L-042

暗号アルゴリズム識別手法の提案

吉田　剛　　河内　清人　　藤井　誠司
三菱電機株式会社　情報技術総合研究所

1. はじめに

近年、P2Pファイル共有アプリケーションのように、技術的な善悪はともかく、社会的に問題となるアプリケーションによる通信が増加している。そのため、多種多様なアプリケーションによる通信の中から問題となるアプリケーションによる通信を識別し、監視・制御したいというニーズが高まっている。一方で、問題となるアプリケーションは通信パケットのデータ部分に独自の暗号化を施し、問題となるアプリケーションによる通信であることを判別しづらくしようとする傾向がある。そこで、問題のあるアプリケーションによる通信を識別する第一歩として、問題となるアプリケーションプログラムが利用する暗号アルゴリズムと暗号ロジックを解析する必要がある。しかしながら、ソースコードが非公開だと人間には理解し難いバイナリコードを解析しなくてはならず、広い知識と多大な時間が必要となる。

プログラム内で利用されている暗号アルゴリズムを識別するためのツールとして、PEiD[1]が知られている。PEiDは、S-BOXや置換表、初期値といったアルゴリズム固有のバイナリパターンを利用してプログラム内の暗号アルゴリズムを瞬時に識別することができる。しかしながら、RC4のようにアルゴリズム固有のバイナリパターンを持たない暗号アルゴリズムを識別できないという課題がある。

そこで、本稿では、オペコード（命令語）の出現頻度に着目した、固定のバイナリパターンを持たない暗号アルゴリズムをも識別可能な手法を提案する。RC4に対する識別率と誤識別率を評価した結果、提案手法によりRC4を実装した関数を識別率93%、誤識別率11%の精度で識別できた。

2. 関数内のオペコードの出現頻度に着目した暗号アルゴリズム識別手法

暗号アルゴリズムがプログラム内で利用されているかどうかを識別するためには、暗号アルゴリズムとそれ以外を区別するための特徴が必要である。暗号アルゴリズムは、暗号化するデータに対して暗号アルゴリズム特有の処理を実施することから、同じ暗号アルゴリズムを実装した関数内のオペコードの出現は、その他の関数のオペコードの出現と比べて類似する傾向にあると考えられる。そこで、関数内のオペコードの出現頻度に着目した、暗号アルゴリズムの識別手法を提案する。

Suggestion of Cryptographic Algorithm Identification Method,
Go YOSHIDA, Kiyoto KAWAUCHI, Seiji FUJII,
Mitsubishi Electric Corporation Inforamtion Technology R&D Center, 5-1-1, Ofuna, Kamakura, Kanagawa, 247-8501, Japan

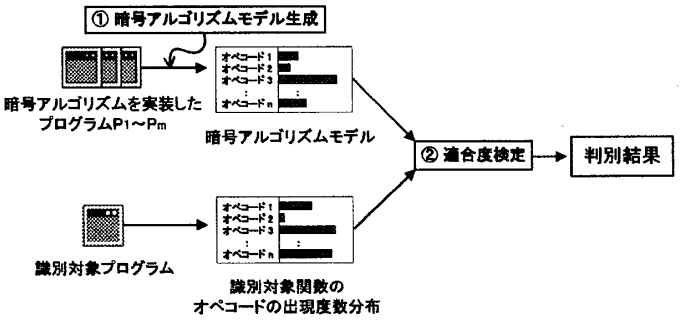

図1：提案手法概要

2.1. 提案手法概要

図1は提案手法の概要を表した図である。まず、暗号アルゴリズムを実装したプログラム$P_1 \sim P_m$から、暗号アルゴリズムを実装した関数のオペコードの出現度数分布のモデルを生成する。次に、識別対象となるプログラム内の関数（以下、識別対象関数と呼ぶ）のオペコードの出現度数分布を作成し、識別対象関数類似度を適合度検定によって算出し、識別対象関数が生成したモデルに適合するかどうか、判定結果を出力する。

2.2. 暗号アルゴリズムモデルの生成

本手法では、識別対象プログラムに暗号アルゴリズムが含まれているかどうかを判定する基準として、暗号アルゴリズムのモデル（以下、モデルと呼ぶ）を生成する。まず、暗号アルゴリズムに対して、ソースコードやコンパイラ、最適化方法（最適化なし、最大限最適化、サイズ最適化）といった実装環境が異なるプログラム$P_1 \sim P_m$を用意する。そして、プログラム毎に暗号アルゴリズムの実行関数を抽出し、実行関数内に出現するオペコード毎の出現度数を計測し、オペコードの出現度数分布を作成する。そして、それらの和をモデルとして用いる。

2.3. 適合度検定

適合度検定は、判定対象が仮定したモデルに適合しているかどうかを判定するための手法として知られている[2]。本手法では、適合度検定を元に、モデルにおけるn種類のオペコードの出現度数を$A_1 \sim A_n$、その和をΣAとし、対象関数におけるオペコードの出現度数を$B_1 \sim B_n$、その和をΣBとして、以下の式1によりモデルに対する対象関数のオペコードの出現度数の分散χ^2を算出する。ここで、E_iは対象関数がモデルに適合する場合のオペコードiの出現度数の期待値を表している。

$$\chi^2 = \sum_{i=1}^{n} \frac{(B_i - E_i)^2}{E_i}, \text{ ただし } E_i \overset{def}{=} \Sigma B \times \frac{A_i}{\Sigma A} \quad [式1]$$

次に、式 1 で得た数値を自由度 n-1 の χ^2 分布と照らし適合度を算出する。本手法では、適合度が予め定めた閾値より高ければ対象関数がモデルに適合すると判定し、閾値より低ければモデルに適合しないと判定する。

表 1：評価関数

プログラム	関数	記述言語/コンパイラ	関数の最適化方法		
			なし	最大限	サイズ
Poeny	RC4 エンコード	D/DMD	○	○	×
OpenSSL		C/GCC	○	×	×
K1		C/GCC	○	○	○
K2		C/GCC	○	○	○
K3		C/GCC	○	○	○
K4		C/GCC	○	○	○
zlib	Deflate	C/GCC	○	○	○
	InflateInit	C/GCC	○	○	○
	Inflate	C/GCC	○	○	○
	InflateEnd	C/GCC	○	○	○
	Compress	C/GCC	○	○	○
	Uncompress	C/GCC	○	○	○

3. 評価

3.1. 評価環境

提案手法の有用性を検証するために評価を実施した。表 1 は評価に利用した関数の内容を表している。RC4 関数には、Poeny[3]、OpenSSL[4]の RC4 エンコード関数と、Koders[5]に投稿されていた 4 種類（K1～K4）の RC4 エンコード関数を利用した。RC4 ではない関数には、圧縮プログラム zlib[6]の Deflate 関数、InflateInit 関数、Inflate 関数、InflateEnd 関数、Compress 関数、Uncompress 関数を利用した。また、各関数は記述言語に対応するコンパイラを用い、表 1 のように、最適化なし、最大限最適化、サイズ最適化の 3 通りの最適化を各関数に適用し、異なる 15 種の RC4 エンコード関数と、18 種の RC4 ではない関数を生成し、評価に利用した。

評価では、まず、15 種の RC4 エンコード関数から RC4 エンコード関数モデルを生成した。次に、RC4 エンコード関数モデルに対して、15 種の RC4 エンコード関数と 18 種の RC4 ではない関数それぞれの適合度を算出した。さらに、手動で誤識別の数が最小となるように閾値を 10^{-6} とした。つまり、適合度が 10^{-6} より高い関数はモデルに適合していると判定し、10^{-6} より低い関数はモデルに適合していないと判定した。そして、提案手法の識別性能の評価を実施した。評価は、15 種の RC4 関数の内、正しく識別できた関数の割合を識別率とし、18 種の RC4 ではない関数の内、RC4 関数であると誤識別した関数の割合を誤識別率として実施した。

表 2：評価結果

種別＼判定結果	RC4 関数	RC4 ではない関数
RC4 関数	14	1
RC4 ではない関数	2	16

3.2. 評価結果

表 2 に評価結果を示す。表より 15 種の RC4 エンコード関数の内、1 種（OpenSSL）が RC4 エンコード関数ではないと誤識別された。これは、OpenSSL が他の RC4 エンコード関数と比べて ADD、AND、MOVZX が多かったためである。また、18 種の RC4 ではない関数の内、2 種（Compress-M、InflateEnd-N）が RC4 エンコード関数であると誤識別された。明確な原因は不明だが、両関数ともオペコードの数が少なかったため、モデルとの差が顕著にでなかったことと、設定した閾値が低いことが原因として挙げられる。結果として、提案手法を用いることで、閾値を 10^{-6} とした時、RC4 を実装した関数として、RC4 エンコード関数を識別率 93%、誤識別率 11%の精度で識別できた。

4. おわりに

本稿では、関数内におけるオペコードの出現頻度に着目する事で、シグネチャでは識別できない暗号アルゴリズムをも識別可能な手法を提案した。評価の結果、提案手法により、識別に利用する閾値を手動で設定することで、RC4 を実装した関数として、RC4 エンコード関数を識別率 93%、誤識別率 11%の精度で識別できることを示した。

今後の課題としては、サンプル数やサンプルの種類を増やした評価を実施する必要がある。また、RC4 以外のアルゴリズムに対する評価、関数の抽出の自動化、閾値の適切な設定方法の検討も実施する必要がある。

5. 参考文献

[1] PEiD, http://peid.has.it/

[2]廣津千尋『自然科学の統計学 - 第 5 章 適合度検定』，東京大学出版会, pp145—176, 1992 年

[3]ポエニー ソースコード, http://www.rootshell.be/~kienzan/poeny/poeny_win32_src.zip

[4] OpenSSL-The Open Source toolkit for SSL/TLS, http://www.infoscience.co.jp/technical/openssl/

[5] Koders - Source Code Search Engine, http://www.koders.com/

[6] zlib Home Site, http://www.zlib.net/