ДОДАТОК А

Програмна реалізація

```
void PRESENT80bitslice8_key_schedule(const u8* masterKey, u8* roundKeys)
{
	word k[P];
	word t[P + 1];
	int r;
	/* Load the keys */
	//LOADKEYS(P, TYPE, masterKey);
	{
		for (int i = 0; i < P; i++){
			t[i] = LOAD(((u16(*)[KEY80])masterKey)[i]);
			inverse_bytes_endian(t[i])
		}
		for (int i = 0; i < P; i++)  {
			if (i < (P / 2))
			{
				k[i] = PUNPCKLQDQ(t[2 * i + 1], t[2 * i]);
			}
			else
			{
				k[i] = PUNPCKHQDQ(t[2 * i - P + 1], t[2 * i - P]);
			}
			//PACK_KEYS8
			packing8(k[0], k[1], k[2], k[3], t[0], t[1], t[2], t[3], t[4], t[5]);
			packing8(k[4], k[5], k[6], k[7], t[0], t[1], t[2], t[3], t[4], t[5]);

		}
	}
	for (r = 0; r <= 31; r++){
		/* Store the keys */
		//STORESUBKEYS(P, TYPE, roundKeys + (P / 2) * r);
		{
			int i;
			for (i = 0; i < (P / 2); i++)
			{
				((word*)roundKeys + (P / 2) * r)[i] = k[i];
			}
		}
		//KEY_SCHEDULE_STEP(P, TYPE);
		{
			word t0, t1, t2, t3, t4;
```

```
                PRESENT80bitslice8_key_schedule_step(k[0], k[1], k[2], k[3],
k[4], k[5], k[6], k[7], t0, t1, t2, t3, t4,
                        round_constants_PRESENT80_8[r][0],
round_constants_PRESENT80_8[r][1], round_constants_PRESENT80_8[r][2],
round_constants_PRESENT80_8[r][3]);
            }
        }

}

void PRESENT80bitslice8_core(const u8* message, const u8* subkeys, u8*
ciphertext)
{
        word s[P / 2];
        word t[P + 1];
        word* k = (word*)subkeys;

        /* load the states */
        //LOADSTATES(P, message);
        {
            int i;
            for (i = 0; i < (P / 2); i++)
            {

                s[i] = LOAD(((u64*)message) + 2 * i);
                inverse_bytes_endian(s[i]);
            }
        }

        packing8(s[0], s[1], s[2], s[3], t[0], t[1], t[2], t[3], t[4], t[5]);

        //PRESENT_CORE_ENCRYPT_BITSLICE8_P
        {
            int r;
            word t0;
            for (r = 0; r < 31; r++)
            {
                XOR_Key(s[0], s[1], s[2], s[3], k[0], k[1], k[2], k[3]);
                Sbox(s[0], s[1], s[2], s[3], t0);
                pLayer_eight_fast(s[0], s[1], s[2], s[3], t0);
                k += (P / 2);
            }
            XOR_Key(s[0], s[1], s[2], s[3], k[0], k[1], k[2], k[3]);
        }
```

```
}


#define PRESENTROUND(state) do {\
        u64 stateIn;\
        stateIn = state;\
        state  = T0_PRESENT[stateIn & MASK8];\
        state ^= T1_PRESENT[(stateIn >> 8) & MASK8];\
        state ^= T2_PRESENT[(stateIn >> 16) & MASK8];\
        state ^= T3_PRESENT[(stateIn >> 24) & MASK8];\
        state ^= T4_PRESENT[(stateIn >> 32) & MASK8];\
        state ^= T5_PRESENT[(stateIn >> 40) & MASK8];\
        state ^= T6_PRESENT[(stateIn >> 48) & MASK8];\
        state ^= T7_PRESENT[(stateIn >> 56) & MASK8];\
} while(0)

#define PRESENTKS80(keyLow, keyHigh, round) do {\
        u64 temp;\
        keyHigh  ^= TroundCounters80[round];\
        temp      = keyHigh;\
        keyHigh <<= 61;\
        keyHigh  |= (keyLow << 45);\
        keyHigh  |= (temp >> 19);\
        keyLow    = (temp >> 3) & 0xffff;\
        temp      = keyHigh >> 60;\
        keyHigh  &= 0x0fffffffffffffff;\
        temp      = TsboxKS80[temp];\
        keyHigh  |= temp;\
} while(0)

#define BSWAP64(in) (u64)((u64)(((u64)(in) & (u64)0x00000000000000ffULL) <<
56) |\
                  (u64)(((u64)(in) & (u64)0x000000000000ff00ULL) << 40) |\
                  (u64)(((u64)(in) & (u64)0x0000000000ff0000ULL) << 24) |\
                  (u64)(((u64)(in) & (u64)0x00000000ff000000ULL) <<  8) |\
                (u64)(((u64)(in) & (u64)0x000000ff00000000ULL) >>  8) |\
                  (u64)(((u64)(in) & (u64)0x0000ff0000000000ULL) >> 24) |\
                  (u64)(((u64)(in) & (u64)0x00ff000000000000ULL) >> 40) |\
                  (u64)(((u64)(in) & (u64)0xff00000000000000ULL) >> 56) )


void Present80_Table_Impl::DoEncrypt(const u64 plaintext_in[1], const u16
keys_in[1][80], u64 ciphertext_out[1])
{
```

```
	u8 subkeys[TABLE_P * PRESENT80_SUBKEYS_SIZE];
	do_key_schedule((const u8*)keys_in, subkeys);
	do_core((const u8*)plaintext_in, subkeys, (u8*)ciphertext_out);
}


void Present80_Table_Impl::do_key_schedule(const u8* masterKey80, u8*
roundKeys80)
{
	u64 currentKeyLow, currentKeyHigh;

	/* get low and high parts of master key */
	currentKeyHigh = BSWAP64(((u64*)masterKey80)[0]);
	currentKeyLow = (BSWAP64(((u16*)(masterKey80 + 8))[0])) >> 48;

	/* get round key 0 and compute round key 1 */
	((u64*)roundKeys80)[0] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 0);

	/* get round key 1 and compute round key 2 */
	((u64*)roundKeys80)[1] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 1);

	/* get round key 2 and compute round key 3 */
	((u64*)roundKeys80)[2] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 2);

	/* get round key 3 and compute round key 4 */
	((u64*)roundKeys80)[3] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 3);

	/* get round key 4 and compute round key 5 */
	((u64*)roundKeys80)[4] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 4);

	/* get round key 5 and compute round key 6 */
	((u64*)roundKeys80)[5] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 5);

	/* get round key 6 and compute round key 7 */
	((u64*)roundKeys80)[6] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 6);

	/* get round key 7 and compute round key 8 */
```

```
((u64*)roundKeys80)[7] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 7);

/* get round key 8 and compute round key 9 */
((u64*)roundKeys80)[8] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 8);

/* get round key 9 and compute round key 10 */
((u64*)roundKeys80)[9] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 9);

/* get round key 10 and compute round key 11 */
((u64*)roundKeys80)[10] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 10);

/* get round key 11 and compute round key 12 */
((u64*)roundKeys80)[11] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 11);

/* get round key 12 and compute round key 13 */
((u64*)roundKeys80)[12] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 12);

/* get round key 13 and compute round key 14 */
((u64*)roundKeys80)[13] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 13);

/* get round key 14 and compute round key 15 */
((u64*)roundKeys80)[14] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 14);

/* get round key 15 and compute round key 16 */
((u64*)roundKeys80)[15] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 15);

/* get round key 16 and compute round key 17 */
((u64*)roundKeys80)[16] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 16);

/* get round key 17 and compute round key 18 */
((u64*)roundKeys80)[17] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 17);

/* get round key 18 and compute round key 19 */
```

```
((u64*)roundKeys80)[18] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 18);

/* get round key 19 and compute round key 20 */
((u64*)roundKeys80)[19] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 19);

/* get round key 20 and compute round key 21 */
((u64*)roundKeys80)[20] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 20);

/* get round key 21 and compute round key 22 */
((u64*)roundKeys80)[21] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 21);

/* get round key 22 and compute round key 23 */
((u64*)roundKeys80)[22] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 22);

/* get round key 23 and compute round key 24 */
((u64*)roundKeys80)[23] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 23);

/* get round key 24 and compute round key 25 */
((u64*)roundKeys80)[24] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 24);

/* get round key 25 and compute round key 26 */
((u64*)roundKeys80)[25] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 25);

/* get round key 26 and compute round key 27 */
((u64*)roundKeys80)[26] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 26);

/* get round key 27 and compute round key 28 */
((u64*)roundKeys80)[27] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 27);

/* get round key 28 and compute round key 29 */
((u64*)roundKeys80)[28] = currentKeyHigh;
PRESENTKS80(currentKeyLow, currentKeyHigh, 28);

/* get round key 29 and compute round key 30 */
```

```
	((u64*)roundKeys80)[29] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 29);

	/* get round key 30 and compute round key 31 */
	((u64*)roundKeys80)[30] = currentKeyHigh;
	PRESENTKS80(currentKeyLow, currentKeyHigh, 30);

	/* get round key 31 */
	((u64*)roundKeys80)[31] = currentKeyHigh;

	return;
}

void Present80_Table_Impl::do_core(const u8* plaintext, const u8* roundKeys80,
u8* ciphertext)
{
	u64* state, * roundKeys;

	/* cast variables */
	*((u64*)ciphertext) = BSWAP64(*((u64*)plaintext));
	state = (u64*)ciphertext;
	roundKeys = (u64*)roundKeys80;

	/* round 1 */
	state[0] ^= roundKeys[0];
	PRESENTROUND(state[0]);

	/* round 2 */
	state[0] ^= roundKeys[1];
	PRESENTROUND(state[0]);

	/* round 3 */
	state[0] ^= roundKeys[2];
	PRESENTROUND(state[0]);

	/* round 4 */
	state[0] ^= roundKeys[3];
	PRESENTROUND(state[0]);

	/* round 5 */
	state[0] ^= roundKeys[4];
	PRESENTROUND(state[0]);

	/* round 6 */
```

```
state[0] ^= roundKeys[5];
PRESENTROUND(state[0]);

/* round 7 */
state[0] ^= roundKeys[6];
PRESENTROUND(state[0]);

/* round 8 */
state[0] ^= roundKeys[7];
PRESENTROUND(state[0]);

/* round 9 */
state[0] ^= roundKeys[8];
PRESENTROUND(state[0]);

/* round 10 */
state[0] ^= roundKeys[9];
PRESENTROUND(state[0]);

/* round 11 */
state[0] ^= roundKeys[10];
PRESENTROUND(state[0]);

/* round 12 */
state[0] ^= roundKeys[11];
PRESENTROUND(state[0]);

/* round 13 */
state[0] ^= roundKeys[12];
PRESENTROUND(state[0]);

/* round 14 */
state[0] ^= roundKeys[13];
PRESENTROUND(state[0]);

/* round 15 */
state[0] ^= roundKeys[14];
PRESENTROUND(state[0]);

/* round 16 */
state[0] ^= roundKeys[15];
PRESENTROUND(state[0]);

/* round 17 */
```

```
state[0] ^= roundKeys[16];
PRESENTROUND(state[0]);

/* round 18 */
state[0] ^= roundKeys[17];
PRESENTROUND(state[0]);

/* round 19 */
state[0] ^= roundKeys[18];
PRESENTROUND(state[0]);

/* round 20 */
state[0] ^= roundKeys[19];
PRESENTROUND(state[0]);

/* round 21 */
state[0] ^= roundKeys[20];
PRESENTROUND(state[0]);

/* round 22 */
state[0] ^= roundKeys[21];
PRESENTROUND(state[0]);

/* round 23 */
state[0] ^= roundKeys[22];
PRESENTROUND(state[0]);

/* round 24 */
state[0] ^= roundKeys[23];
PRESENTROUND(state[0]);

/* round 25 */
state[0] ^= roundKeys[24];
PRESENTROUND(state[0]);

/* round 26 */
state[0] ^= roundKeys[25];
PRESENTROUND(state[0]);

/* round 27 */
state[0] ^= roundKeys[26];
PRESENTROUND(state[0]);

/* round 28 */
```

```
        state[0] ^= roundKeys[27];
        PRESENTROUND(state[0]);

        /* round 29 */
        state[0] ^= roundKeys[28];
        PRESENTROUND(state[0]);

        /* round 30 */
        state[0] ^= roundKeys[29];
        PRESENTROUND(state[0]);

        /* round 31 */
        state[0] ^= roundKeys[30];
        PRESENTROUND(state[0]);

        /* last addRoundKey */
        state[0] ^= roundKeys[31];

        /* endianness handling */
        state[0] = BSWAP64(state[0]);
}
```

ВІДОМІСТЬ МАГІСТЕРСЬКОЇ АТЕСТАЦІЙНОЇ РОБОТИ

№ док.	*Позначення*	*Найменування*	*Дод. від*
1		Текстові документи	
	ГЮІК.ХХХХ 1412Ст.02ПЗ	Пояснювальна записка	83 стр.
2		Графічні документи	
		Презентаційний матеріал	15 слайдів.
3		Інші документи	
		Електронна версія ПЗ	
		Рецензія	1 стр.
		Відгук керівника	1 стр.

					ГЮІК.ХХХХ 1412Ст.02ВД
З	Л	№ докум.	Підпис	Дата	
Розроб		Бакаєв С.В.			Програмна оптимізація алгоритму шифрування PRESENT
Переві		Руженцев В.І.			
Н.конт		Конева Н.Ф.			
Затвер		Халімов Г.З.			

Літ.	*Лис*	*Лист*
Т	1	1

ХНУРЕ
Кафедра БІТ