Forensic Science International: Digital Investigation 46 (2023) 301572

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

Evaluation of live forensic techniques, towards Salsa20-Based cryptographic ransomware mitigation

Luis Fernandez de Loaysa Babiano, Richard Macfarlane*, Simon R. Davies

School of Computing, Engineering & the Built Environment, Edinburgh Napier University, Edinburgh, UK

ARTICLE INFO

Article history:
Received 22 September 2022
Received in revised form
6 April 2023
Accepted 26 May 2023
Available online xxx

Keywords:
Ransomware
Stream encryption
Memory forensics
Malware analysis

ABSTRACT

Ransomware has been established as one of the largest current threats to organisations, small businesses, governments, and individuals alike. The appearance of cryptocurrencies and the enhancement of encryption key management schemes increased the capacity of this malicious software to compromise the victim's data and demand ransom payments. The variety of ransomware families and their continued evolution make the task of detecting and mitigating these attacks extremely difficult. Current ransomware typically uses complex multi-layer hybrid encryption methods, which cannot be mitigated using conventional methods such as attacking the encryption keys directly. Recent studies have shown that when using live forensic techniques, it is possible to find the ransomware data encryption keys in the volatile memory of an infected machine while the ransomware is being executed, in a form of a side-channel attack. However, the related work in the field does not address the most recent cryptography typically now used by ransomware, including stream ciphers such as Salsa20. Related work has also not fully explored the typical use of unique keys per victim's file which is now common with current ransomware. The work described in this paper reproduces these latest cryptographic management techniques being used and explores methods for both, Salsa20 key extraction from memory, and one key per file ransomware encryption key recovery. The methods have been evaluated against recent real-world ransomware samples with various victim file data sets. The method has been shown in some cases to successfully recover over 90% of Salsa20 key and nonce pairs from volatile memory, which in turn have been used to decrypt victim files to validate the extracted pairs. This method could facilitate the recovery of victims' files without the need for paying a ransom and bypasses the complex hybrid encryption methods typically used by current ransomware. The findings from the experiments show that it is possible to use live memory forensics to extract multiple ransomware symmetric encryption keys during execution, and then use these to successfully decrypt a large percentage of the victim's encrypted files without requiring the master key. The developed method could be used to help recover from the most advanced current ransomware attack and can prove useful when developing new cryptographic ransomware mitigation techniques.

1. Introduction

Ransomware ranks high among current cyber threats around the world, with institutions such as Interpol classifying it as a threat that requires the same international collaboration as currently exists with fighting terrorism, mafia groups, and human trafficking (Interpol News, 2021). The first cryptographic ransomware appeared in 2013 and has become the dominant form of ransomware. Crypto-ransomware typically encrypts a selection of a victim's data files and demands a ransom in exchange for the decryption of the data. If the ransom is paid, a decryption key or tool is provided allowing the recovery of the files.

The crypto-ransomware threat model and associated structured attacks have been aligned to attack models such as with the Cyber-Kill-Chain (CKC) based taxonomy detailed in (Dargahi et al., 2019). This maps ransomware attacks to the six of the seven Lockheed Martin CKC stages: (1) Reconnaissance, (2) Weaponization, (3) Delivery, (4) Exploitation, (5) Installation, (6) Command and Control, and (7) Actions on Objectives. Ransomware planning on how the attack is carried out including the payload delivery, encryption,

* Corresponding author.
E-mail address: R.Macfarlane@napier.ac.uk (R. Macfarlane).

https://doi.org/10.1016/j.fsidi.2023.301572

and defence evasion is aligned with the Weaponization stage. The Delivery stage includes the payload delivery via methods such as phishing, and the Exploitation phase is concerned with the initial compromise of the target such as with a specific exploit. The Installation phase includes the ransomware payload execution which performs various malicious tasks, including victim file encryption and with recent strains additionally, exfiltrating victim data (McIntosh et al., 2021). The Command and Control stage can involve communications with attacker controlled servers such as communicating encryption keys or payment information. The last phase, actions on objectives, includes the demand for ransom payment from the victim. In the Weaponization stage, crypto-ransomware can use a variety of different cryptographic management strategies including which types of encryption algorithm should be used to encrypt the victim's files. A range of ransomware behaviour is defined at this stage, including victim file system directories to skip, victim file types to target, the sizes of files to target, and the order files are attacked. The installation stage contains the execution of the ransomware itself and includes the encryption of victim files, with the method and algorithms used from the weaponization phase. These weaponization and installation stages are specific to each crypto-ransomware strain and their methods are continually changing to combat detection and mitigation. Zimba (Zimba et al., 2021a) describes a similar threat model, and in addition categorises ransomware types based on the attack structures used, including categories based mainly on the cryptographic systems used for victim files encryption CAT1-CAT5.

Over the last decade, crypto-ransomware has evolved, adopting more sophisticated cryptographic methods to make its detection and mitigation more difficult (Oz et al., 2021; Ramsdell and Esbeck, 2021). This includes the use of more robust and faster encryption algorithms such as Advanced Encryption Standard (AES) and Salsa20. A recent high-profile example ransomware attack against *Kaseya* was carried out using the *Sodinokibi* ransomware strain (US Dept. of Justice Office of Public Affairs, 2022), which has been documented to use Salsa20 for victim file encryption (Guillois, 2020). *RansomCartel* is another example of a recent ransomware strain that is known to use Salsa20 encryption (Amer Elsad, 2022).

In addition to this, most ransomware typically now uses an individual unique encryption key for each file attacked along with hybrid encryption methods employed for key management, protecting the per-file encryption keys against recovery. This is achieved by using additional asymmetric encryption of the per victim file symmetric encryption keys (Zimba et al., 2021a). A consequence of this additional encryption is that it becomes extremely challenging to recover the original victim's files once the encryption process has been completed. As highlighted recently, research has often focused on the prevention, detection and post-execution ransomware responses, while reactive methods have not been explored by many, especially by those involved in the digital forensics community (Huck and Breitinger, 2022).

This paper builds on previous work applying live forensic methods for ransomware file encryption key recovery (Davies et al., 2020) focusing on stage five of the Cyber Kill Chain. This is where the ransomware has already bypassed security mechanisms and is being executed on the victim's system, encrypting victim files (Dargahi et al., 2019).

- The work explores current ransomware cryptographic methods leading to evaluating a method for Salsa20 encryption algorithm memory artefact extraction.
- Experiments are carried out to evaluate this method for crypto-ransomware per file encryption key recovery from memory, with recovery and validation of multiple per file keys.
- These methods are applied, demonstrating a successful side-channel attack against recent ransomware samples using individual Salsa20 keys per file encryption, as well as successful file decryption for large numbers of files.
- Salsa20 key exposure times in memory are also evaluated for a one key per victim file crypto-ransomware.

These methods aim to bypass the multi-layer and hybrid key management incorporated into current ransomware families with the goal of developing an effective ransomware encryption key recovery method that facilitates the decryption of victim files. Such a method could then be used in mitigation systems, providing victims with a route to file recovery without requiring the payment of a ransom.

The remainder of the paper is organised as follows. Section 2 contains a discussion about related work in the area. Section 3 describes the developed methodology with Section 4 detailing the experiments that were conducted and the results achieved. Section 5 provides a critical analysis of the experimental results and a comparison to similar work in the field. Section 6 contains a discussion of the findings, limitations of the research and possible future work. The conclusions drawn from this research are represented in Section 7.

2. Background and related work

This section introduces and discusses the background of crypto-ransomware at the time of writing. Literature related to this work is explored specifically concerning crypto ransomware key management and cryptographic artefact retrieval using memory forensics. This section also contains a discussion on the Salsa20 encryption algorithm.

2.1. Cryptographic ransomware key management

Due to the changing nature of crypto-ransomware, it can be useful to classify ransomware according to the cryptographic scheme they employ, typically split into single key or hybrid encryption systems (Zimba et al., 2021b). For instance, a single-key attack model uses a symmetric or asymmetric cipher to encrypt the victim's data. Whereas, a hybrid system uses both symmetric and asymmetric ciphers simultaneously. With hybrid encryption, user data is typically encrypted using a symmetric encryption key, and this symmetric key is then encrypted using the public key to an asymmetric key pair, the corresponding private key is never present on the infected machine. It is important to note that in principle the data cannot be recovered without the private key (Bajpai and Enbody, 2020b, 2020c; Zimba et al., 2021b).

Bajpai (Bajpai and Enbody, 2020b, 2020c) and Oz et al. (2021) found that popular symmetric and asymmetric ciphers used by crypto-ransomware are Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) respectively. However, as highlighted by Bajpai (Bajpai and Enbody, 2020c), symmetric ciphers introduce a weakness as the same key is used to both encrypt and decrypt the data and as the encryption key is exposed in memory, it is possible to capture the key at this point and then use it to subsequently decrypt and recover the data. Although, this window is minimised if a unique encryption key is used for each victim file, as they highlight can be the case with recent hybrid ransomware.

Typically, more recent crypto-ransomware has been found to now use this hybrid scheme (Bajpai and Enbody, 2020a, 2020b, 2020c; Genç, 2020; Zimba et al., 2021b), where asymmetric and symmetric ciphers are combined (Moussaileb et al., 2021; Oz et al., 2021; Ramsdell and Esbeck, 2021). These types of attack structures are also the most damaging and difficult to recover from without

paying the ransom (Bajpai and Enbody, 2020a, 2020c; Zimba et al., 2021b). However, these classifications help to understand how crypto-ransomware acts and can point toward how these might be detected or mitigated.

2.2. Reactive mechanisms

Reactive mechanisms are used as a last resort when mitigation and prevention solutions have not halted the attack and the ransomware is executing in the infected system. These often include identifying and recovering encryption keys. Some of these mechanisms also use key escrow, to store recovered encryption keys being used by ransomware, to then attempt to recover the encrypted data (Bajpai and Enbody, 2020b, 2020c; Genç, 2020; Oz et al., 2021). Berrueta et al. (2019) point out two major studies from 2017 that take advantage of the ransomware's need to access random numbers to generate encryption keys in an infected host. The first study proposes controlling the Pseudo Random Number Generator (PRNG) of the cryptographic API in Windows systems. This approach allows the user to control all random numbers generated by the PRNG. Therefore, the encryption keys can be regenerated from these captured random numbers to recover the encrypted files (Kim et al., 2017). The second study presents *PayBreak*, a key escrow solution that hooks the cryptographic API to gain access to the keys generated in an infected host. However, the authors of *PayBreak* noted how malware could intentionally create false encryption keys to fill the vault where the keys are stored (Kolodenker et al., 2017). A problem with both proposals is that ransomware might import its own cryptographic library, such as with its own version of a PRNG (Bajpai, 2020; Bajpai and Enbody, 2020b; Moussaileb et al., 2021; Oz et al., 2021). Moreover, as Genç (2020) points out, if *PayBreak* were to become ubiquitous it would present a new target for cybercriminals as the encryption keys needed for other legitimate operations could be stored in the same vault. After the appearance of *PayBreak*, Genç (Genç et al., 2018) developed *UshallNotpass* which allows or denies access to the PRNG of the cryptographic API by applications based on their legitimacy. This means that only processes that have been whitelisted or certified in some manner can access the PRNG function. For the practical testing, they set a system policy based on locally whitelisted applications so any program that does not belong in this list is denied access to the cryptographic API. *UshallNotpass* is able to mitigate ransomware such as *NotPetya* as it stops its execution before the crypto-API can be used. However, Genç (2020) notes that *UshallNotpass* is susceptible to ransomware hijacking the process of a whitelisted application and being accepted as a legitimate application. For this reason, an improved version *NoCry* was developed which unifies the interceptor and controller of the process to avoid the hijacking. While successful results are produced, a similar problem to that of *PayBreak* or *UshallNotpass* remains unsolved, in that ransomware can build in its own encryption code (Bajpai, 2020; Bajpai and Enbody, 2020b; Moussaileb et al., 2021; Oz et al., 2021). Genç (2020) argues that neither *UshallNotpass* nor *NoCry* was designed to stop all types of ransomware and that these applications are to be used as complementary defence solutions. Furthermore, their implementations require either a local whitelisted database or a method to certify applications adopted by major manufacturers.

Another key escrow-based system is *PickPocket* (Bajpai, 2020). In this solution, a bait file is used to trigger the dumping of keys from the ransomware process which is encrypting the files. Genç (2020) highlights the limitation of such a solution, as any files which were encrypted prior to the bait file being accessed would still be unrecoverable. Also to note, ransomware which encrypts victim files with an asymmetric cipher would bypass *PickPocket* and similar systems as the private key would typically not be available on the victim machine. However, as highlighted by Bajpai, single-key asymmetric encryption tend to be slow and resource intensive resulting in these types of ransomware being much less common (Bajpai, 2020; Bajpai and Enbody, 2020a). Bajpai (Bajpai, 2020; Bajpai and Enbody, 2020b, 2020c) and Genç (2020) highlight that crypto-ransomware using a symmetric cipher to create a unique key for each file encrypted, can make it difficult to capture all keys due to the speed in which these ciphers encrypt data. Therefore, Bajpai (2020) developed *PickPocket* to extract asymmetric keys and those AES keys that are found to be in the same process memory space, identifying the presence of these keys via the existence of their key schedule.

2.3. Extracting keys from memory

Several related studies have demonstrated the retrieval of encryption keys from the volatile memory of target machines using digital forensic methods (Halderman, Schoen, Heninger, Clarkson, Paul, Calandrino, Feldman, Appelbaum, Felten; Hargreaves and Chivers, 2008; Kaplan and Geiger, 2007; Maartmann-Moe et al., 2009). Encryption keys are vulnerable to side-channel attacks as the keys and any supporting artefacts have to be held in memory while the encryption is being performed. Symmetric encryption algorithms are typically used for data encryption, with the same key being used for both encryption and decryption. Retrieving a key during the encryption process allows it to be used to decrypt the encrypted files. Some related work has focused on retrieving keys while ransomware is executing, and using those extracted keys to decrypt data encrypted by the ransomware (Bajpai, 2020; Bajpai and Enbody, 2020c; Davies et al., 2020). Digital forensics and ransomware research around extracting keys from memory are overlapping fields. The method used by Davies (Davies et al., 2020) and around the same time Bajpai (Bajpai, 2020; Bajpai and Enbody, 2020c) seems to be the first work to show successful decryption of compromised ransomware encrypted files using keys extracted from the memory of running ransomware. These approaches focused on AES ransomware key extraction from memory as ransomware around that time typically used AES to encrypt victim files.

Significant recent surveys identify that RSA and AES are the most popular ciphers among crypto-ransomware (Beaman et al., 2021; Humayun et al., 2021; Mohammad, 2020; Moussaileb et al., 2021; Oz et al., 2021), but recent technical analysis does indicate that newer strains are evolving to use faster stream type ciphers (Craciun et al., 2019; Lee et al., 2019; Yuste and Pastrana, 2021). Some versions of the *Petya* ransomware used the Salsa20 symmetric cipher to encrypt the Master Boot Record (MBR) and Master File Table (MFT) of infected machines (Aidan et al., 2018; Fayi, 2018). *Anatova* (Poudyal, 2021), *Darkside* (Makrakis et al., 2021), and *Ranzy Locker* in 2020 (Özarslan, 2021), *Karma* and *BlackMatter* in 2021 (Mundo, 2022; Team, 2021), and *RansomCartel* in 2022 (Amer Elsad, 2022) also use the Salsa20 cipher or a variant to encrypt files of their victims, as well as some others using Chacha20 which is modified version of Salsa20 (Yuste and Pastrana, 2021). Current methods of extracting RSA, and AES keys from memory are well-defined, but these techniques are not able to identify Salsa20 keys and nonces. These existing cipher key recovery methods use key schedule and entropy-related methods. Salsa20 stream ciphers can not be recovered using the same techniques, but a method using the initialisation matrix is proposed in this work. The identification and recovery of Salsa20 keys from memory do not seem to have been explored in the literature and this is the focus of the initial phase of this research, followed by applying the developed method to real-world ransomware in the subsequent phases of this research.

2.4. Salsa20 encryption

Salsa20 is a stream cipher, that uses a key, nonce and counter to generate a key stream which is then aligned to the plain-text stream and used to create an encryption stream the same size as the input plain-text stream. The author defines this as a hash function which combines the key, nonce and counter along with some constant values, to create each 64-byte block of the stream. The block is organised into a 4×4 array which has 32 bytes Salsa20 key, 8 bytes nonce, and 8 bytes block counter (or 'position'), with additional 16 bytes of constants also included. The algorithm then performs addition, XOR, and rotation operations on the 64 bytes block in various rounds to create the Salsa20 stream. The stream can then be used to encrypt the associated plain-text stream, using XOR operations, up to the length of bytes of the plain text with the rest of the stream being disregarded (Bernstein. and Snuffle, 2005).

Bernstein (2008) defines the 32-word array as being built from four constant values of two words each, 16 words making up the 32-byte key, four words making up the nonce, and four words making up the block counter/position. The order they are positioned in the array is fixed, with a two-word constant *0x61707865*, followed by 8 keywords making up the first 16 bytes of the key value, then a constant two-word *0x3320646e*, four words making up the 8-byte nonce, four words for the 8-byte block-counter/position value, a constant two-word *0x79622d32*, the last 8 words of the key value, and a constant *0x6b206574*. These values are in little-endian format. The constant values in ASCII are **expa**, **nd 3**, **2-by**, and **te k** and are the same for every Salsa20 array. These are arranged in a diagonal, in row 1 column 1, row 2, column 2, row 3 column 3, and row 4 column 4, making the complete array as shown in Fig. 1.

3. Methodology

This work builds on the previous related work but focuses on the extraction of encryption keys for the Salsa20 encryption algorithm, which has at the time of writing is an encryption method of choice for many of the recent crypto-ransomware strains. Firstly a method for identifying the Salsa20 array and extracting the key and nonce is evaluated. Subsequently, the method is then evaluated with samples of Salsa20-based crypto-ransomware, while they perform encryption of a mixed file data set. According to the literature the most current ransomware strains now employ a one key per file technique (Bajpai, 2020; Bajpai et al., 2018; Kolodenker et al., 2017), so the ransomware key extraction experiments were designed to include multiple key captures facilitating the evaluation of key exposure in memory during the execution of the ransomware. Samples of crypto-ransomware documented as using unique Salsa20 keys and nonce's for victim file encryption were selected for the experiments. The evaluation of the Salsa20 algorithm was selected for this research due to its increasing use by ransomware developers, possibly due to the enhanced speed of the algorithm (Bernstein. and Snuffle, 2005). Salsa20 is a stream cipher, which encrypts data faster than traditional block ciphers such as some modes of AES (Sharif and Mansoor, 2010). The encryption speed of the Salsa20 algorithm could be leveraged by the ransomware developers as it allows the attack to complete sooner, thus aiding ransomware attacks by avoiding detection (Salvio, 2018). Methods for finding and extracting Salsa20 keys from memory do not seem to have been attempted in any related work.

'expa'	key-dword1	key-dword2	key-dword3
key-dword4	'nd 3'	nonce-dword1	nonce-dword2
counter-dword1	counter-dword2	'2-by'	key-dword5
key-dword6	key-dword7	key-dword8	'te k'

Fig. 1. Salsa20 initialisation matrix in memory.

The research has been divided into two main parts. A method for the identification and retrieval of Salsa20 crypto artefacts including the key and nonce, which would typically be unique for each file encrypted by current ransomware. Followed by the application of the method to specific ransomware samples, including extraction of the encryption keys from the memory of running ransomware and validation of the extracted keys by decryption of files previously encrypted by the ransomware.

3.1. Method for identifying Salsa20 key and nonce pairs in memory

Using the pattern of the structure described in Section 2.4, it should be possible to identify these Salsa20 array artefacts in memory. Once identified and extracted, these Salsa20 keys and nonces can be used to decrypt any data which has been encrypted using those arrays.

An experiment was designed to evaluate the method, based on the previous work on AES ransomware keys discovery (Bajpai, 2020; Bajpai and Enbody, 2020c; Davies et al., 2020). Firstly a synthetic ransomware program which mimics ransomware behaviour, encrypting files using the Salsa20 algorithm, will be developed by the authors and subsequently executed in a virtual environment. During the execution of this program, while the synthetic ransomware is carrying out the encryption, the machine's volatile memory is acquired. The memory capture is then searched in an attempt to identify and extract candidate Salsa20 array structures, from which the encryption keys and nonces can be determined. The found key and nonce pairs will then be used to decrypt the files which have been encrypted, thus validating that the correct encryption keys/nonces have been identified and that the method is viable. The developed synthetic ransomware software will also display the keys as it uses them, so the keys will be 'known' and act as an additional method of validation. The software will be executed for a specific length of time sufficient for the memory to be captured, and also no attempt at key hygiene will be performed by the synthetic ransomware. Typically modern ransomware will attempt to remove encryption keys from memory once file encryption is completed, and so the length of time the keys are in memory may vary but typically will be only until the key has been successfully used to encrypt a victim file, after which, the key is removed from memory (The BlackBerry Research and Intelligence Team, 2019).

3.2. Experiments to evaluate the Salsa20 method with real ransomware

Further experiments were then designed to assess the method's success against a real-world ransomware sample. An air-gapped virtualised environment was used to run a real-world ransomware sample from the *Sodinokibi* family. Using a similar experimental process to the above, to encrypt mixed file data sets and extract and validate Salsa20 cryptography artefacts to allow decryption of the victim's files. The ransomware sample selected uses complex multilevel hybrid cryptography to protect its symmetric Salsa20 file encryption keys by encrypting them with asymmetric encryption using public/private key pairs. A public key is used to encrypt each file's Salsa20 encryption key. The associated private key, required to decrypt the Salsa20 key and thus recover the victim's encrypted file, is never present on the victim's machine and is held until the ransom is paid (Guillois, 2020). This is typical of current ransomware, but the hypothesis is that the side-channel attack on the memory of the running ransomware should be able to

extract Salsa20 artefacts and these can then be used to decrypt the victim's files successfully without the need for determining the asymmetric key pairs.

An initial experiment is carried out to encrypt files, capture the memory and identify Salsa20 keys and nonce values and validate that these can be used to decrypt the files. It is similar in nature to the previous Salsa20 identification method experiment in using a small set of files to evaluate the method, except in this case a real-world ransomware sample is being used to encrypt the files and create the Salsa20 memory artefacts. The keys are not known as the ransomware does not disclose these as our initial experiment software does. Real ransomware may not hold the keys in memory for the duration of the process such as with the synthetic ransomware software, so memory captures will be carried out at time intervals to evaluate the key's exposure to the side channel type attack.

Further experiments are then carried out to assess the method for a larger more realistic victim file data set. A set of 4000 files from the NapierOne mixed file data set are used to represent user data files (Davies et al., 2021) and are added to a standard Windows 10 machine file system. Real-world Salsa20-based ransomware samples are then executed. Ransomware typically ignores some directories, and file types so the files created from the execution of the ransomware are reviewed to assess how many have been encrypted. Again periodic memory captures are carried out and the Salsa20 keys and nonces are identified and validated as being able to decrypt the encrypted victim files. The total number of valid keys is recorded for each memory capture, as well as unique keys found at each capture, to assess the total number of keys found and compare these to the number of files encrypted. When keys are present in memory and how long they are exposed before removal are also recorded.

4. Implementation and initial evaluation

Initial experiments were conducted into the evaluation of the method for identification of Salsa20 encryption artefacts from running memory, and then further experiments in applying this method against real-world ransomware samples. Two experimental environment setups were used to perform these investigations; both using VMware Workstation type 2 hypervisor and Virtual Machines (VM) to isolate the ransomware activity and associated memory, with the second also air-gapped and using additional isolation techniques to ensure a safe and ethically sound environment for ransomware analysis.

4.1. Identifying Salsa20 Keys/nonce's in memory

The experiment was designed to explore the method for identifying Salsa20 keys in memory, using a similar technique to the previous work in extracting AES key schedules (Bajpai, 2020; Bajpai and Enbody, 2020c; Davies et al., 2020; Hargreaves and Chivers, 2008). Simple software to mimic ransomware behaviour was created, which generated Salsa20 keys/nonces and encrypted some files, during its execution, it also displays these keys to the user. This synthetic ransomware is run in the virtual environment and the virtual machine's memory is captured by taking snapshots of the VM to create.vmem files, which contain the VM memory contents. The known keys can then be searched for in the volatile memory.vmem files, by identifying the Salsa20 array pattern defined in Section 2.4. The keys and nonces found can then be validated, by decrypting the associated files and comparing those to the original files.

4.1.1. Experimental setup

The experimental setup comprised of a Windows 10 host running within a virtual machine (1 GB RAM + 1 core of a i7-8700k processor and 30 GB of disk space). The technical specification of the environment are detailed in Table 1.

4.1.2. Salsa20 key extraction tool

A simple tool was created to identify and extract Salsa20 initialisation matrix and the key and nonce pairs from the memory captures. The tool searched for the Salsa20 Key State Matrix matching the pattern of constants, key nonce pair and block counter. The script accepted binary files as input which can be in different formats such as.dmp, .vmem, and.core. The tools matches on the 64 byte Salsa20 initialisation matrix pattern: the ASCII string **'expa'** hexadecimal equivalent *0x65787061* at byte 1 and ASCII **'nd 3'** hexadecimal *0x6e642033* at byte 21, **'2-by'** or hexadecimal *0 × 322d6279*, and at byte 61 ASCII **'te k'** or *0 × 7465206b* (Bernstein. and Snuffle, 2005). The process for identifying and extracting the Salsa20 Key and Nonce values from the memory capture is shown in Algorithm 1.

Algorithm 1. Identify Salsa20 Keys in Memory Capture

Algorithm 1 Identify Salsa20 Keys in Memory Capture

```
while not end of Memory Capture do
    PossMatrix ← Read 64 Bytes
    if PossMatrix = Salsa20 Matrix Pattern then
        Salsa20Key ← PossMatrix[5:20] + PossMatrix[45:60]
        Nonce ← PossMatrix[25:32]
        Keys ← Keys + 1
        if Salsa20Key not in Keystore then
            KeyStore ← Add Salsa20Key and Nonce
            UniqueKeys ← UniqueKeys + 1
            Move 64 bytes forward
        end if
    else
        Move forward 1 byte
    end if
end while
```

4.1.3. Synthetic ransomware file encryption

The behaviour of the synthetic ransomware is that it reads all the files in a specified directory, encrypts each file's content using a unique Salsa20 key and nonce per file and then pauses for a small amount of time before writing the encrypted files to the directory and terminating the process. The program does not overwrite the original files but creates a new encrypted version adding an incremented counter to the name of the created file. Fig. 2 shows this program generating four Salsa20 key and nonce pairs, encrypting four files and terminating the process after 30 s.

4.1.4. Extraction of Salsa20 keys from memory

While the synthetic ransomware is running the memory of the VM was captured and saved to a.vmem file. The Salsa20 identification tool was run then against the memory dump file to identify and extract any Salsa20 keys and nonces and record the number of keys found. Fig. 3 shows the output of the extraction tool and the key and nonce pairs being successfully identified and Fig. 4 shows a key as it is presented in the memory dump.

Table 1
Experimental environment technical details.

Software	Version and additional information
Hypervisor type 2	VMware Workstation 15 Pro - 15.5.6 build-16341506
Python3	Version 3.9, Modules imported: PyCryptodome, Time and Os
findaes	https://github.com/mmozeiko/aes-finder
Operating System	Windows 10 Education build 19041.1415
Virtual Machine	1 processor with 1 cores (i7-8700K), 1024 MB of RAM and 30 GB of disk space.
Hex Editor	HxD version 2.5.0.0
Sodinokibi	Hash: 96dde0a25cc6ca81a6d3d5025a36827b598d94f0fca6ab0363bfc893706f2e87

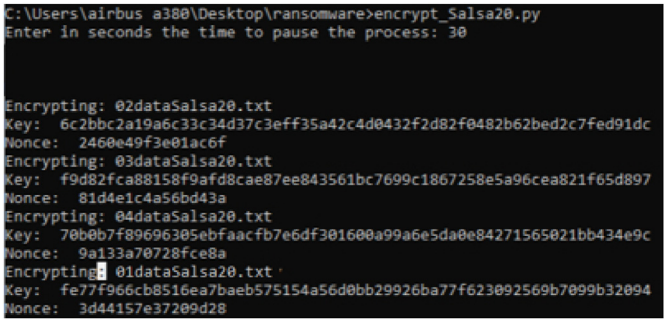

Fig. 2. Synthetic Ransomware Encrypting Files and Salsa20 Key and Nonce pairs used.

4.1.5. *Validation of found Salsa20 keys and nonces*

To validate the extracted keys, the files encrypted by the synthetic ransomware were decrypted using the Salsa20 keys and nonces found by the extraction tool. The decrypted files were then compared to the original files. When using the synthetic ransomware program, the keys are known as the program displays them, so the encrypted files can be easily matched to the appropriate keys and nonces. With real-world ransomware, typically the nonce is appended to the encrypted file in plain text along with the asymmetrically encrypted Salsa20 encryption key (Guillois, 2020; Hasherezade, 2021). If this is the case, for larger scale experiments, or experiments using real ransomware, the nonce should be able to be used to match the extracted Salsa20 key and nonce to the appropriate encrypted files for decryption. The decryption tool outputs the data in plain text and hexadecimal to the console and also writes the file to the directory. Fig. 5 shows a comparison of one of the files *2data_enc_Salsa20.txt* after decryption and the original file *02dataSalsa20.txt* encrypted in stage 1 and in Fig. 6 the hashes of these files to verify their integrity and similarity in bytes before and after encryption. This confirms that the files are the same, validating the decryption process.

4.2. *Evaluation of method for real-world Salsa20-based ransomware*

These experiments were designed to evaluate the method for identifying Salsa20 keys in memory against a typical current ransomware sample which uses Salsa20 to encrypt victim files. As with most modern ransomware strains, multiple, unique, Salsa20 keys/nonces are used to encrypt the files.

A similar process to the initial experiments was followed with the real-world ransomware strain replacing the synthetic ransomware used previously. This sample creates and uses a unique Salsa20 key/nonce to encrypt each victim file, and targets a range of file types on the victim file system where it is run.

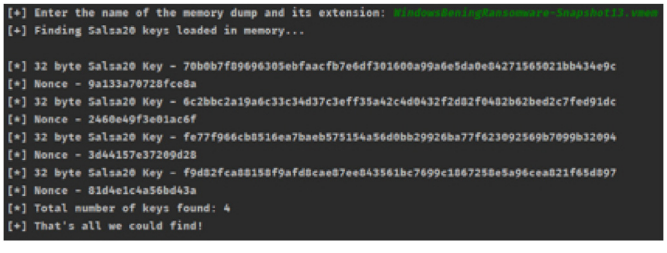

Fig. 3. Key and nonce pairs identified in memory.

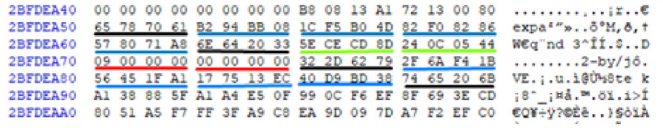

Fig. 4. Salsa20 initialisation matrix in memory.

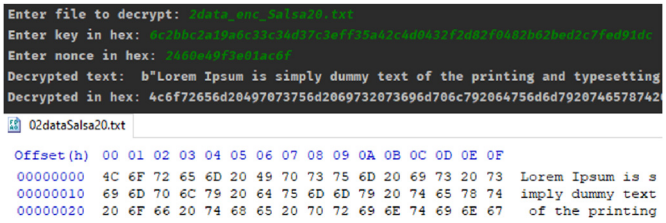

Fig. 5. Validate keys by decryption of files.

4.2.1. *Ransomware sample*

Sodinokibi, also sometimes referred to as REvil, was identified from the literature as using Salsa20 for victim file encryption along with a complex hybrid cryptographic scheme to protect the Salsa20 victim file encryption keys (Tiwari and Koshelev, 2019). It was first identified in 2019 and was one of the most pervasive ransomware of 2021 (Blog, 2022). Sodinokibi uses a unique Salsa20 key to encrypt each of the victim's files after infection (Guillois, 2020). It also uses a range of asymmetric and symmetric encryption algorithms; orchestrating them to avoid leakage of relevant information

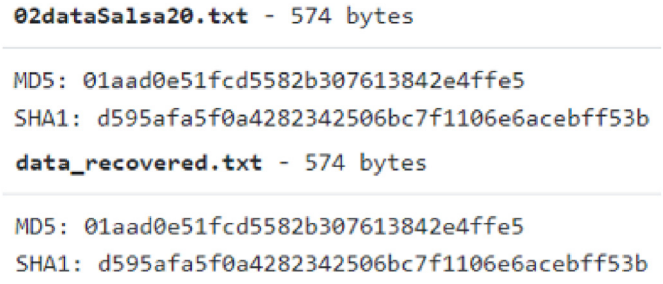

Fig. 6. Comparing file hashes to verify integrity of the files.

and to ensure that the files can only be recovered with one of the two master keys. These master keys are never present on the victim machine which in normal circumstances means that decryption of these files would be impossible without paying the ransom to acquire these keys. From a technical analysis of the specific ransomware, the encryption of files appears to be performed in a multi-threaded manner with the probable aim of making the file encryption process as fast as possible (Tiwari and Koshelev, 2019). This could result in more than one set of keys being in memory at the same time, but perhaps would not be in memory for as long as they would be if the process was serialised. However, as mentioned previously, other factors such as the size of the victim files would also affect the key exposure time.

The following is a summary of the cryptography used by the Sodinokibi ransomware strain. At first, the ransomware creates a per-file public and private key and a session key. It then uses the private key and the session public key with the Elliptic-Curve Diffie-Hellman (ECDH) algorithm to generate a file encryption key which is used to derive a Salsa20 key and nonce pair, unique to each victim file, that is used to encrypt the data content of each file. This process is shown in Fig. 7.

After encryption of the content of the victim file, the ransomware process appends metadata to the encrypted file including the encrypted Salsa20 key, encrypted by two separate asymmetric master keys, the nonce in plain text, as well as some configuration parameters, along with a CRC value and constant value (Guillois, 2020). While reversing the asymmetric encryption of the Salsa20 keys is impossible, given that neither of the two private master keys is ever present on the victim machine, a side channel attack should be able to capture the Salsa20 keys directly from the volatile memory while the ransomware encrypts the files, as demonstrated in the earlier experiments.

4.2.2. Experimental setup

The machine configuration for these experiments was changed so that the test machine was air-gapped, with the ransomware sample being delivered to the air-gapped machine via a USB memory stick. Any files for further analysis were moved back from the air-gapped machine to an analysis machine using the same technique. The configuration of the air-gapped machine again had a Windows 10 OS running within a VM (1 GB RAM + 1 core of an i7-8700k processor and 30 GB disk).

The experiment is carried out with the real-world ransomware sample running within the VM with memory captures from the VM being taken at time intervals during the period that the ransomware is encrypting the victim files. Similar to the previous experiment, Salsa20 encryption artefacts are identified by searching the memory captures using the new method explored in the initial experiment. Unlike the previous experiment, the victim file Salsa20 encryption keys and nonces are not known, requiring that for the found keys and nonces to be validated, the associated encrypted files must first be identified, and then decryption attempted.

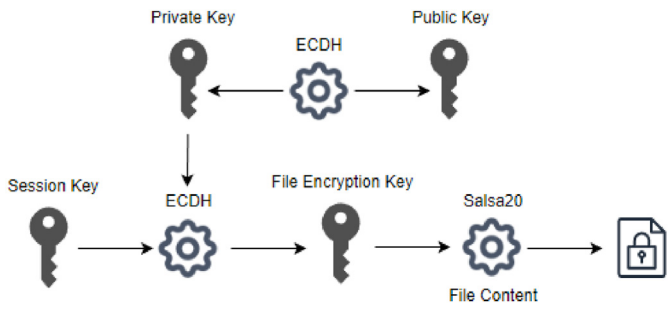

Fig. 7. Sodinokibi key generation.

In the initial experiments with the real-world ransomware sample, a small number of files were used, to assess the viability of the Salsa20 identification method, and to explore the decryption process. This decryption process is often not as straightforward as it was with the synthetic ransomware program, as real-world ransomware-encrypted victim files typically have a range of additional metadata appended to them. Also, sometimes, parts of the original file are removed, such as the header sections, and encryption may also be applied to portions of the victim's files. Once a robust decryption process was developed during the small-scale initial experiment, this was then automated, and larger-scale experiments on a victim file data set made up of 4000 files were carried out. This data set was created from the NapierOne dataset (Davies et al., 2021) and contained mixed file types.

4.2.3. Sodinokibi ransomware execution to encrypt victim files (Salsa20 key recovery. Step 1)

To encrypt the victim file data set, the Sodinokibi ransomware sample was executed on the VM, hosted on the air-gapped machine. For the initial experiment, only three files were used to explore the ransomware implementation, along with the method for decryption. The command to execute the ransomware, with flags and parameters:

C:\Sodinokibi.exe -path files.

The ransomware reads the files in the directory specified after the *-path* argument and creates encrypted files with metadata appended to them, overwriting or deleting the original files. The ransomware is run in default mode, rather than the faster partial encryption mode which is also available.

To evaluate the new Salsa20 key identification and extraction method when executing the real-world Sodinokobi ransomware sample, the same three-stage experiment was performed in the air-gapped experimental environment. Furthermore, a data set of files to be encrypted was created on the virtual machine. These target files were sourced from a publicly available dataset of mixed files (Davies et al., 2021). Initially, only three files were used in the target data set in order to confirm the main experimental steps of key extraction and validation of decryption functioned correctly, prior to moving onto a larger data set of victim files.

4.2.4. Extract Salsa20 keys from memory (Sodinokibi ransomware Salsa20 key recovery. Step 2)

While the ransomware is executing, the memory of the VM was captured at 10-s intervals; during this period the ransomware encrypts the victim files. The Salsa20 Identification tool was then run against the memory captures to identify and extract any Salsa20 keys and nonces, and keep a count of keys found and unique keys. Successful extraction of Salsa20 keys and nonces were found for the memory captures at 20, 30 and 40 s after the start of the ransomware execution and are shown in Figs. 8–10. Two key and nonce pairs were found in the memory capture at the 20-s interval, and then a new pair is additionally found at the 30-s interval, with the original first two found keys still exposed in memory. At the 40-s interval, only one key remains in memory, this being the last of the three previously found keys. Presumably, the first two keys being used to encrypt the first two files are then subsequently being removed from memory after a period of time. This aligns with technical analysis that has been published on this ransomware strain where the researchers claim that the keys are being removed from memory after each file has finished being encrypted (The BlackBerry Research and Intelligence Team, 2019).

Also pointed out by the literature, the appended data at the end of the files contains the nonce used to encrypt them, as well as

```
[*] 32 byte Salsa20 Key - 232371c7f5194c40f2c7696b4e48604439a8da6a32290044f4dd1b3a387e00f4
[*] Nonce - 3d61027bb8265ce8
[*] 32 byte Salsa20 Key - edd8a9f8e446c41b312649df34f5237740db6b2a5923a5a44300389c9c3d3f54
[*] Nonce - 094f60cabbcfb4c9
```

Fig. 8. Identification of Salsa20 key and nonce pairs at 20 seconds.

```
[*] 32 byte Salsa20 Key - 232371c7f5194c40f2c7696b4e48604439a8da6a32290044f4dd1b3a387e00f4
[*] Nonce - 3d61027bb8265ce8
[*] 32 byte Salsa20 Key - edd8a9f8e446c41b312649df34f5237740db6b2a5923a5a44300389c9c3d3f54
[*] Nonce - 094f60cabbcfb4c9
[*] 32 byte Salsa20 Key - 6d38d2f2d0318e53712156ad06f6813349be934a83b15f723f039a82fa3f2fb9
[*] Nonce - 1c5cd305932039f5
```

Fig. 9. Identification of Salsa20 key and nonce pairs at 30 seconds.

```
[*] 32 byte Salsa20 Key - 6d38d2f2d0318e53712156ad06f6813349be934a83b15f723f039a82fa3f2fb9
[*] Nonce - 1c5cd305932039f5
```

Fig. 10. Identification of Salsa20 key and nonce pairs at 40 seconds.

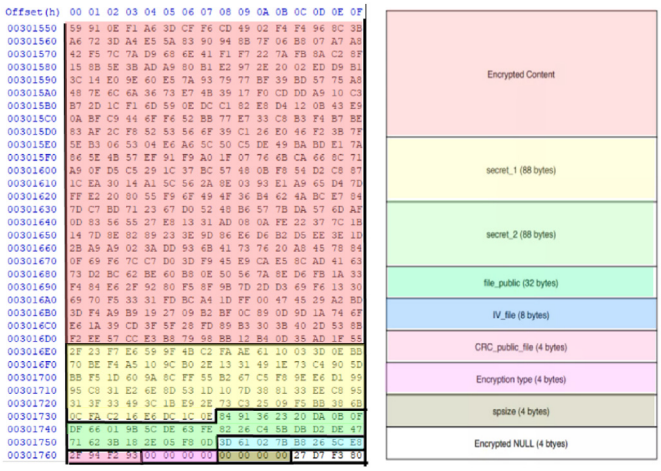

Fig. 11. Sodinokibi encrypted file content and metadata.

some other metadata information related to the ransomware and encryption keys (Guillois, 2020; Hasherezade, 2021).

4.2.5. Validate Salsa20 keys and nonces found in memory (Sodinokibi ransomware Salsa20 key recovery. Step 3)

To validate the keys identified from the captures, the same process of decrypting the files encrypted by the ransomware, and then comparing them to the original files, was followed. For the Sodinokibi encrypted files, the keys used for each file encryption are not known. As a result, the extracted keys and nonces have to be matched to the corresponding encrypted files. Alternatively, some type of brute force decryption would need to be undertaken where all found keys were tested against every file until successful decryption was carried out. As discussed previously, in this case, the nonce should be able to be used to match the Salsa20 key and nonce pair to the appropriate encrypted files for decryption via the appended metadata by Sodinokibi. After analysis of the encrypted files and the appended metadata, the matching nonces in plaintext for the three found in the memory captures were located in the metadata of encrypted victim files. Details of the 232 bytes of appended metadata are discussed in Sodinokibi technical papers such as (Guillois, 2020). The encrypted files and their appended metadata were analysed using a hex editor to validate the 232 bytes of metadata and the individual items were consistent for all three files. An example is shown in Fig. 11 which highlights the 8-byte nonce used for encryption of the file 0307- pptx.pptx.gp7k2 in the initial Sodinokibi experiments and can be located 24 bytes from the end of the metadata, or as a 208-byte offset from the beginning of the metadata (from the end of the encrypted file's data).

The last 4 bytes of the metadata are a 4-byte NULL word which has been encrypted using Salsa20. The 4 bytes are 0×00 values encrypted with the same Sodinokibi key and nonce used to encrypt the file contents.

A decryption tool was created, to first match a nonce identified in memory to a Sodinokibi encrypted file via the plain text nonce in its metadata, then to decrypt the victim file's encrypted content.

The 232 bytes of metadata were removed, with the encrypted file contents matching in size to the original files in all three cases. The decryption of the file contents was then attempted with the Salsa20 key and nonce matched via the previous process and the output compared to the original files. However, the file contents did not match any of the three files. The last 4 bytes of the encrypted file's metadata were extracted and decryption was successful, with the output 0×0000 as expected. This pointed towards the key and nonce being those used to encrypt the file, but perhaps some other step has been added to the encryption process.

In previous work (Davies et al., 2020), file headers and trailers had to be manipulated during the decryption process, but in this case, the file sizes including these match between original and encrypted files, indicating that the encrypted files contain header/footer data of at least the same size. Several manual manipulations of the data were made, such as reversing the data before decryption and altering the headers and footers, but none were successful. To check for Salsa20 encryption stream having been aligned to a different offset, rather than the first byte, a script was created to try decryption incrementally starting at each byte in the file, with the file being rotated by adding a byte from the start to the end of the file after each attempt; thus attempting decryption of the entire file contents starting at a different byte each time. The script matched the bytes decrypted at each offset against the first 25 bytes of the original file including the file header values. For example, a JPG file would include the signature 0xffd8ff and the 19 bytes following. It was found that the decryption back to the original files was accomplished successfully with an offset of -64 bytes or 64 bytes from the end of the file. Thus, the first byte of the file needs to be aligned with the 65th byte of the Salsa20 stream to perform the decryption successfully.

This offset was added to the decryption tool, and the extracted keys and nonces were validated. The decryption of the three files encrypted by the Sodinokibi ransomware was successful using the Salsa20 keys and nonces found by the extraction tool and the method above and was an exact match when compared to the original files. Fig. 12 shows the hash of the original and decrypted files to verify their. This encryption/decryption offset does not seem to be documented in any other academic literature or technical

```
0307-pptx.pptx - 3151496 bytes
MD5: 33a16136aaeac99def2388ab24f045cf
SHA1: b5368089a6bb347bb78469f35e661ef4bebdd754
recovered_0307-pptx.pptx - 3151496 bytes
MD5: 33a16136aaeac99def2388ab24f045cf
SHA1: b5368089a6bb347bb78469f35e661ef4bebdd754
```

Fig. 12. Verifying integrity of files after decryption.

papers. The process of decryption is outlined in Algorithm 2.

Algorithm 2. Decrypt Files with Extracted Keys and Nonces

Algorithm 2 Decrypt Files with Extracted Keys and Nonces

```
Read Encrypted File
if Nonce in Enc File Metadata then
    Enc Content ← Enc File less 232 Bytes of Metadata
    Salsa20 Stream ← CreateStream(Key + Nonce)
    Align Enc Content to 65th Byte of Salsa20 Stream
    Decrypted Bytes ← Decrypt(Enc Content)
    Write Decrypted File ← Decrypted Bytes
else
    Read Next File
end if
```

5. Evaluation

5.1. Sodinokibi large scale key recovery

The Salsa20 extraction method was shown to be able to identify and extract the encryption key and nonce pairs from the cryptographic ransomware sample's memory while it was executing. The decryption tool could then validate the keys and nonces as being able to successfully perform decryption, and thus recover the original victim files.

To be able to determine how many keys could be successfully extracted using this method, and also to produce a timeline of when these might be found in memory, the same experiment was run with a significantly larger data set of 4000 files of mixed types and sizes, with a total size of 2.7 GB. These were sourced from the NapierOne mixed file data set which has been created from real files (Davies et al., 2021) and aims to represent victim user's data files in a data set of similar construction to other researcher's (Berrueta et al., 2020; Kolodenker et al., 2017). Table 2 shows the details of the file types, minimum and maximum file sizes, and the number of each file type contained in the 4000 file data set. These file types were selected as they typically represent files that are targeted by crypto-ransomware, and as the focus was on testing the method while files were being encrypted, these should trigger the encryption. The mixed file data set was added to a Windows 10 file system, under a single directory, which could be specified for this particular ransomware sample. The Windows 10 file system contained a total of 88,265 files and the total size of the file system was approximately 22 GB. Note these were not targeted by the ransomware during the experiment, as the ransomware sample under test was specifically targeted at the 4000 file data set.

After some initial experimentation, a 10-s interval between memory captures starting at 5 s was decided on, with the same method of key and nonce extraction carried out at each time interval.

Table 2
Details of mixed file data set.

File Type	Quantity	Max Size in KB	Min Size in KB
.ppt	556	11161	58
.pdf	780	10111	5
.docx	1220	7340	12
.xlsx	240	7081	11
.jpg	980	4495	3
.tiff	224	2939	872

Fig. 13, shows results from the experiment, with a total number of keys found in memory at each memory capture, for the same Sodinokibi sample encrypting 4000 files. After 15 s of execution, the first Salsa20 encryption keys/nonces are identified in memory. At this point over a thousand keys are identified from the memory capture. The total number of keys found increases for the next 3 intervals until 45 s where 1789 keys are found, after which the number of keys found decreases until none are present after 95 s. The keys appear to be removed from memory as the program executes, most likely after the files have been encrypted and written to disk. It was identified that this strain of ransomware creates the ransom note just prior to the point where the keys start to be identified in memory, at around 15 s after execution, which aligns with documented behaviour (Guillois, 2020).

However, after analysis of the extracted keys, it was discovered that more than 8000 Salsa20 keys were found over the experiment's total time. After, comparing key and nonce values, it was observed that some of the keys identified in the memory snapshots were not unique as one key and nonce pair may appear within multiple memory dumps. The numbers of unique keys found in each memory capture were then analysed, as shown in Fig. 14.

Fig. 14 highlights the unique key and nonce pairs that pertain to each of the intervals. For instance, interval 25 has 796 pairs that are not found in any other memory dump. The total number of unique Salsa20 keys and nonces recovered from the memory captures was 3,612, which is over 90% of the probable 4000 keys used by the ransomware to encrypt the 4000 files. Furthermore, Fig. 14 also demonstrates that Sodinokibi initialises a significant amount of the keys in the first interval and does not gradually increase the number of keys as might be expected. This could be due to the ransomware's multi-threaded behaviour for the key creation and encryption (Tiwari and Koshelev, 2019). It also initialises a lower number of pairs in the following intervals; however, interval 55 has an increased number of new unique pairs when compared to interval 45. This could be due to attempts by the ransomware to delete keys from memory once the file encryption has been completed. Fig. 15, attempts to show the persistence of keys found at each memory capture. It shows the total amounts of keys at each interval and highlights the number of unique new keys created at each interval and the number of keys still in memory from the previous captures. This shows the exposure time of the keys found across several memory captures, and the persistence of the keys in memory as the encryption of the files is carried out. From the 1045 unique keys found at 15 s, 449 were shown still to be in memory at 25 s, and 280 at 35 s, and this indicates a pattern of removal from memory after use. As mentioned previously this will depend on several factors, including the implementation of the ransomware software such as threading and key hygiene processes (The

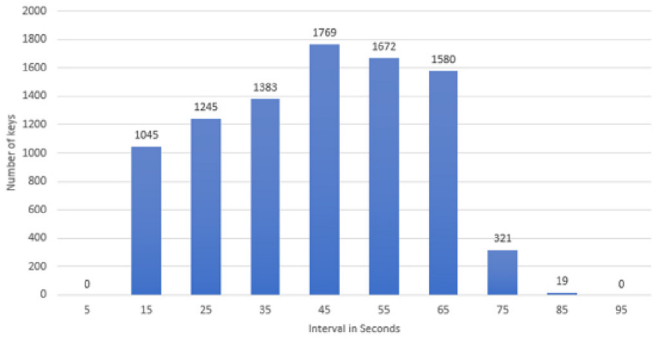

Fig. 13. Total Sodinokibi Salsa20 keys identified in memory.

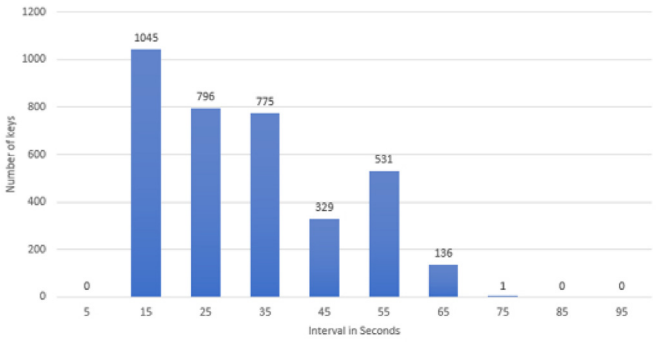

Fig. 14. Unique Sodinokibi Salsa20 keys identified in memory.

BlackBerry Research and Intelligence Team, 2019), the size of victim files being encrypted, and the victim system resources as well as other software running on the system.

Showing how long the pairs found at each memory dump are present in memory is interesting and represents the pattern of chunks of keys being created and then gradually deleted after use. An anomalous pattern is observed among keys from intervals 15, 25, and 35 s which can also be seen in Fig. 14 in that some of the unique keys found seem to be removed from memory and then are found at a later capture. This could indicate swapping of the memory from RAM to disk and back, and experimentation into an analysis of the paging files might be interesting to explore, as highlighted in previous work (Hargreaves and Chivers, 2008).

This experiment showed that out of 4000 files encrypted by this sample of cryptographic ransomware 90.3% were recovered using live memory forensics techniques at 10-s intervals. It also showed that the encryption of the files by this sample of Sodinokibi commences at around 15 s with the described setup and this value is consistent over multiple executions of this experiment. Also, the time that the keys are exposed in memory is between 10 and 60 s from the start of the ransomware's execution.

5.2. Sodinokibi encryption key exposure time lining

Another smaller experiment was performed to attempt to analyse the exact timeline of exposure of specific Salsa20 file encryption keys in memory for the sample ransomware. It was executed using a smaller data set of 20 files of exactly the same size and with snapshots taken at 1-s intervals. The aim was to attempt to identify specific file encryption keys exposure in memory; highlighting individual keys being created and destroyed while the ransomware was carrying out the encryption. This experiment was performed six times and on only one occasion were all 20 key and nonce pairs identified, which are the results discussed. For the remaining executions, a maximum of no more than four keys were found, possibly as a consequence of the limited number of files being used and the multi-threaded nature of the ransomware sample. Fig. 16 shows the results of the experiment where all 20 Salsa keys were found. The Salsa20 encryption keys found in each memory capture are shown, along with the encrypted files the keys were used to create, and these are ordered by the last modified date. This shows the key's timeline in the memory captures against the encrypted files in the order they were written to disk.

The figure shows all of the keys are in memory at 12 s, with no keys appearing in any of the memory captures prior to this point. The ransom note was also created at 12 s. As with the previous experiment, no keys are present in memory for the first 11 s, with perhaps the ransomware process performing reconnaissance and preparation prior to the encryption phase, which aligns with documented behaviour (Guillois, 2020; Tiwari and Koshelev, 2019). For the next 3 s, all 20 keys can be found in memory. At 15 s the first key and nonce pairs for the first 10 files were no longer available and had probably been removed from memory by the ransomware after the encryption of those files had been completed and the files are written. Three seconds later the remaining keys were also removed from the memory This shows a similar pattern of

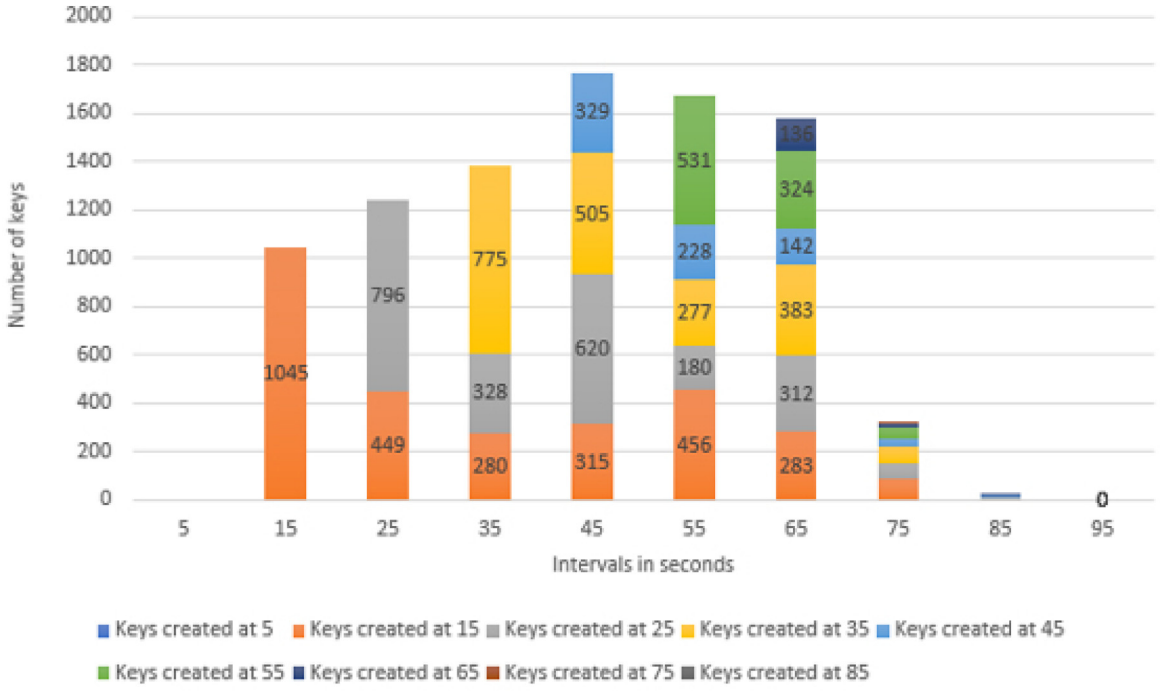

Fig. 15. Sodinokibi Salsa20 keys persistence in memory.

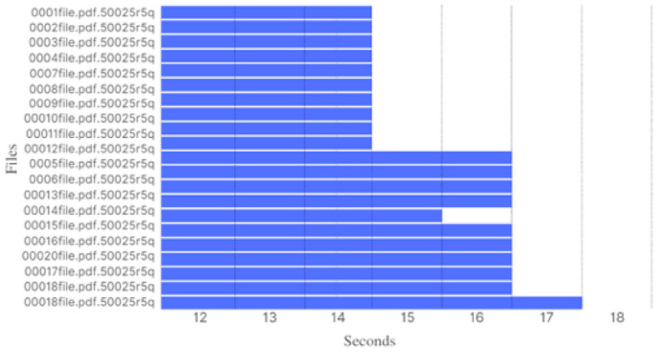

Fig. 16. Sodinokibi encrypted file Salsa20 keys exposure.

behaviour, where keys are being created in bulk and then subsequently removed from memory once those files have been encrypted.

The timeline experiment showed that the symmetric encryption keys are exposed in memory for a short amount of time in which they can be recovered. However, it is likely that this time shrinks if the infected system has more resources (Bajpai, 2020; Tiwari and Koshelev, 2019), on the other hand in a more realistic environment other applications would be taking up resources of that system, therefore more experiments would be needed to accurately determine the time in which these keys are available in a real-world system.

5.3. Ransom Cartel key recovery evaluation

For comparison, a second large-scale key recovery experiment was performed using the Ransom Cartel strain of crypto-ransomware. This strain was selected because it is documented to use the standard Salsa20 encryption algorithm for victim file encryption (Amer Elsad, 2022), the details of the specific version used are presented in Table 3. Ransom Cartel first appeared in December 2021, and like Sodinokibi, was active throughout 2022 and remains active at the time of writing. The analysis states that it uses a unique Salsa20 key nonce pair for each victim file encrypted, and similar to Sodinokibi protects these with symmetric and asymmetric encryption preventing the decryption of files without paying a ransom for a master private key.

The same 4000 file mixed file data set was added to the same base Windows 10 file system, under a single directory. However, for this particular ransomware sample, no option for specifying the directories to encrypt was available resulting in this ransomware theoretically targeting the entire file system of the VM. This included the 88,265 files from the Windows 10 file system as well as the 4000 mixed file data set mentioned previously. During the execution of this ransomware, in theory, the entire file system could be encrypted, however, it is known that normally system folders are excluded and thus the majority of Windows 10 OS files were expected to remain untouched. The anticipated behaviour was that the ransomware would mainly target files under the *C:\Users* directory, where user data files are typically stored in a Windows file system.

After execution of the ransomware program, it was identified that all 4000 files within the test dataset had been encrypted, together with an additional 18 files that were also present under the *C:\Users* directory. The total execution time taken for the ransomware sample was 4.16 min, and as shown in Fig. 17, no keys were identified in memory after 220 s. This is slower than the Sodinokibi sample, although Ransom Cartel was potentially targeting many more directories for encryption as well as generating more ransom note files. This behaviour could explain the difference in execution times. The total Salsa20 key and nonce pairs identified in the memory captures, using 10-s capture intervals, was 14,175. This is significantly more than was identified during the execution of the Sodinokibi sample. However, the total number of unique keys recovered was 3,182, which is only 79% of the possible 4018 file encryption key and nonce pairs generated. Fig. 17 illustrates the presence of Salsa20 encryption keys identified in the ransomware's volatile memory and it can be seen that the process cleans up and removes used keys periodically, with the final clean-up occurring between 220 and 230 s.

It's interesting to note that the time taken during the reconnaissance phase, before the start of encryption, is similar to that of Sodinokibi, even though this sample is targeting the entire file system. This aligns with the expected behaviour mentioned earlier where the majority of the files on the base file system are ignored and the program focuses on user data files. It is known that ransomware is evolving so that it focuses its targeting of files in an attempt to execute as fast as possible and thus evade detection. This behaviour highlights the relatively small period of time detection systems have to identify these types of malware once they start executing.

6. Discussion

Currently, no single solution to mitigate cryptographic ransomware exists. However, progress is continually being made in the effort to combat this type of malware. This work has presented a method for identifying and recovering Salsa20 cryptographic artefacts from volatile memory and applying this to the mitigation of one key per file Salsa20-based ransomware. An evaluation of periodic sampling for key recovery with real ransomware samples has been conducted in an attempt to map the life cycle of cryptographic keys in volatile memory. Furthermore, this paper also demonstrates a successful technique for the successful decryption of ransomware-encrypted victim files validating that the keys being

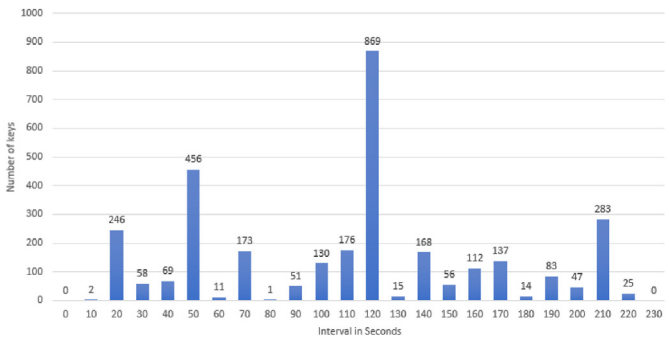

Fig. 17. Unique ransom cartel Salsa20 keys identified in memory.

Table 3
Ransom cartel crypto-ransomware sample.

Ransomware Strain	SHA256 Hash
Ransom Cartel	55e4d509de5b0f1ea888ff87eb0d190c328a559d7cc5653c46947e57c0f01ec5

recovered are in fact the ones used to encrypt the targeted files. No description of such a technique has been found in the literature.

The initial experiments in this research build on the previous work in the established area of the encryption key extraction from memory (Halderman et al., 2009; Hargreaves and Chivers, 2008), by developing, evaluating and validating a method of identifying Salsa20 encryption artefacts in volatile memory. The developed method was then applied to the analysis of Sodinokibi, which is an actual example of Salsa20-based crypto-ransomware. The work has similarities to previous AES-based related work (Bajpai and Enbody, 2020c; Davies et al., 2020; Yuste and Pastrana, 2021), evaluating a side-channel attack on the running ransomware to retrieve AES encryption keys from the ransomware process memory. However, many of the new strains of ransomware are now employing faster encryption algorithms such as Salsa20 resulting in the need to expand the research into developing the methods for recovery of Salsa20 encryption keys. The identified keys are validated by the successful decryption of files targeted by the ransomware. The working decryption method for the Sodinokibi ransomware strain's victim files is also described in detail, as descriptions of this type of process do not seem to be documented anywhere in the literature.

The developed methods of key recovery were then evaluated against two Salsa20-based ransomware strains, targeting a data set of mixed data victim files. The method used periodic sampling and analysis of the ransomware's volatile memory. Using a 10-s sampling interval the method successfully recovers a high percentage of Salsa20 encryption keys demonstrating the viability of this live forensics approach in the mitigation of some strains of crypto-ransomware. However, using periodic sampling could potentially miss keys depending on factors such as the environment resources, the ransomware's cryptographic management implementation, the size of the files being encrypted, and faster-emerging methods such as partial and intermittent encryption. However, the method can produce very good results, if the sampling interval could be matched to the speed of keys being created and could provide a good last line of defence. The technique could theoretically also be incorporated as a part of other detection and mitigation solutions.

There appears to be little related research concerning ransomware that uses one unique encryption key per file, with the majority of the related work focusing on ransomware that uses a single key for all the encryption (Bajpai and Enbody, 2020c; Davies et al., 2020; Yuste and Pastrana, 2021). Using a unique encryption key per file is now the preferred method used by most modern ransomware. This work evaluates the identification and collection of the file encryption keys across a mixed file data set and shows that by periodic sampling of the memory of the malicious process, many of these keys can be identified and extracted. The exposure and lifecycle of the keys in memory are explored with timeline experiments, and these show the creation of keys as the ransomware executes, and what looks like removal from memory once the key had been used, This behaviour aligns with other technical analysis of the ransomware behaviour (Tiwari and Koshelev, 2019). Analysis of key exposure could aid the development of new useful techniques to extract important information from the memory of an infected host. Investigation into events that could be used to trigger memory analysis could also be researched as a method for determining optimal times for key extraction. New strains of cryptographic ransomware could also be analysed using this sampling technique and may provide a better way of understanding the ransomware threat model, through the evaluation of key persistence in memory as new ransomware methods evolve.

6.1. Limitations and future work

Limitations of the work are mainly related to the narrow focus on the later parts of the crypto-ransomware threat model. The methods of recovering keys and decryption of victim files are a last defence mechanism, once the ransomware is already encrypting the user's data. As highlighted by recent surveys (McIntosh et al., 2021), ransomware can also use the exfiltration of victim data as an additional method of extortion. However, victim file encryption is still performed and used for extortion by all the main ransomware strains currently in operation so any mechanisms of mitigation and recovery would be beneficial.

There are some areas of research related to the work described in this paper that could benefit from further investigation. One direction could be memory analysis based on trigger events, to try to identify when to capture keys, such as with previous research hooking on certain API function calls (Genç et al., 2018; Kim et al., 2017; Kolodenker et al., 2017; Mehnaz et al., 2018; Seo and Kim, 2012), though these single trigger events can, in theory, be evaded such as with custom cryptography or direct system calls (McIntosh et al., 2021). Evaluating the scanning of more specific areas of memory for encryption keys, such as within the process memory, could also be interesting. Methods for monitoring in near real-time, or capturing memory within smaller time intervals, may also be needed to mitigate against the emerging faster crypto-ransomware strains (Anand et al., 2022).

The development of tools for the identification and extraction of Salsa20 cryptographic artefacts has shown to be useful, and additional methods could be developed to extract other similar symmetric encryption algorithms, such as variants of Salsa20 and custom implementations. Limitations with the methods currently identified could be encountered, due to custom versions of these algorithms being implemented, such as with custom initialisation arrays reported to be used by some of the latest ransomware strains (Yuste and Pastrana, 2021), so further work on identifying these could also be undertaken.

7. Conclusion

The work presented in this paper demonstrates that methods for ransomware encryption key identification and subsequent extraction can be successfully extended and applied to current techniques being used by ransomware. Previous work in this area has focused on investigating AES symmetric encryption and key extraction. However, some strains of ransomware are now using alternative encryption algorithms such as Salsa20, as well as typically using a unique encryption key per file. This paper presents a method that can successfully identify Salsa20 cryptographic artefacts in the volatile memory of a ransomware process. The research builds on previous work to facilitate key extraction from ransomware strains that employ more complicated techniques of encryption key management, including multiple symmetric encryption keys.

The paper initially describes the methods required to allow for Salsa20 key identification and extraction, as well as per file encryption keys. These methods were then tested and validated using a synthetic ransomware sample with the results confirming that the identified keys were able to decrypt the files affected by the ransomware. Two samples of different strains of real-world ransomware programs were then run on a forensically safe, isolated machine and again the developed method was able to successfully identify and extract the majority of the keys used by the

ransomware. These keys were then validated by decrypting the affected files and recovering the victim's data. In a real-world scenario, this would effectively relieve the victim from having to pay any ransom to recover their data.

A related part of the research performed during this work was to analyse and map the symmetric encryption key life cycle and monitor how long specific keys were exposed in memory. It was noted that key generation appears to be handled on a block basis where blocks of keys appear to be created at the same time, are present in the memory at the same time and destroyed simultaneously. It is hypothesised that this could be related to the ransomware executable being multi-threaded. Due to the granularity of the technique used to capture the memory during this work, it was not possible to precisely map individual key life cycles, but this observation raises some interesting questions and will most certainly be followed up in subsequent research.

The method of identifying and extracting Salsa20 cryptographic artefacts from memory, and using these to decrypt files has been shown to be successful, and useful in the mitigation of modern ransomware which used hybrid cryptography. The method of using live forensics to extract victim file encryption keys directly from the memory of running ransomware has been shown to produce successful results, in terms of being able to decrypt victim files without the need for the master asymmetric key, and so help prevent the need to pay a ransom.

The main focus of this paper was to explore the possibility of being able to extract encryption keys from the volatile memory of running ransomware programs, and the tools developed to identify Salsa20 key identification, key extraction and file decryption are available on request.

Declaration of competing interest

This paper has not been previously published or considered or is being considered for publication elsewhere. There are also no conflicts of interest.

Data availability

Data will be made available on request.

References

Aidan, J.S., Verma, H.K., Awasthi, L.K., 2018. Comprehensive survey on petya ransomware attack, 2017. In: Proceedings - 2017 International Conference on Next Generation Computing and Information Systems. ICNGCIS, pp. 131–136. https://doi.org/10.1109/ICNGCIS.2017.30.

Amer Elsad, D.B., 2022. Ransom cartel ransomware: a possible connection with revil. URL: https://unit42.paloaltonetworks.com/ransom-cartel-ransomware/.

Anand, P.M., Charan, P.S., Shukla, S.K., 2022. A comprehensive api call analysis for detecting windows-based ransomware. In: 2022 IEEE International Conference on Cyber Security and Resilience (CSR). IEEE, pp. 337–344.

Bajpai, P., 2020. Extracting ransomware's keys by utilizing memory forensics. URL: https://d.lib.msu.edu/etd/48467.

Bajpai, P., Enbody, R., 2020a. An Empirical Study of Key Generation in Cryptographic Ransomware. International Conference on Cyber Security and Protection of Digital Services. https://doi.org/10.1109/CYBERSECURITY49315.2020.9138878. Cyber Security 2020.

Bajpai, P., Enbody, R., 2020b. Attacking Key Management in Ransomware, vol. 22. IT Professional, pp. 21–27. https://doi.org/10.1109/MITP.2020.2977285.

Bajpai, P., Enbody, R., 2020c. Memory Forensics against Ransomware. International Conference on Cyber Security and Protection of Digital Services. https://doi.org/10.1109/CYBERSECURITY49315.2020.9138853. Cyber Security 2020.

Bajpai, P., Sood, A.K., Enbody, R., 2018. A key-management-based taxonomy for ransomware. In: 2018 APWG Symposium on Electronic Crime Research (eCrime). https://doi.org/10.1109/ecrime.2018.8376213.

Beaman, C., Barkworth, A., Akande, T.D., Hakak, S., Khan, M.K., 2021. Ransomware: recent advances, analysis, challenges and future research directions. Comput. Secur. 111, 102490. https://doi.org/10.1016/j.cose.2021.102490.

Bernstein, D.J., 2008. The salsa20 family of stream ciphers. In: New Stream Cipher Designs. Springer, pp. 84–97.

Bernstein, D.J., Snuffle, 2005. URL: https://cr.yp.to/snuffle.html (Last Accessed: 2021-11-14).

Berrueta, E., Morato, D., Magana, E., Izal, M., 2019. A survey on detection techniques for cryptographic ransomware, 1–1 IEEE Access 7. https://doi.org/10.1109/ACCESS.2019.2945839.

Berrueta, E., Morato, D., Magaña, E., Izal, M., 2020. Open repository for the evaluation of ransomware detection tools. IEEE Access 8, 65658–65669.

Blog, Threatcop, 2022. Notorious ransomware attacks by revil in 2021. URL: https://threatcop.com/blog/revil-group/.

Craciun, V.C., Mogage, A., Simion, E., 2019. Trends in design of ransomware viruses. URL:. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Cham, pp. 259–272 https://link.springer.com/chapter/10.1007/978-3-030-12942-2_20.

Dargahi, T., Dehghantanha, A., Bahrami, P.N., Conti, M., Bianchi, G., Benedetto, L., 2019. A cyber-kill-chain based taxonomy of crypto-ransomware features. Journal of Computer Virology and Hacking Techniques 15, 277–305. https://doi.org/10.1007/s11416-019-00338-7.

Davies, S.R., Macfarlane, R., Buchanan, W.J., 2020. Evaluation of live forensic techniques in ransomware attack mitigation. URL: Forensic Sci. Int.: Digit. Invest. 33, 300979 https://linkinghub.elsevier.com/retrieve/pii/S2666281720300858.

Davies, S.R., Macfarlane, R., Buchanan, W.J., 2021. Differential area analysis for ransomware attack detection within mixed file datasets. URL: Comput. Secur. 108, 102377 https://www.sciencedirect.com/science/article/pii/S0167404821002017.

Fayi, S.Y.A., 2018. What petya/NotPetya ransomware is and what its remidiations are. URL: Adv. Intell. Syst. Comput. 738, 93–100 https://link.springer.com/chapter/10.1007/978-3-319-77028-4_15.

Genç, Z.A., 2020. Analysis, detection, and prevention of cryptographic ransomware. URL: https://orbilu.uni.lu/handle/10993/44662.

Genç, Z.A., Lenzini, G., Ryan, P.Y., 2018. No random, No ransom: a key to stop cryptographic ransomware. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) 10885 LNCS, 234–255. URL: https://link.springer.com/chapter/10.1007/978-3-319-93411-2_11.

Guillois, N., 2020. Sodinokibi/revil malware analysis. URL: https://www.amossys.fr/fr/ressources/blog-technique/sodinokibi-malware-analysis/.

Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W., 2009. Lest we remember: cold-boot attacks on encryption keys. Commun. ACM 52, 91–98.

Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.. Lest we remember: cold boot attacks on encryption keys. Technical Report. URL: http://citp.princeton.edu/memory.

Hargreaves, C., Chivers, H., 2008. Recovery of encryption keys from memory using a linear scan. In: ARES 2008 - 3rd International Conference on Availability, Security, and Reliability, Proceedings, pp. 1369–1376. https://doi.org/10.1109/ARES.2008.109.

Hasherezade, 2021. A deep dive into phobos ransomware. URL: https://blog.malwarebytes.com/threat-analysis/2019/07/a-deep-dive-into-phobos-ransomware/.

Huck, J., Breitinger, F., 2022. Wake up digital forensics' community and help combating ransomware. IEEE Security Privacy 2–11. https://doi.org/10.1109/MSEC.2021.3137018.

Humayun, M., Jhanjhi, N.Z., Alsayat, A., Ponnusamy, V., 2021. Internet of things and ransomware: evolution, mitigation and prevention. Egyptian Informatics Journal 22, 105–117. https://doi.org/10.1016/J.EIJ.2020.05.003.

Interpol News, 2021. Immediate action required to avoid ransomware pandemic. URL: https://www.interpol.int/en/News-and-Events/News/2021/Immediate-action-required-to-avoid-Ransomware-pandemic-INTERPOL.

Kaplan, B., Geiger, M., 2007. URL: https://cryptome.org/0003/RAMisKey.pdf.

Kim, H.E., Yoo, D., Kang, J.S., Yeom, Y., 2017. Dynamic ransomware protection using deterministic random bit generator. In: 2017 IEEE Conference on Applications, Information and Network Security, AINS 2017 2018-January, pp. 64–68. https://doi.org/10.1109/AINS.2017.8270426.

Kolodenker, E., Koch, W., Stringhini, G., Egele, M., 2017. Paybreak : defense against cryptographic ransomware. In: ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security, pp. 599–611. https://doi.org/10.1145/3052973.3053035.

Lee, S., Youn, B., Kim, S., Kim, G., Lee, Y., Kim, D., Park, H., Kim, J., 2019. A study on encryption process and decryption of ransomware in 2019. URL: Journal of the Korea Institute of Information Security & Cryptology 29, 1339–1350. https://doi.org/10.13089/JKIISC.2019.29.6.1339.

Maartmann-Moe, C., Thorkildsen, S.E., Årnes, André, 2009. The persistence of memory: forensic identification and extraction of cryptographic keys. Digit. Invest. 6, S132–S140. https://doi.org/10.1016/J.DIIN.2009.06.002.

Makrakis, G.M., Kolias, C., Kambourakis, G., Rieger, C., Benjamin, J., 2021. Industrial and critical infrastructure security: technical analysis of real-life security incidents. IEEE Access 9, 165295–165325. https://doi.org/10.1109/ACCESS.2021.3133348.

McIntosh, T., Kayes, A., Chen, Y.P.P., Ng, A., Watters, P., 2021. Ransomware mitigation in the modern era: a comprehensive review, research challenges, and future directions. ACM Comput. Surv. 54, 1–36.

Mehnaz, S., Mudgerikar, A., Bertino, E., 2018. RWGuard: a real-time detection system against cryptographic ransomware. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in

bioinformatics) 11050 LNCS, 114–136. URL: https://link.springer.com/chapter/10.1007/978-3-030-00470-5_6.
Mohammad, A.H., 2020. Analysis of ransomware on windows platform comparing two feature selections methods (information gain and gain ratio) on three different classification algorithms using Arabic dataset. View project analysis of ransomware on windows platform. URL: IJCSNS International Journal of Computer Science and Network Security 20 https://www.researchgate.net/publication/343194067.
Moussaileb, R., Cuppens, N., Lanet, J.L., Bouder, H.L., 2021. A survey on windows-based ransomware taxonomy and detection mechanisms. ACM Comput. Surv. 54, 1–36. https://doi.org/10.1145/3453153.
Mundo, A., 2022. Blackmatter ransomware analysis; the dark side returns. URL: https://www.trellix.com/en-gb/about/newsroom/stories/threat-labs/blackmatter-ransomware-analysis-the-dark-side-returns.html.
of Public Affairs, O, 2022. Ukrainian arrested and charged with ransomware attack on kaseya. URL: https://www.justice.gov/opa/pr/ukrainian-arrested-and-charged-ransomware-attack-kaseya.
Oz, H., Aris, A., Levi, A., Uluagac, A.S., 2021. A survey on ransomware: evolution, taxonomy, and defense solutions. ACM computing surveys 1. URL: http://arxiv.org/abs/2102.06249. arXiv:2102.06249.
Özarslan, S., 2021. A detailed walkthrough of ranzy locker ransomware ttps. URL: https://www.picussecurity.com/resource/blog/a-detailed-walkthrough-of-ranzy-locker-ransomware-ttps.
Poudyal, S., 2021. Multi-Level Analysis of Malware Using Machine Learning. Ph.D. thesis. The University of Memphis.
Ramsdell, K.A.W., Esbeck, K., 2021. Evolution of ransomware - mitre corporation. URL: https://healthcyber.mitre.org/wp-content/uploads/2021/08/Ransomware-Paper-V2.pdf.
Salvio, J., 2018. Gandcrab v4.0 analysis: new shell, same old menace. URL: https://www.fortinet.com/blog/threat-research/gandcrab-v4-0-analysis-new-shell-same-old-menace.
Seo, H.J., Kim, H.W., 2012. Network and data link layer security for DASH7. Journal of information and communication convergence engineering 10, 248–252. URL: http://jicce.org.
Sharif, S.O., Mansoor, S.P., 2010. Performance analysis of stream and block cipher algorithms. In: ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings, vol. 1. https://doi.org/10.1109/ICACTE.2010.5578961.
Team, T.B.R.I., 2021. Threat thursday: Karma ransomware. URL: https://blogs.blackberry.com/en/2021/11/threat-thursday-karma-ransomware.
The BlackBerry Research and Intelligence Team, 2019. Threat spotlight: revil/sodinokibi ransomware - blackberry. URL: https://blogs.blackberry.com/en/2019/07/threat-spotlight-sodinokibi-ransomware.
Tiwari, R., Koshelev, A., 2019. Taking deep dive into sodinokibi ransomware. URL: https://www.acronis.com/en-gb/blog/posts/sodinokibi-ransomware/.
Yuste, J., Pastrana, S., 2021. Avaddon ransomware: an in-depth analysis and decryption of infected systems. URL: Comput. Secur. 109. https://doi.org/10.1016/J.COSE.2021.102388 www.sciencedirect.com. arXiv:2102.04796.
Zimba, A., Chishimba, M., Chihana, S., 2021a. A ransomware classification framework based on file-deletion and file-encryption attack structures. URL: https://doi.org/10.48550/arXiv.2102.10632.
Zimba, A., Wang, Z., Chishimba, M., 2021b. Addressing crypto-ransomware attacks: before you decide whether to-pay or not-to. J. Comput. Inf. Syst. 61, 53–63. https://doi.org/10.1080/08874417.2018.1564633.