2014 Tenth International Conference on Computational Intelligence and Security

Cryptographic Algorithms Analysis Technology Research Based on Functions Signature Recognition

Rui Chang
State Key Laboratory of Mathematic Engineering and Advanced Computing
MEAC
Zhengzhou, China
crix1021@163.com

Liehui Jiang
State Key Laboratory of Mathematic Engineering and Advanced Computing
MEAC
Zhengzhou, China
jiangliehui@163.com

Hui Shu
State Key Laboratory of Mathematic Engineering and Advanced Computing
MEAC
Zhengzhou, China
crix1021@163.com

Hongqi He
State Key Laboratory of Mathematic Engineering and Advanced Computing
MEAC
Zhengzhou, China
jiangliehui@163.com

***Abstract*—At present, the research on software reverse engineering pays more attention to cryptographic algorithms recognition and analysis. Malwares are becoming increasingly stealthy, more and more malwares are using cryptographic algorithms to protect themselves from being analyzed. There are no mature theory and tools to analyze cryptographic algorithms from the executable files effectively and exactly. In order to solve the problem efficiently, this paper discusses functions signature recognition technology, creating the cryptographic algorithms library including more than 3000 signature characteristics, and explores a way to recognize cryptographic algorithms based on functions signature recognition. Finally, the experiment results indicate the method is effective and feasible.**

Keywords-signature library, cryptographic algorithms recognition, minimum perfect Hash function

I. INTRODUCTION

In the field of communications and computer, data transmission and the safety of software system often rely on cryptographic algorithms. Cryptographic algorithms recognition belongs to the category of program understood [1]. The research on the category is mainly about code optimization and program analysis [2].

Malware analysis and reverse engineering seek to understand the inner workings of malware, which are invaluable to defending against malware. To prevent themselves from being analyzed and reversely engineered, more and more malwares are using cryptographic algorithms to protect the malicious code and communication. Recognizing cryptographic algorithms from the executable binary files plays an important role in analyzing malicious code and protecting computer system.

Ian Harvey first proposed the problems on cryptographic algorithms recognition in his report in 2001[3]. He summarized the local characteristic and overall characteristic of cryptographic algorithms in binary codes. However, the simple linear weighting cryptographic algorithm proposed in the report located the location of cryptographic algorithms roughly, not specifically.

Jizhong Li proposed judging methods based on Bayes decision-making model [4] and vector angle cosine [5]. These kinds of models could recognize some cryptographic functions effectively, mainly including Hash functions and block cipher. However, the recognition accuracy was limited and the results need manual analysis by analyst.

Tieming Liu established static characteristics library of cryptographic algorithms and designed Boyer-Moore algorithms for pattern matching [6]. Nevertheless, the method couldn't recognize and handle variant characteristics and characteristics dynamically generated.

Jason distinguished cryptographic functions and analyzed the results by Neural Net for Locating Cryptography [7] (NNLC for short) based on statistical characteristics of cryptographic functions. It is indicated that statistical characteristics are effective for several logic instructions, taking no account of other computing instructions. The reliability of the method needs to be further verified, especially for the recognition of stream ciphers and public-key ciphers.

On the above basis, we established cryptographic algorithms signature library based on signature recognition technology and implemented the recognition algorithms based on minimum perfect hash function. We can recognize the cryptographic algorithms (including Hash functions, block cipher, stream ciphers, public-key ciphers and others) from binary codes and locate the typical functions exactly at a reasonable overhead. It is an intuitive and effective method for recognizing and analyzing cryptographic algorithms by functions signature recognition.

The rest of this paper is organized as follows. Next section discusses the signature matching mechanism of Fast

978-1-4799-7434-4/14 $31.00 © 2014 IEEE
DOI 10.1109/CIS.2014.20

Library Identification and Recognition Technology (FLIRT) [8]. Then we build a large cryptographic algorithms signature library for recognizing core cryptographic algorithms and encryption-decryption functions from the executable files. In Section 3, we give the recognition algorithm based on Minimum Perfect Hash Function (MPHF). And then we implement the algorithms and analyze the algorithms performance in Section 4. The results of the experiment in Section 5 indicate that cryptographic algorithms from executable binary unknown files can be recognized effectively and exactly. Finally, we conclude this paper in Section 6.

II. FUNCTIONS SIGNATURE RECOGNITION TECHNOLOGY

Functions signature recognition technology uses signature matching mechanism of Fast Library Identification and Recognition Technology to recognize core cryptographic algorithms and encryption-decryption functions from the executable files. FLIRT analyzes the startup code of executable file which is waiting for recognition, identifying different compilers by their different characteristics, and loading corresponding signature file of the compiler for pattern matching through functions characteristics. FLIRT uses the first 32 bits of the function as pattern characteristic. The auxiliary characteristics are bits length, external symbolic reference and so on. FLIRT matches the function characteristic using the Hash algorithm, and then gets the signature file. Usually each kind of complier has a signature file. Therefore, we can recognize the functions by loading corresponding signature file.

The core of functions signature recognition technology is pattern match algorithms. The main problem is to quickly make sure whether the disassembled function matches one of known cryptographic algorithms library. Owing to the powerful multi-platform disassemble capability of IDA (Interactive Disassembler), FLIRT is operating in multi architectures and has strong universality.

Nevertheless, it is unrealistic to load the exclusive signature library manually when we meet the unknown binary files. Thus, we make a large cryptographic algorithms signature library, which includes several static algorithms libraries, to recognize functions signature.

Cryptographic algorithms signature library is the union of signature files, which are produced with several frequently-used static algorithms libraries, such as miracle, FGInt, openssl, crypto++, libtomcrypt, and so on. The information of these static algorithms libraries is shown in Table I. They are of different versions, different compiling environment and different compiler optimizations. O1 means minimize size. O2 means maximize speed. Ox means optimize fully.

The steps of making large cryptographic algorithms signature library are given as follows.

- The first stage is to compile the static libraries with different compiler optimizations in different compiling environment. Ordinarily, we can get some "*.obj" files and "*.lib" files from a static library. The static libraries are shown in table 1.
- In the second stage, the program "pcf.exe" is used. It preprocesses "*.obj" or "*.lib" files to produce a pattern-file. All of the pattern-files generate "all.pat" by automatic program synthesis. The pattern-file actually is a text file, which contains the patterns of the functions, their names, their CRC16, their bits length and all other information necessary to create the signature file.
- In the third stage, the "sigmake.exe" program builds the signature file "all.sig" from the pattern-file "all.pat".
- At the last stage, copy the "all.sig" to <IDADIR>/sig.

TABLE I. THE INFORMATION OF THE STATIC ALGORITHMS LIBRARIES

Static libraries	Compiling environment	Compiler optimizations
miracl4.7.4	vc6/vs2008	O1/O 2
miracl5.2	vc6/vs2010	O1/ O 2/Ox
miracl5.3	vc6/vs2008	O1/O 2
miracl5.4	vc6/vs2008	O1/ O 2
OpenSSL0.9.5	vc6	O1/ O 2
OpenSSL0.9.6a	vc6	O1/O 2
OpenSSL0.9.6m	vc6	O1/O 2
OpenSSL0.9.7	vc6	O1/O 2
OpenSSL0.9.8	vs2008	O1/O 2
OpenSSL0.9.8k	vc6/vs2008	O1/O 2
OpenSSL1.0.1	vc6/vs2008	O1/O 2
OpenSSL1.0.1c	vc6/vs2008/vs2010	O1/O 2
crypto5.0	vs2008	O1/O 2
crypto5.1	vc6/vs2008	O1/O 2
crypto5.2.1	vc6	O1/O 2
crypto5.4	vc6	O1/O 2
crypto5.5.2	vc6/vs2008	O1/O 2
crypto5.6.0	vc6/vs2008	O1/O 2
crypto5.6.1	vc6/vs2008	O1/O 2
ECDSA	delphi7	none
ECElGamal	delphi7	none
ECGFp	delphi7	none
FGInt	delphi7	none
FGIntDSA	delphi7	none
FGIntElGamal	delphi7	none
FGIntGOSTDSA	delphi7	none
FGIntPrimeGeneration	delphi7	none
FGIntRSA	delphi7	none
libtomcrypt0.96	vc6	Ox
libtomcrypt1.17	vc6/vs2010	O1/ O 2

We use the cryptographic algorithms signature library to recognize core cryptographic algorithms and encryption-decryption functions from the executable files. Therefore, the recognition results rely on cryptographic algorithms signature library to a large extent. The cryptographic algorithms library created as outlined above includes more than 3000 signature characteristics.

III. RECOGNITION ALGORITHMS BASED ON MINIMUM PERFECT HASH FUNCTION

A series of binary code make up library function feature sequences. We recognize library function by matching feature sequences of library functions and function modules in application program. When the number of functions is small, the matching is fast [9]. If the number of functions is large, the matching speed must be improved.

In order to find a faster matching method, we construct Hash signature [10] for schema files and match feature sequences of library functions by Minimum Perfect Hash Function Recognition Algorithms (MHRA). We regard the feature sequences union as edges of a graph. The values of the schema files random transformation are regarded as nodes of a graph. Then construct one-to-one mapping of feature sequences signature and node pair. Finally, match feature sequences by nodes of graph. The whole recognition process can be divided into three parts: mapping process, assignment process and matching process.

A. Mapping Process

Mapping process of MHRA is initializing the graph. The number of edges is kinds of the schema files. Firstly, analyze the elements of schema files and get the whole alphabet. Then construct two random lists on every element of alphabet, and generate two Hash values as corresponding nodes for every feature sequence. After that, analyze whether the random lists are feasible to avoid only if an edge corresponds to a node. Finally, generate the graph and detect the circuits until generating the suitable graph. The process is shown in Figure 1.

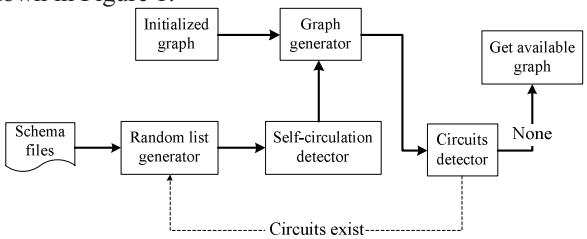

Figure 1. Mapping process model of MHRA.

Mapping process needs detecting the generated graph. Because the signature is minimum perfect Hash function signature, the constructed graph is standard, in which the feature key value and the index are one-to-one mapping. If there is self- circulation in the graph, it means one edge corresponds to two same nodes. It is hard to find the standard corresponding relationship about edge and node. Thus construct graph fails. In the same way, if there is a circuit in the graph and the whole graph is a circuit, it is allowable. There are two possible reasons for the circuits in generated graph. One is there are duplicate key. A method for the duplicate key is to delete one and reserve one. In the process of schema extraction, it is hard to avoid the two different functions getting the same characteristic. It is quite difficult to extract peculiar characteristic from two similar functions automatically. Therefore, using the strategy of ambiguity, the similar functions are recognized as a function. The other reason is that the mistakes in the random process could result in the same vertex of different key values. The only method is remapping until the efficient graph is generated.

B. Asssignment Process

Assignment process of MHRA is structuring one-to-one mapping of vertex and key value. It demands a given key value and an available specific index. After assignment process, the edge and the two corresponding nodes are one-to-one mapping relationship. First of all, we select one vertex and give it a value of 0. Then calculate the value of the other vertex in accordance with the connected edges. Finally, get the only index of edge according to the two vertexes. The process is shown in Figure 2.

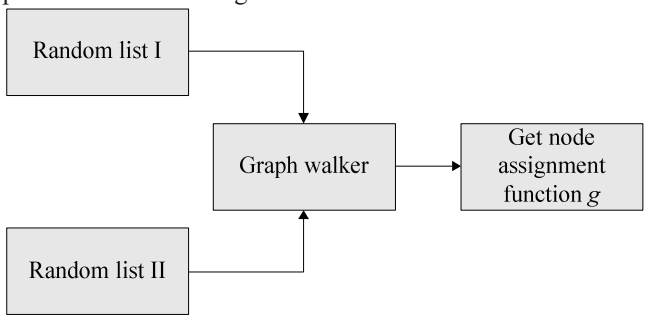

Figure 2. Assignment process model of MHRA.

Assignment process of MHRA demands each edge and its two corresponding nodes are one-to-one mapping relationship. If we get the value of edge according to calculating the random value of vertex and map them, it is highly possible to get the same edge in accordance with different vertexes in rare situation. Thus, we achieve the uniqueness edge index according to the value of edge used by function g. Consequently, the node and its corresponding edge are one-to-one mapping. The assignment algorithm is shown in Figure 3.

```
ass(v), v is a vertex of graph;
{
      visited[v]=true;
      for all <v, w> ∈ G
         if(visited[w]==false)
            g(w) = (h(e=(v,w)) – g(v))mod m;
            ass(w);
         end if;
      end;
}
visited[i] = false; i is the vertex of graph;
for all v ∈ V, V is the union of all vertexes in graph;
{
      if (visited[v] == false)
         g(v)=0;
         ass (v);
      end if;
}
```

Figure 3. Assignment algorithm of graph.

C. Matching Process

Matching process of MHRA needs to select signature library. There are three methods to choose the signature library. The first method is to select library files in known system and generate the signature directly. It is applied to the application program in known system. The second method is to obtain library files and compiler releases information according to recognize dynamic library functions. It is applied to the situation that dynamic library functions are adequate and easy to be recognized. The third method is to get different library files information according to signature of the entry code in application program and signature library selected. We use the third method. The recognition process is automatic.

The feature sequences of candidate functions are transformed by Hash function and then identified whether it belongs to signature library. Matching process of MHRA is shown in Figure 4. Firstly, analyze and obtain the feature sequences of files by features extraction algorithm. Secondly, look up random table generated by alphabet according to feature sequences and get the nodes of the graph, which are two random numbers of key values. And then obtain index of edges according to these nodes. Thus we can successfully recognize by getting the key values of corresponding edges and related information. The key is finding the corresponding value of two vertexes from assignment table G. The key to success is that the graph is acyclic. Obviously, there is only one assignment per node in the assignment process of acyclic graph. For any two vertexes, it guarantees the unique index of the corresponding edge.

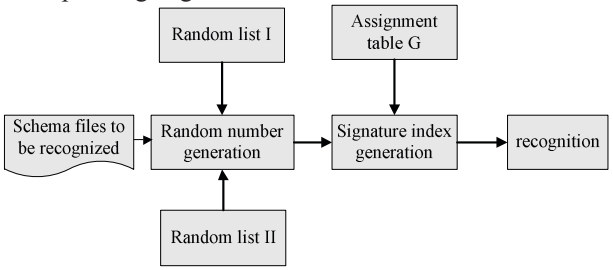

Figure 4. Matching process model of MHRA.

Set w is the feature sequences which are waited recognition. Construct two corresponding vertexes of graph are u, v. $w=\{w[1], w[2],...w[i]..., w[m]\}$

$$u = (\sum_{i=1}^{w} T_1(\mathrm{i}, \mathrm{w[i]}))\ \mathrm{mod}\ \ \mathrm{n} \quad (1)$$

$$v = (\sum_{i=1}^{w} T_2(\mathrm{i}, \mathrm{w[i]}))\ \mathrm{mod}\ \ \mathrm{n} \quad (2)$$

Then the Hash value of feature sequence is $h(w)$.

$$h(w)=(g(u)+g(v))\ mod\ m \quad (3)$$

Finally, compare the Hash value with that in known library. If it presents, it is the recognized result.

IV. Implementation and Analysis of MHRA

A. Algorithms Implementation

MHRA provides a fast characteristics matching method by structuring Hash graph. The process is described in detail as follows.

The hypothesis is that a library file L contains several library functions. $L=\{ l_1, l_2,\ldots, l_n \}$. l_i is a library function of the library file L.

Firstly, obtain the feature sequence $T(l_i)$ by features extraction method. Then take the function feature sequence as its input for the started executing of MHRA.

- For feature sequence of every function, construct two random number tables T_1,T_2 by random number generator according to alphabet of feature sequence. Generate a Hash value according to the two tables and Hash function. Detect the generated Hash value in order to avoid that a feature sequence has two same Hash value.
- Initialize Hash graph. The number of edges in Hash graph is the number of feature sequences. The number of nodes is twice more than the number of edges. Store graph by taking the array subscript as the index of edge and storing nodes in the array. And that relate each node by several affiliated array. Store the order number of feature and corresponding Hash value in the graph according to the order of feature sequences Hash function transformation. Traverse and look up the Hash graph until there is no single circuit and no multiple circuits in it. Otherwise, reconstruct the Hash graph.
- Construct function g by traversing the whole graph. Get the assignment of function g by Hash value of every node and then construct the one-to-one mapping of an edge and two vertexes.
- For a feature sequence which are waiting recognition, obtain the index of corresponding edge by Hash function $h(w)=(g(u)+g(v))\ mod\ m$. If the index presents, it is a successful recognition.

Extracting the feature sequence of function at the binary level effectively ensure the integrity of features extraction. To some extent, it reduces the conflict between different function bodies. However it can't solve all the conflicts. For example, _strupr() and _strlwr() have the same function body but different names, because they call different functions. _strupr() calls _toupper(), and _strlwr() calls _tolower(). Another example, _remove() and _unlink() in C++ library have same functions, and the call cases in the variable area are same. And yet they have different names in different usage situations. To solve these problems, we should take double recognition on static library functions by constructing auxiliary characteristics for accurate results.

B. Algorithms Performance Analysis

1. Program complexity analysis in the process of constructing Hash signature

Set the number of characteristics' key value is *m*. Set the number of the nodes is *n*.

In the mapping process, it must map the key value of every function to graph so that it needs to estimate the generation of *m* edges and *n* nodes. In the estimation, the number of random computation, the size of alphabet and the length of key all need to be estimated. It contains three parts as follows: the generation of random tables, the computing and judging every key value, and detecting whether it is acyclic. The time random table generation needed is a constant, which is equal to the size of alphabet multiplies by the length of key value. The time complexity of computing and judging key value is $O(m)$. The time complexity of detecting is $O(m+n)$. As a result, the time complexity of mapping process for a graph is $O(m+n)$.

In the assignment process, it mainly constructs mappings of function *g* by traversing the whole graph. Thus the time complexity of assignment process is $O(m+n)$.

2. Backtracking analysis in the process of constructing the graph

There are repetitions in mapping process. The number of repetitions is closely related to the number of vertexes in graph.

Suppose there are *m* edges and *n* vertexes in the graph. The probability of generating the graph is *P*. The number of repetitions in mapping process is *X*.

$$P(\mathrm{X}=\mathrm{i})=\mathrm{p}^{*}(1-\mathrm{p})^{\mathrm{i}-1} \tag{4}$$

The expected value of X is $1/p$, and its variance is $(1-p)/p^2$. The probability of the repetitive times achieving *k* is $(1-p)^k$.

The probability of generating a cyclic graph with spares graph is closely related to *n*.

Suppose $n=cm$. *c* is a constant. When *n* goes to infinity, the number of circuits of length *k* will go to $2^k/(2kc^k)$. The formula applies to self-circulation and multiple edges.

The probability of having acyclic graph is

$$\exp(-\sum_{k=1}^{n}\frac{2^{k}}{2kc^{k}}) \tag{5}$$

Hence, if $c>2$, the probability of generating acyclic graph is

$$\sqrt{\frac{c-2}{c}}$$

If $c<=2$, the probability of generating acyclic graph is 0.

After above mentioned analysis, it can be known that the time complexity of the algorithm is $O(m+n)$.

If $n=cm$, the time complexity is $O(m)$.

Meanwhile, the probability of generating acyclic graph is

$$\sqrt{\frac{c-2}{c}}$$

The expected value of the repetitive times in mapping process is

$$\sqrt{\frac{c}{c-2}}$$

V. The Result of Analysis Based on Functions Signature Recognition

By running the quick scan program, we can identify and recognize more than 3000 signature characteristics from target software by the cryptographic algorithms signature libraries, including miracle, FGInt, openssl, crypto++, libtomcrypt, and so on. We can also recognize 94 kinds of cryptographic algorithms, which are shown in Table II. Furthermore, the position of corresponding function can be accurately located.

TABLE II. The recognized cryptographic algorithms

The type	The name of recognized cryptographic algorithms	The number
Hash functions	TIGER/MD2/MD4/MD5/WHIRLPOOL/RIPEMD128/RIPEMD160/RIPEMD256/RIPEMD320/SHA0/SHA1/SHA224/SHA256/SHA384/SHA512/PANAMA	16
Block ciphers	3WAY/AES|RIJNDAEL/ANUBIS/KHAZAD/DES/DESX/BLOWFISH/CAMELLIA/CAST128/CAST5/CAST256|CAST6/RAWDES/RC2/GOST/TWOFISH/SHARK/NOEKEON/KASUMI/KSEED|SEED/SAFER/SERPENT/SKIPJACK/SQUARE/TEA/XTEA/RC5/RC6/SHACAL2/3DES/IDEA/MARS	30
Stream ciphers	XSALSA20/SALSA20/SEAL/WAKE/SOBER128/SOSEMANUK/PANAMA/ RC4	8
Public-key ciphers	GDSA/GOSTDSA/DSA/ECC/ECDH/ELGAMAL/ECDSA/ECELGAMAL/RSA	9
Others	BASE32/BASE64/BigNumCalc/CMS/GUNZIP|GZIP/X509/CRC32/TTMAC/CMAC/DIFFIE-HELLMAN/HMAC/OMAC/HUFFMAN/GCD/PELICAN/KRB5/PKCS/Crypto_Related/FGInt_Related/SSL_RELATED/PMAC/VMAC/SRP/ZLIB/POWMOD/YARROW/FORTUNA/INFLATOR/NR/SPRNG/SHA_Transform	31
Sum		94

Take the unknown executable file "cryptest.exe" as an example. We can recognize cryptographic algorithms such as sha512, tiger, md2, rc6 used in the file. The recognition results include cryptographic algorithms, encryption-decryption functions, function address, static algorithms

libraries, compiling environment and compiler optimizations, and so on. The results of signature recognition are shown in Figure 5.

Function address	Algorithm	Algorithm characteristics	Type
0x005DD7A5	SHA512	public: static CryptoPP::SHA512::Transform(unsigned __int64 ...	Sig
0x005E8A67	RIJNDAEL	protected: static CryptoPP::Rijndael::Base::FillEncTable(void)	Sig
0x005E8ACB	RIJNDAEL	protected: static CryptoPP::Rijndael::Base::FillDecTable(void)	Sig
0x005E8C37	RIJNDAEL	public: virtual CryptoPP::Rijndael::Base::UncheckedSetKey(un...	Sig
0x005EEF34	RAWDES	public: CryptoPP::RawDES::RawSetKey(enum CryptoPP::CipherDir...	Sig
0x005F048D	WHIRLPOOL	public: static CryptoPP::Whirlpool::Transform(unsigned __int...	Sig
0x005F1687	TIGER	public: static CryptoPP::Tiger::Transform(unsigned __int64 *...	Sig
0x005FF120	MD2	public: virtual CryptoPP::Weak1::MD2::Update(unsigned char c...	Sig
0x00603243	SHACAL2	public: virtual CryptoPP::SHACAL2::Base::UncheckedSetKey(uns...	Sig
0x00603D43	CAMELLIA	public: virtual CryptoPP::Camellia::Base::UncheckedSetKey(un...	Sig
0x00608DFF	WAKE	protected: CryptoPP::WAKE_Base::GenKey(unsigned int,unsigned...	Sig
0x0060B9A0	TWOFISH	public: virtual CryptoPP::Twofish::Base::UncheckedSetKey(uns...	Sig
0x0060BBB8	TWOFISH	protected: static CryptoPP::Twofish::Base::h0(unsigned int,u...	Sig
0x0060BD01	TWOFISH	protected: static CryptoPP::Twofish::Base::h(unsigned int,un...	Sig
0x0060EF88	CAST128	public: virtual CryptoPP::CAST128::Base::UncheckedSetKey(uns...	Sig
0x006108AF	CAST256	protected: static CryptoPP::CAST256::Base::Omega(int,unsigne...	Sig
0x00610CA3	RC6	public: virtual CryptoPP::RC6::Base::UncheckedSetKey(unsigne...	Sig
0x006110AA	MARS	public: virtual CryptoPP::MARS::Base::UncheckedSetKey(unsign...	Sig
0x00612127	SKIPJACK	public: virtual CryptoPP::SKIPJACK::Base::UncheckedSetKey(un...	Sig
0x00613703	BLOWFISH	public: virtual CryptoPP::Blowfish::Base::UncheckedSetKey(un...	Sig
0x006139C6	SEED	public: virtual CryptoPP::SEED::Base::UncheckedSetKey(unsign...	Sig
0x0061E786	BASE64	public: virtual CryptoPP::Base64Encoder::IsolatedInitialize(...	Sig
0x00621BA7	RC2	public: virtual CryptoPP::RC2::Base::UncheckedSetKey(unsigne...	Sig
0x006221CE	SAFER	public: virtual CryptoPP::SAFER::Base::UncheckedSetKey(unsig...	Sig
0x00622F00	GOST	protected: static CryptoPP::GOST::Base::PrecalculateSTable(v...	Sig
0x00623A0F	SHARK	CryptoPP::SHARKTransform(unsigned __int64)	Sig
0x00623A87	SHARK	public: CryptoPP::SHARK::Enc::InitForKeySetup(void)	Sig
0x005B7CCC	ECC	public: CryptoPP::ECPPoint::ECPPoint(struct CryptoPP::ECPPoi...	Sig
0x005B7D13	ECC	public: CryptoPP::EcPrecomputation<class CryptoPP::ECP>::Set...	Sig
0x005B7DD2	ZLIB	public: CryptoPP::ZlibDecompressor::UnsupportedPresetDiction...	Sig

Figure 5. The results of signature recognition about unknown executable file “cryptest.exe”.

VI. CONCLUSIONS

The cryptographic algorithms analysis technology is presented based on functions signature recognition. We can recognize core cryptographic algorithms from unknown binary files and the identify results rely on cryptographic algorithms signature library we established. An efficient and feasible method to implement it is given. Our experiments demonstrate that current software implementations of cryptographic algorithms hardly have any secrecy in the field of software reverse engineering. We could know so much information from binary code, not only the cryptographic algorithms name, but also pinpointing the functions and parameters. The major advantage of this method is that it can recognize cryptographic algorithms and encryption-decryption functions from any unknown executable files effectively and exactly. Hence, it can be used to find malicious code, protect computer system and deal with some other problems in the field of cryptographic algorithms recognition.

REFERENCES

[1] C.Alias. Program Optimization by Template Recognition and Replacement. University of Versailles Saint-Quentin Press. 2005.

[2] Calvet.J, Fernandez.J.M, Marion.J.Y. Aligot: cryptographic function identification in obfuscated binary programs. Proceedings of the 2012 ACM conference on Computer and communications security. pp.169-182, 2012.

[3] Ian Harvey. Cipher Hunting: How to Find Cryptographic Algorithms in Large Binaries. nCipher Corporation Ltd. pp.46-51, 2001.

[4] Jizhong Li, Liehui Jiang. Cryptogram Algorithm Recognition Technology Based on Bayes Decision-making. Computer Engineering. pp.159-163, 2008.

[5] Jizhong Li. Cryptogram Algorithm Recognition Technology Research Based on Similarity Judgement. Information Engineering University. pp.57-68, 2009

[6] Tieming Liu, Liehui Jiang. Researching on Cryptographic Algorithm Recognition based on Static Characteristic-Code. Future Generation Information Technology Conference(FGIT). pp. 140-147, 2009.

[7] Jason L. Wright, Milos Manic. Neural Network Approach to Locating Cryptography in Object Code.Emerging Technologies & Factory Automation (ETFA). 2009.

[8] Igor Skochinsky. Hex-RaysHome/IDA/Technology/IDA/FLIRT. https://www.hex-rays.com/products/ida/tech/flirt/index.shtml, 2013

[9] Ruoxu Zhao, Dawu Gu, et al. Detection and Analysis of Cryptographic Data inside- software. Journal of Lecture Notes in Computer Science. pp.182-196, 2011.

[10] Kwok-Wo Wong. A combined chaotic cryptographic and hashing scheme. Physics Letters. 2003