A Key-Management-Based Taxonomy for Ransomware

Pranshu Bajpai
Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
Email: bajpaipr@cse.msu.edu

Aditya K Sood
SecNiche Security Labs
San Jose, California
Email: soodadit.msu@gmail.com

Richard Enbody
Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
Email: enbody@cse.msu.edu

***Abstract*—Ransomware encrypts user files making management of the encryption key(s) critical to its success. Developing a better understanding of key management in ransomware is a necessary prerequisite to finding weaknesses that can be exploited for defensive purposes. We describe the evolution of key management as ransomware has matured and examine key management in 25 samples. Based on that analysis, we introduce a ransomware taxonomy that is analogous to hurricane ratings: a Category 5 ransomware is more virulent from a cryptographic standpoint than a Category 3. In our analysis of samples in light of the taxonomy, we observed that poor cryptographic models appear as recently as 2018.**

I. Introduction

Ransomware, or cryptovirii, are malicious programs that encrypt user data with the goal of extorting money from the victim in exchange for file decryption. Ransomware operate on the fundamental principle that recovering locked data without the proper decryption essentials is an intractable problem [1] and hence the victim has no choice but to pay the ransom for data recovery. `WannaCry`, `Petya`, `CryptoLocker` and `TeslaCrypt` are some of the more notable examples of such ransomware. In general, modern ransomware are known to only encrypt user data files (e.g. `.xlsx,.docx,.jpg,.pptx` etc.) and leave system files (e.g. `.dll`) untouched to permit use of the system to meet the ransom demand. The growing popularity of cryptocurrency allows ransomware developers to extort money anonymously. In recent news [2] [3] [4] [5] [6], ransomware have made frequent appearance because of their virulent impact and continuously evolving encryption models that are a cause of major concern to individuals and organizations alike.

In this paper, we present key management and cryptography models that are deployed in these ransomware with the objective of providing a deeper comprehension of potential flaws in cryptoviral infections. Our goal is to show how key management in cryptoviral extortions has evolved over time. Exploring and discovering vulnerabilities in key management in these ransomware helps to mitigate the threat. Certainly 'prevention is better than a cure' holds for ransomware, but we assume that the ransomware has successfully compromised the victim's computer. We explore options from this point forward to combat the ransomware infection other than restoring from backups. In theory, regular backups facilitate easy restoration. However, the sad truth is that backups are not always available, are partial, or are unacceptably outdated due to infrequent sync ups. In some cases, ransomware are known to explicitly search for backups over the network and encrypt any discovered backups as well [7]. Network shares are usually mapped to drive letters on host systems and discoverable by the ransomware. The trade-off between security and backup cost in organizations is favoring ransomware developers for now. So a better defense is to exploit weaknesses in design and implementation of cryptographic models deployed by ransomware which in turn warrants the need to comprehend the evolution of key management in cryptoviral extortions.

Note that terms such as 'cryptoviral extortion' or 'cryptovirus' can be used interchangeably with 'ransomware' throughout this paper. Also, we do not recommend paying the ransom since there is no guarantee of file recovery even after payment is successfully made according to terms [8] and because payment invigorates the ransomware business model by making it profitable. Furthermore, note that the term 'decryption essentials' is used to refer to any knowledge that is required to decrypt victim's encrypted files. For instance, this could refer to either a symmetric AES key or the knowledge of how a custom encryption algorithm functions or both a key and an algorithm. Usually cryptovirii deploy well known algorithms (e.g. RSA or AES) and some form of key is the secret needed for data decryption.

In brief, the main contributions of our paper are:

- Analyzing the evolution of key management in ransomware.
- Introducing a classification system that groups ransomware according to the potency of their encryption model (summarized in Figure 3).
- Classifying 25 ransomware samples using our proposed classification system as shown in Table I.

978-1-5386-4922-0/18/$31.00

The rest of this paper is structured as follows. In Section 2, we briefly discuss related work. In Section 3, we discuss the basics of cryptography in the context of ransomware. In Section 4, we introduce the types of key management observed in ransomware. In Section 5, we provide pseudo code pertaining to a ransomware using a hybrid cryptosystem. In Section 6, we discuss data collection and research methodology. In Section 7, we discuss our findings. Finally, we conclude the paper in Section 8.

II. Related Work

We found a small number of discussions on key management in ransomware scattered across a few papers. In many papers, specific key management techniques were discussed indirectly as part of behavior analysis while dissecting a particular ransomware variant. Young and Yung [9] first discussed key management approaches such as public key encryption and key splitting among peers (discussed in detail later in this paper). Since then ransomware have adopted more resistant key management models. Kharraz *et al.* [10] focus on all aspects of ransomware of which key generation and management techniques are a part—whereas our entire emphasis is on that. They consider a wide variety of variants, including `GPCode`, `CryptoWall` and `CryptoLocker`, across 15 ransomware families. Cabaj *et al.* [11] discuss network activity of `Cryptowall`. Further insights into key derivation by ransomware on a Windows hosts are provided by Palisse *et al.* [12], while Puodzius [13] discusses how cryptography was pivotal in shaping ransomware using specific case studies. Young [14] demonstrated the use of Microsoft's `CryptoAPI` in cryptoviral extortions. Gazet [15] presents a comparitive analysis of several ransomware variants as seen prior to 2008.

To the best of our knowledge, there is no previous work on elaborately classifying ransomware samples. In general, the term 'scareware' is used while referring to malicious software that prey upon victim's fear of loosing data or private information. Ransomware is a special type of scareware [10] that encrypts user data and demands payment. Broadly speaking, two main classes of ransomware have been discussed [16]: 1) locker ransomware, that focus on locking users out of the host machine, and 2) crypto ransomware, that focus on denying users access to their files or data on the host machine.

III. cryptography basics revisited

This section serves as a refresher towards cryptography types that are popularly used by modern ransomware. Broadly, cryptographic algorithms are divided into the following two types:

A. Symmetric key

As the name suggests, symmetric key cryptography uses the same key for encryption and decryption. For example, Advanced Encryption Standard (AES) [17] is a symmetric cipher we have found to be deployed by many ransomware variants. A clear advantage that symmetric key encryption offers cryptovirii is that encryption is a lot faster than in asymmetric algorithms. Like any other crime, the goal is to quickly intimidate the victim and extort money before any interruptions occur so speed is of the essence. For example, an antivirus program may notice file access and modification patterns and quarantine the cryptovirus. The more user data that has been encrypted before such quarantine, the more leverage the cryptovirus has and hence the better the chances of getting the ransom. Therefore, symmetric key cryptography is enticing to ransomware developers. A disadvantage of symmetric key cryptography, however, is that improper key management results in key disclosure. Ransomware needs to securely deploy the key for performing the encryption and then conceal the key in a way that it is out of reach of the victim until payment is made.

B. Asymmetric key

Also known as public key cryptography, asymmetric key cryptography uses a mathematically-related key pair, e.g. a public key for encryption and the paired private key for decryption (or the reverse). The RSA algorithm is an asymmetric cipher popularly used by ransomware. It is currently not feasible to decipher the encryption or recover the private key relying solely on the public key and the algorithm. When implemented correctly, this approach offers more flexibility to attackers and makes it impossible to reverse the encryption without knowledge of attacker's private key as shown later in this paper. However, a major disadvantage to attackers is that asymmetric key encryption is slow and increases the size of the cryptogram when compared to the corresponding plaintext. Thus, the encryption process is lengthy and the encrypted data requires more storage space on the host. For this reason, asymmetric encryption is mainly deployed to securely encrypt a symmetric 'session key', after said session key has been used by the ransomware to encrypt user data as explained in the hybrid approach below.

Elliptic Curve Diffie-Hellman (ECDH) asymmetric encryption is deployed by some of the more recent variants, such as `PetrWrap`, in place of the more common RSA encryption. The cryptographic model behind a cryptoviral infection based on ECDH typically operates similar to RSA encryption except that the ransomware developer decides on a predefined elliptic curve (e.g. `secp192k1`) needed to generate the keys on both sides. Encryption trends in modern cryptoviral extortions have thus shifted from RC4 to RSA+AES to ECDH+AES [18]. An obvious question is: Why use ECDH over RSA? Although ECDH has shorter key size while providing comparable security to RSA and a slight performance boost, ECDH does not offer any major cryptographic advantages over RSA. It is speculated by Kotov and Rajpal [18] that ECDH is not

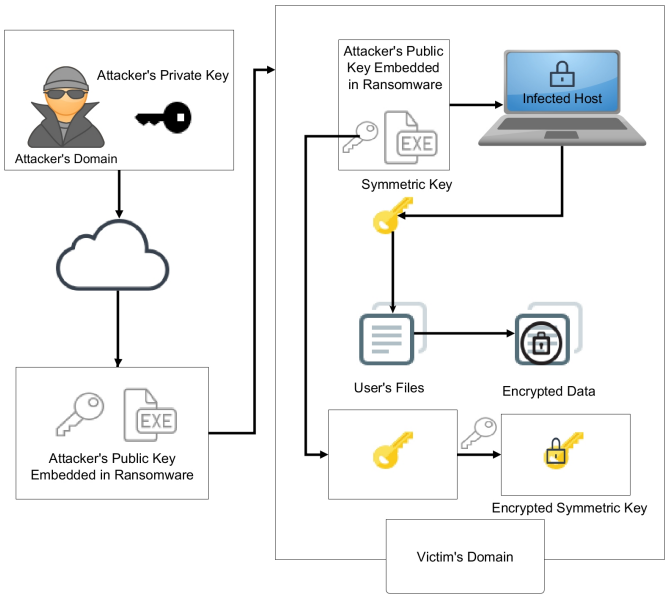

Fig. 1. Basic hybrid encryption model in ransomware

as well scrutinized for security flaws and is consequently better for marketing in the underground communities.

C. Hybrid of Symmetric and Asymmetric

Generally, we found that a hybrid of the two are deployed by recent ransomware to take advantage of the best of both types. User data is encrypted using a symmetric cipher for speed, while the symmetric key used for the encryption is then encrypted using the public key of the attacker. The public key may come embedded in the ransomware as shown in Figure 1.

This basic hybrid (symmetric+asymmetric) key model works in the following steps:

1) Ransomware compromises host and commences execution.
2) Cryptographic APIs available on the host are used to generate an encryption key such as an `AES-256` key.
3) Ransomware encrypts this symmetric key with a hard-coded asymmetric key (e.g. `RSA-2048`) and communicates a copy of the now encrypted symmetric key to the attacker.
4) User data is encrypted using the symmetric key.
5) Ransomware securely destroys the symmetric key on the host machine, now making the attacker the sole possessor of the decryption key.
6) A ransom note is displayed to the user while ransomware awaits payment.

There are variations where the encrypted key is securely stored on the host machine so the only communication with the attacker is ransom payment.

IV. Types of key management in ransomware

Key management in ransomware has gone through several changes during the years as cryptovirii developers learn from past oversights. The result is an ever evolving cybercrime operation that continues to be profitable as long as it is correctly implemented. In effect, all cryptoviral infections follow these very elementary steps:

1) Infect host and commence execution.
2) Acquire encryption secret (key).
3) Encrypt user data.
4) Demand ransom.

The 'encryption secret' is usually a symmetric key. Protecting this secret is crucial for the attacker to have leverage over the victim and this is where key management comes in. Here we present the primary key management techniques as observed in several cryptoviral extortion programs. We will discuss the following main types of key management in this paper:

(A) No key or no encryption
(B) Decryption essentials in user domain
  a) Decryption essentials on host machine
  b) Decryption essentials distributed among peers
(C) Decryption essentials in attacker domain
  a) Decryption essentials on a command and control, C&C, server—single encryption
  b) Decryption essentials on C&C server—hybrid encryption
    - hybrid encryption model with multiple layers

A. No key or no encryption

Some scareware are used mainly to deceive people into believing their security is compromised. They deploy scam tactics to frighten users into making hasty decisions while under stress. It is beneficial for fake scareware to piggyback on the recent success of large-scale ransomware infections and pose as functional ransomware. Being a fake, the software will not actually encrypt files. Instead, it might simply obfuscate user data on the host or delete it and display a ransom note asking for payment. The fake `AnonPop` "ransomware", which deleted user files and asked for $125 for "decryption" is an example. In reality, there is no file restoration procedure in this fake scareware. However, because the files were not securely deleted, they can easily be recovered. There is no reason to make a ransom payment. Since there is no actual encryption, there is no key management. The motive behind such fake ransomware is to make a quick buck without going through the actual acrobatics of performing secure file encryption, decryption and the relevant key management. It is a low-effort operation for cybercriminals to pursue while authorities are busy working on actual, bigger malware threats. Moreover, not performing encryption operations means that scareware have a greater chance of slipping through heuristics-based detection procedures deployed by antivirus solutions such as a trigger caused when a program demands access to `CryptoAPI` in Windows.

Examples of ransomware that follow this model: `AnonPop` and original variants of `ConsoleCrypt` and

Nemucod and, more recently, certain WannaCry imitators such as Aron WanaCrypt0r 2.0.

B. Decryption essentials in user domain

Certain ransomware strains have failed to protect decryption essentials such as the decryption key from the user. Note that when saying 'decryption essentials in user domain,' we are including cases where the "one key" that is essential for decryption can be discovered by reverse engineering the ransomware code or analyzing a hidden file in the system or network where the ransomware has "secretly" stored the key. As long as decryption essentials are within a user's reach, the cryptovirus variant fits this category.

1) Decryption essentials on host machine: If the decryption key can be gleaned from analysis of the host machine either during or after the ransomware encryption process, then it fits this category. Use of a static hard-coded key significantly weakens an encryption model. For example, a key hard-coded in the JigSaw ransomware was recovered by reverse engineering the ransomware. The section of de-obfuscated code that holds the AES key and initialization vector (IV) [19] is shown in Listing 1. Note that the AES key binary data is encoded as base-64 digits which can be decoded on the host using a standard method FromBase64String(). The result is an 8-bit integer array that contains the AES key.

This category also includes cases where a symmetric key is generated uniquely on the infected host and then protected using the hard-coded public key available in the ransomware. The attacker holds the private key corresponding to this public key at a remote location. However, this is discussed under 'key on host machine' since the key was ineffectively concealed on the host machine, which enables easy key recovery by the victim. Since the symmetric encryption key was generated within the user's domain, it may be possible to access this key without paying the ransom. At times, programming blunders in ransomware coding have made such key retrieval fairly straightforward [20]. For example, CryptoDefense ransomware never executed the crucial step—securely destroying the key on host. Thereafter, retrieving the decryption key was as easy as looking in the right folder [20].

Examples of ransomware that follow this model: JigSaw, CryptoDefense, AIDS.

2) Decryption essentials distributed among peers: In this model, attackers attempt to obfuscate the decryption key by breaking it into parts, potentially encrypting those parts, and distributing it among a peer group such as compromised hosts in an organizational network [9]. The clear advantage of this approach for the attacker is that the key does not reside with one host making reverse engineering more difficult. Furthermore, attackers rest easy knowing that encryption does not depend on successful communication with a C&C server post-infection which can prove fatal for the ransomware as explained later in this paper. There is a risk, however, that one of the users restores their host machine from a backup and loses their part of the key, rendering it impossible to decrypt the rest of the infected peers since the key cannot be reconstructed. This is a serious concern for attackers since the overall success of a cryptoviral extortion campaign depends on successful decryption upon payment—otherwise, future victims have no motivation to pay. Ransomware authors now emphasize in the ransom notes that attempted restoration will result in loosing critical information needed for decryption and cause loss of data for other nodes as discussed by Young and Yung [9].

Examples of ransomware that follow this model: (None seen so far).

C. Decryption essentials in attacker domain

This model covers all instances where the attacker has the only copy of decryption essentials. In general, this model offers the tactical advantage to the attacker of safeguarding the key with themselves.

1) Decryption essentials on a C&C server: single encryption: Certain variants are known to deploy only public key cryptography in their encryption models. A ransomware following this model may have a hard-coded public key or acquire an infection-specific public key in the following manner:

1) Upon initial infection, proceed to encrypt user files using the public key embedded within (alternatively, acquire this public key via communication with a C&C server).
2) Display a ransom note to the victim.
3) Send private key for decryption after receiving payment.

Clearly, the model's strength lies in its simplicity and the fact that the decryption key never leaves the attacker until the ransom is paid. However, this model is weak for the following reasons:

- Only one key pair exists, if the public key came hard-coded in the ransomware, so all victims can be decrypted by the same private key. Hence, if one victim makes the payment and obtains the private key to decrypt files, this user can share it with all victims and neutralize the ransomware's entire campaign.
- Asymmetric key encryption is slow when compared to symmetric key encryption.

CryptoLocker is a prime example of a single encryption approach that works in the following steps:

1) Ransomware compromises the host system and sends a notification to a C&C server.
2) A C&C server acknowledges the client and requests an ID from the client.
3) Ransomware sends a unique ID derived from the compromised host and campaign ID to a C&C server.
4) A C&C server uniquely generates an asymmetric key pair for that particular host and sends the public key to the host.

```
using(AesCryptoServiceProvider aesCryptoServiceProvider = new
    AesCryptoServiceProvider()) {
          aesCryptoServiceProvider.Key = Convert.FromBase64String(‘‘
             OoIsAwwF23cICQoLDA0ODe==’’);
          aesCryptoServiceProvider.IV = new byte[] {
              0, 1, 0, 3, 5, 3, 0, 1, 0, 0, 2, 0, 6, 7, 6, 0 }; }
```

Listing 1. Key and IV embedded in the Jigsaw ransomware

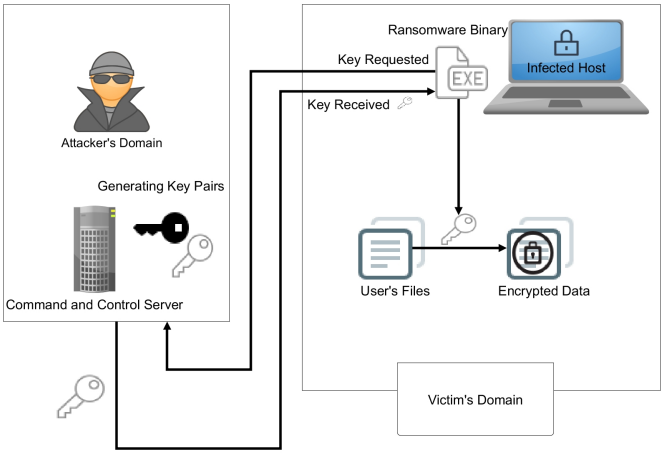

Fig. 2. Single encryption model in ransomware

5) Ransomware sends a final acknowledgment of having received the public key and closes the connection.
6) Ransomware proceeds to encrypt user data using the acquired public key.

In this way `CryptoLocker` encrypts user data using a host-specific asymmetric public key which prevents victims from sharing decryption keys as shown in Figure 2.

Depending on the particular ransomware strain, this communication between a C&C server and a host machine may or may not be encrypted. Older variants of `CryptoLocker` used custom encryption or obfuscation to secure this communication. However, newer variants are using standard schemes such as transport layer security (TLS). Using TLS hinders any kind of network analysis so it provides ransomware with a layer of protection.

This model followed by `CryptoLocker` does not have any cryptographic flaws when implemented correctly. Nevertheless, it is disappearing in the more modern variants of ransomware such as `WannaCry`. One cause is that asymmetric key encryption is slow. However, the ultimate reason is the fundamental operational constraint: connection to a C&C server. Encryption does not start until the ransomware has received the public key from the C&C server. It is possible to block this communication by identifying a request being sent to a potential C&C server. Network administrators maintain and share a list of such blacklisted IP addresses where C&C servers are known to exist [21]. Over time, such crowd-sourced lists of identified C&C servers grow and can be used to effectively set blocks at border firewalls. If the communication is not successful, the cryptoviral infection stays dormant and the overall ransomware operation crumbles. `CryptoDefense` tried to fix this operational flaw by generating keys on host machines by following these steps:

1) Ransomware compromises host system and sends a notification to C&C server.
2) C&C server acknowledges client and requests ID from client.
3) Ransomware exploits cryptographic APIs at host to generate an `RSA-2048` key pair.
4) Ransomware proceeds to encrypt user files using the public key and transfers the private key to the attacker.
5) Ransomware destroys private key in host machine, making attacker the sole possessor of decryption key.
6) Ransomware displays a ransom note to user.

The clear advantage of this approach is that the ransomware is fully-independent in its extortion operation in that it does not need to reach an external server to obtain the encryption key after initial infection as it does in the case of `CryptoLocker`. Such independent ransomware does not reach out to an external entity for an encryption key; rather it generates a key locally. However, `CryptoDefense` had a flaw in that it did not effectively remove the private key from the host machine. `Cerber` on the other hand, implemented the model correctly and has no known flaws.

There are several examples of ransomware that follow this model. `zCrypt` attempts to use the public key to encrypt user data if the connection to a C&C server fails. Initial variants of `CryptoWall` and `CryptoLocker` used a public key to encrypt files.

2) Decryption essentials on a C&C server: hybrid encryption: We previously described a hybrid encryption model. Here we present the case of ransomware that deploys a slightly modified hybrid model.

Hybrid encryption with multiple layers: `WannaCry` gained particular attention because its distribution did not require active user involvement such as clicking the wrong link. It exploited an unpatched vulnerability on a host machine and propagated like a worm. However, the encryption model differed little from earlier hybrid models. The following steps detail the encryption procedure in a `WannaCry` infection.

1) Ransomware compromises a host machine and generates an infection-specific RSA key pair, (K_s, K_p).
2) A public key hard-coded (K_A) in the ransomware is then used to encrypt the private key, K_s, from the key pair generated in step 1. Note that the attacker holds the private key, K_B, corresponding to this public key hard-coded in the ransomware.
3) Ransomware generates AES keys using a cryptographically-secure pseudorandom number generator (CSPRNG) [22] on the host to encrypt files (one AES key per file-to-be-encrypted) and commences file encryption.
4) Ransomware encrypts all AES keys, $S = \{K_1, K_2, ...K_n\}$, using the infection-specific public key, K_p, generated in step 1.
5) Ransomware displays ransom note.

`WannaCry` thus follows a hybrid encryption model with the added step of generating an infection-specific asymmetric key pair on the host. Upon successful payment, the attacker can use their unique private key to decrypt the infection-specific private key. This decrypted private key can be used to decrypt AES keys that are then used to decrypt user files as shown below.

$$\{\{K_s\}_{K_A}\}_{K_B} = K_s \quad (1)$$

$$\{\{S\}_{K_p}\}_{K_s} = S \quad (2)$$

$$\{\{data\}_S\}_S = data \quad (3)$$

One obvious advantage of this model is that it does not suffer from drawbacks of any of the previous models. The advantages of this model are highlighted below:

- Encryption is fast since symmetric encryption (such as AES) is used to encrypt files.
- Communication with an external entity such as C&C server only happens at the time of payment.
- Attacker's private key is never sent anywhere and is kept safe with the attacker.

Note that while the same AES key could be used to encrypt all files on an infected host, attackers use different AES keys—one to encrypt each file, possibly due to these reasons:

- Ransomware developers are wary of the scenario where an antivirus's heuristic detection engine notices the ransomware mid-encryption and hibernates the machine to extract a key from swap storage. If a different key is used to encrypt each file, all the victim would be able to extract is the particular symmetric key being used to encrypt one file. The victim can decrypt one file using this key but others are still held hostage. In other words, such ransomware are doing contingency planning for the event where the encryption operation is interrupted.
- Reusing key and IV pairs for encrypting different files in block ciphers leaves them vulnerable to attacks and make it possible to recover plaintext from ciphertext without knowledge of key [23].

V. Pseudo code of ransomware deploying hybrid cryptosystem in Windows context

The encryption procedure of ransomware using a hybrid encryption approach on a Windows host is illustrated using pseudo code in Listing 2. Modern ransomware such as `Petya` are known to encrypt Windows hosts using this procedure:

1) Generate Symmetric Key
   a) The ransomware's encryption thread creates a handle for an AES key using `HCRYPTKEY`, while a handle to the cryptographic service provider (CSP) is created using `HCRYPTPROV`, and a cryptographic context is generated using `CryptAcquireContext`. A call to a key generation function is then made using the key handle.
   b) The AES key generation function in turn calls `CryptGenKey` with `hProv`, a handle to the CSP, and `CALG_AES_128` (algorithm ID), as parameters. The attacker now sets cipher mode to Cipher Block Chaining (CBC). The symmetric key is then returned to the calling function.
2) Encrypt Files
   The calling function now invokes a file encryption function with `hProv` and AES key as parameters. This file encryption function then performs batch encryption of specific file types using the symmetric key with the standard `CryptEncrypt` function. Control is then returned to the calling function.
3) Encrypt Symmetric Key
   The AES key and handle to the CSP are passed to this function.
   a) The function grabs the RSA public key shipped with the ransomware using `CryptImportKey`.
   b) The AES key is encrypted with the RSA public key and then Base64 encoded. During these operations, `LocalAlloc` is used to allocate memory to hold key blobs and keys are securely exported using `CryptExportKey`.
   c) This encrypted and base64 encoded version of AES key is stored on disk in a file along with a ransom note. A crucial call to `LocalFree` frees all associated memory and invalidates handles. Control is then returned to the calling function.
4) Clean Up
   The calling function now invokes `CryptDestroyKey`, which frees the *hkey* handle to the AES key. After this step is executed, the key is destroyed and all associated memory is freed. Depending on the CSP, the memory area where key was held is also scrubbed before freeing it. This scrubbing ensures that the user cannot recover the key from memory. Finally,

`CryptReleaseContext` is used to release handle to CSP.

Note that details of these Windows CryptoAPI functions are available in Microsoft's documentation [24].

VI. Research Methodology

We analyzed samples from 25 ransomware families and classified them into the following categories that are similar to the well-known hurricane classification. Samples were collected from several malware repositories [25] [26] [27]. During sample collection, we gave preference to the ransomware that were well-known for their impact and we included some recently seen ransomware variants as well. We assigned 'year' to a ransomware based on when its first variant was reported. Note that while we realize multiple strains exist for a particular case of ransomware, we analyzed one variant per ransomware, but kept the analysis broad enough to cover all potential characteristics that would impact classification. We performed static and dynamic analysis of ransomware binaries in many cases to comprehend their functionality and execution behavior. Static analysis included reverse engineering the binary and dynamic analysis included executing samples under Cuckoo Sandbox [28] on a Windows XP SP3 hosted on a virtual machine. Internet connectivity was simulated to observe ransomware behavior.

The objective of this classification is to provide a sense of how virulent a ransomware infection is in terms of its encryption model. In other words, our classification system is designed to gauge how time consuming and challenging it is to reverse encryption without paying the ransom. Note that it is possible for a ransomware strain to shift up or down the categories over time. For example, a new ransomware instance with no apparent vulnerability might be in a severe category at first but shift down if a flaw is eventually discovered in its encryption model. A summary of ransomware categories is shown in Figure 3.

A ransomware belongs to one of these categories if one or more of the following conditions are true for that ransomware:

Category 1
- No actual encryption (fake scareware)
- Demanded ransom before encryption

Category 2
- Decryption essentials extracted from binary
- Derived encryption key predicted
- Same key used for each infection instance
- Encryption circumvented (decryption possible without key)
- File restoration possible using Shadow Volume Copies

Category 3
- Key recovered from file system or memory
- Due diligence prevented ransomware from acquiring key
- Click-and-run decryptor exists
- Kill switch exists outside of attacker's control

Category 4
- Decryption key recovered from a C&C server or network communications
- Custom encryption algorithm used

Category 5
- Decryption key recovered under specialized lab setting
- Small subset of files left unencrypted

Category 6
- Encryption model is seemingly flawless

Fig. 3. Key points in ransomware categories

A. Category 1

- Fake scareware (no real encryption): infection merely poses as a ransomware by displaying a ransom note while not actually encrypting user files
- Displaying the ransom note before encryption process commences. As seen in the case of `Nemucod`, some ransomware will display a ransom note before file encryption [20]. This is a serious operational flaw in the ransomware. The victim or their antivirus solution could effectively take prompt evasive action to prevent ransomware from commencing encryption.

B. Category 2

- Decryption essentials can be reverse engineered from ransomware code or the user system. For example, if the ransomware uses a hard-coded key, then it becomes straight-forward for malware analysts to extract the key by disassembling the ransomware binary. Another possibility of reverse engineering the key is demonstrated in the case of the `Linux.Encoder.A` ransomware where a timestamp on the system was used to create keys for encryption resulting in easy decryption provided that the timestamp is still accessible [20].
- Ransomware uses the same key for every victim. If the same key is used to encrypt all victims during a campaign, then one victim can share the secret key with others.
- Files can be decrypted without the need for a key due to poor choice or implementation of the encryption

```
void thread_encrypt() {                                    // main calling function
...
HCRYPTKEY symKey;                                          // handle to key
HCRYPTPROV hProv = [...];                                  // handle to CSP
symKey = generateKey(hProv);                               // call to key generation function
encryptData(hProv, symKey);                                // call to file encryption procedure
cleanup(symKey);                                           // clean up procedure
CryptDestroyKey(symKey);                                   // destroy key in memory
CryptReleaseContext(hProv, 0);                             // release handle to CSP
}

HCRYPTKEY generateKey(hProv) {
HCRYPTKEY symmKey;                                         // handle to key
CryptGenKey(hProv, CALG_AES_128, 1u, &symmKey);            // generate AES-128 key
DWORD mode = CRYPT_MODE_CBC;                               // use CBC cipher mode
CryptSetKeyParam(symmKey, KP_MODE, &mode, 0);
DWORD padData = PKCS5_PADDING;                             // PKCS 5 padding method
CryptSetKeyParam(symmKey, KP_PADDING, &padData, 0);        // set the padding mode
return symmKey;                                            // return generated key
}

void encryptData(hProv, symKey) {
for each file type F:                                      // search for specific file types
    cryptFile(hProv, symKey);                              // locate and encrypt files
}

void cleanup(hProv, symKey) {
HCRYPTKEY asymPubKey = getasymPubKey(hProv):               //acquire RSA public key
void* symKeyEncryptb64 = exportKey(symKey, asymPubKey);    //encrypt and encode AES key
    //...write ransomnote.txt...
    //...write base64 encoded encrypted AES key...
    //...
LocalFree(symKeyEncryptb64);                               //free allocated memory
}
```

Listing 2. Pseudo code of hybrid encryption in Windows CryptoAPI context

algorithm. Consider the case of `desuCrypt` that used an RC4 stream cipher for encryption. Using a stream cipher with key reuse is vulnerable to known plaintext attacks and known-ciphertext attacks due to the key-reuse vulnerability [29] and hence this is a poor implementation of the encryption algorithm.

- Files can be restored using system backups, e.g. `Shadow Volume Copies` on the New Technology File System (NTFS), that were neglected by the ransomware.

C. Category 3

- Decryption key can be retrieved from the host machine's file structure or memory by an average user without the need for an expert. In the case of `CryptoDefense`, the ransomware did not securely delete keys from the host machine. The user can look in the right folder to discover the decryption key [20].
- User can prevent ransomware from acquiring the encryption key. Ransomware belongs in this category if its encryption procedure can be interrupted or blocked by due diligence on part of the user. For example, `CryptoLocker` discussed above cannot commence operation until it receives a key from the C&C server. A host or border firewall can block a list of known C&C servers hence rendering ransomware ineffective.
- Easy 'Click-and-run' solution such as a decryptor has been created by the security community [42] such that a user can simply run the program to decrypt all files.
- There exists a kill switch outside of attacker's control that renders the cryptoviral infection ineffective. For example, in the case of `WannaCry`, a global kill switch existed in the form of a domain name. The ransomware reached out to this domain before commencing encryption and if the domain existed, the ransomware aborted execution. This kill switch was outside the attacker's control as anyone could register it and neutralize the ransomware outbreak [6].

D. Category 4

- Key can be retrieved from a central location such as a C&C server on a compromised host or gleaned with some difficulty from communication between ransomware on the host and the C&C server. For instance, in the case of `CryptoLocker`, authorities were

TABLE I
RANSOMWARE CLASSIFICATION

Ransomware Variant	Year	Classification	Primary Reasoning
Nemucod	2016	Category 1	Displays ransom note before actual encryption [20]
AIDS	1989	Category 2	Decryption key extracted from ransomware code [30]
DirCrypt	2014	Category 2	Used same RC4 keystream for multiple files [20]
Poshcoder	2014	Category 2	Decryption key extracted from ransomware code [20]
TorrentLocker	2014	Category 2	Used same key and IV for multiple files [31]
Linux.Encoder.1	2015	Category 2	Timestamp used to generate keys can be used for decryption [20]
Jigsaw	2016	Category 2	Decryption key extracted from ransomware code [19]
desuCrypt	2018	Category 2	Used same RC4 keystream for multiple files [32]
RaRuCrypt	2018	Category 2	Decryption key extracted from ransomware code
CryptoDefense	2014	Category 3	Decryption key not securely deleted on host [13]
CryptoWall	2014	Category 3	Ineffective if it cannot reach the C&C server [11]
CTB-Locker	2014	Category 3	Ineffective if it cannot reach the C&C server [33]
Locky	2016	Category 3	Ineffective if it cannot reach the C&C server [34]
KeRanger	2016	Category 3	Ineffective if it cannot reach the C&C server [35]
zCrypt	2016	Category 3	Ineffective if it cannot reach the C&C server [36]
HydraCrypt	2016	Category 3	Decryptor available [37]
WannaCry	2017	Category 3	Global killswitch renders ransomware ineffective [6]
GPCoder	2005	Category 4	Weak custom encryption algorithm [16]
PowerWare	2016	Category 4	Decryption key extracted from plaintext communication with C&C server [38]
CryptoLocker	2013	Category 6	No known weakness exists in the ransomware [39]
Petya	2016	Category 6	No known weakness exists in the ransomware
Crysis	2016	Category 6	No known weakness exists in the ransomware
Cerber	2016	Category 6	No known weakness exists in the ransomware [40]
RAA	2016	Category 6	No known weakness exists in the ransomware [41]
NotPetya/GoldenEye	2017	Category 6	No known weakness exists in the ransomware

able to seize a network of compromised hosts used to spread CryptoLocker and gain access to decryption essentials of around 500,000 victims [43].

- Ransomware uses custom encryption techniques and violates the fundamental rule of cryptography: *"do not roll your own crypto."* It is tempting to design a custom cipher that one cannot break themselves, however it will likely not withstand the scrutiny of professional cryptanalysts [44] [45]. Amateur custom cryptography in the ransomware implies there will likely soon be a solution to decrypt files without paying the ransom. An example of this is an early variant of the `GPCoder` ransomware that emerged in 2005 with weak custom encryption [16].

E. Category 5

- Key can only be retrieved under rare, specialized laboratory settings. For example, in the case of `WannaCry`, a vulnerability in a cryptographic API on an unpatched Windows XP system allowed users to acquire from RAM the prime numbers used to compute private keys and hence retrieve the decryption key [46]. However, the victim had to have been running a specific version of Windows XP and be fortunate enough that the related address space in memory has not been reallocated to another process. In another example, it is theoretically possible to reverse `WannaCry` encryption by exploiting a flaw in the pseudo-random-number-generator (PRNG) in an unpatched Windows XP system that reveals keys generated in the past [47]. Naturally, these specialized conditions are not true for most victims.
- A small subset of files left unencrypted by the ransomware for any number of reasons. Certain ransomware are known to only encrypt a file if its size exceeds a predetermined value. In addition, ransomware might decrypt a few files for free to prove decryption is possible. In such cases, a small number of victims may be lucky enough to only need these unencrypted files and can tolerate loss of the rest.

F. Category 6

- Encryption model is resistant to cryptographic attacks and has been implemented seemingly flawlessly such that there are no known vulnerabilities in its execution. Simply put, there is no proven way yet to decrypt the files without paying the ransom.

If a ransomware satisfies specified conditions in multiple categories above, it should be categorized as the lowest of those set of categories. For example, `Apocalypse` ransomware uses custom encryption (Category 4) but also has

a symmetric key hard-coded in the ransomware (Category 2) and a decryptor is available online (Category 3). Hence, it becomes $min\{4, 2, 3\}$ = Category 2.

Note that such classification becomes challenging not just because a ransomware variant can change categories over time, but also because the same ransomware may have different variants, each belonging to a different category according to its encryption model. Hence, it makes sense to keep track of which variant was grouped under a certain category by specifying an MD5 or SHA checksum while performing the classification.

This classification system is not meant to provide a quantitative score to the ransomware, rather it is an indication of the cryptographic strength of the cryptoviral infection. Hence, we do not consider cases where all master keys were released by ransomware developers due to one reason or the other [43]. While release of all master keys by attackers turns the ransomware from a deadly infection into a mere annoyance, such a condition does not reflect on the cryptographic model of the ransomware—which is what our methodology rates.

Being hit by a Category 3 ransomware implies that files can be potentially successfully recovered without paying the ransom whereas a Category 6 indicates that there is no known method of recovering files without payment. By 'current,' we mean how potent the ransomware presently is. For example, a ransomware variant might have a seemingly effective encryption model and hence a Category 6 at one time, but eventual discovery of implementation flaws in the encryption model might bring it down to a Category 3 or 2.

VII. Classification Results

We classified 25 ransomware samples as shown in Table I using the methodology described above. In the samples classified, we discussed the primary reasoning behind why they belong in a category as a ransomware may meet conditions across different categories.

A classification system can indicate one of the following characteristics:

1) Technical prowess: potency of the cryptosystem.
2) Overall effectiveness: potential ways to recover files without paying the ransom.

The difference between these two becomes clear with the following example. Consider a ransomware variant that has an extremely effective cryptographic model, but master decryption keys have been released by ransomware developers. The technical prowess makes the infection potent however overall effectiveness presently is extremely low because of the availability of master decryption keys. Our classification system only considers the technical prowess of initial cryptographic implementation.

Classification results demonstrate that though Category 6 cryptoviral infections raised the bar in 2016, certain recent ransomware seen as late as 2017 or 2018 are Category 2 or Category 3, as shown in Figure 4, due to poor design or implementation by ransomware developers. For example, `desuCrypt`, seen in 2018, uses the same keystream in encrypting different files while using RC4 encryption which allows victims to decrypt files without proper decryption essentials as long as a victim can produce a file that is encrypted with the same key in its decrypted form [29]. This implies that victims should not be quick to pay the ransom when hit by a new ransomware since it may not be a Category 6 infection and certain file recovery procedures may exist that do not require paying the ransom.

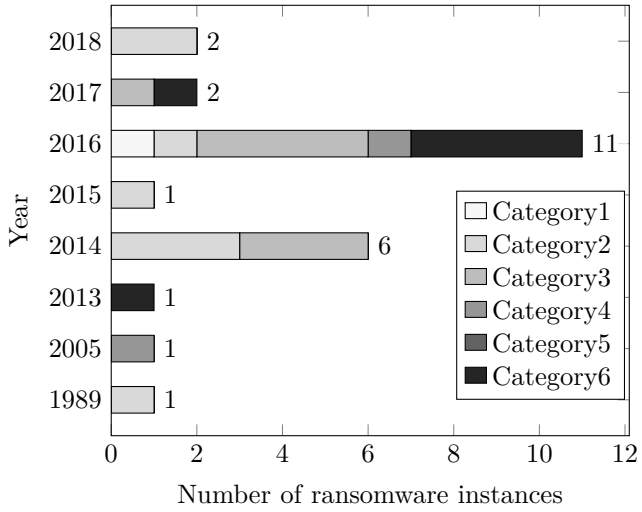

Fig. 4. Ransomware infections over the years.

It is worth noting that this process of classification reveals the true potency of a cryptoviral infection in terms of its encryption model. For example, despite all of the media attention focused on `WannaCry`, it fits in Category 3 due to the embedded global kill switch. `WannaCry` was special primarily because its infection vector exploited an unpatched vulnerability which made it worm-like and differentiated it from other ransomware. Its encryption model, however, was not exceptionally different. Figure 4 also indicates that only a few ransomware variants possess high potency, while the rest contain serious cryptographic flaws. Both Figure 4 and Table I indicate a continued lack of technical prowess in the majority of cryptoviral infections.

VIII. Conclusion

A crucial factor that differentiates a cryptoviral extortion program from a regular on-the-fly encryption program, such as `TrueCrypt` or `VeraCrypt`, is that the decryption key is unknown to the owner. Moreover, the encryption was not commenced or authorized by the user. If a user can obtain access to the decryption essentials in some manner, the ransomware becomes ineffective. It is crucial for modern ransomware to generate a unique encryption key for each victim so that victims cannot cooperate and share decryption keys. Every infection instance needs to be different from the other in terms of decryption essentials,

such as the key. In fact, `WannaCry` uses a different key for every file likely so that if at any point the operation is interrupted and a key is acquired from the host, only one key is compromised which implies that the victim can at most decrypt one file using that current key in memory. Key management is thus a crucial component of effective and potent ransomware operation.

In this paper, we presented the evolution of key management in ransomware. We also introduced a methodology to classify ransomware and used the methodology to classify 25 ransomware samples. Our classification system only considers the technical prowess of the ransomware and not the overall effectiveness. To this effect, a Category 6 ransomware will remain a Category 6 even after its master decryption keys are leaked online since such a leak does not reflect upon the technical capability of the cryptosystem implemented initially. In the future, we plan to expand this work to reflect the overall effectiveness of a ransomware variant so that the general public can use it as a reference to comprehend the potency of a ransomware variant. This will facilitate informed decision making. One could imagine an online ransomware observatory that anybody could query. By acquiring the category of a ransomware, one could comprehend immediately if an easy fix is available or not. In our future work, we also wish to perform an extended analysis on variants to observe if most variants stay in the same category as the original ransomware or if they tend to introduce new vulnerabilities that weakens their category. Table I and Figure 4 indicate that many ransomware developers seem to have little comprehension of cryptographic implementations: we observed that poor cryptographic models appear as recently as 2018. Although with time, it is inevitable that Ransomware-as-a-Service (RaaS) will evolve to the point where more Category 6 ransomware with worm-like propagation capabilities will haunt the Internet. At that time, unauthorized file encryption prevention techniques [10] and detection measures will be the best defense. Scrounging to discover cryptographic flaws in ransomware implementations will be less rewarding.

Our discussions in this paper were focused on the *aftermath* of successful ransomware execution. We assumed that the ransomware has already infiltrated a host machine. It is clear that an error made by the attacker in implementing the cryptosystem, such as neglecting key security, is the only way to reverse the damage without paying the ransom. Ultimately, the solution to the threat of ransomware lies in comprehending key management in ransomware operations.

Acknowledgment

The authors would like to thank the information security team at Michigan State University for their support throughout this project. They are also grateful to the anonymous reviewers for their valuable suggestions towards improving this paper.

References

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Introduction to automata theory, languages, and computation," *Acm Sigact News*, vol. 32, no. 1, pp. 60–65, 2001.
[2] K. Zetter, "Why hospitals are the perfect targets for ransomware," *Wired*, 2016.
[3] A. Greenberg, "The wannacry ransomware hackers made some real amateur mistakes," *Wired*, 2017.
[4] K. Zetter, "What is ransomware? a guide to the global cyberattack's scary method," *Wired*, 2017.
[5] ——, "Hacker lexicon: A guide to ransomware, the scary hack that's on the rise," *Wired*, 2015.
[6] L. H. Newman, "How an accidental 'kill switch'slowed friday's massive ransomware attack'," *Wired*, vol. 13, 2017.
[7] K. Zetter, "4 ways to protect against the very real threat of ransomware," 2016. [Online]. Available: https://www.wired.com/2016/05/4-ways-protect-ransomware-youre-target/
[8] G. O'Gorman and G. McDonald, *Ransomware: A growing menace*. Symantec Corporation, 2012.
[9] A. Young and M. Yung, "Cryptovirology: Extortion-based security threats and countermeasures," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 1996, pp. 129–140.
[10] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 3–24.
[11] K. Cabaj, P. Gawkowski, K. Grochowski, and D. Osojca, "Network activity analysis of cryptowall ransomware," *Przeglad Elektrotechniczny*, vol. 91, no. 11, pp. 201–204, 2015.
[12] A. Palisse, H. Le Bouder, J.-L. Lanet, C. Le Guernic, and A. Legay, "Ransomware and the legacy crypto api," in *International Conference on Risks and Security of Internet and Systems*. Springer, 2016, pp. 11–28.
[13] C. Puodzius, "How encryption molded crypto-ransomware," *welivesecurity Blog, September*, 2016. [Online]. Available: https://www.welivesecurity.com/2016/09/13/how-encryption-molded-crypto-ransomware/
[14] A. L. Young, "Cryptoviral extortion using microsoft's crypto api," *International Journal of Information Security*, vol. 5, no. 2, pp. 67–76, 2006.
[15] A. Gazet, "Comparative analysis of various ransomware virii," *Journal in Computer Virology*, vol. 6, no. 1, pp. 77–90, Feb 2010. [Online]. Available: https://doi.org/10.1007/s11416-008-0092-2
[16] K. Savage, P. Coogan, and H. Lau, "The evolution of ransomware," *Symantec, Mountain View*, 2015.
[17] N. F. Pub, "197: Advanced encryption standard (aes)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.
[18] V. Kotov and M. Rajpal, "Understanding crypto-ransomware," *Bromium whitepaper*, 2014.
[19] P. Aiyyappan, "Jigsaw ransomware demystified," *Vinransomware Blog, November*, 2016. [Online]. Available: http://www.vinransomware.com/blog/jigsaw-ransomware-demystified
[20] B. Herzog and Y. Balmas, "Great crypto failures," 2016.
[21] L. Zeltser, "Blocklists of suspected malicious ips and urls," 2017. [Online]. Available: https://zeltser.com/malicious-ip-blocklists/
[22] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.
[23] J. R. Vacca, *Computer and information security handbook*. Newnes, 2012.
[24] R. Coleridge, "The cryptography api, or how to keep a secret," 1996. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms867086.aspx
[25] 2018. [Online]. Available: https://minotr.net/
[26] 2018. [Online]. Available: http://vxvault.net/
[27] 2018. [Online]. Available: http://thezoo.morirt.com/
[28] 2018. [Online]. Available: https://cuckoosandbox.org/

[29] L. R. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege, "Analysis methods for (alleged) rc4," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 1998, pp. 327–341.

[30] A. L. Young and M. Yung, "Cryptovirology: The birth, neglect, and explosion of ransomware," *Communications of the ACM*, vol. 60, no. 7, pp. 24–26, 2017.

[31] M. Léveillé, "Torrentlocker: Ransomware in a country near you (2014)."

[32] L. Abrams, "desucrypt ransomware in the wild with deuscrypt and decryptable insane variants," *Bleeping Computer Blog, January*, 2018. [Online]. Available: https://www.bleepingcomputer.com/news/security/desucrypt-ransomware-in-the-wild-with-deuscrypt-and-decryptable-insane-variants/

[33] J. Wyke, S. E. T. Team, and A. Ajjan, "The current state of ransomware," *SophosLabs technical paper*, 2015.

[34] R. Richardson and M. North, "Ransomware: Evolution, mitigation and prevention," *International Management Review*, vol. 13, no. 1, p. 10, 2017.

[35] C. Xiao and J. Chen, "New os x ransomware keranger infected transmission bittorrent client installer," *Palo Alto Networks Blog, March*, 2016. [Online]. Available: https://researchcenter.paloaltonetworks.com/2016/03/new-os-x-ransomware-keranger-infected-transmission-bittorrent-client-installer/

[36] "zcrypt ransomware: under the hood," *Malwarebytes Labs Blog, June*, 2016. [Online]. Available: https://blog.malwarebytes.com/threat-analysis/2016/6/zcrypt-ransomware/

[37] L. Abrams, "Emsisoft releases a decrypter for hydracrypt and umbrecrypt ransomware," 2016. [Online]. Available: https://www.bleepingcomputer.com/news/security/emsisoft-releases-a-decrypter-for-hydracrypt-and-umbrecrypt-ransomware/

[38] S. Mansfield-Devine, "Ransomware: taking businesses hostage," *Network Security*, vol. 2016, no. 10, pp. 8–17, 2016.

[39] J. Cannell, "Cryptolocker ransomware: What you need to know," *Malwarebytes Labs*, 2013.

[40] hasherezade, "Cerber ransomware: new, but mature," 2016. [Online]. Available: https://blog.malwarebytes.com/threat-analysis/2016/03/cerber-ransomware-new-but-mature/

[41] L. Abrams, "The new raa ransomware is created entirely using javascript," 2016. [Online]. Available: https://www.bleepingcomputer.com/news/security/the-new-raa-ransomware-is-created-entirely-using-javascript/

[42] ——, "Decryptor for the apocalypse ransomware released by emsisoft," 2016. [Online]. Available: https://www.bleepingcomputer.com/news/security/decryptor-for-the-apocalypse-ransomware-released-by-emsisoft/

[43] M. Ward, "Cryptolocker victims to get files back for free," *BBC News, August*, vol. 6, 2014.

[44] P. Zimmermann *et al.*, "An introduction to cryptography," *Network Associates*, 1999.

[45] R. Verdult, *The (in) security of proprietary cryptography*. Sl: sn, 2015.

[46] A. Guinet, "Wannacry in-memory key recovery," 2018. [Online]. Available: https://github.com/aguinet/wannakey

[47] L. Dorrendorf, Z. Gutterman, and B. Pinkas, "Cryptanalysis of the random number generator of the windows operating system," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, p. 10, 2009.

[48] D. Boneh *et al.*, "Twenty years of attacks on the rsa cryptosystem," *Notices-American Mathematical Society*, vol. 46, pp. 203–213, 1999.

[49] R. Valdez and M. Sconzo, "Threat alert: Powerware, new ransomware written in powershell, targets organizations via microsoft word — carbon black," 2016. [Online]. Available: https://www.carbonblack.com/2016/03/25/threat-alert-powerware-new-ransomware-written-in-powershell-targets-organizations-via-microsoft-word/