卒業論文紹介

Feistel構造をもつ軽量暗号の系統的な可逆化

名古屋大学　大学院　情報学研究科　複雑系科学専攻

修士1年　鈴木琳久

目次

- ①前提知識の紹介
- ②研究目的
- ③提案手法の紹介
- ④デモでの実例紹介

※2人での共同研究ですが、②以降は私が担当した部分について紹介します。

①暗号と変換プロセス

- 暗号とは？

情報が第三者には内容が分からないように変換する技術

暗号化鍵

平文
（元のデータ）

鍵を使って暗号化

鍵を使って復号

暗号文
（変換されたデータ）

復号鍵

長い暗号の安全性は高い！
→でも処理に時間がかかる、、、

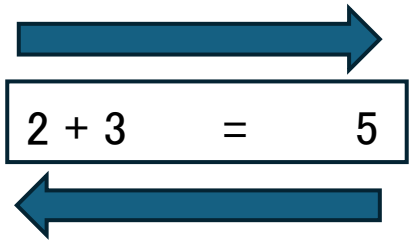
安全性と処理効率はトレード・オフ

①可逆計算と暗号分野の親和性

- 可逆計算とは？

計算過程において、計算前と計算後の結果が一意に定まるような計算

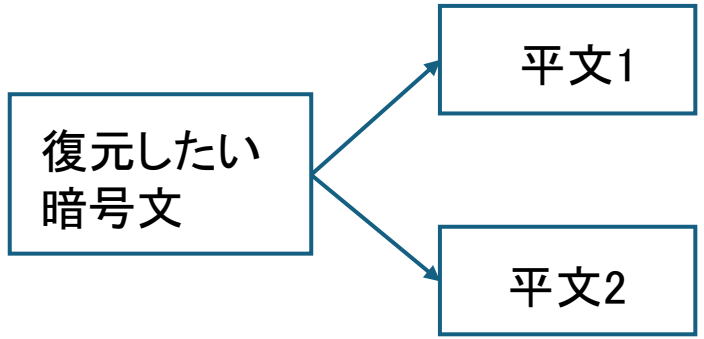
計算結果は必ず1つ



1 + 4
などの反例あり

このような計算は可逆でない！

1対1でないと、、、



→普段、私たちが日常で行う計算のほとんどは「不可逆」な計算

→暗号は平文と暗号文が1対1の関係である必要があるため、可逆計算と相性◎

②既存の問題と研究目的

- Feistel構造は右図のような対称的な構造をもつ

→暗号化と復号の手順が逆

→可逆プログラミング言語を使うのが自然

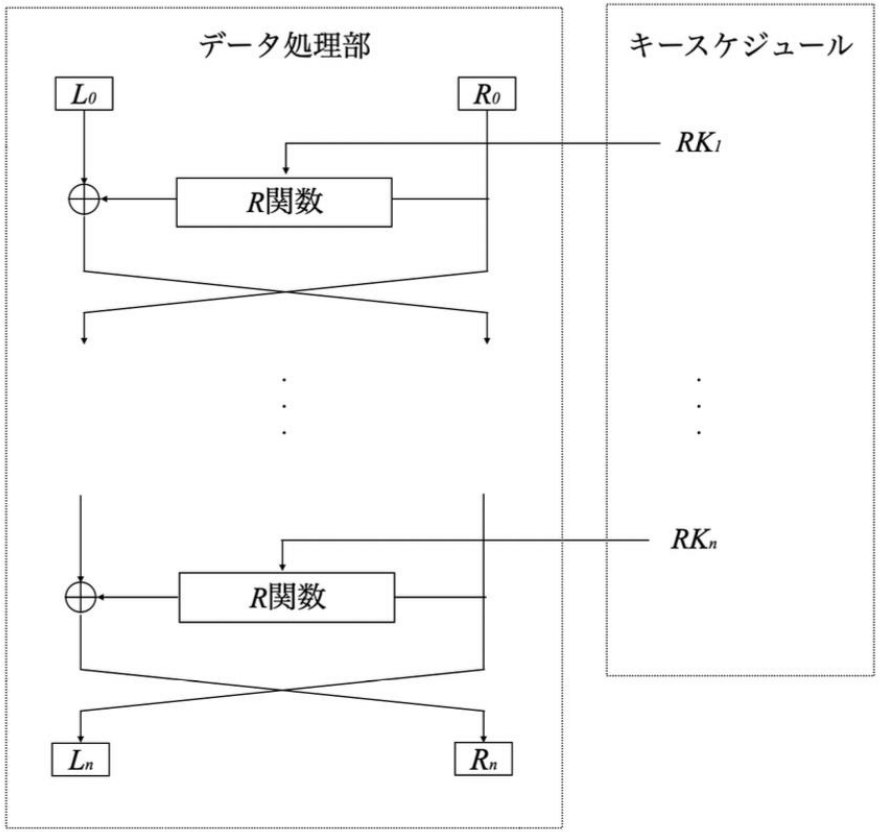
- 可逆プログラミング言語Hermesで暗号アルゴリズムを記述したい！

→Hermesでは可逆な文しか書けない

→暗号アルゴリズムには可逆性が無い場合がある

→手動で可逆化するのも煩雑

Feistel構造の煩雑な可逆化を自動したい！



③提案手法

手順1：仕様の単射化

可逆性をもたせるために、暗号鍵も出力するように変更する

$$encrypt(t,k) = c \longrightarrow encrypt'(t,k) = (c,k)$$

手順2：プログラムの可逆化

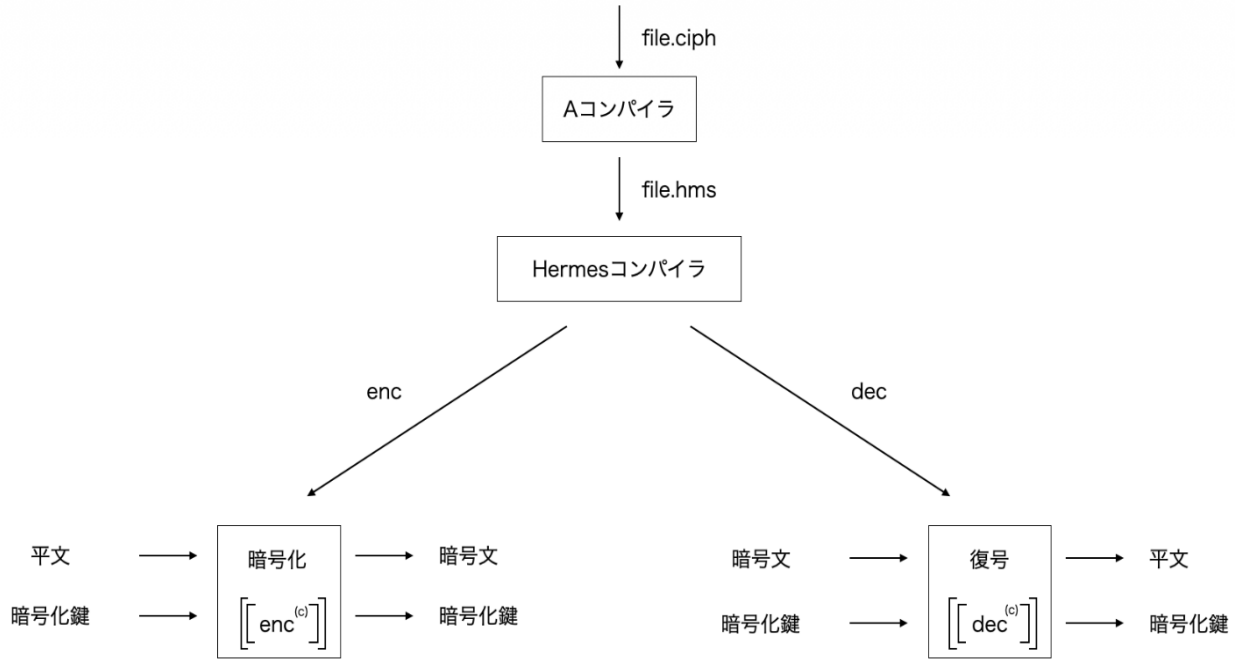
各ステップで1対1の関係を保つようにする

手順3：変数の依存関係をグラフ化

トポロジカルソートによるアプローチを使用

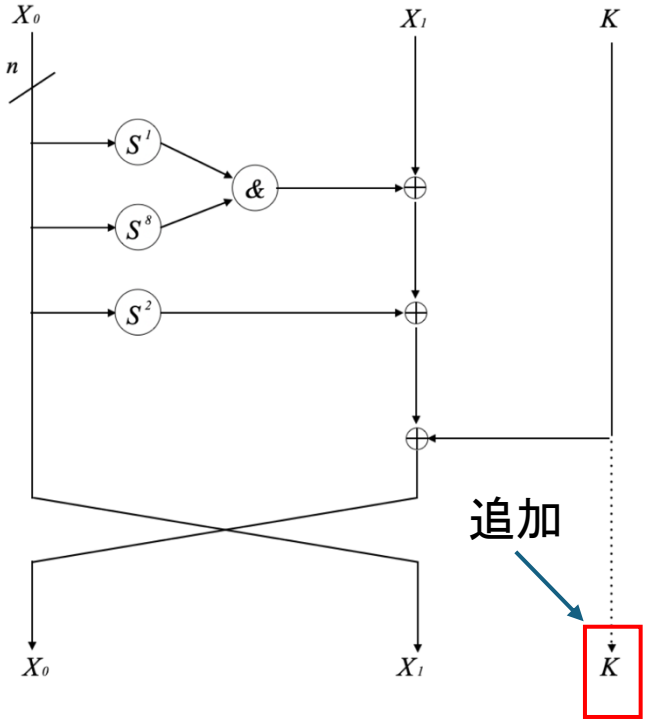
手順4：Hermesプログラムの生成

擬似言語プログラムから機械的に変換する



提案手法の全体図

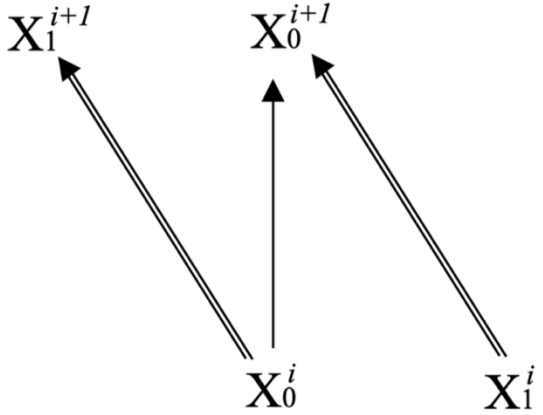
④軽量暗号「SIMON」での実行例



手順1

$$X_0^{i+1} \leftarrow X_1^i \oplus ((S^1 X_0^i \& S^8 X_0^i) \oplus S^2 X_0^i) \oplus RK_i,$$
$$X_1^{i+1} \leftarrow X_0^i.$$

手順2



手順3

$$X_1 \leftarrow X_1 \oplus ((S^1 X_0 \& S^8 X_0) \oplus S^2 X_0) \oplus RK_i,$$
$$X_1 \leftrightarrow X_0.$$

```
simon(u64 X0,u64 X1,u32 RK[68])
{
  for (i = 0; 68) {
    X1 ^= (X0 << 1 & X1 << 8) ^ (X0 << 2) ^ RK[
    i]
    X1 <-> X0;
  }
}
```

手順4

提案した手順に沿って進めれば、
正しいコードを取得可能に！