軽量暗号化アルゴリズムに対する可逆プログラミング言語 Hermes の有効性の評価

2018SE076 佐々木龍之介
指導教員 横山哲郎

1 はじめに

近年，可逆プログラミング言語 Janus を軽量暗号に用いることが検討されている．しかし，Janus ではタイミング攻撃を防ぐことが困難であったため，新たな言語として Hermes が提案された．Hermes は軽量暗号アルゴリズムに焦点を当ててつくられた可逆プログラミング言語である．しかし，現状 Hermes を用いて記述を行った軽量暗号アルゴリズムのプログラムの数が少ないため，より多くの軽量暗号アルゴリズムを，Hermes で記述を行えるかを検証する必要がある．そこで，軽量暗号を Hermes プログラムで読み書きできるように Hermes の構文と意味論について理解する．特定の計算機に依らない形式意味論で厳密で曖昧さのない記述をし，プログラムの推論・解析をするための準備をする。また，文献 [1] に記載されている軽量暗号 TEA，RC5，Speck128 の Hermes プログラムから暗号化，復号の C プログラムが生成されることを確認する．他にも，これら以外の軽量暗号についても調査を行う．そして，それらが Hermes で記述可能であるか，Hermes プログラムから C プログラムを生成し，生成された C プログラムが脆弱性を持つかを評価する．

2 関連研究

2.1 暗号

暗号とは情報を伝達する際に第三者から情報を守るために使われている技術である．また，暗号化された文自体を暗号と呼ぶ場合もある．今回は，意味の混同を避けるため，暗号化された文を暗号文と呼ぶこととする．基本的には，平文を暗号化した文を他者へ伝達した後，受け取った相手がその暗号文を復号し元の文へと戻すことで情報のやりとりを行う．平文とは第三者が読んでも理解できるような文のことである．暗号化とは平文を暗号文に変換することである．復号とは暗号文を平文に戻すことである．

代表的な暗号方式として，共通鍵暗号と公開鍵暗号が存在する．共通鍵暗号では事前に 1 つの秘密の暗号鍵を共有しておき，暗号化，復号のどちらの場合も同じ鍵（共通鍵）を利用する．公開鍵暗号では公開鍵と秘密鍵の 2 つの鍵を使用して暗号化と復号を行うので，それぞれ異なる鍵を利用する．

共通鍵暗号には暗号化，復号の処理速度が速いというメリットがあるが，予め秘密裏に暗号鍵を共有しなければならないというデメリットがある．公開鍵暗号は予め秘密裏に暗号鍵を共有しなくても良いというメリットがあり大規模拡張性に優れている．しかし，共通鍵暗号と比べ暗号化，復号の処理速度が極端に遅いというデメリットがある．したがって，共通鍵暗号，公開鍵暗号の 2 種類を組み合わせることでそれぞれの欠点を補ったハイブリッド暗号が使用されることが多い．

2.2 暗号に対する攻撃

暗号に対する攻撃とは暗号文を何かしらの方法を使用することによって不正に情報を入手しようとすることである．攻撃の例としてサイドチャネル攻撃などが挙げられる．

サイドチャネル攻撃とは暗号化・復号の処理時間，消費電力量，パソコンのノイズなど，物理的な要因から不正に情報を手に入れようとする攻撃である．暗号化・復号する処理時間を計測しその時間から解析を行うことをタイミング攻撃，消費電力量を計測することで解析を試みることを電力解析攻撃，微弱な電磁波を測定することで解析しようとすることをテンペスト攻撃という．

今後，暗号に対する攻撃の手段は増加していくことが予想される．したがって，攻撃から情報を守るため，随時セキュリティを更新していく必要がある．

2.3 軽量暗号

軽量暗号とは限られた条件の中でも実装可能な暗号である．どのようなものが軽量暗号なのかは厳密に定義されていないが，軽量暗号は ISO/IEC によって標準化が進んでいる．現在，軽量暗号として認められているものは主に共通鍵暗号方式である．軽量暗号の例として，AES,Camelia,SPECK などが挙げられる．軽量暗号で求められる条件として，以下のようなものが挙げられる．

1. 消費電力量についての制約
2. メモリサイズについての制約
3. 処理時間についての制約
4. 回路規模についての制約

軽量暗号の利用例として，RFID などが挙げられる．RFID とは専用の IC チップと無線通信を利用した非接触型の情報管理技術である．RFID は主に物流管理として大量に利用されるので，搭載されるチップは出来るだけコストを抑えなければならない．回路規模を小さくすることで生産コストを抑えることが出来る．したがって，軽量暗号が利用されている．

2.4 プログラミング言語

プログラミング言語とは，コンピュータのプログラムを記述するために用いられる形式的な言語である．コンピュータが処理を行うには，プログラムの意味は厳密である必要がある．自然言語では，しばしば意味に曖昧さが残ることがあるため，プログラムの記述に利用するのはあまり適切ではない．したがって，構文論と意味論を用いて厳密な定義を行う必要がある．

構文論とは，どのように文字や記号などを並べれば単語（トークン）として成立するか，また，単語をどのように並べれば文として成立するかを定めた規則である．構文論はあくまでどのような文字列の並び方が正しいかを定めるだけであり，文字列が何を意味するのか，どのような意味が与えられるかについては定めていない．それらは意味論によって定義される．意味論とは構文論において文として認められたものに対して，それが何を意味するのかを対応させる規則である．意味論は主に表示的意味論，操作的意味論，公理的意味論に分類される．

2.5 命令型言語

命令型言語とは，目的を達成するにあたり，どのような処理を行うかを明確に記述する言語である．手続き型言語とも呼ばれる．命令型言語という名称は，宣言型言語と区別するために用いられる．命令型言語が処理方法を記述するのに対し，宣言型言語は，処理方法を記述せず，目的のみを宣言する．

構造化命令型言語とは，構造化プログラミングの考え方を取り入れた命令型言語である．すなわち，目的を達成するにあたり，どのような処理を行うかを明確に記述し，その処理を順次，選択（分岐），反復のいずれかで表す言語である．

2.6 可逆計算

可逆計算とは，任意の計算過程の直前の状態が，高々一つに定まるような計算である．可逆プログラミング言語とは，可逆なプログラムのみを記述することができる言語である．プログラムの任意の実行過程において，直前の状態が高々一つに定まるとき，そのプログラムは可逆であるという．可逆でないプログラムを記述できないように制約がかけられていることにより，その分安全性が高まっており，可逆であることが保証されている．また，モジュール性も高まっており，あるプログラムを別のプログラムで再利用しやすくなっている．

3 Hermes

可逆プログラミング言語 Hermes は Janus から着想を得て作られた，軽量暗号アルゴリズムに対するドメイン固有言語である．[1] ドメイン固有言語とはある特定の分野に利用する目的で作られた言語であり，それ以外の目的で利用されることを想定していない言語である．Janus とは可逆であるプログラムしか記述することができない命令型プログラミング言語である．軽量暗号に Janus を用いる利点として，暗号化，復号を 1 つのプログラムで実行できる点などが挙げられる．Janus は暗号化，復号の処理時間が暗号鍵と平文の値によって変化する制御構造を持っている，したがって，サイドチャネル攻撃の一種であるタイミング攻撃から守られていない．一方，Hermes は暗号化，復号の処理時間が暗号鍵と平文の値と独立している制御構造を持っていることによって，タイミング攻撃から守られている．[2]

- Hermes の構文
  Hermes の構文の一部を EBNF で記述したものは図 1 のようになる．
- Hermes の意味論（判断，意味規則）
  Hermes の意味論の一部を表示的意味論で記述したものは図 2 のようになる．

4 結果と今後の課題

文献 [1] に記載されている軽量暗号 TEA，RC5，Speck128 の Hermes プログラムから暗号化，復号の C プログラムが生成されることを確認した．今後は，TEA，RC5，Speck128 以外の軽量暗号について調査し，それらは Hermes で記述可能であるかと，Hermes プログラムから C プログラムを生成し，生成された C プログラムが脆弱性を持つかを評価する．また，軽量暗号を Hermes プログラムで読み書きできるように，Hermes の構文と意味論について理解する．そして，特定の計算機に依らない形式意味論で，厳密で曖昧さのない記述をしプログラムの推論・解析をするための準備をする．

参考文献

[1] Mogensen, T.Æ.: Hermes: A Reversible Language for Writing Encryption Algorithms (Work in Progress), *Perspectives of System Informatics* (Bjørner, N., Virbitskaite, I. and Voronkov, A., Eds.), Cham, Springer International Publishing, pp.243–251 (2019).

[2] Mogensen, T.Æ.: Hermes: A Language for Light-Weight Encryption, *Reversible Computation* (Lanese, I. and Rawski, M., Eds.), Cham, Springer International Publishing, pp.93–110 (2020).

$$
\begin{array}{rcll}
\mathit{program} & = & \{\mathit{procedure}\}\ ; & \text{(program)} \\
\mathit{procedure} & = & \mathit{id}\ ,\ \text{"("}\ ,\ \mathit{args}\ ,\ \text{")"}\ ,\ \mathit{stat}\ ; & \text{(procedure)} \\
\mathit{args} & = & \mathit{type}\ ,\ (\ \mathit{id} \mid \mathit{id}\ ,\ \text{"["}\ ,\ \text{"]"}\ )\ ,\ \mid \mathit{args}\ ,\ \text{","}\ ,\ \mathit{args}\ ; & \text{(args)} \\
\mathit{type} & = & (\ \mathit{secret} \mid \mathit{public}\ )\ ,\ \mathit{inttype}\ ; & \text{(type)} \\
\mathit{stat} & = & \texttt{[}\ \mathit{lval}\ ,\ \mathit{update}\ ,\ \mathit{exp} \mid \texttt{[}\ 'if'\ ,\ \text{"("}\ ,\ \mathit{exp}\ ,\ \text{")"}\ \texttt{]}\ ,\ \mathit{lval}\ ,\texttt{"<->"}\ ,\ \mathit{lval} & \\
 & & \mid 'for'\ ,\ \text{"("}\ ,\ \mathit{id}\ ,\ \texttt{"="}\ ,\ \mathit{exp}\ ,\ \text{";"}\ ,\ \mathit{exp}\ ,\ \text{")"}\ ,\ \mathit{stat} & \\
 & & \mid (\ \mathit{call} \mid \mathit{uncall}\ )\ ,\ \mathit{id}\ ,\ \text{"("}\ ,\ \mathit{lval}\ ,\ \texttt{[}\{\quad\text{","}\quad,\quad \mathit{lval}\quad\}\quad\texttt{]}\ ,\ \text{")"} & \\
 & & \mid \{\ \mathit{decls}\ ,\ \mathit{stat}\ \}\ \texttt{]}\ ,\ \text{";"}\ ; & \text{(stat)} \\
\mathit{exp} & = & \mathit{lval} \mid \mathit{numconst} \mid \mathit{size}\ ,\ \mathit{id} \mid (\ \mathit{exp}\ ,\ \mathit{binop} \mid \mathit{unop}\ )\ ,\ \mathit{exp}\ ; & \text{(exp)} \\
\mathit{decls} & = & \mathit{type}, \mathit{lval}, \text{";"}, \mathit{decls} | \mathit{const}, \mathit{id}, \text{" = "}, \mathit{numconst}, \text{";"}, \mathit{decls}\ ; & \text{(decls)}
\end{array}
$$

図 1 Hermes の構文

$$
\begin{array}{rl}
\mathcal{S}[\![\textbf{if}\ (e) l_1\texttt{<->}l_2; : \sigma]\!](\Delta,\eta) = \sigma & \text{(CondSwap1)} \\
\quad where\ \mathcal{E}[\![e]\!](\sigma,\eta) = v \quad v = 0 & \\
\mathcal{S}[\![\textbf{if}\ (e) l_1\texttt{<->}l_2; : \sigma]\!](\Delta,\eta) = \sigma[\lambda_1 := v_2, \lambda_2 := v_1] & \text{(CondSwap2)} \\
\quad where\ \mathcal{E}[\![e]\!](\sigma,\eta) = v \quad v \neq 0 & \\
\quad \mathcal{L}[\![l_1]\!](\sigma,\eta) = (z,\lambda_1) \quad \mathcal{L}[\![l_2]\!](\sigma,\eta) = (z,\lambda_2) & \\
\quad \sigma(\lambda_1) = v_1 \quad \sigma(\lambda_2) = v_2 & \\
\mathcal{S}[\![\textbf{for}\ (x = e_1, e_2)\ s : \sigma]\!](\Delta,\eta) = \sigma_4 & \text{(Forloop)} \\
\quad where\ \mathcal{E}[\![e_1]\!](\sigma,\eta) = v_1 \quad \mathcal{E}[\![e_2]\!](\sigma,\eta) = v_2 & \\
\quad (\sigma_1,\lambda) = newlocation_{64}(\sigma) \quad \sigma_2 = \sigma_1[\lambda := v_1] & \\
\mathcal{F}[\![s : \sigma_2]\!](\Delta,\eta[x \mapsto (64,\lambda)],\lambda,v_2) = \sigma_3 & \\
\quad \sigma_4 = disposelocation_{64}(\sigma_3,\lambda) & \\
F[\![s : \sigma]\!](\Delta,\eta,\lambda,v) = \sigma & \text{(Loop1)} \\
\quad where\ \sigma(\lambda) = v & \\
F[\![s : \sigma]\!](\Delta,\eta,\lambda,v) = \sigma_2 & \text{(Loop2)} \\
\quad where\ \sigma(\lambda) \neq v & \\
\quad S[\![s : \sigma]\!](\Delta,\eta) = \sigma_1 \quad F[\![s : \sigma_1]\!](\eta,\lambda,v) = \sigma_2 &
\end{array}
$$

図 2 Hermes の意味論