可逆プログラミング言語 Hermes

2019SE040　長瀬由直
2019SE075　嶋田龍貴

はじめに

- Hermes とは、可逆プログラミング言語であり、軽 量暗号アルゴリズムを記述するためのドメイン特化言 語である。Janus という可逆プログラミング言語には いくつか問題点があり、それを解決することができる Hermes において、軽量暗号アルゴリズムを実装する 必要がある。

背景

- Janus の問題点は、タイミング攻撃による攻撃を防 げないという点である。暗号の処理時間が予測されて しまい、そこから攻撃を受けてしまう。Hermes は暗号 の処理時間に依存しないため、タイミング攻撃からの 攻撃を防ぐことができる。この Hermes での軽量暗号 アルゴリズムの実装はまだ少ないことが背景である。

目的

- Hermes で軽量暗号アルゴリズムを実装することを 目的とする。

関連研究

- 軽量暗号 について

軽量暗号とは、メモリサイズや消費電力といった制 約がある中でのデバイスに暗号技術を実装するための 技術である。Iot や CPS に対して有効であると考えら れている。軽量暗号の例として、AES、Camelia、SPECKなどが挙げられる。

関連研究

- サイドチャネル攻撃 について

デバイスの物理的な特性から情報を抜き取り、盗聴 などを行おうとする攻撃手段のことである。

関連研究

- タイミング攻撃 について

サイドチャネル攻撃の一つである。暗号処理、復号 処理に要する時間をもとに暗号鍵を推定しようとする 攻撃手段のことである。

HermesとCの違い

- C言語の暗号化アルゴリズムは図1のようになる。Hermesの暗号化アルゴリズムは図2のようになる。大きな違いとしては二つ挙げられる。一つは、Cでは明示的な関数宣言が必要であるが、Hermesでは必要ではないということである。もうひとつは、Hermesではローカル変数がuncomputationによって0に上書きされるのに対して、C言語では、上書きがされないということである。また、C言語とHermesは、構文が似ているためC言語を理解している人にとってHermesは理解がしやすい。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <inttypes.h>

#define __STDC_FORMAT_MACROS
#include <inttypes.h>

void rc5_F(), rc5_B();
void main_F(), main_B();

void rc5_F(uint32_t ct[], uint32_t S[])
{
  {
    uint32_t A = 0;
    uint32_t B = 0;
    A ^= ct[0]; ct[0] ^= A; A ^= ct[0];
    B ^= ct[1]; ct[1] ^= B; B ^= ct[1];
    A += S[0];
    B += S[1];
    { int32_t i = 2;
      int32_t i_end = 1024;
      while (i != i_end)
        {
          A ^= B;
          { int tmp_ = B & 0x1f;
            A = ((A << tmp_) | (A >> (32 - tmp_))) & 0xffffffff;
            tmp_ = 0; }
          A += S[i];
          B ^= A; A ^= B; B ^= A;
          i++;
        }
    }
}
```

```
    ct[0] ^= A; A ^= ct[0]; ct[0] ^= A;
    ct[1] ^= B; B ^= ct[1]; ct[1] ^= B;
    if (B != 0)
      fprintf(stderr,"B not zero at end of block starting at line 3\n");
    if (A != 0)
      fprintf(stderr,"A not zero at end of block starting at line 3\n");
  }
}
void rc5_B(uint32_t ct[], uint32_t S[])
{
  {
    uint32_t A = 0;
    uint32_t B = 0;
    ct[1] ^= B; B ^= ct[1]; ct[1] ^= B;
    ct[0] ^= A; A ^= ct[0]; ct[0] ^= A;
    { int32_t i = 1024;

    int32_t i_end = 2;
    while (i != i_end)
      {
        i--;
```

```
        B ^= A; A ^= B; B ^= A;
        A -= S[i];
        { int tmp_ = B & 0x1f;
          A = ((A >> tmp_) | (A << (32 - tmp_))) & 0xffffffff;
          tmp_ = 0; }
        A ^= B;
      }
  }
  B -= S[1];
  A -= S[0];
  B ^= ct[1]; ct[1] ^= B; B ^= ct[1];
  A ^= ct[0]; ct[0] ^= A; A ^= ct[0];
  if (B != 0)
    fprintf(stderr,"B not zero at end of block starting at line 3\n");
  if (A != 0)
    fprintf(stderr,"A not zero at end of block starting at line 3\n");
}
```

```
void main_F()
{
}
void main_B()
{
}
int main(int ac, char **av)
{
  if (ac>1 && strcmp(av[1],"-r")==0) main_B();
  else main_F();
  exit(0);
}
```

図1　C言語で記述したRC5

```
rc5 (u32 ct[1024], u32 S[1024])
{
u32 A, B;
A <-> ct[0]; B <-> ct[1];
A += S[0]; B += S[1];

for(i=2; 1024){
A ^= B; A <<= B; A += S[i];
B <-> A;
i++;
}
ct[0] <-> A; ct[1] <-> B;
}
main()
{
}
```

図2　Hermesで記述したRC5

今後の課題

- Hermesの構文について理解し、新たな暗号化アルゴリズムを考案したり、既存の暗号化アルゴリズムを改善する必要がある。また、Janusについて理解し、Hermesの利点について理解する。

まとめ

- 今回はHermesの実行環境を構築し、文献[1]に記載されている軽量暗号アルゴリズムの一例を記述した。さらに、C言語でも同じプログラムを記述し、両者の違いを知った。

参考文献

- 1] Mogensen, Torben Ægidius,Hermes: A language for light-weight encryption. In Andrei Voronkow Nikolaj Bjrner,Irina Virbitskaite,editor,Reversible Computation,pp.243-251,Cham,2019.Springer International Publishing.
- [2] CRYPTREC 暗号技術ガイドライン (軽量暗号),https://www.cryptrec.go.jp/report/ cryptrec-gl-2003-2016jp.pdf