On Randomization in MTD Systems

Majid Ghaderi
mghaderi@ucalgary.ca
University of Calgary

Samuel Jero
samuel.jero@ll.mit.edu
MIT Lincoln Laboratory

Cristina Nita-Rotaru
c.nitarotaru@northeastern.edu
Northeastern University

Reihaneh Safavi-Naini
rei@ucalgary.ca
University of Calgary

ABSTRACT

Randomization is one of the main strategies in providing security in moving-target-defense (MTD) systems. However, randomization has an associated cost and estimating this cost and its impact on the overall system is crucial to ensure adoption of the MTD strategy. In this paper we discuss our experience in attempting to estimate the cost of path randomization in a message transmission system that used randomization of paths in the network. Our conclusions are (i) the cost crucially depends on the underlying network control technology, (ii) one can reduce this cost by better implementation, and (iii) reducing one type of cost may result in increased costs of a different type, for example a higher device cost. These suggest that estimating the cost of randomization is a multivariable optimization problem that requires a full understanding of the system components.

CCS CONCEPTS

• **Security and privacy → Network security**.

ACM Reference Format:
Majid Ghaderi, Samuel Jero, Cristina Nita-Rotaru, and Reihaneh Safavi-Naini. 2022. On Randomization in MTD Systems. In *Proceedings of the 9th ACM Workshop on Moving Target Defense (MTD '22), November 7, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3560828.3564016

KEYWORDS

Network Security, Moving target defence, post-quantum security, information theoretic security

MTD'22, November 7, Los Angeles, CA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9878-7/22/11…$15.00
https://doi.org/10.1145/3560828.3564016

1 INTRODUCTION

Securing information communication systems, or what is generally referred to as cybersecurity, has been a continuous battle between attackers and defenders. Gene Spafford's frequently cited quote "The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts." makes it abundantly clear that this battle is not going to end and all the defenders can hope for is to stay ahead of the attackers.

Cryptography provides a set of powerful algorithms and protocols that can be used to provide security for networks and controlling access to the system. The well-known and accepted Kerckhoff's principle that is used in the design of a cryptographic system, rules out the notion of "security by obscurity" that promotes hiding the details of the cryptographic mechanisms to achieve security, and requires all algorithmic and implementation details of the system except the secret key, to be public and available to the attacker, such that security depends on the secrecy of the cryptographic key(s) only. A more recent requirement on the security of cryptographic systems is security against an adversary that has access to a quantum computer. This effectively excludes all the existing cryptographic infrastructure of the Internet, which relies on the hard problems of integer factorization and discrete logarithms, for which efficient quantum algorithms are known.

Under such a stringent set of conditions, and since the designers cannot hide the details of the system, introducing uncertainty in the attackers' view of the system by introducing changes to the system configuration and physical description without relying on intractability assumptions, becomes an attractive proposition.

Moving target defence (MTD) systems seek to increase the work of the attackers in breaking the system by creating additional uncertainty for them. Moving target (MT) approaches make the system less static, less predictable, and add diversity to the system components, configuration, and physical layer properties to increase the cost of attacks.

Randomization is the primary technique for introducing uncertainty for the attackers. Randomization not only reduces the success probability of attacks but also, because of the changing nature of the system, reduces the lifetime of the attack and the system compromise. That is, even if the system will be compromised, it could recover from the compromise as time passes. MT techniques have been proposed and implemented even before the term moving target was coined [13].

Evaluating MTD systems for dynamic networks. To evaluate an MTD system one needs to consider their effectiveness and the cost of employing them. That is, one needs to examine possible attacks that would be prevented by the system, and the cost of the dynamic changes to the system that would be added in terms of system performance measures and the cost of additional devices and infrastructure. The effectiveness of the system can be evaluated by developing an abstract model that captures the essential components of the system and their interaction, modeling the adversarial strategies, defining success measures for the adversary, and analyzing the success of the specific MTD strategy against the attacker. Evaluating the cost, that almost always accompanies an MTD strategy, would be measured in terms of parameters such as response time and/or throughput of the system or in terms of additional cost of equipment required for the implementation of the strategy in a realistic setting and evaluated by performing sufficient experiments to provide a realistic cost of using the system in practice.

MTD techniques and our work. In [12], techniques are categorized into five major groups, "dynamic networks, dynamic platforms, dynamic runtime environments, dynamic software, and dynamic data." The focus of this paper is on dynamic networks. We report on our work on *implementing an MTD strategy* that had a theoretical analysis of its security strategy and the very attractive promise of providing post-quantum secure communication without relying on any intractability assumption, the challenges we faced, and the lessons that we learned. Although our focus is on introducing dynamicity into the networks, we believe the general concepts, observations and takeaways would have parallels in other types of MTD systems and would be applicable to them as well.

Outline. In section 2, we introduce the MTD system and its theoretical analysis that motivated our implementation. The system is proved to provide post-quantum security without relying on a secret key or intractability assumption. The system was published in [17] and used an evaluation framework for the MTD strategy that was proposed in [1, 9]. We motivate the need for implementation and experiments to (i) understand the cost of employing the system in practice to refine the model and make the results closer to a real-life implementation, and (ii) to validate the assumptions and evaluate performance of the system in practice.

In section 3, we argue and discuss the implementation of the system using Software Defined Networking (SDN) [10], using a software platform as well as a physical testbed. We show that by improving the implementation strategy, some of initially perceived costs (e.g., overhead of data processing and packet loss) could be removed. However, we note that this improved implementation came at the cost of extra assumptions on the knowledge of network topology and traffic, as well as more sophisticated networking infrastructure (costly switches), pointing out the trade-offs between different cost factors.

In section 4, we reflect on our experience and offer some lessons learned for future MTD systems.

2 PATH-HOPPING AS AN MTD STRATEGY

Securing transmission between two parties requires assumptions on the adversary's computational capabilities.

Shannon's ground-breaking paper in 1949 [18] provided the first formal model for cryptographic systems (encryption) with security against a *computationally unbounded* eavesdropping adversary, and proved information theoretic perfect secrecy of the One-Time-Pad (OTP) encryption system. OTP however has limited use in practice because it requires a new fresh random key for each message, and requires the key length to be the same length as the message entropy.

Computationally secure encryption systems assume the adversary's computational power is bounded polynomially in the size of input, and under this assumption they achieve secure communication using keys that are short and fixed in length, irrespective of message size. Securing communication using these encryption systems starts with a *key exchange protocol* that establishes a shared key between the two parties that will provide a shared secret key that will be used in the encryption algorithm. Secure symmetric key encryption algorithm such as AES (Advanced Encryption Standard) have been efficiently implemented and widely used in practice. Today's key exchange protocols rely on assumptions about the hardness of mathematical problems such as Integer Factorization and Discrete Logarithm problem. Both these problems that are the basis of security of all key establishment protocols that are actively used over the Internet today, have efficient solutions using quantum algorithms [19], and so post-quantum secure key establishment has been an active area of research and development.

2.1 Physical layer assumption for securing transmission

Dolev, Dwork, Waarts, and Yung [5] proposed *Secure Message Transmission (SMT)* systems that provide a model for secure communication over partially corrupted networks. The model provides information theoretic security (i.e., assumes a computationally unbounded adversary) and very importantly, (i) does not require any shared secret key and so does not need any initial key exchange protocol and, (ii) because it does not make any assumption on the computational power of the adversary, it will remain secure against an adversary with access to a quantum computer. These very desireable properties come at the cost of the requiring the physical environment to satisfy a certain assumption, referred to as the *physical layer assumption*. In particular, the assumption is that there exists a set of n node-disjoint paths, called *wires*, between the sender and the receiver, such that only a subset of size t of the wires can be controlled by the adversary. The model can be seen as the continuation of the pioneering work of Aaron Wyner [21] on wiretap channels that used *physical layer assumptions*, in particular noise in the channel, for securing transmission over a (noisy) communication channel.

2.1.1 *SMT Privacy and Reliability.*

SMT protocols are interactive protocols in synchronous networks, and consist of rounds where each round includes two "phases" where in one phase one party sends a transmission to the other. Let Π be a message transmission protocol, M^A denote the message that is selected by Alice, and M^B denote the message outputted by Bob when Π is completed. Security of SMT systems is defined using two properties of *privacy* and *reliability* [6].

Privacy. Π is called ϵ-private if for any two messages m_0, m_1, and for any random choices (coin tosses) r of the adversary, we have

$$\Sigma_c |\Pr[View_A(m_0, r) = c] - \Pr[View_A(m_1, r) = c]| \leq \epsilon,$$

where $View_A(m_i, r)$ is the adversary's view when $m^A = m_i, i = 0, 1$ is sent, the probabilities are taken over the coin flips of the honest parties, and the sum is over all adversary's view.

Reliability. Π is called δ-reliable if, $\Pr(M^A \neq M^B) \leq \delta$, where the probability is taken over the choices of M^A and the coin flips of all parties.

SMT protocols are widely studied and elegant theoretical constructions for optimal protocols have been proposed.

2.2 SMT with Passive Adversary and Dynamic Path Selection

Security of SMT protocols has been studied against passive, blocking, and Byzantine adversaries, as well as a combination of these adversary types. Protection against active adversaries, however is costly. It was proved that reliability requires $n \geq 2t + 1$, and perfect security and reliability require at least two phases. The cost can be seen in terms of high connectivity (e.g., $n \geq 3t + 1$ for 1-phase protocols with perfect secrecy and reliability), the need for interaction (e.g., at least two phases to achieve perfect secrecy and reliability when $n \geq 2t + 1$), and the need for complex processing (encoding and decoding in each phase).

Considering passive adversaries significantly improves the efficiency of SMT protocols. The number of corrupted wires can be increased to $t = n - 1$, and a one phase protocol with very efficient encoding and decoding can be designed.

For example to encode a message m for a set of n wires, one can use an (n, n) secret sharing to generate a vector of n shares, $SMT.enc(m) = SS_{n,n}.Share(m) = (S_1, \cdots S_n)$, and send the share S_i over the wire w_i. Let $m \in Z_q$ where q is an integer. The encoder randomly selects $S_i \in Z_q, i = 1, \cdots n - 1$, and calculates $S_n = m - \sum_{i=1}^{n-1} S_i$ in Z_q. The decoding is by simply adding all components of the vector $SS_{n,n}.Share(m)$. Perfect security of the message transmission follows from perfect security of (n, n) secret sharing.

Considering passive adversaries is also well-motivated by protection against silent APT (Advanced Persistent Threat) adversaries that stay dormant in the network with the goal of collecting information for multistage attacks. By dividing a message into shares and spreading the shares over multiple paths the success chance of the APT adversary will significantly reduced.

2.2.1 *Moving Target Defence (MTD)* .

The *information rate* of an SMT system is defined as the number of bits that must be transmitted to transmit a single message bit. The information of the above encoding system for n wires is $1/n$. For an adversary that eavesdrops t paths however, one can select a subset of $t + 1$ wires and use a $(t + 1, t + 1)$-secret sharing instead. This will improve the information rate of the SMT system to $1/(t + 1)$.

To further improve security and efficiency of the system against a dynamic adversary that changes the set of their eavesdropped paths over time, a Moving Target Defence (MTD) approach to SMT systems was proposed [17].

The MTD system considered transmission of a sequence of messages that are sent one per time interval. While the number of compromised wires t was assumed to be strictly less than n, the encoding considered a parameter k that was not directly linked to t, for sharing and spreading the message in the network. The system works as follows. For each message, a random subset G of k target wires are chosen, and $SMT.enc(m) = SS_{k,k}.Share(m)$ is used to generate k shares that will be sent over the selected target wires. The subset G will be renewed for each message, and as long as the elements of G are not all within the set of adversary's accessible wires, the message will remain perfectly private. This means that even if the attacker can capture $t \geq k$ wires, because of the randomness in the system, the transmission may stay secure. This allows security for the message stream to stay secure as long as the adversary cannot capture all shares of all messages. The security depends on the value k, the number of shares of the message, and $t \geq k$, the number of wires that the adversary can capture at a time. For simplicity of analysis, $t = k$ is assumed.

The analysis of the system, following the theoretical framework of [9], models the MTD system as a stateful game between a defender and an attacker that use random probabilistic strategies. The game changes state after a pair of "actions" taken by the defender and the attacker, and the new state is determined by the previous state and the actions of the two players. The game has $k + 1$ states, labeled by $i \in \{0, 1, \cdots, k\}$, where state i corresponds to the adversary successfully finding the i'th target wires. State transitions characterize a Markov chain, with state k being the winning state in which the adversary learns all the target wires that carry shares of a message, and so the message is compromised.

The concrete analysis considered a defender that transmits one message in each time interval, and *takes action* with probability λ, in which case they modify the set of target wires that are used in the previous time interval by randomly choosing one of the k target wires, and replacing it with another randomly selected wire from the remaining $n - k$ wires. The adversary also acts with probability μ in each time interval, in which case they randomly select one of the wires from those that they have not learned yet. The probability μ depends on λ and a second parameter system τ that is the probability of being detected because of the move.

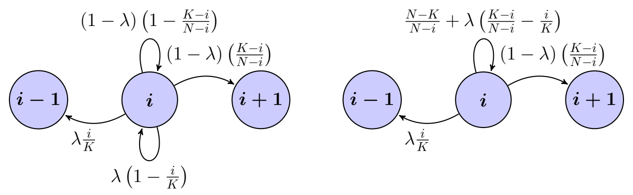
$(1-\lambda)\left(1-\frac{K-i}{N-i}\right)$
$(1-\lambda)\left(\frac{K-i}{N-i}\right)$
$i-1$
i
$i+1$
$\lambda\frac{i}{K}$
$\lambda\left(1-\frac{i}{K}\right)$
$\frac{N-K}{N-i}+\lambda\left(\frac{K-i}{N-i}-\frac{i}{K}\right)$
$(1-\lambda)\left(\frac{K-i}{N-i}\right)$
$i-1$
i
$i+1$
$\lambda\frac{i}{K}$

Figure 1: State transition probabilities. The left diagram shows the defender's moves, λM^D, with arrows bellow and attacker's moves, $(1 - \lambda)M^A$), with arrows above. The right diagram shows the combined transition matrix $M = \lambda M^D + (1 - \lambda)M^A$)

2.2.2 *Security.*

The system evaluation uses two security measures, (i) *the expected number of times that the adversary wins in the first T time intervals,* given by $T.\pi(k)$ where $\pi(k)$ is the Markov chain stationary probability of state k, and (ii) *the expected number of time intervals until the first compromise happens,* denoted by $E^{(1)}_{win}$. These measures were computed using transition probability matrix of the

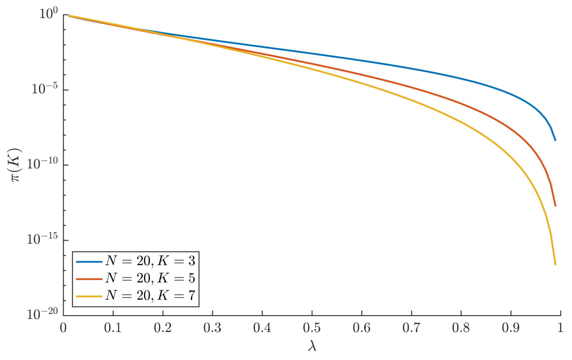

Figure 2: Numerical results for $\pi(K)$ as a function of λ for $N = 20$ and different values of K.

Markov chain, that is obtained using the above random strategies of the defender and the attacker.

Graphs of the numerical values of $\pi(k)$ and $E^{(1)}_{win}$ (Figures 3 and 4,[17]) for typical choices of n, k, and other system parameters showed that increasing λ resulted in the decrease of $\pi(k)$ and the increase of $E^{(1)}_{win}$, and thus higher security. This matches the intuition that faster changes in the system will "confuse" the adversary more and so "increase" security. Figures 3 in [17] is reproduced in Figure 2 below for completeness.

The analysis, however, abstracts out the concrete length of hopping interval and allows the hopping probability to be close to one. That is, it does not consider any cost for path-hopping. To find the concrete security of the system, however, one needs to estimate the actual cost of changing paths in each time interval.

3 IMPLEMENTATION AND EVALUATION

3.1 Implementation Challenges

To implement the path hopping system in practice, one needs to map the theoretical model of "wire" and the notion of "compromised wires" to real-life systems. A "wire" abstracts a network path. A path is a sequence of links that are joined by switches. Path hopping requires route planning, communication of the required information to switches along the paths, and sufficient time for switches to activate the new paths. Note that correct decoding of a (k, k) secret sharing based encoded message requires all shares to be received by the receiver, which means that messages are likely to be dropped due to dropped packets while communicating and activating new paths. To determine reasonable values of λ, then, we needed to implement the system and understand the constraints of path hopping in a real network. Our research question was: *What is the cost of hopping paths in networks, and what is the highest hopping rate that can be used while message recovery stays at an acceptable rate?*

One significant challenge is that networks often do not contain a high degree of path diversity. Although having more than a single path is common, path hopping really requires larger N's (e.g., 6 to 10) than are common in most networks. This is increasingly true towards the edge of the network and especially at the end host, which rarely has more than 1 or 2 NICs. As a result, our evaluation relied on "simplistic" network topologies, like fully connected meshes.

Our most significant challenge was ensuring that shares actually traverse disjoint paths in the network. In traditional IP-based networks, hosts have little to no say about how their traffic is actually routed. In particular, IP routing is hierarchical and based on the destination address only. This means that all traffic for a given destination IP address will be routed in the same manner. This is at odds with path hopping's need to send multiple packets to a destination over different paths. In particular, it means that one cannot convince an IP-based network to send packets to a destination IP address over different routes. There are, however, several possible approaches to ensuring a disjoint set of paths, including source routing and SDN. Using source routing, the sender computes the entire path to reach a destination, and thus can change the path for each packet on the fly, as needed in path hopping. Source routing, however, is not widely supported in practice for a variety of reasons including the additional overhead of encoding path information in the header of every packet. Software Defined Networking (SDN), on the other hand, is a recent paradigm that enables control of the whole network from a logically centralized controller. This controller has the ability to dynamically control paths for specific flows in the network, enabling it to construct the disjoint paths required for path hopping.

3.2 SDN-Based Implementation

We decided to implement our system using software-defined networking (SDN), which enables dynamic reconfiguration of network paths using standard APIs for communication between software-based network controllers and network switches [8]. In SDN, the control and data planes of the network are decoupled from each other. The data plane is implemented on programmable switches that are able to match on a variety of packet fields and perform basic packet forwarding, while the control plane is implemented through a logically centralized SDN controller that coordinates and controls these switches. The SDN controller provides a framework for defining, enacting, and enforcing per-flow policies in a dynamic manner. SDN is widely adopted in practice in both wide area [7] and datacenter networks [20], and SDN-capable switches are commercially available from many vendors [15], making it relatively easy to build an SDN network testbed to experiment with our path hopping protocol. Using the flexible, programmatic control of the network provided by SDN, we can provide the disjoint paths needed for path hopping. In particular, we can observe a path hopping flow, compute disjoint paths between the source and destination, and install routes for each flow that follow disjoint paths through the network, thereby ensuring secure operation of path hopping.

An important question for any path hopping design is how packets for different paths are identified so that the network can forward them appropriately. If the sender and receiver actually have N Network Interface Cards (NICs), then no explicit differentiation is needed. However, most systems have only one or two NICs, so to perform path hopping with $N > 2$, some part of the network packet must identify its path. In our design, we encapsulate message shares in UDP datagrams and use UDP ports to distinguish between paths. This works well with common OpenFlow-based SDNs [14], which

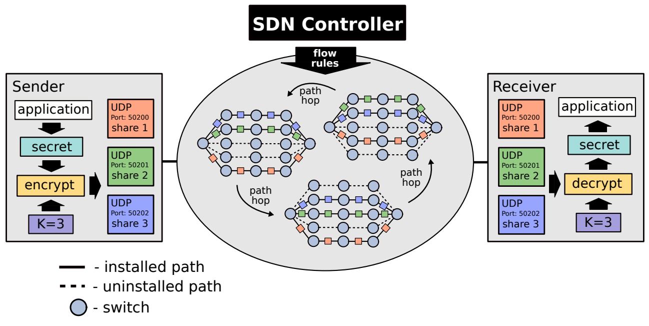

Figure 3: Network hopping.

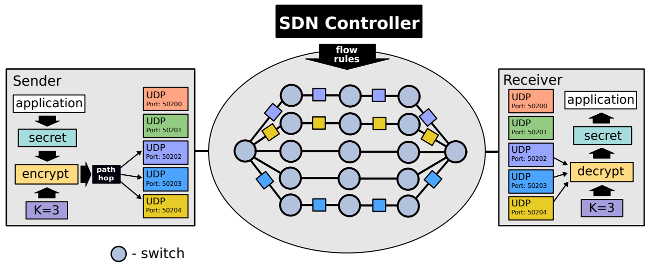

Figure 4: Host hopping.

are limited to matching fields in Ethernet, ARP, IP, IPv6, TCP, and UDP.

3.3 Path Hopping Approaches

We identified two distinct mechanisms by which the routing constraints of path hopping could be enforced in an SDN. We refer to these different mechanisms as *network hopping* and *host hopping* and discuss them below.

Network Hopping. In network hopping, shown in Figure 3, an initial set of K disjoint paths is installed in the network and the network is continuously reconfigured throughout the course of a path hopping flow by the SDN controller switching the set of K paths in use. This is arguably the most SDN way to implement path hopping. While it has the advantage of ensuring that the path hopping happens irrespective of the software running on the end hosts, it suffers from a number of crucial shortcomings. The foremost of which is the fact that performing a path hop requires reconfiguring the flow table of every switch along each of the disjoint paths used by a path hopping flow. This need for runtime reconfiguration of switch flow tables has the potential to introduce transient packet losses and inconsistent routing behavior [4]. These drawbacks are particularly limiting given that the security of a path hopping flow is proportional to the frequency with which hopping can take place.

Since it takes time for switches to change their configuration, every time hopping occurs, some packets may be dropped as their path may not be available in the network. Recall that, with network hopping, the SDN controller establishes new paths and tears down old paths to implement hopping. Every time hopping occurs, the SDN controller has to execute this procedure, which takes a non-negligible amount of time depending on the size of the network, number of paths, and switch hardware. The result of this limitation is that the sender cannot hop arbitrarily fast, due to the time the SDN controller needs to setup new paths and tear down old paths.

We attempted to measure the time required for the SDN controller to establish new paths and tear down old paths, but found that it was not straightforward to measure. The reason is that after the SDN controller sends control commands to switches, it does not receive any feedback from switches to know when their reconfiguration is complete. Hence, one must actively probe switches to find out if they have updated their configuration, which does not produce accurate results, as it takes some time to probe each switch. As a result, in our testbed experiments (described later), we measured *packet loss* as an indirect measure of the hopping delay. If the hopping frequency is higher than what can be achieved given the hopping delay, then some packets are lost in the network due to the old paths being torn down before packets reach the receiver.

Host Hopping. In host hopping, shown in Figure 4, an initial set of N disjoint paths is installed in the network and does not change over the course of the flow. The act of hopping is achieved on the hosts by changing the set, K, of these paths used for different shares. In this configuration, the SDN essentially configures a set of tunnels such that shares with a particular tag are always forwarded to their destination via a particular tunnel. The host then picks the tags to attach to each share. This allows us to enforce the routing constraints entailed by the path hopping system while obviating the need for runtime reconfiguration of the network. As a result, the source can hop essentially arbitrarily fast without suffering from the adverse effects of path reconfiguration needed with network hopping.

However this solution is not without drawbacks. Namely, the installation of such tunnels results in increased consumption of flow table space, since N paths need to be installed instead of K paths with network hopping. As flow table space is in limited supply on modern switches, sufficiently large values of N or sufficiently many path hopping flows can require more expensive switches with larger flow tables.

3.4 Cost of Path Hopping

We implemented our path hopping mechanism as a module on top of the ONOS SDN Controller [3]. The ONOS API provides a general purpose mechanism to both query and manipulate the flow table state of OpenFlow switches as well as carry out tasks necessary to the implementation of path hopping such as topology discovery. Our path hopping implementation relies on two components: 1) a host application that is responsible for encoding/decoding operations at the sender/receiver, and 2) an SDN controller module to configure the substrate network.

Testbed Setup. We used four Aruba 2930F switches [2] to construct the substrate network in which our testbed experiments were conducted. Each of the physical switches supports OpenFlow version 1.3 [14] and can host up to 16 distinct OpenFlow agent instances. From the perspective of the SDN controller, each OpenFlow agent instance appears as a distinct OpenFlow enabled switch in the substrate network. Each of the OpenFlow agent instances hosted by a particular switch is assigned a subset of the physical

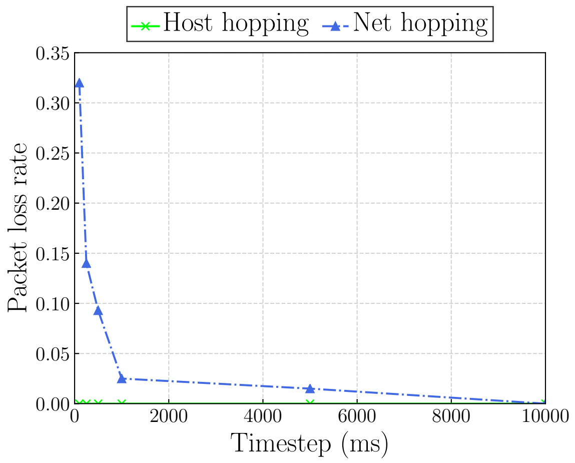

Figure 5: Effect of hopping interval on packet loss.

ports present on the switch. This design allows for the construction of diverse network topologies using relatively small amounts of physical switching hardware. Specifically, for the experiments presented in this section, we configure the testbed to have a complete graph topology with 10 nodes. All internal links interconnecting the nodes in the substrate topology have 1 Gbps capacity.

Results and Discussion. To provide some insight into the hopping intervals possible with each hopping method, we examined the the packet loss rates for a single path hopping flow when either network hopping or host hopping are used. The experiment consisted of transferring a 20MB file from a source node to a destination node on the SDN testbed. The path hopping flow used a message size of 256B and a K value of 3.

The results are shown in Fig. 5. It can be observed that network hopping results in extremely high packet loss rates with hopping intervals under 1 second and packet loss rates that are high enough to be severely problematic for many applications for hopping intervals as long as 8 seconds. Additionally, other experiments we performed indicate that the packet loss rate also increases with the number of switches in the topology. Given the relationship between the security of the system and the hopping frequency, a mechanism which imposes such high costs in exchange for a high hopping frequency is impractical. In contrast, we see that even for very high hopping frequencies, host hopping does not cause any packet loss to occur, demonstrating that the quality of service provided by path hopping flows does not degrade as the hopping frequency of the flow is increased.

4 REFLECTIONS

This project set out with a fairly simple goal: to measure the cost of hopping paths in real networks, and to determine an acceptable hopping rate. Ultimately, the project took over three years and became a true exploratory work.

Our attempts to have a realistic estimate of the cost factors of the MTD system in practice remained elusive. The project, however, taught us important lessons.

Firstly, it underlined the importance of vetting the assumptions of theoretically sound protocols in real-world settings. The network model for the original protocol required that (i) paths are truly disjoint (e.g., not implemented through overlay networks that share nodes and links in lower network layer), and (ii) a set bound (parameter t) on the adversary's power that could be supported by secondary evidences (e.g., monitoring the network activities). After carefully investigating alternative networking technologies, we concluded that providing guarantee for path disjointness is a complicated task in traditional TCP/IP networking systems and instead chose to focus on modern SDN-based systems that provided us with the knowledge of the physical connections between switches that would ensure path disjointness.

Secondly, we found that the natural approach to hopping paths in an SDN, dynamically reconfiguring the network or network hopping, results in extremely poor performance. Instead, we settled on a scheme where the paths in the network are fixed and the end hosts dynamically switch between them. Note that both of these approaches randomize the paths through the network. By changing *where* we implement the randomness in our MTD defense we dramatically improved performance. We believe this is an important observation for MTD systems.

Thirdly, although the new implementation lowered the cost in terms of end-to-end delay, it incurred other costs. For example, it required high end switches that could store large routing tables and rapidly search through them. There are also other assumptions and costs, such as the need to compute all the disjoint paths at regular intervals, and update the routing tables at the switches. This led us to believe that having a realistic estimate of the costs of path hopping requires large scale implementations and experiments that could realistically model the working of the system in practice and under normal network traffic conditions, leaving such estimation in practice an elusive goal.

Although we failed to achieve our initial research goals, we ended up with a surprising discovery along the way that changed the course of the project. We found that real networks do not obey the properties assumed by the original network model, resulting in a side channel. We discovered that packets from shares of multiple consecutive messages were present in the network simultaneously. This is perhaps obvious from a networking perspective, as packets must travel through all switches on a given path, which takes time. However, the theoretical underpinnings of the MTD-based SMT system assumed that packets travel instantly from the source to the destination and, thus attackers have only one opportunity to capture a given packet. That turned out to be false. In reality, attackers may have multiple opportunities to capture a given packet. This resulted in the discovery of a side channel we called Network Data Remenance (NDR) [16]. Our work has led to the discovery of NDR side channels in other protocols [11].

On a final note, the project showed us the crucial role of collaboration of researchers from across computer science disciplines, including cryptography, networking, and systems. Implementing cryptographic systems that rely on physical assumptions are particularly challenging (compared to the implementation of a computationally secure systems) because of possible implementation choices, discipline specific requirements, and evaluation criteria and methods. The same research question finds different statements, different evaluation criteria, and different approaches to evaluation in these disciplines. In physical security, we have found that collaboration is essential, as security is tightly related to the lower-level network properties.

Acknowledgement. The work of Safavi-Naini is in part supported by Natural Sciences and Engineering Research Council of Canada and Telus Communications under Industrial Research Chair Program.

REFERENCES

[1] Hadi Ahmadi and Reihaneh Safavi-Naini. 2013. Detection of algebraic manipulation in the presence of leakage. In *International Conference on Information Theoretic Security*. Springer, 238–258.

[2] Aruba Networks. 2022. ARUBA 2930F SWITCH SERIES. https://www.arubanetworks.com/assets/ds/DS_2930FSwitchSeries.pdf accessed August 22, 2022.

[3] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. 2014. ONOS: Towards an Open, Distributed SDN OS. In *ACM Workshop on Hot Topics in Software Defined Networking*.

[4] Mahdi Dolati, Ahmad Khonsari, and Majid Ghaderi. 2018. Consistent SDN Rule Update with Reduced Number of Scheduling Rounds. In *Proc. IEEE CNSM*.

[5] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. 1993. Perfectly secure message transmission. *Journal of the ACM (JACM)* 40, 1 (1993), 17–47.

[6] M. Franklin and R. Wright. 2000. Secure Communication in Minimal Connectivity Models. *Journal of Cryptology volume* 13, 1 (2000), 9–30.

[7] C. Hong et al. 2018. B4 and After: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN. In *Proc. SIGCOMM*.

[8] HPE. 2022. SDN Switches Portfolio. https://techlibrary.hpe.com/ie/en/networking/solutions/technology/sdn/portfolio.aspx accessed July 18, 2022.

[9] Hoda Maleki, Saeed Valizadeh, William Koch, Azer Bestavros, and Marten van Dijk. 2016. Markov Modeling of Moving Target Defense Games. In *ACM Workshop on Moving Target Defense*. 81–92.

[10] Thomas D Nadeau and Ken Gray. 2013. *SDN: Software Defined Networks: an authoritative review of network programmability technologies.* " O'Reilly Media, Inc.".

[11] Pushpraj Naik and Urbi Chatterjee. 2022. Network Data Remanence Side Channel Attack on SPREAD, H-SPREAD and Reverse AODV. In *Security, Privacy, and Applied Cryptography Engineering*. Springer, 129–147.

[12] H. Okhravi, T. Hobson, D. Bigelow, and W. Streilein. 2014. Finding Focus in the Blur of Moving-Target Techniques. *Security Privacy, IEEE* 12, 2 (Mar 2014), 16–26. https://doi.org/10.1109/MSP.2013.137

[13] Hamed Okhravi, MA Rabe, TJ Mayberry, WG Leonard, TR Hobson, David Bigelow, and WW Streilein. 2013. *Survey of cyber moving target techniques.* Technical Report. MIT Lincoln Laboratory.

[14] Open Networking Foundation. [n. d.]. OpenFlow Switch Specification. https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf accessed August 22, 2022.

[15] Open Networking Foundation. 2022. Software-Defined Networking (SDN) Definition. https://opennetworking.org/sdn-definition/ accessed July 18, 2022.

[16] Leila Rashidi, Daniel Kostecki, Alexander James, Anthony Peterson, Majid Ghaderi, Samuel Jero, Cristina Nita-Rotaru, Hamed Okhravi, and Reihaneh Safavi-Naini. 2021. More than a Fair Share: Network Data Remanence Attacks against Secret Sharing-based Schemes. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society.

[17] Reihaneh Safavi-Naini, Alireza Poostindouz, and Viliam Lisy. 2017. Path Hopping: An MTD Strategy for Quantum-safe Communication. In *ACM Workshop on Moving Target Defense*. 111–114.

[18] Claude E Shannon. 1949. Communication theory of secrecy systems. *The Bell system technical journal* 28, 4 (1949), 656–715.

[19] Peter W. Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* 41, 2 (1999), 303–332.

[20] A. Singh et al. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *Proc. SIGCOMM*.

[21] A.D. Wyner. 1975. The wire-tap channel. *Bell Systems Technical J.* (1975).