Dynamic Distributed Secure Storage Against Ransomware

Jason Castiglione, *Member, IEEE*, and Dusko Pavlovic, *Member, IEEE*

***Abstract*— In just a few years, ransomware evolved into one of the most pernicious threats on the web. From hijacking private disks, the cybercriminals moved to disabling hospital networks, while the cyberwarriors launched destructive cyberwar exercises masquerading as ransomware. To match the variety of attacks, there is also a variety of promising proposals for the mitigation of the ransomware problem by disrupting the attack cycle at various points. None of them seems to be eliminating the vulnerability of static nodes in dynamic networks. We put forward the idea that ransomware is a symptom of a broader problem of architectural imbalance in social computation, while the processes are dynamic and nonlocal, the storage is static and local. We study and discuss some paths toward dynamic, nonlocal, and secure storage. Furthermore, we provide a toy method for locally encrypting the data that can provide a balance of high security and encryption speed.**

***Index Terms*— Computer crime, computer security, decoding, distributed computing, encoding.**

I. Introduction: Why Is There Ransomware?

In THE recent years, ransomware has emerged as a significant threat across all levels of use, from individuals, hospitals, and banks, to government institutions and organizations [1]–[4]. It continues to spread, since ultimately it is profitable. The passive storage architecture enables malicious executables to hijack the locally stored data. Until recently, the stolen data was siphoned away and monetized on the black market. Ransomware enabled criminals to profit from hacking into systems where the data could not be sold to other criminals. They realized the data had value to the owners.

Why is hijacking local storage and ransoming it to the owner much easier than attempting to ransom the communication links? The answer to this question is full of intrigue and goes back to the design of the Internet. The origin story of the Internet is motivated by the cold war [5], [6]. The United States of America and the Union of Soviet Socialist Republics were in an arms race. They were designing nuclear command and control systems to insure if one side launched nuclear ballistic missiles then the other would respond in kind. The main concern was designing a communications network that was resilient to nuclear attack so that it may delay the launch of a retaliation strike in case of any erroneous warnings. P. Baran of *RAND Corporation*[1] studied the problem and, in multiple reports [7], started architecting a new system where the switching nodes stored minimal information. The work at RAND in the 1960s made a paradigm change in the communications systems so that a large message would be broken up into smaller ones of a fixed size and be passed along until it reached its destination.

As mentioned above, the Internet was born of social and political challenges. It answered the requirement of a robust communications infrastructure that could survive an all-out nuclear war. A consequence of designing such a robust communications network is that criminals now have access and are able to reach beyond their backyard. We argue that ransomware and denial of access to data present us with the next challenge. It is imperative to design a resilient and secure storage system that can survive local attacks and the degradation of service.

In this paper, we focus on ransomware [8], [9], which is a type of digital crime that is essentially theft[2] of information followed by demanding a ransom from the victim to regain access. We recommend a paradigm change, akin to the ARPANET project, with regards to a broadly deployed network storage system. The intent is to find a solution which addresses: 1) the financial incentive for ransomware attacks and 2) the difficulty of securing a system from an ever-evolving social/technical attack matrix. In addition, we take into account the restraint that any solution must be cost-effective.

Thus, we submit that the architecture with: 1) local, single-copy storage of valuable confidential data and 2) local control of program executions, often irreversible, is indefensible. Either 1) needs to be relaxed, so that the confidentiality requirement can be satisfied by nonlocal, cloud storage. Otherwise, 2) needs to be relaxed, and the executions of potentially harmful payloads should be removed from the user's hands.

The confluence of problems (1) and (2) is due to legacy and commercial interests. Changing the architecture is much

Manuscript received November 7, 2018; revised April 27, 2019; accepted June 18, 2019. Date of publication July 24, 2019; date of current version January 13, 2021. This work was supported in part by NSF through the Scholarship for Service Award under Grant 1650862 and in part by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-15-1-0263. *(Corresponding author: Dusko Pavlovic.)*

J. Castiglione is with the Department of Electrical Engineering, University of Hawai'i at Mānoa, Honolulu, HI 96822 USA.

D. Pavlovic is with the Department of Information and Computer Sciences, University of Hawai'i at Mānoa, Honolulu, HI 96822 USA (e-mail: dusko@hawaii.edu).

Digital Object Identifier 10.1109/TCSS.2019.2924650

[1] *"RAND"* originally referred to the *"Research ANd Development Project,"* established in 1945 by the commander of the United States Air Forces (USAF), General H. H. Arnold, as a framework for planning the cooperations between the military and private researchers and developers. The project evolved into a contract that USAF gave to the Douglas Aircraft Company in 1946, which spun out into an independent nonprofit organization in 1948. With many legendary researchers on staff, and even with its name pointing to the historical roots of the defense-funded research, the RAND Corporation has played a crucial role in strategic thinking and technologies ever since.

[2] It is noted that this differs from just illegally accessing and copying information, in the sense that the information owner is denied access by ransomware.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

easier and cheaper than defending it. Resolving problem (2) by restricting what the users can execute has been introduced effectively and resourcefully when this was needed for digital rights management (DRM) and for the software IP protections, as well as when such restrictions provided a foundation for the market of *approved* apps. Resolving problem (1) by assuring the confidentiality of nonlocal (cloud) storage is an interesting challenge which requires reconciling technical investments into the solution and the rational confidentiality requirements.

II. What Is Ransomware?

We attempt to define ransomware as an attack strategy and to place it in the space of other similar attack strategies. The goal is to classify all possible defense vectors. Security can be viewed as an adversarial process involving an attacker, a defender, and an asset. In most cases, the asset is of value both for the attacker and for the defender, and that provides the incentives for both. The characteristic property of a *ransom attack* is that the asset itself has no direct utility for the attacker, but only for the defender. Attacker's interest in the asset is indirect: his goal is to hijack the asset temporarily and to sell it back to the defender. In cybersecurity, such ransom attacks originated from research in *cryptovirology* [8].

The experience of piracy, from ancient to modern times, suggests that the hijackings for ransom continue as long as there are:

1) attack vectors against the defender;
2) safe havens the attacker against retaliation;
3) market to monetize the ransom.

Eliminating 2) the safe havens for the perpetrators of ransomware or disrupting 3) their monetization channels requires large-scale *legal* operations. We devote attention to 1) the *technical* problem of attack vectors. Our intention is to maintain the availability of the data by fortifying our system and replicating the information across multiple servers. We simultaneously drive up the cost to attack while lowering the value of the data to the defender.

The first observation is what truly separates ransomware from ransom attacks seen prior to the digital age. Julius Caesar was kidnapped and a ransom was requested around 75 BCE. There were only one Julius Caesar and no method to clone him which would have rendered the captured Julius unnecessary. With regards to the information on a computer, if there is a copy of the ransomed data available to the defender, then there will be no need to pay any ransom. Thus, we see the replication lessens as the value of the ransomed data to the defender.

The second observation is that ransomware infections occur the same as in the broader class of malware. A user may receive targeted spam email luring them to click on a link or download a file which subsequently infects their computer, e.g., Cryptolocker, Cerber, Jaff, and Sage. There have also been instances where the malware propagates as a computer virus or worm using the same attack vectors like other forms of malware, e.g., Spora, WannaCrypt (also known as WannaCry), and Petya. Preventing an infiltration thus boils down to a mixture of the general measures for intrusion prevention with an emphasis on the social aspect of informing users to be more security conscious.

Once the infiltration occurs, ransomware can deny access using multiple strategies. Crypto ransomware, e.g., Cryptolocker, uses encryption to deny the victim access to their data. The malware will search the victim's device for the files matching certain patterns, then locally encrypt the files, and then will either exfiltrate the key or hide it locally. Locker ransomware, e.g., CryLocker, denies the victim access to their computer system. This can happen in multiple ways, e.g., resetting user passwords, locking the user interface, disabling keyboard input, and so on. There are also variants of malware which appear to be ransomware on the surface, but may actually delete the data without the ability to recover. They still demand a ransom, but never allow the victim to regain access. The common thread of all variants of ransomware is they *locally* deny access to resources valuable to the victim.

Immediately, we observe multiple security issues which are completely social in nature. No matter how secure is a system, it is not feasible for it to deduce whether every user's action is legitimate. In addition, we must also convince a user to adhere to good security policy, e.g., applying updates as available. Thus, we see that no matter how effective a firewall may be, its protections become irrelevant once the malware penetrates the system.

However, implementing a ransom attack requires a peculiar interaction protocol: the attacker must announce the hijacking and require the ransom; and he must be able to securely collect the ransom. In other words, the attacker must design, propose, and implement an *exchange contract*, whereby the defender should receive the hijacked asset, and the attacker should receive the ransom. This contract is in principle *asymmetric*, in the sense that the defender runs the risk of losing the valuable asset, whereas the attacker has a strategic initiative and often even requires that the defender's part of the contract is executed first. Since defender's valuation of the asset surpasses the requested ransom, accepting the contract is the rational choice. Disrupting the establishment of the contract or the execution of the exchange is thus not a rational strategy for either player.[3]

III. Ransomware Defense Strategy

The upsurge of attacks within a crime type at a given state of development of a social system is usually:

not *exclusively caused* by an invention of a new attack technique;

but a *combined result* of technical and economic developments in the social system as a whole.

If the defenders' possessions gain value, they must spend more to defend and maintain possession. Similarly, as the attackers have more incentive to attack, they will use more resources to capture the defender's valuables. Thus, we see the impetus for change with regard to security can be linked

[3]The governments often disrupt the execution of the exchange protocol to deter repeated hijackings. Although the cost of loss for the defender of an executed hijacking is infinite, the risk is therefore usually unacceptable; unless the attacker makes a mistake and implements a vulnerable exchange protocol, the governments trade in the high *individual* risk of the victims of the *present* hijackings against a decrease of the *social* risk from the *future* hijackings, which is purported to occur when a "we-do-not-negotiate" social strategy is imposed.

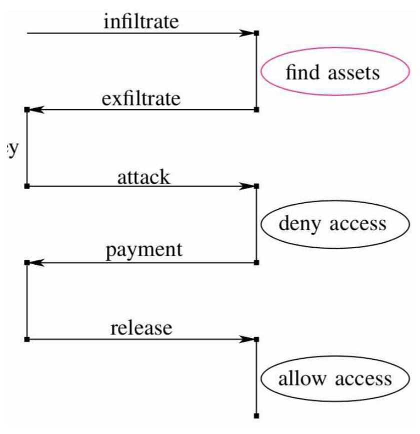

Fig. 1. Ransomware process.

to technical and economic rationale. In the technical domain, new attacks on the defender or vulnerabilities in the defense may be discovered. Otherwise, economic incentives may lead to increased defenses or the use of previously costly attacks. Studying only one side of the problem is equivalent to developing tactics without strategy.

For a particular security problem, this means that the efforts toward confronting a particular newly emerged family of attacks through a particular family of defenses, invented for the purpose, is the familiar strategic mistake, akin to confronting battling the cavalry frontline of a military expeditionary force, while ignoring the infantry behind the hill, and the navy behind the horizon, that is, the tactics without strategy phenomenon. This leads us to a balanced strategy in deterring crime, by either making it harder to commit the crime or reduce the financial incentive for committing a crime. We propose that applying this mindset to digital crimes will provide new methods of reducing crime.

The task of defending against ransomware is of particular interest from the standpoint of *social computation*, because it is clear that a solution cannot be found either in the realm of *social* interactions alone, where the ransom attacks have been launched since the early days of mankind, and have not been eliminated; *or* in the realm of *computation* alone, since a purely technical solution of ransomware could only be based on a purely technical solution of malware in general. The problem of ransomware is a typical security problem that straddles both the social and the computational component of *social computation*.

The ransomware protocol is illustrated in Fig. 1 and provides multiple opportunities for defending from ransomware. For example, protections from *infiltrate* vary from firewalls, spam filters, antivirus, and so on. Two points of concern with defense at this stage are in regards to social engineering and the social processes which result in imperfect code. A significant source of ransomware intrusions can be traced to emails with either malicious links or attached documents. The only vulnerability in the system was that the actual user falls prey to the attack by clicking on a malicious link or document. This is an example of a *level-above attack*, i.e., an attack that is completely legitimate with regards to the functionality of the computer yet violates the intended operation of the system.

An underlying motivator for this paper is to pursue strategies that help eliminate the game between attacker and defender, instead of the endless cycle of increased attacks/defenses. A prime example of a failed methodology is antivirus software which uses a database to lookup known fingerprints of malware to determine whether a piece of code is malicious. The attacker in this game simply rewrites the malware to perform the same function yet appear slightly different. The antivirus company then eventually receives a report of this new malware and adds a new signature to their table. We argue to reconstruct the attack surface, so it is not profitable for ransomware to attack.

Each stage in the ransomware process shares the same defense strategy as any intrusion-based cybercrime. The difference in ransomware is whether the defender will pay the ransom. Thus, we attempt to quantify what would the defender pay for security to negate the effects of any ransomware. We present two strategies to a defender. One attempt to assess the cost of paying a ransom to regain system access, and the other proposes a solution based on backing up the associated computer systems.

The first strategy is to pay the ransom. The average ransom demand per infection [4] was approximately $544 in 2017. Although we note, a ransom of about one million dollars was paid by South Korean Web Hosting Company Nayana for 153 Linux servers infected with an Erebus ransomware variant. Along with the ransom demand is the business cost associated with system recovery, which is nominally more expensive than the actual ransom demand. Furthermore, paying the ransom is not a guarantee of regaining access to locked systems, as seen in the Petya infections.

The second strategy is to frequently back up data and operating system (OS)/program files on distributed servers. The OS and specific program files will be part of a backup image stored on distributed servers, not necessarily encrypted. The sensitive data are encrypted and stored in remote secure storage facilities. In an attempt to protect from ransomware, which is an attack on availability, we must be careful not to expose the system to attacks on confidentiality and integrity. For replication and remote storage of potentially sensitive personal data to be an acceptable solution, fast encryption can address confidentiality.

Given a cyber intrusion where the system is either locked or data are encrypted, then a full system reset is performed. Storage estimates at a dedicated cloud provider are approximately $100 per terabyte per year. Given that backup storage is recommended regardless of ransomware, one might consider the cost as part of the business. The remaining associated cost is with downtime attributed to discovery, reimaging hard drives, and decrypting sensitive data.

From a financial perspective, we argue that the second strategy is advantageous in the long run. Since this is a case of a repeated game, it might be considered as a minimal cost to pay a ransom, but in the long term, any ransom paid to attackers will only further advance their campaign. In this paper, we address encryption which will take into account considerations that arise from requiring frequent backups of a large amount of data.

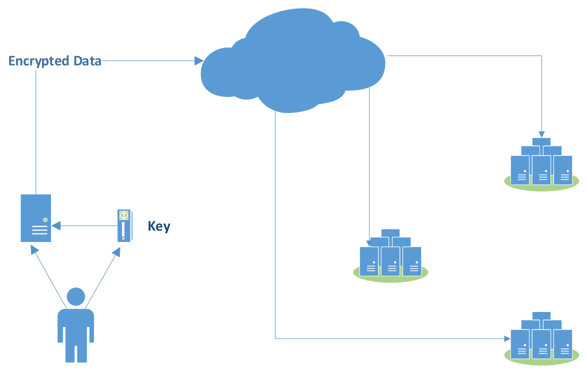

Fig. 2. Dynamic distributed secure storage.

IV. Dynamic Distributed Secure Storage

Now, we architect multiple components of a system for dynamic distributed secure storage (DDSS). The main points of concern are confidentiality, resiliency, and security of the stored data. Confidentiality will be assured by fast local encryption, so commercial cloud providers do not have access to sensitive data. To address the resiliency of data, we reference recent work on the replication of data over multiple servers with protection against erasure data. We do not address the security of cloud enclaves in this paper and, therefore, reference current overviews of infrastructure security, such as [10].

The model of DDSS is to securely encrypt the data prior leaving the local network. In order to encrypt, there will be a separate device which permanently stores the key and has a limited interface to the local computer to mitigate ransomware attacks on the key.[4] Data is then sent to cloud providers that distribute the encrypted data to multiple servers in such a way, dependent on the desired redundancy, that if up to a certain percentage of the nodes are compromised, the data can be restored then redistributed to new nodes. This is depicted in Fig. 2.

A. Fast and Secure Local Encryption

We consider the following requirements that cannot be satisfied by any of the current crypto concepts:

1) the encryption should not incur any efficiency cost;
2) speed should not incur any security loss.

The apparent contradiction of these two requirements is resolved by observing that a third requirement that is normally imposed on standard cryptosystems can be relaxed for cloud cryptosystems: the decryption can be significantly slower than the encryption. The reason is that the encrypted backups only need to be decrypted when the user needs to revert to an earlier version, because of an error or accident or because of a ransomware attack.

We provide a form of encryption that is based on a cryptosystem which achieves *perfect secrecy*, i.e., the one-time pad. The one-time pad is chosen for its simplicity and ease of analysis.

[4]Access to the key is required for data recovery.

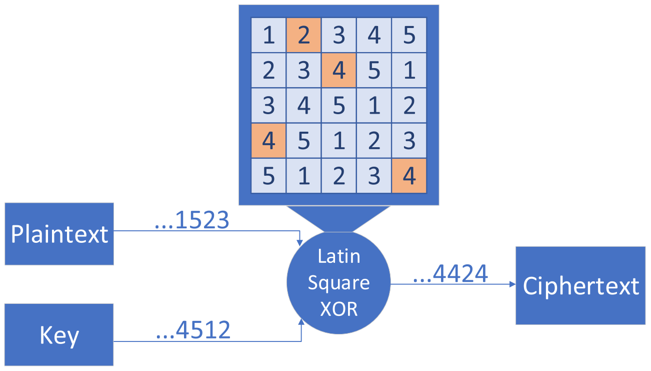

Fig. 3. One-time pad with corresponding Latin square.

B. One-Time Pad and Latin Squares

The one-time pad is first known to be documented by Frank Miller [11] in the late 1800s. The essence of the cryptosystem is that one shares a sequence of numbers, i.e., the key, to be used one time to encrypt a message, as shown in Fig. 3. The system consists of an alphabet, plaintext, key, ciphertext, and a table, called a Latin square. The *alphabet* determines the symbols used in the plaintext, key, ciphertext, and Latin square. The *plaintext* is the message that is desired to be secretly transmitted. The *key* is the shared secret between users, which can be used only one time, so they may secretly transmit information. The *Latin square* is a lookup table which determines how to combine any two elements of the alphabet and has no repeated symbols in a row or column. As shown in Fig. 3, 2 and 3 combine to give 4, which is highlighted in the table. For shorthand moving forward, given a Latin square as a matrix, say L with entries $l_{i,j}$ we represent the lookup as $L(2, 3) = l_{2,3} = 4$. Thus, the column is chosen based on the plaintext, and the row based on the key.

It is essential in generating the key, such that each symbol is equally similar so that the system has perfect secrecy. We say that a cryptosystem has *perfect secrecy*, if the ciphertext provides no more information about the plaintext that was already known. In particular, the cryptosystem has perfect secrecy if and only if

$$\text{Prob}(\text{Plaintext} \mid \text{Ciphertext}) = \text{Prob}(\text{Plaintext}).$$

Another question may be asked, suppose the attacker has the key and ciphertext, but only partial information about which Latin square was chosen. How well can the attacker guess the plaintext? This question was posed in [12] and leads us to multiple cryptosystems with fast encryption, potentially slow decryption, and high secrecy.

C. One-Time Pad and Deletion-Based Cryptosystem

We now provide a high-level description of the algorithm as shown in Fig. 4. We now show how to encrypt, as shown in Algorithm 1, a block of N symbols with an alphabet of size k, where P_i is for individual plaintext, and $\ell_{i,j}$ is a $k \times k$ binary matrix which represents the key, and the positions of the Latin square that will be deleted. Let d denote the number of zeros in the key, i.e., the number of deletions.

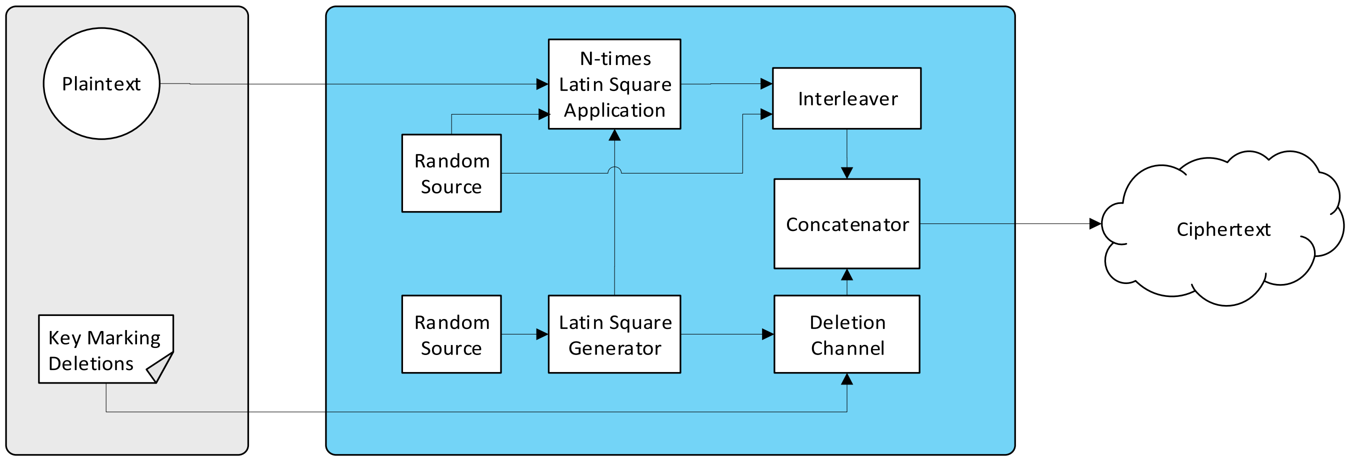

Fig. 4. Fast encryption based on one-time pad and deletions.

Prior to encryption, the intended receiver and sender share a key consisting of a $k \times k$ binary matrix. We will see shortly the ones in this binary matrix will determine what is transmitted to the sender. Then, the sender randomly generates a $k \times k$ Latin square, say L, with the given alphabet. Methods to uniformly generate Latin squares are addressed in [13] and more recently [14]. In addition, the sender generates N uniformly random symbols from the alphabet, say, $K_1, K_2, \ldots, K_N$. Using the previously generated $k \times k$ binary matrix, the sender transmits the elements in the Latin square, L, where the associated binary symbol is a 1. Now, the sender uses the Latin square to calculate $L(K_i, P_i)$, where $P_1, P_2, \ldots, P_N$ represents the plaintext. After which the sender transmits the concatenation $K_i||L(K_i, P_i)$.

Then to decrypt, the intended receiver had recorded the received symbols of the Latin square and places them in an empty grid using the key, i.e., the binary $k \times k$ matrix. The receiver then uses techniques from Sudoku to solve for the remaining symbols of the Latin square, such as Algorithm X from [15]. For the finale, the receiver simply uses the solved Latin square to calculate the plaintext.

When transmitting N plaintext from an alphabet of k elements, the speed of encryption is dependent on the ability to: 1) uniformly generate a $k \times k$ Latin square; 2) uniformly generate N symbols from the alphabet; and 3) N Latin square lookups. Generation of the Latin square and N symbols can be done beforehand. Thus, the complexity is mainly dependent on the implementation of the Latin square and requires memory of $k^2 \times \log_2(k)$. The complexity of a lookup is constant time and is hardware dependent. The complexity of decryption requires solving of a $k \times k$ Latin square with missing d symbols. In using *Dancing Links* [15], it can be implemented with a "grid-shaped" linked list with $3n^2 + 3^2d$ nodes, where each node has four incoming and outgoing pointers. The Latin square is elegantly solved by a brute force search that removes links representing inconsistent dependencies. When it reaches a dead end, it backtracks to the last choice made and chooses differently.

Algorithm 1 Encryption Using One-Time Pad With Deletions

```
Data: $P_1, P_2, \ldots, P_N; \ell_{1,1}, \ldots, \ell_{k,k}$
Result: Transmit
begin
    Generate uniformly random $K_1, K_2, \ldots, K_N$ in
      $\{1, 2, \ldots, k\}$ ;
    Generate uniformly random $k \times k$ Latin square,
      $L = (l_{i,j})$ ;
    for $i, j \in \{1, 2, \ldots k\}$ do
        if $\ell_{i,j} == 1$ then
            Transmit $l_{i,j}$
        end
    end
    while $P_i$ at input do
        Transmit $K_i$ ;
        Transmit $L(K_i, P_i)$ ;
    end
end
```

The crucial feature of the problem at hand—that the decryption is performed only for attack recovery, whereas the encryption is performed continuously, and must incur no significant computational expense on the cloud storage—limits the applicability of the standard complexity on the proposed framework. While the detailed security analysis is left for future work, it is immediately clear that increasing the block size improves the efficiency of the system, at the cost of security. A straightforward quantitative analysis would allow the user to determine the system parameters adequate to the requirements of the application at hand.

D. Resilient Distributed Storage

In recommending distributed storage to all users, the amount of data being stored throughout networks with limited bandwidth increases, new problems to solve arise in how to accomplish distributed storage in a dynamic environment. Four main issues arise in this scenario:

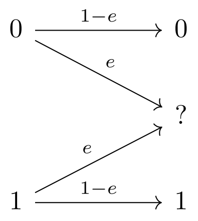

Fig. 5. Erasure channel with erasure probability of e.

1) How do we add redundancy to the stored data?
2) How do we distribute the data over multiple nodes?
3) How do we recover corrupted data?
4) How do we redistribute the data as servers are removed and added?

Thus, the questions posed illustrate that the methodology must adapt as the environment changes. It is not immediately clear whether the aforementioned problems can be solved separately. In particular, if one first calculates the required redundancy and designs an error correction code that achieves the specification, then would simply spreading the data over servers as they go online/offline provide the optimal solution?

In [16], they consider how to add redundancy in such a way that helps to increase the efficiency of redistributing encoded data as additional servers come online. The model of adding redundancy is based on error correcting codes for the erasure channel. A brief overview of considerations in applying erasure error correcting codes to distributed storage is provided in [17] and [18]. In [19], they design a new class of codes and compare them with Reed–Solomon codes. Their codes are applied to the use case of a Hadoop HDFS, and they claim a reduction in repair disk I/O and network traffic associated with the repairs. Their codes require 14% more storage than Reed–Solomon codes. Both of the previously mentioned code designs have benchmarks focused on minimizing I/O and network traffic associated with the repair. They are interested in minimizing the effect of correcting errors on the system as a whole. Below, we provide an example to illustrate the process.

1) Background: When data is stored on a file system, there may be errors when writing to a disk, or there may be an external trigger like a power surge which may cause bits to be erased. The simplest channel model for erased bits is the erasure channel, as shown in Fig. 5. It is simple in that sense that it is memoryless and erases the current symbol independently of its previous actions.

A natural question arises, is it possible to reliably transmit information across this channel? Shannon answered this question affirmatively in [20] and provided a rate at which one could reliably transmit information. The capacity of the binary erasure channel is $C = 1 - e$. This rate will help us evaluate how well a particular code performs. The most optimal of storage schemes for k bits, given an erasure probability of e, would then have a total storage requirement of approximately $n = (k/(1 - e))$ bits.

2) Using a Repeat Code for Distributed Storage: We now perform a simple analysis of a r-Repeat code, which simply means we repeat each symbol r times. First, we note the rate of this code is $(1/r)$ and calculate the probability of an error as $Prob(\text{error}) = p(r \text{ erasures}) = e^r$

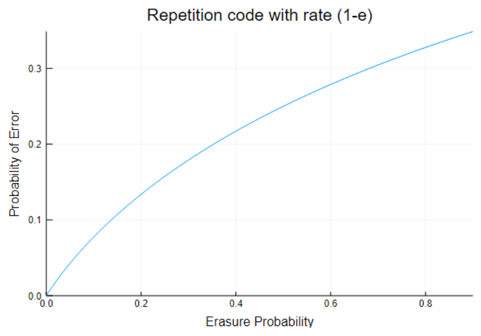

Fig. 6. Probability of error versus erasure probability for repetition code.

We can quickly observe for $e \approx 1$ the probability of error is high. This will clearly not be an optimal code which is illustrated in Fig. 6, although we will use the repetition code to illustrate the basic concepts of [16].

Suppose we would like to encode a data bit using an r-repeat code. Then, there will be a total of $n = r$ bits to store. Next, we distribute the data onto n nodes. If we have a total of m data bits to encode, then we may use the repeat code to store $m \cdot n$ data bits on n nodes. If the data server provider adds a new server for extra redundancy, then we may choose to transfer the data from the n existing nodes into the new server. There are multiple strategies we may choose.

3) Broadcast: In the broadcast case, we can have each existing server send their m bits to the new server, which will ultimately have $m \cdot n$ bits transferred to the new server. This method is decentralized and requires no coordination between servers.

4) Coordination: On the other hand, we may implement a distributed consensus algorithm, where the individual (honest) servers each play a game of matching pennies, i.e., draw random bits, and if they match one player wins, if not then the other wins. Given $n = 2^j$, this algorithm will require j rounds, and the ith round transmits 2^i bits. Thus the total bits transmitted will be $m + \sum_{i=1}^{j} 2^i = m + 2^j - 1 = m + n - 1$. This is a little bit of an improvement. It is also possible to always choose a particular server, and then the overhead is simply m bits, the ability to select a random server has multiple benefits, e.g., security and data diversity.

5) Precoordination: Another possibility is to distribute *a priori*, a rank m, $m \times n$ binary matrix. Then, the ith server may calculate the ith bit of the product and send it to the new server. The new server may then recover the m bits. This requires sending only m bits to the new server and has the advantage that it requires no coordination.

V. Related Work

Recent studies in defending against ransomware have focused on detection and prevention. In [21], an extensive study of 1359 ransomware samples from 2006 to 2014 was performed. In their analysis, they found multiple similarities in the activities of the different ransomware families collected. They provided examples for detection, such as monitoring abnormal file system activity and certain Windows application programming interface (API) calls.

In [22], crypto ransomware is specifically targeted. They design a system, PAYBREAK, to detect and securely store the keys associated with calls to cryptographic primitives on the host system. They assume the ransomware authors either dynamically link to system-provided libraries or statically link to external libraries that provide cryptographic functions. This allows PAYBREAK to monitor calls to cryptographic functions and record the keys that the ransomware used to encrypt the user's files.

We see the move to network detection of ransomware in [23], where the focus is on using software-defined networking to actively monitor and react to malicious activity. In one example, they update a blacklist database with malicious domains. Their intention is to disrupt the connection between an infected host and Command and Control center.

An approach with similar intent to this paper is found in [24]. The focus of their paper is locally duplicating data in available storage space. They attempt to remove the incentive for users to pay ransomware by making extra copies of the information in a local secure file system. The user may then recover any ransomed data that was deemed important.

VI. Conclusion

In conclusion, we have argued for a paradigm change with DDSS as a solution for all the families of ransomware and local data compromises. This is not solely a technical solution, but a strategic decision to lessen the value of ransomed data to the owner since they may recover it elsewhere. A method for fast and secure encryption was presented. In addition, we provide a way to distribute data throughout multiple servers dynamically, so that as nodes become compromised the data can redistribute to new nodes.

References

[1] Reuters, (2018). *Atlanta Officials Reveal Worsening Effects of Cyber Attack*. [Online]. Available: https://preview.tinyurl.com/yd2msqze

[2] The Denver Post. (2018). *Ransomware Strikes CDOT for Second Time Even as Agency Still Recovering from First Samsam Attack*. [Online]. Available: https://preview.tinyurl.com/y7zb5tz3

[3] D. Y. Huang *et al.*, "Tracking ransomware end-to-end," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 618–631.

[4] D. O'Brien, "Internet security threat report: Ransomware," Symantec, Mountain View, CA, USA, Tech. Rep., 2017.

[5] G. Dyson, *Darwin Among the Machines*. Boston, MA, USA: Penguin Books Limited, 2012. [Online]. Available: https://books.google.com/books?id=NY7GlDGToRIC

[6] RAND. (2009). *Paul Baran and the Origins of the Internet*. [Online]. Available: https://www.rand.org/about/history/baran.html

[7] P. Baran, "On distributed communications I-XI," RAND Corp., Santa Monica, CA, USA, Tech. Rep., 1964.

[8] A. Young and M. Yung, "Cryptovirology: Extortion-based security threats and countermeasures," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 1996, pp. 129–140. [Online]. Available: http://dl.acm.org/citation.cfm?id=1947337.1947357

[9] A. L. Young and M. Yung, "Cryptovirology: The birth, neglect, and explosion of ransomware," *Commun. ACM*, vol. 60, no. 7, pp. 24–26, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3097347

[10] G. Cloud. (2017). *Google Infrastructure Security Design Overview*. [Online]. Available: https://cloud.google.com/security/security-design/resources/google_infrastructure_whitepaper_fa.pdf

[11] F. J. Miller, *Telegraphic Code to Insure Privacy and Secrecy in the Transmission of Telegrams*. New York, NY, USA: C. M. Cornwell, 1882.

[12] P. Cameron. (2012). *Bluebells*. [Online]. Available: https://cameroncounts.wordpress.com/2012/05/10/bluebells-2/

[13] M. T. Jacobson and P. Matthews, "Generating uniformly distributed random latin squares," *J. Combinat. Des.*, vol. 4, no. 6, pp. 405–437, 1996.

[14] R. Fontana, "Random latin squares and Sudoku designs generation," *Electron. J. Statist.*, vol. 8, no. 1, pp. 883–893, 2014.

[15] D. E. Knuth, "Dancing links," 2000, *arXiv:cs/0011047*. [Online]. Available: https://arxiv.org/abs/cs/0011047

[16] X. Zhang, Y. Hu, P. P. C. Lee, and P. Zhou, "Toward optimal storage scaling via network coding: From theory to practice," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 1808–1816.

[17] J. S. Plank, "Erasure codes for storage systems," in *Proc. Tutorial USENIX Conf. File Storage Technol.* Berkeley, CA, USA: USENIX, 2005, pp. 44–50.

[18] J. Li and B. Li, "Erasure coding for cloud storage systems: A survey," *Tsinghua Sci. Technol.*, vol. 18, no. 3, pp. 259–272, Jun. 2013.

[19] M. Sathiamoorthy *et al.*, "Xoring elephants: Novel erasure codes for big data," in *Proc. 39th Int. Conf. Very Large Data Bases*, 2013, pp. 325–336. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488335.2488339

[20] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

[21] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Proc. 12th Int. Conf. Detection Intrusions Malware, Vulnerability Assessment (DIMVA)*, vol. 9148. Berlin, Germany: Springer, 2015, pp. 3–24. doi: 10.1007/978-3-319-20550-2_1.

[22] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "Paybreak: Defense against cryptographic ransomware," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, New York, NY, USA, 2017, pp. 599–611. doi: 10.1145/3052973.3053035.

[23] K. Cabaj and W. Mazurczyk, "Using software-defined networking for ransomware mitigation: The case of cryptowall," *IEEE Netw.*, vol. 30, no. 6, pp. 14–20, Nov./Dec. 2016.

[24] K. P. Subedi, D. R. Budhathoki, B. Chen, and D. Dasgupta, "RDS3: Ransomware defense strategy by using stealthily spare space," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–8.

Jason Castiglione (M'15) received the B.S. degree in physics and mathematics from the University of Florida, Gainesville, FL, USA, in 2004, and the M.S. degree in mathematics from the University of Virginia, Charlottesville, VA, USA, in 2006. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Hawai'i at Mānoa, Honolulu, HI, USA.

Dusko Pavlovic (M'95) is Professor of Computer Science at University of Hawai'i at Mānoa, where he is the Director of Adaptive Security and Economics Lab (ASECOLab). His research interests have spanned a broad area of mathematics and computer science, with an emphasis on security.