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