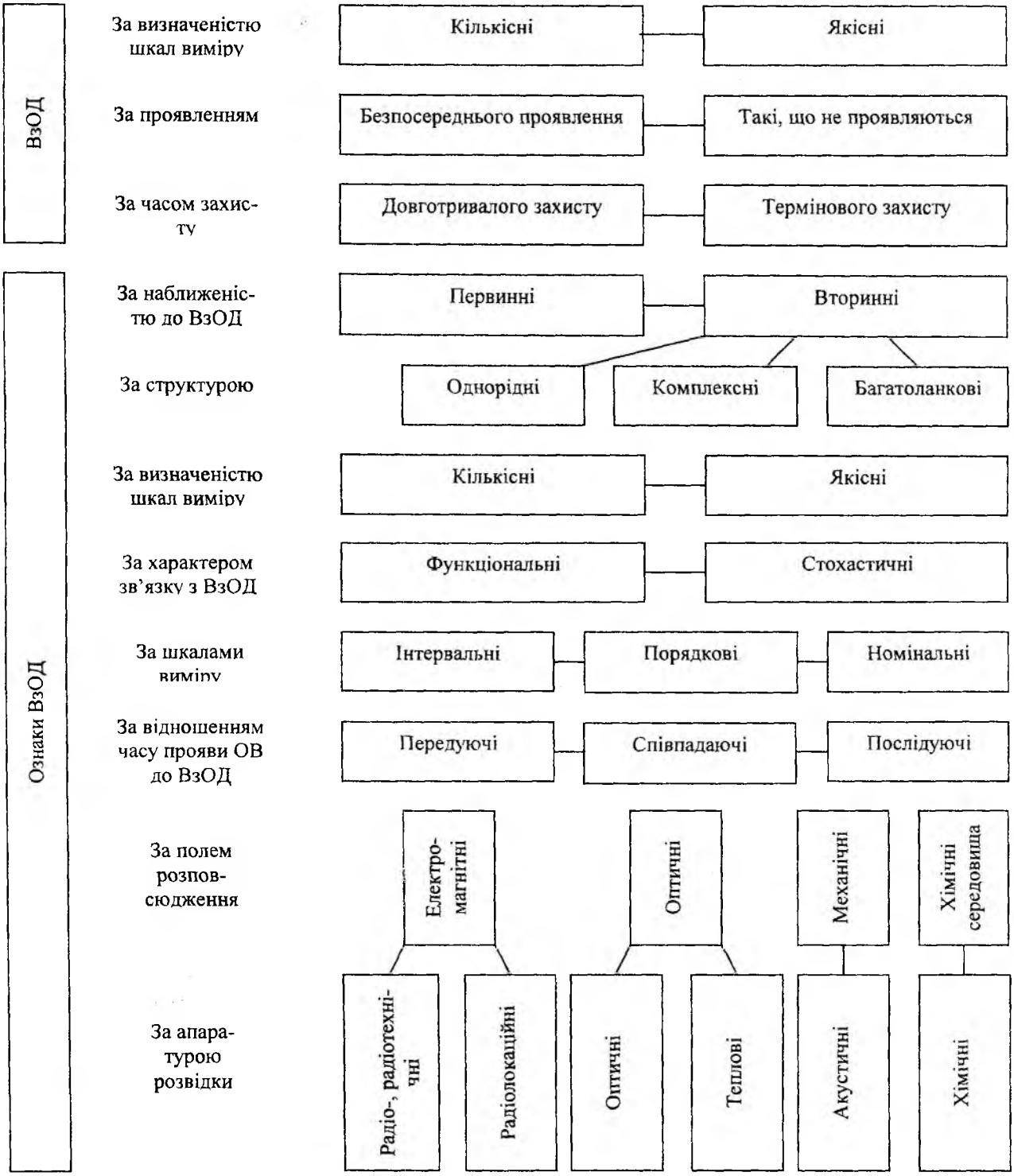


Рис. 3

За структурою вторинні ознаки можуть бути однорідні, одержані шляхом обробки однієї первинної ознаки характерної ВзОД (рівень радіосигналу, що приймає апаратура розвідки, може нести дані про діаграму спрямованості антени розвідуваного радіоелектронного засобу). Існують і комплексні ОВ, одержані на основі спостереження за комплексом різноманітних первинних або однорідних вторинних ознак, що характеризують відомості. Це, наприклад, за шумовими звуками та за хімічним складом вихлопних газів можна встановити тип двигуна, який випробовують на стенді у конструкторському бюро. Також можна відзначити і багатоланкові вторинні ОВ. Їх хронологічна або просторова прояви почергово породжує таку ознаку, що розкриває ВзОД. Прикладом цього може бути ВзОД «об'єм підземного сховища, що будується» і ланцюг ланок ОВ – кількість ходок автомашин, що вивезли ґрунт із котловану, кількість куп ґрунту на звалищі, розміри котловану, що утворився.

Як і у класифікації ВзОД ОВ можуть бути *за визначеністю шкал виміру* кількісними та якісними.

За характером зв`язку ОВ з ВзОД можуть бути функціональними (ВзОД – дальність радіозв'язку, ОВ – потужність радіопередавача) або стохастичними (ВзОД – ресурс двигуна, ОВ – час роботи двигуна до виходу з ладу).

За шкалами виміру ОВ відносять до інтервальних, порядкових і номінальних. Інтервальні ОВ відображаються дійсними числовими величинами, що характеризують ступінь прояву визначеної властивості. До інтервальних ознак можна віднести «рівень шуму двигуна» по відношенню до ВзОД «потужність двигуна». Порядкові ОВ відображаються числами – номерами, які показують місце, що займає реалізація даної ознаки у впорядкованому ряді ступеня прояви властивості. Приклад порядкових ОВ відносно ВзОД як «призначення двигуна, що розробляється» за потужністю:

для авто масового користування – 70...120 кінських сил,
для представницького авто – 150...250 кінських сил,
для спортивного авто – 250...600 кінських сил.

Номінальні ознаки, що характеризують відомості, позначаються умовними символами та відображають факт наявності або відсутності визначеної властивості. Наприклад, для відомості «тип авто, що розробляється» однією з ознак буде «відкритий кузов», що характерно для спортивного авто.

Розгляд ОВ у статичному варіанті, без урахування співвідношення їх з часом прояву ВзОД не завжди дає повну картину. Тому доцільно проводити класифікацію *за відношенням часу прояви ОВ до ВзОД*. Таким чином, можна виділити такі класи. Передуючі – такі ОВ, які виникають в часі до реальної прояви ВзОД. До такого класу належить ознака «глибина котловану» відносно до ВзОД «об'єм сховища». Співпадаючі – такі ознаки, що виникають і діють під час прояви відомості. Це – «час роботи двигуна» при його випробуванні на ресурс. Послідуючі – такі ознаки, що характеризують відомості, які виникають після прояви ВзОД. До них належать «тривалості роботи контрольних двигунів», за якими оцінюють статистичні параметри ресурсу. Кількість подібних класів, при яких враховуються співвідношення часів початків і закінчень існування відомостей та ознак, може скласти до 15-ти.

За полем розповсюдження ОВ, можуть розрізнятися на поля:

електромагнітні,
оптичні,
механічні,
хімічні середовища.

За апаратурою розвідки, яка здатна фіксувати ОВ, класифікація ТКВІ слідуюча :

радіо-, радіотехнічні,
радіолокаційні,
оптичні,
теплові (інфрачервоні),
акустичні (звукові, гідроакустичні),
хімічні.

Наведений перелік критеріїв класифікації та прийнятих для цього класів не є остаточним. Він може доповнюватися та корегуватися в залежності від поставлених завдань класифікації та розвитку науки й техніки у відповідних галузях.

Висновки

Дані про результати виробничої, проектної та іншої діяльності установ, підприємств, юридичних та фізичних осіб, які потрібно захищати від витоку по технічних каналах доцільно поділяти на інформацію та відомості (ІзОД та ВзОД).

Запропоновані підходи до класифікації технічних каналів витоку інформації і відомостей дають змогу створювати окремі моделі загроз, для їх подальшого кількісного опису і розробки відповідних заходів захисту.

Список літератури: 1. *ДСТУ3396.0-96* Захист інформації. Технічний захист інформації. Основні положення. 2. *ДСТУ3396.1-96* Захист інформації. Технічний захист інформації. Порядок проведення робіт. 3. *ДСТУ3396.2-97* Захист інформації. Технічний захист інформації. Терміни та визначення. 4. *Положение* о технической защите информации в Украине // Безопасность информации. 1996. №2. С 56-67. 5. *Закон* Украины «Об информации» // Безопасность информации. 1995. №1. С 63-72. 6. *Родионов Ю.Н.* Обеспечение ЭМС РЭС. М.: Сов. радио, 1989. 256 с. 7. *Горелик А.Л. и др.* Методы распознавания. Учебник для вузов. М.: Высшая школа, 1984. 368 с.

Харківський національний
університет радіоелектроніки

Надійшла до редколегії 19.05.2003

УДК 681.3.06

А. А. КУЗНЕЦОВ, канд. техн. наук

ЭНЕРГЕТИЧЕСКИЙ ВЫИГРЫШ АЛГЕБРОГЕОМЕТРИЧЕСКОГО КОДИРОВАНИЯ

Одним из эффективных средств защиты информации от ошибок, возникающих при передаче по сетям связи, является помехоустойчивое кодирование. Основными требованиями к помехоустойчивому кодированию являются высокая обнаруживающая и исправляющая способность кода, низкая вносимая избыточность, высокое быстродействие и низкая сложность реализации процедур кодирования-декодирования. Недвоичные алгебраические блоковые коды, построенные по алгебраическим кривым (алгеброгеометрические коды), обладают высокой исправляющей способностью при небольшой доле вносимой избыточности [1].

Основная задача помехоустойчивого кодирования информации состоит в повышении энергетической эффективности телекоммуникационных систем. Под энергетической эффективностью систем связи понимают минимально необходимое соотношение сигнал/шум, которое требуется для обеспечения заданной достоверности приема цифровых сообщений. В качестве показателя достоверности используют вероятность ошибочного приема знаков (показатель потери достоверности), которая определяется отношением количества принятых знаков с ошибками к общему количеству переданных знаков за достаточно большой промежуток времени. Снижение минимально необходимого соотношения сигнал/шум при обеспечении заданной вероятности ошибочного приема знаков, которое позволяет получить применяемая система помехоустойчивого кодирования информации, называют энергетическим выигрышем от кодирования.

В статье проводится оценка энергетического выигрыша алгеброгеометрического кодирования информации, сравнение потенциальной помехоустойчивости М-ичных ортогональных сигналов и той помехоустойчивости, которая достигается при использовании алгеброгеометрических кодов.

1 Оценка потенциальной помехоустойчивости М-ичных ортогональных сигналов

Рассмотрим вариант некодированной передачи сообщений М-ичных символов и оценим помехоустойчивость когерентного приема ортогональных сигналов.

Вероятность ошибочного приема символа при когерентном приеме ортогональных сигналов определяется выражением [2-3]:

$$P_c = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{u^2}{2}} \left[\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u+\sqrt{2\gamma}} e^{-\frac{z^2}{2}} dz \right]^{M-1} du,$$

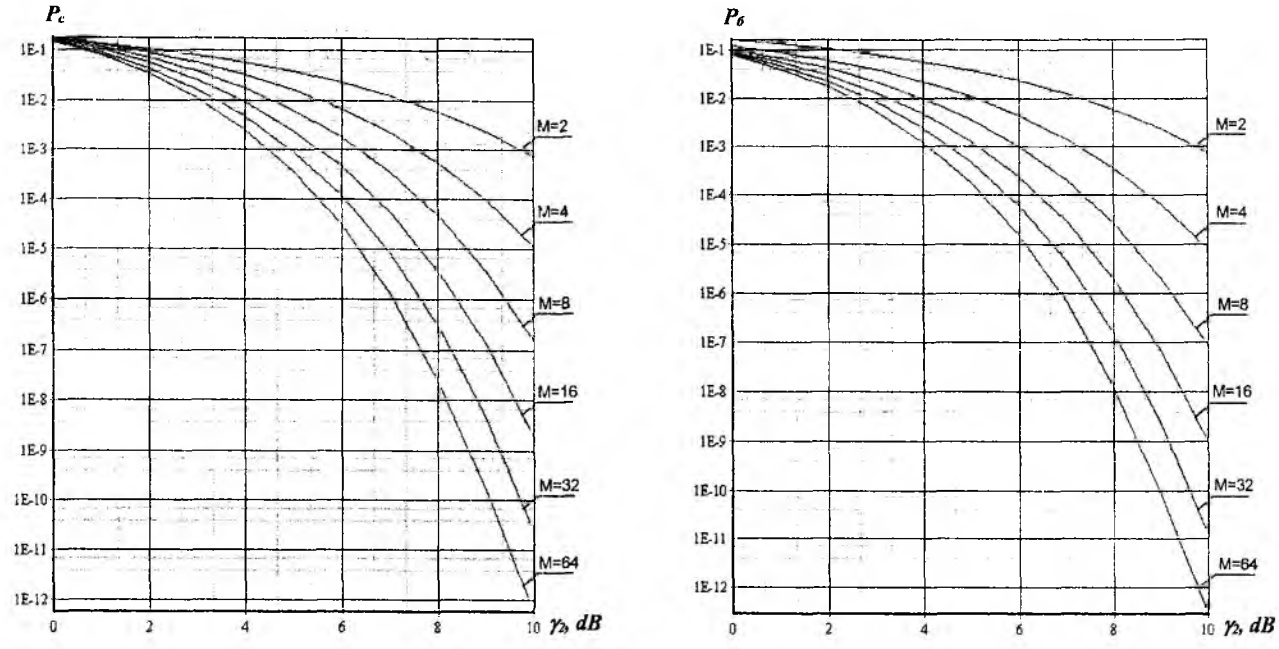
где γ – отношение сигнал/шум для М-ичного символа, $M = 2^m$.

Нормированное отношение сигал/шум на двоичную единицу запишется в виде $\gamma_2 = \gamma / m$.

На рис. 1а представлены зависимости вероятности ошибочного приема М-ичных символов для случаев М=2...64. Средняя вероятность ошибочного приема отдельного бита определяется выражением [3]:

$$P_б = \frac{2^{m-1}}{2^m - 1} P_c.$$

На рис. 1б представлены зависимости средней вероятности ошибочного приема отдельных бит М-ичного символа для случаев М=2...64.



а б

Рис. 1

Зависимости, представленные на рис. 1, свидетельствуют о том, что передача М-ичных ортогональных сигналов позволяет получить значительный выигрыш помехоустойчивости при фиксированном соотношении сигнал/шум или существенный энергетический выигрыш при фиксированной вероятности ошибки символа. При увеличении мощности ансамбля сигналов этот выигрыш возрастает. Целью данной работы является оценка энергетического выигрыша алгеброгеометрического кодирования информации при передаче М-ичных ортогональных сигналов.

2 Алгеброгеометрическое кодирование информации

Зафиксируем конечное поле $GF(q)$. Пусть X – гладкая проективная алгебраическая кривая в проективном пространстве P^n, $g = g(X)$ – род кривой, $X(GF(q))$ – множество ее точек над конечным полем, $N = |X(GF(q))|$ – их число. Пусть C – класс дивизоров на X степени α. Тогда C задает отображение $\varphi : X \to P^m$, набор генераторных функций $y_i = \varphi(x_i)$ задает алгеброгеометрический код длины $n \le N$. Кодовые характеристики (n, k, d) связаны соотношением $k + d \ge n - g + 1$.

Если $2g - 2 < \alpha \le n$, код связан характеристиками $(n, \alpha - g + 1, d), d \ge n - \alpha$. Дуальный к нему код также является алгеброгеометрическим с характеристиками $(n, n - \alpha + g - 1, d_\perp), d_\perp \ge \alpha - 2g + 2$ [4].

Рассмотрим многообразия, соответствующие проективным гиперповерхностям, заданным в P^n уравнениями $f = 0$, где f – однородные одночлены степени d в P^n. Тогда степень α класса дивизоров C на X определим как $\alpha = (X, f) = \deg X \cdot \deg f$.

Пример. Кривая X, заданная однородным многочленом $xz^2 + y^2 z + x^2 y + x^3 = 0$ над $GF(16)$ дает $N = |X(GF(16))| = 25$ точек, $\deg X = 3$, $g(X) = 1$. Точки кривой приведены в табл. 1.

Таблица 1

	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
x	0	1	0	1	1	13	10	2	9	13	4	2	3	10	11
y	1	1	0	0	1	2	3	4	4	4	5	7	7	7	8
z	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	***17***	***18***	***19***	***20***	***21***	***22***	***23***	***24***	***25***
x	7	5	7	9	6	3	4	5	11	6
y	9	10	10	10	12	13	13	13	14	15
z	1	1	1	1	1	1	1	1	1	1

Пусть $\alpha > g-1$, отображение $\varphi: X \to P^{k-1}$ задает порождающую матрицу G алгебро-геометрического кода с конструктивными характеристиками $(n \le N, k \ge \alpha - g + 1, d \ge n - \alpha)$, число генераторных функций $y_i = f_i(x, y, z)$ соответствует числу информационных кодовых символов k. Конструктивные характеристики кодов для случаев $\deg f = 1...7$ сведены в табл. 2.

Таблица 2

$\deg f$	α	Через G: $(n \le N, k \ge \alpha - g + 1, d \ge n - \alpha)$				
1	3	(25, 3, 22)	(24, 3, 21)	(22, 3, 19)	(20, 3, 17)	(18, 3, 15)
2	6	(25, 6, 19)	(24, 6, 18)	(22, 6, 16)	(20, 6, 14)	(18, 6, 12)
3	9	(25, 9, 16)	(24, 9, 15)	(22, 9, 13)	(20, 9, 11)	(18, 9, 9)
4	12	(25, 12, 13)	(24, 12, 14)	(22, 12, 10)	(20, 12, 8)	(18, 12, 6)
5	15	***(25, 15, 10)***	***(24, 15, 9)***	***(22, 15, 7)***	***(20, 15, 5)***	***(18, 15, 3)***
6	18	(25, 18, 7)	(24, 18, 6)	(22, 18, 4)	–	–
7	21	(25, 21, 4)	(24, 21, 3)	–	–	–

Пусть $\alpha > 2g - 2$, отображение $\varphi: X \to P^{r-1}$ задает проверочную матрицу H алгебро-геометрического кода с конструктивными характеристиками $(n \le N, k \ge n - \alpha + g - 1, d \ge \alpha - 2g + 2)$ число генераторных функций $y_i = f_i(x, y, z)$ соответствует числу проверочных символов кода $r = n - k$. Конструктивные характеристики кодов для случаев $\deg f = 1..7$ сведены в табл. 3.

Таблица 3

$\deg f$	α	Через H: $(n \le N, k \ge n - \alpha + g - 1, d \ge \alpha - 2g + 2)$			
1	3	(25, 22, 3)	(24, 21, 3)	(21, 18, 3)	***(18, 15, 3)***
2	6	(25, 19, 6)	(24, 18, 6)	***(21, 15, 6)***	(22, 12, 6)
3	9	(25, 16, 9)	***(24, 15, 9)***	(21, 12, 9)	(22, 9, 9)
4	12	(25, 13, 12)	(24, 12, 12)	(21, 9, 12)	(22, 6, 12)
5	15	(25, 10, 15)	(24, 9, 15)	(21, 6, 15)	(22, 3, 15)
6	18	(25, 7, 18)	(24, 6, 18)	(21, 3, 18)	–
7	21	(25, 4, 21)	(24, 3, 21)	–	–

В телекоммуникационных системах специального назначения циркулируют короткие формализованные кодограммы (10-15 четырехбитных символа). Для обеспечения высокой помехоустойчивости передаваемых сообщений предлагается применение помехоустойчивых кодов, выделенных в таблицах 2-3 курсивом.

Пример проверочной матрицы H в систематическом виде $H = [P\ I]$ кода (24, 15, 9) для случая $\deg F = 3$, $\alpha = \deg F \cdot \deg f = 9$ над полем $GF(16)$ приведен в табл. 4 (единичная матрица I опущена).

Таблица 4

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	5	6	13	6	13	8	6	10	5	12	1	10	1	7	12
2	5	10	0	12	9	11	3	2	6	13	3	10	10	13	15
3	6	14	1	0	11	8	5	6	14	7	10	3	13	8	1
4	10	5	14	13	6	12	5	15	8	7	0	10	14	5	4
5	1	13	7	4	15	7	8	2	6	3	13	11	0	2	8
6	6	3	15	11	7	4	8	4	15	15	8	13	1	2	9
7	15	10	9	6	14	1	6	6	0	11	7	1	13	6	13
8	11	10	1	5	3	0	2	5	8	2	5	10	7	4	5
9	9	15	5	7	6	14	11	15	6	4	9	7	10	3	14

Проведем оценку энергетического выигрыша предлагаемых кодовых конструкций.

3 Оценка энергетического выигрыша алгеброгеометрических кодов при когерентном приеме 16-х ортогональных сигналов

Рассмотрим код (n,k,d). Полагаем, что ошибки в последовательно передаваемых кодовых символах происходят независимо с вероятностью P_o. Тогда вероятность ошибки кратности i на длине блока n будет

$$P_i = C_n^i P_o^i (1-P_o)^{n-i}.$$

Если декодер исправляет $t=(d-1)/2$ ошибок, то вероятность ошибочного декодирования блока запишется в виде выражения

$$P_{бл} = \sum_{i=t+1}^{n} P_i = \sum_{i=t+1}^{n} C_n^i P_o^i (1-P_o)^{n-i}.$$

Если принять предположение о случайном возникновения $2t+1$ и более ошибок в результате ошибочного декодирования кодового слова, то математическое ожидание ошибочных информационных символов на выходе декодера определяется выражением [5]

$$m_{ош} = \sum_{i=t+1}^{n-t} \frac{(i+t)k}{n} P_i + k \sum_{i=n-t+1}^{n} P_i,$$

а вероятность ошибочного декодирования информационного символа –

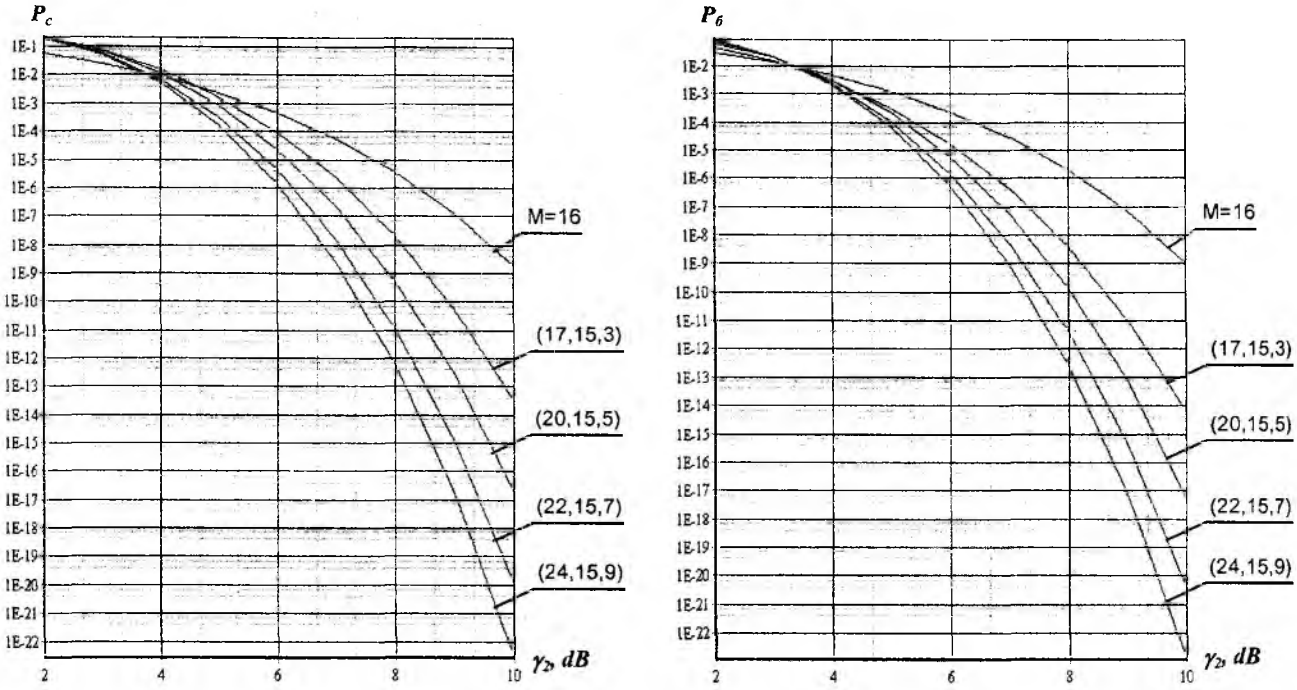
$$P_{оо} = m_{ош} P_{бл}.$$

Использование помехоустойчивого кодирования сопряжено с внесением избыточности в передаваемые данные. Если зафиксировать энергию сообщения, передаваемого в канал связи, то энергия, приходящаяся на один символ, уменьшится пропорционально внесенной избыточности. В выражении для расчета вероятности ошибки символа отношение сигнал/шум γ уменьшится в $R=k/n$ раз.

Зависимости вероятностей ошибок на выходе декодера приведены на рис. 2:

- на рис. 2а представлены зависимости вероятности ошибочного приема 16-ичных символов от нормированного энергетического отношения сигнал/шум, приходящегося на один бит;
- на рис. 2б представлены зависимости средней вероятности ошибочного приема отдельных бит 16-ичных символов от нормированного энергетического отношения сигнал/шум, приходящегося на один бит

Передача символов осуществляется 16-ичными ортогональными сигналами. Зависимость, отмеченная «M=16», соответствует некодированной передаче. На рис.2 приведена также зависимость вероятности ошибки символа при использовании кода Рида-Соломона.



а б

Рис. 2

Как следует из представленных на рис. 2 зависимостей, применение алгеброгеометрических кодов приводит к значительному энергетическому выигрышу. Так, для вероятности ошибочного приема символа 10^{-6} применение кода (24, 15, 9) позволяет получить энергетический выигрыш ≈2,5dB по сравнению с некодированной передачей и ≈0,8dB по сравнению с использованием расширенного кода Рида-Соломона (17, 15, 3). При значении нормированного энергетического отношения сигнал/шум 8dB удается понизить вероятность ошибки более чем на 6 порядков по сравнению с некодированной передачей и на 4 порядка по сравнению с использования кода Рида-Соломона.

Список литературы: 1. *Цфасман М.А.* Коды Гоппы, лежащие выше границы Варшамова – Гилберта. // Проблемы передачи информации. 1982. № 3. С. 3 – 6. 2. *С. Стейн, Дж. Джонс.* Принципы современной теории связи и их применение к передаче дискретных сообщений. М.: Связь, 1971. 376 с. 3. *В.И. Долгов.* Основы статистической теории приема дискретных сигналов. Харьков: ХВВКИУРВ, 1989. 448 с. 4. *Влэдуц С.Г., Манин Ю.И.* Линейные коды и модулярные кривые // Современные проблемы математики. М.: ВИНИТИ, 1984. Т. 25. С. 209 – 257. 5. *Т. Касами, Н. Токура, Е. Ивадари, Я. Инагаки.* Теория кодирования. М.: Мир, 1978. 576 с.

Харьковский военный университет

Харьковский национальный

университет радиоэлектроники

Поступила в редколлегию 20.11.2002