RAD HAZU. MATEMATIČKE ZNANOSTI
Vol. 25 = 546 (2021): 1-13
DOI: https://doi.org/10.21857/mnlqgc58gy

ANALYSIS OF ENCRYPTION SCHEMES IN MODERN RANSOMWARE

Roderik Ploszek, Peter Švec and Patrik Debnár

Abstract. In the past few years, activity of ransomware increased. As new variants and families of ransomware are developed, security systems have to keep up. Well designed encryption system is at the heart of ransomware and even a small mistake in the algorithm can break it. This paper analyzes 10 ransomware samples from various families. The goal of the analysis is to describe encryption schemes used in current ransomware. This includes key generation and storage, symmetric and asymmetric ciphers and their chosen implementation.

1. Introduction

This paper focuses on cryptographic primitives used by modern ransomware. Ransomware is a special kind of malware that prevents users from using the computer until the user pays a ransom. This can be done in two ways. First one, locker ransomware, locks the user out of the device. The user may continue using the device only after he pays the ransom and the attacker unlocks his device.

The second type, crypto ransomware, is more dangerous – it encrypts all valuable user files, including documents, images, videos, etc. Files can be returned into their original state only if user pays the ransom and if the attacker provides a decryption tool.

Ransomware requires two conditions for correct operation: firstly, a connection to the Internet in order to communicate with the command server (this is not required for all ransomware) and secondly, a method of anonymous payment so that the attacker cannot be easily identifiable. The second condition was fulfilled with the development and expansion of cryptocurrencies such as Bitcoin [19]. This significantly accelerated the spread of ransomware. According to 2019 IOCTA report [12], ransomware was rated top threat of 2019.

2020 *Mathematics Subject Classification.* 68M25, 94A60.
Key words and phrases. Ransomware, computer security, encryption.

Motivation. The actual way the ransomware encrypts and transforms the files can be very important. Incorrect implementations of cryptographic algorithms can lead to situations where the files are damaged and no longer recoverable. In other situations, the implementation can be exploited and files can be decrypted without a key (or the key can be derived from some known information).

The main theme of this paper are encryption algorithms used in modern ransomware. We analyze the samples and describe used encryption schemes. These schemes are then compared and analyzed for any weaknesses. Samples were chosen from last two years, so they represent the newest trends used by malware authors. This allows us to compare our findings to related works that analyzed older families of ransomware in a similar manner.

Contribution. Our main contribution is the analysis of current (2019–2020) usage of cryptographic primitives in ransomware. We provide a complex analysis, including ciphers used, implementations and key generation. Finally, the analyzed set of 10 ransomware samples[1] chosen at random represents different programming languages (compiled C/C++, Go and runtime C#) with different encryption algorithms.

Organization. Our paper is organized as follows. Section 2 presents previous research on malware including current state of the art research outputs. Section 3 introduces methodology used for the analysis. Section 4 introduces notation used in Section 5 that presents schemes that were analyzed. Section 6 presents results of the analysis and Section 7 discusses those results in the context of previous analyses mentioned in related works. Finally, Section 8 sums up the findings.

2. Related work

One of the first ransomware analyses was done by Gazet [14] in 2008, about three years after the ransomware phenomenon emerged. In the analysis, they focused on quality of code, malware functionalities and usage of cryptographic primitives. They analyzed four ransomware families. Many analyzed samples showed various mistakes, such as not using encryption at all and instead changing the access rights of the files, bugs that can destroy files of specific sizes, easily breakable custom cryptographic functions and usage of constant seed for the PRG (pseudo-random generator).

In the last four years, partly intensified by the WannaCry outbreak, interest in ransomware grew rapidly. Scopus analytics [11] shows 12 documents containing word ransomware in the title in 2015 and 252 documents in year 2017.

[1] samples are available at `https://github.com/UIM-SEC/ransomware-samples`

Palisse et al. [18] showed that either by monitoring Microsoft's cryptographic API present in Windows operating system or by exploiting weak cipher modes, one can detect or revert actions done by ransomware and get their files back. The success rate was 50%.

Craciun et al. [9] presents trends in ransomware design, including coexistence with service providers on black market, ways of distribution and propagation, types of encryption used including design mistakes. Notable mistakes found are usage of the `rand()` function, incorrect usage of encryption algorithms or bugs that damage the files.

Akbanov et al. [1] presents a detailed analysis of the infamous WannaCry ransomware that infected hundreds of thousands computers in 2017. The analysis was conducted in an isolated lab environment. In this environment, researchers could easily trace API calls of encryption components and identify functions used to generate keys and encrypt files.

Bajpai and Enbody [4] focused their research on ransomware that uses .NET framework. This type of ransomware was chosen because the decompiled code is much closer to the actual source code of the malware and subsequent manual analysis is easier and faster. Main focus of the study was key generation, but in addition, they analyzed other life stages of ransomware, including delivery and preparation, file enumeration and post-encryption. Their other work [5] is more tightly focused on key generation strategies, where they found empirical evidence that insecure key generation schemes are still present in some types of modern ransomware.

Cicala et al. [8] analyzed .NET ransomware using static and dynamic analysis. They describe four encryption models (symmetric, asymmetric, hybrid and full hard disk encryption). Their analysis focused mainly on key generation and encryption methods with a description of reverse engineering techniques. In conclusion, they formulate a few defense strategies that can be used against malware that include exploiting weak encryption process or crypto API logging that can revert cryptographic operations done by ransomware.

3. Methodology

In this section, we introduce the methodology used to analyze the ransomware samples.

3.1. *Sample Retrieval.*

Samples were retrieved mostly from websites offering malware for analysis, specifically Hybrid Analysis [16] and any.run [2]. One sample was captured live from infected computer. Hashes of the samples were selected based on their capture date so that the set of samples consisted of current ransomware.

3.2. *Analysis Process.* The main goal of the analysis is to find all encryption operations. The first important area is the beginning of the program, where the public key may be loaded or asymmetric key pair may be generated. The second area of interest is the encryption loop, where each file with suitable extension on the file system is encrypted. The encryption algorithm, encryption mode and key generation can be usually analyzed here.

There are multiple strategies that can be used during analysis. The following is a summary of techniques that can be used during malware analysis. For more in-depth information, we recommend specialized literature such as Sikorski and Honig [21].

3.3. *Dynamic analysis.* This strategy requires the ransomware to be executed. As this is potentially destructive operation, it needs to be performed in a safe environment. For this purpose, we have used a Windows 10 virtual machine with Flare[2] distribution. Dynamic analysis runs the malware inside a debugger that allows to step through the ransomware one instruction at a time.

Main advantage of dynamic analysis compared to static one is the ability to view information that is available only at runtime. For example, some ransomware may contain encrypted strings that are decrypted only when the ransomware uses them. However, there are also some drawbacks. Some malware may contain anti-debugging measures that prevent the program from being debugged or change its original behavior making the analysis more challenging.

3.4. *Static analysis.* This strategy is safer than the previous one as it does not involve execution of the ransomware. The sample is imported into disassembler such as IDA Pro [13] or Ghidra [15]. At this point, external library calls can be analyzed (e.g. CryptoAPI) along with strings present in the binary. Strings that are most valuable include hard-coded public keys or error messages which might point to a specific library used or even contain information about position (line number) in the original source code of the library.

In addition to disassembler, a decompiler may also be used. Code generated by a decompiler is different from the original source code of the ransomware, but flow control constructs such loops, branches and switches are easier to analyze in C than in assembly.

There are some further tricks that can be done at this stage. Signature search [17] in IDA Pro allows to search for common compression and encryption algorithms. Function ID in Ghidra can identify library functions from prepared function databases [22]. Plugin Karta [7] for IDA Pro has a similar functionality.

[2] `https://github.com/fireeye/flare-vm`

4. Notation

In this section we provide a description of the notation used throughout this paper. The notation is summarized as follows:

- The symbols A and V denote attacker and victim respectively. In our case, attacker represents a cybercriminal group, responsible for delivering a malicious sample to the victim. Victim, on the other hand, represents a single machine, infected by ransomware.
- Keys are represented as symbol K_n. The index n describes, who is the owner of the mentioned key (attacker or victim) and whether the key is symmetric or asymmetric. For example, K_A is an attacker's symmetric key and $K_{V_{pub}}$ is a public key for the victim.
- Function $Exe(K)$ represents a special process of embedding a key K directly into the executable. For example, output from the function $Exe(K_{V_{pub}})$ is a malicious sample with embedded victim's public key.
- The key generation function is used as $K = Gen()$ when generating a symmetric key and $K_{pub}, K_{priv} = Gen()$ when generating an asymmetric public/private key pair.
- The encryption function is denoted as $Enc(pt, K)$, where pt is plaintext and K is the key used for encryption.
- Each file to be encrypted is represented as f_i, where $i \in \{1, \ldots, N\}$ and N is the total number of files that ransomware selects for encryption.

5. Encryption Schemes

Cracium et al. [9] identified three encryption schemes used by ransomware until the year 2018. During the analysis of the latest samples, we also observed one additional scheme (encryption scheme $\mathcal{B}$). This chapter is devoted to description of all four major ransomware encryption schemes.

First encryption scheme $\mathcal{A}$ is shown in Figure 1 (left). We can see that attacker firstly generates a key pair $(K_{V_{pub}}, K_{V_{priv}})$ for a specific victim and embeds the public key $K_{V_{pub}}$ directly into the malicious executable. When this sample is transferred and executed by the victim, a custom symmetric key K_i is generated for each file f_i. The key K_i is used for encrypting the file content. K_i is then encrypted with victim's public key $K_{V_{pub}}$ and attached to the end of an encrypted file.

As we can see in Figure 1 on right side, in second encryption scheme $\mathcal{B}$, malicious sample is distributed with an attacker's key K_A. Compared to the previous scheme, the victim's key pair $(K_{V_{pub}}, K_{V_{priv}})$ is generated directly on the infected machine. Private key $K_{V_{priv}}$ is encrypted with the key K_A, sent back to the attacker and removed from memory. File encryption process is the same as in scheme $\mathcal{A}$.

Encryption scheme $\mathcal{C}$ (left side of Figure 2) is very similar to scheme $\mathcal{A}$. Main difference is that instead of generating a custom key for each file, one

global symmetric key K is generated. Then, every file is encrypted with the same key.

Last scheme is known as a three-tier trust model. As we can see in Figure 2 (right), malicious executable is shipped with the attacker's public key $K_{A_{pub}}$. Then, victim's key pair $(K_{V_{pub}}, K_{V_{priv}})$ is generated and the private key $K_{V_{priv}}$ is encrypted with attacker's public key $K_{A_{pub}}$. File encryption process is the same as in schemes $\mathcal{A}$ and $\mathcal{B}$, where custom symmetric key is generated for each individual file. This scheme was used by the WannaCry ransomware [1] and can be considered as the most secure one. Main advantage of this scheme is that victims affected by the same binary cannot share their private keys, as they are protected by another asymmetric layer.

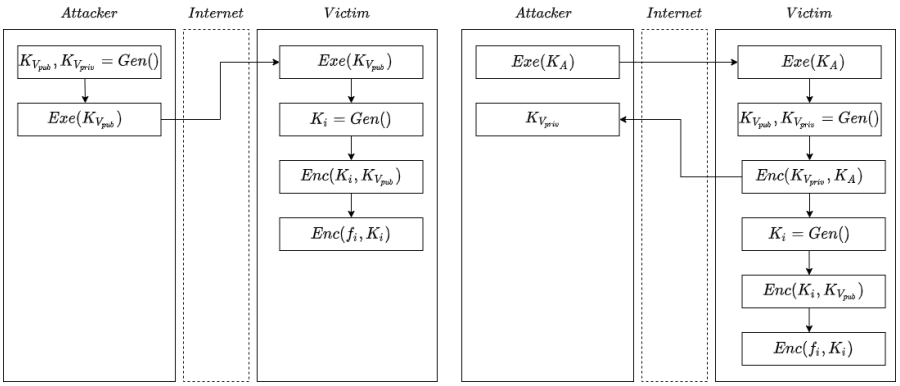

FIGURE 1. Encryption schemes $\mathcal{A}$ (left) and $\mathcal{B}$ (right).

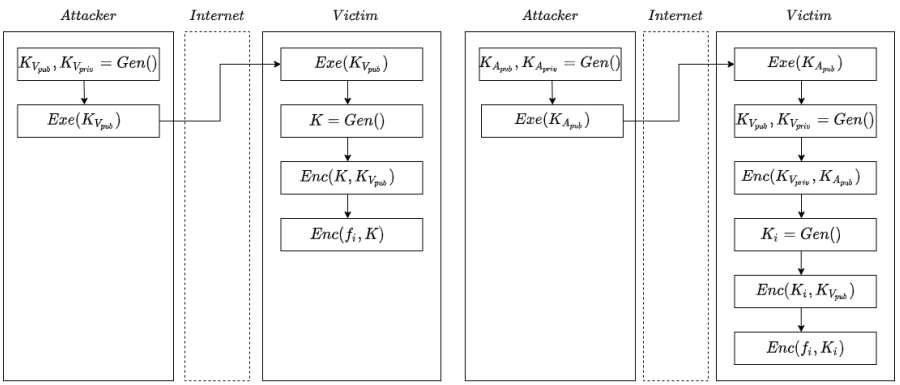

FIGURE 2. Encryption schemes $\mathcal{C}$ (left) and $\mathcal{D}$ (right).

6. Results

In this section we summarize the results we achieved during the analysis process. As stated in the previous sections, we mainly focused on: identification of the encryption scheme, symmetric and asymmetric encryption algorithms and its parameters, key generation functions and implementation details of the cryptographic primitives. We performed an in-depth analysis of ten samples intercepted during the years 2019 and 2020. It is important to note, that we focused on different families rather than individual variants as malware authors tend to release a new version of their ransomware every few weeks. Therefore two different variants of ransomware from a single family can differ significantly in terms of their cryptographic primitives usage.

Identified encryption schemes can be seen in Table 1. We can see that significant amount of samples use scheme $\mathcal{A}$, where $K_{V_{pub}}$ is generated by an attacker for a specific victim. These findings directly correspond to the IOCTA report from 2019 [12], where they stated, that modern ransomware is targeting private companies instead of a regular citizens. The prevalence of this scheme indicates, that attackers are no longer interested in affecting as many victims as possible, but rather focusing on the specific targets. An interesting fact is that we observed only one sample using the most secure scheme $\mathcal{D}$. Ransomware *Nemty* used the scheme $\mathcal{D}$ with the modification from scheme $\mathcal{C}$, where one global key K is used for encryption. We found global key K usage in three samples (*Nemty*, *Katyusha* and *Phobos*). It has to be noted that in two cases (*Nemty* and *Phobos*), the IV was randomly generated for each file, while on the other hand, the *Katyusha* sample used same key and IV for all files. We also identified one sample using the scheme $\mathcal{B}$ (*GandCrab*). This scheme can be considered as one of the weakest, as $K_{V_{priv}}$ is generated on the victim's machine and sent back to the attacker, encrypted with a known key K_A. Hence if the victim has network logs at his disposal, encrypted files are easily recoverable.

In the case of symmetric ciphers (see Table 2), we can see that most samples utilize AES [10], with key sizes of 128 and 256 bits. In terms of encryption modes, majority of cases used CBC mode with randomly generated IV. Ransomware *Ryuk* is the only one that used zero IV, which can leak information from ciphertext if the same key is used for different files, however this is not the case in scheme $\mathcal{A}$. One special case includes *Snatch* sample, using the OCFB mode. We observed only two samples that used stream ciphers for file content encryption. *GandCrab* used combination of Salsa20 [6], with 256 bit key and RC4 with 2048 bit key. *Clop*, on the other hand used only RC4 cipher with 2048 bit key.

TABLE 1. Encryption schemes observed in the latest samples.

Ransomware name	Encryption scheme
Ryuk	$\mathcal{A}$
Dharma	$\mathcal{A}$
LockBit	$\mathcal{A}$
SamSam	$\mathcal{A}$
GandCrab	$\mathcal{B}$
Clop	$\mathcal{A}$
Katyusha	$\mathcal{A}/\mathcal{C}$
Snatch	$\mathcal{A}$
Phobos	$\mathcal{A}/\mathcal{C}$
Nemty	$\mathcal{C}/\mathcal{D}$

TABLE 2. Symmetric ciphers in modern ransomware and its parameters.

Ransomware name	Algorithm	Mode	IV
Ryuk	AES-256	CBC	Zero
Dharma	AES-128	CBC	Random
LockBit	AES-128	CBC	Random
SamSam	AES-128	CBC	Random
GandCrab	Salsa20, RC4	—	—
Clop	RC4	—	—
Katyusha	AES-256	CBC	Random
Snatch	AES-128	OCFB[1]	Random
Phobos	AES-256	CBC	Random
Nemty	AES-128	CBC	Random

[1] OpenPGP variant of the standard Cipher-Feedback(CFB) mode.

Public-key cryptography is an important part of the encryption schemes in modern ransomware. As we can see in Table 3, solely the RSA algorithm is used nowadays. Key size varies from 1024 bits up to 8192 bits. In our dataset, *Nemty* was the only sample that implemented encryption scheme $\mathcal{D}$, hence it uses two public/private key pairs. In this case, the RSA-8192 is used for $(K_{A_{pub}}, K_{A_{priv}})$ and RSA-2048 for $(K_{V_{pub}}, K_{V_{priv}})$.

During our analysis we also focused on key generation functions, specifically for symmetric and asymmetric keys. As mentioned earlier, since almost all samples have pre-embedded public keys, no key generation function for asymmetric keys is needed. Only two exceptions are *GandCrab* and *Nemty* samples, where both use secure key generation function `CryptGenKey()`. In case of a symmetric key generation, various library functions were used. Most

TABLE 3. Asymmetric ciphers in modern ransomware.

Ransomware name	Algorithm
Ryuk	RSA-2048
Dharma	RSA-1024
LockBit	RSA-2048
SamSam	RSA-2048
GandCrab	RSA-2048
Clop	RSA-1024
Katyusha	RSA-2048
Snatch	RSA-2048
Phobos	RSA-1024
Nemty	RSA-8192/RSA-2048

common functions were `CryptGenKey()`/`CryptGenRandom()` from Windows CryptoAPI. Others include key generation functions from various cryptographic libraries such as `get_random_NZ()` from axTLS library, `rand()` from OpenPGP Go package, etc. During the research we observed two samples (*Katyusha* and *Nemty*) using an insecure C `rand()` function for symmetric key generation.

TABLE 4. Key generation functions in modern ransomware.

Ransomware name	Symmetric key	Asymmetric key
Ryuk	`CryptGenKey`	hardcoded
Dharma	`get_random_NZ`	hardcoded
LockBit	`CryptGenRandom`	hardcoded
SamSam	`RNGCryptoServiceProvider`	hardcoded
GandCrab	`CryptGenRandom`	`CryptGenKey`
Clop	`CryptGenKey`	hardcoded
Katyusha	`rand()`	hardcoded
Snatch	`rand()`[1]	hardcoded
Phobos	`CryptGenRandom`	hardcoded
Nemty	`rand()`	`CryptGenKey`

[1] From `crypto/rand` package, which implements a cryptographically secure random number generator.

We also analyzed the implementation details of the ciphers used. Results can be seen in Table 5. As expected, the Windows CryptoAPI is the most common library. We can also see that adversaries prefer secure implementations from various libraries, as custom cipher implementations are prone to mistakes that can potentially lead to file recovery. Ransomware *Nemty* is the only sample that used custom AES implementation, while the RSA was

implemented securely with Windows CryptoAPI. An interesting sample was *LockBit*, which used multiple AES implementations. In this case, if the victim's machine supported fast AES instruction set, then files were encrypted using these instructions, otherwise, an optimized Rijndael implementation was used.

TABLE 5. Cryptography implementation in modern ransomware samples.

Ransomware name	**Implementation**
Ryuk	Windows CryptoAPI
Dharma	axTLS embedded SSL
LockBit	Windows CryptoAPI + AES-NI instruction set + Optimized Rijndael
SamSam	.NET System.Security.Cryptography
GandCrab	Windows CryptoAPI
Clop	Windows CryptoAPI
Katyusha	OpenSSL
Snatch	OpenPGP Go package
Phobos	Windows CryptoAPI + syslinux RSA implementation
Nemty	Windows CryptoAPI + custom AES implementation

7. Discussion

In the previous section, we presented our analytical findings. Compared to the most similar work by Craciun et al. [9], we can notice that trends have changed in the last few years. In their work, authors identified significant amount of ransomware samples, that used hardcoded symmetric key or used insecure C `rand()` function for key generation (which can be bruteforced in reasonable time). We observed this weakness only in the minority of samples and in most cases the key generation method was implemented properly. Another weakness present in the older samples was using ECB mode or CBC mode with IV equal to zero. We found no samples using the ECB mode and only one sample used the CBC mode with zero IV, however, the scheme should be secure as the symmetric keys are not being reused. Older ransomware samples also used to manually implement the cryptographic primitives, or even create a custom cipher, potentially resulting in an exploitable scheme. However, we noticed that only one current sample used a custom AES implementation. Generally, from these observations, we can state that malware authors are slowly getting better at cryptography, resulting in a more secure scheme implementations.

An interesting aspect to study would be the key management in modern ransomware, i.e. storage and protection of keys. Bajpai et al. [3] studied this topic and showed that depending from the scheme and its implementation, symmetric and asymmetric keys can be potentially extracted directly from the memory during the encryption process, subsequently enabling the file recovery.

Since the powerful quantum computers are becoming more and more closer to reality, it also presents a problem for ransomware authors. It is a known fact, that with a reasonably powerful quantum computer, all the current public-key algorithms would be easily breakable. This means that the schemes presented in this paper become vulnerable, as they heavily rely on the security of RSA. However, efficient factorization algorithms for quantum computers [20] currently exist, hence we could extract a private key from a victim's public key (which is known). With an observed trend, that ransomware authors are getting more skilled in cryptography, we can expect adoption of a new post-quantum algorithms in a near future.

8. Conclusion

This paper presented an analysis of cryptographic functions used in modern ransomware. Ten selected samples from various families and programming languages were analyzed using static and dynamic analysis. Analysis identified four separate encryption schemes. Scheme where the key was generated by the attacker for each infected computer was most prevalent. Most of the encryption algorithms used in the samples had better security when compared with older samples analyzed in related works.

As the area of malware is always evolving, there are many topics that would benefit from further research. One of them is ransomware detection. Knowing the libraries and encryption schemes, one can better generalize how ransomware behaves on the system. Thanks to this, better detection tools may be developed.

Another topic is the analysis itself. There are some operations in the analysis that can be automated. Currently, tools can identify which libraries are used by the malware sample, but sometimes the detection may fail. An interesting project would be to create a system that could automatically detect encryption algorithms, modes and schemes. Parts of such system could be also used for detection of ransomware.

Acknowledgements.

This research was sponsored by Slovak Republic under grants VEGA 1/0159/17 and APVV-19-0220.

References

[1] M. Akbanov and V. Vassilakis, *WannaCry Ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms,* Journal of Telecommunications and Information Technology **1** (2019), 113–124.

[2] ANY.RUN, *Interactive Online Malware Analysis Sandbox* [Online], Available at: `https://app.any.run/`.

[3] P. Bajpai and R. Enbody, *Attacking key management in ransomware,* IT Professional **22 (2)** (2020), 21–27.

[4] P. Bajpai and R. Enbody, *Dissecting .NET ransomware: key generation, encryption and operation,* Network Security **2020 (2)** (2020), 8–14.

[5] P. Bajpai and R. Enbody, *An empirical study of key generation in cryptographic ransomware,* in: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Dublin, 2020, pp. 1–8.

[6] D. J. Bernstein, *Salsa20 specification. eSTREAM Project algorithm description,* 2005, `http://www.ecrypt.eu.org/stream/salsa20pf.html`.

[7] Check Point Software Technologies Ltd. *Karta* [Online], Available at: `https://github.com/CheckPointSW/Karta`.

[8] F. Cicala and E. Bertino, *Analysis of encryption key generation in modern crypto ransomware,* in: IEEE Transactions on Dependable and Secure Computing, 2020.

[9] V. C. Craciun, A. Mogage and E. Simion, *Trends in design of ransomware viruses,* in: J.-L. Lanet and C. Toma C. (eds), Innovative Security Solutions for Information Technology and Communications, SECITC 2018, Lecture Notes in Comput. Sci. **11359**, Springer, Cham, 2019, pp. 259–272.

[10] J. Daemen and V. Rijmen, *The Design of Rijndael. AES – The Advanced Encryption Standard*, Springer-Verlag, Berlin, 2002.

[11] Elsevier. *Analyze search results* [Online]. Scopus. Available at: `https://www.scopus.com/term/analyzer.uri?sid=56a6bc5b10958348ce1b177abf5dd58d&origin=resultslist&src=s&s=TITLE-ABS-KEY\%28ransomware\%29&sort=plf-f&sdt=b&sot=b&sl=25&count=845&analyzeResults=Analyze+results&txGid=7a910e8c6b468a1518301b0710a09aa3`.

[12] Europol, *Internet Organised Crime Threat Assessment (IOCTA)* [Online], Available at: `https://www.europol.europa.eu/activities-services/main-reports/internet-organised-crime-threat-assessment-iocta-2019`.

[13] Hex Rays, *IDA Pro - Hex Rays* [Online], Available at: `https://www.hex-rays.com/products/ida/`.

[14] A. Gazet, *Comparative analysis of various ransomware virii,* Journal in Computer Virology **6** (2010), 77–90.

[15] *Ghidra* [Online], Available at: `https://ghidra-sre.org/`.

[16] Hybrid Analysis, Free Automated Malware Analysis Service - powered by Falcon Sandbox [Online], Available at: `https://www.hybrid-analysis.com/`.

[17] M. Gothe *IDA Signsrch* [Online], Available at: `https://github.com/nihilus/IDA_Signsrch`.

[18] A. Palisse, H. Le Bouder, JL. Lanet, C. Le Guernic and A. Legay, *Ransomware and the Legacy Crypto API,* in: F. Cuppens, N. Cuppens, J.-L. Lanet and A. Legay (eds.), Risks and Security of Internet and Systems, CRiSIS 2016, Lecture Notes in Comput. Sci. **10158**, Springer, Cham, 2017, pp. 11–28.

[19] R. Richardson and M. North, *Ransomware: Evolution, mitigation and prevention,* International Management Review **13** (2017), 10–21.

[20] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,* SIAM Rev. **41** (1999), 303–332.

[21] M. Sikorski and A. Honig, Practical malware analysis: The hands-on guide to dissecting malicious software, No Starch Press, San Francisco, 2012.
[22] ThreatTrack Security, *Ghidra Function ID dataset repository* [Online], Available at: https://github.com/threatrack/ghidra-fidb-repo.

Analiza shema šifriranja u suvremenom ransomwareu

Roderik Ploszek, Peter Švec i Patrik Debnár

Sažetak. U posljednjih nekoliko godina povećala se aktivnost ransomwarea. Kako se razvijaju nove inačice i familije ransomwarea, sigurnosni sustavi moraju to pratiti. U srcu ransomwarea je dobro osmišljen sustav šifriranja i čak i mala pogreška u algoritmu može ga slomiti. Ovaj rad analizira 10 uzoraka ransomwarea iz različitih familija. Cilj analize je opisati sheme šifriranja koje se koriste u trenutnom ransomwareu. To uključuje stvaranje i pohranu ključeva, simetrične i asimetrične šifre i njihovu odabranu implementaciju.

Roderik Ploszek
Institute of Computer Science and Mathematics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava, Slovakia
E-mail: roderik.ploszek@stuba.sk

Peter Švec
Institute of Computer Science and Mathematics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava, Slovakia
E-mail: peter.svec1@stuba.sk

Patrik Debnár
Institute of Computer Science and Mathematics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava, Slovakia
E-mail: xdebnar@stuba.sk

Received: 30.10.2020.
Revised: 1.3.2021.
Accepted: 16.3.2021.