More than a Fair Share: Network Data Remanence Attacks against Secret Sharing-based Schemes

Leila Rashidi*, Daniel Kostecki†, Alexander James*, Anthony Peterson†,
Majid Ghaderi*, Samuel Jero‡, Cristina Nita-Rotaru†, Hamed Okhravi‡ Reihaneh Safavi-Naini*,
*University of Calgary, {leila.rashidi, alexander.james, mghaderi, rei}@ucalgary.ca
†Northeastern University, {kostecki.d, peterson.ant, c.nitarotaru}@northeastern.edu
‡MIT Lincoln Laboratory, {samuel.jero, hamed.okhravi}@ll.mit.edu

***Abstract*—With progress toward a practical quantum computer has come an increasingly rapid search for quantum-safe, secure communication schemes that do not rely on discrete logarithm or factorization problems. One such encryption scheme, Multi-path Switching with Secret Sharing (MSSS), combines secret sharing with multi-path switching to achieve security as long as the adversary does not have global observability of all paths and thus cannot capture enough shares to reconstruct messages. MSSS assumes that sending a share on a path is an atomic operation and all paths have the same delay.**

In this paper, we identify a side-channel vulnerability for MSSS, created by the fact that in real networks, sending a share is not an atomic operation as paths have multiple hops and different delays. This channel, referred to as Network Data Remanence (NDR), is present in all schemes like MSSS whose security relies on transfer atomicity and all paths having same delay. We demonstrate the presence of NDR in a physical testbed. We then identify two new attacks that aim to exploit the side-channel, referred to as NDR Blind and NDR Planned, propose an analytical model to analyze the attacks, and demonstrate them using an implementation of MSSS based on the ONOS SDN controller. Finally, we present a countermeasure for the attacks and show its effectiveness in simulations and Mininet experiments.

I. Introduction

The common approach for achieving secrecy and integrity over an untrusted network is the usage of standard protocols such as TLS [27] to establish a secure and authenticated communication channel. The security of such standard protocols is predicated on several assumptions: ❶ (perhaps trivially) the protocol can be deployed on all platforms, ❷ the implementation of the protocol is correct, ❸ implementation of cryptographic schemes will not affect their guaranteed security, and ❹ the adversary cannot break the underlying cryptographic primitives (*e.g.*, the RSA, AES, DH, and DSA schemes). Several challenges have undermined the validity of these assumptions. Low-resourced devices (*e.g.*, IoT devices) often do not have the computational power to implement the standard protocols (challenging assumption ❶) [17]; various implementation flaws have been found [23] in even the most heavily-vetted protocols (challenging assumption ❷); despite secure designs, implementations of cryptographic schemes contain side-channels such as cache attacks [16] which can be used to break the security of the system (challenging assumption ❸) and the emergence of quantum computing has challenged the long-term security of existing crypto primitives (challenging assumption ❹).

These motivations have led to the development of novel secure communication protocols that attempt to provide secrecy using physical properties such as multiple network paths [33], [12], [22], [35], [36] and/or introducing dynamism in the system to stay ahead of an adversary trying to guess what paths are used for communication [2], [8], [31], [34], [37], [38], [39], [40], [41], [20], [21], [26], [3]. A class of these novel protocols leverages the concept of breaking a message into shares and sending each share over a randomly chosen path, possibly changing with time. For example, Lou and Fang [36] propose a secret sharing scheme over multiple network paths to provide confidentiality with significantly lower computational requirements (i.e., for low-resourced devices). Dolev and Tzur-David [13] propose a secret sharing scheme over multiple network paths to mitigate the problem of stolen or short keys (i.e., implementation vulnerabilities). Ahmadi et al. [1] and later Safavi-Naini et al. [28] considered models where shares of the message are sent over multiple paths that are changing (switching) in each time interval, and proved that the system provides information theoretic security and so stays secure against a quantum computer. Applications on real networks for such schemes, which combine secret sharing with multi-path switching, have also started to be proposed, including [10], which applies secret sharing to provide security for controller and switch communication in Software-Defined Networks (SDNs) in the face of an adversary with quantum computing capabilities.

In this work, our goal is to examine the real-world security of schemes that combine secret sharing with multi-path switching. We consider the scheme in [28] (referred to in the rest of this paper as Multi-path Switching with Secret

 This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

The research of Leila Rashidi and Reihaneh Safavi-Naini is in part supported by Natural Sciences and Engineering Research Council of Canada and Telus Communications, under Industrial Research Chair Program.

Network and Distributed Systems Security (NDSS) Symposium 2021
23-26 February 2021, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2021.23xxx
www.ndss-symposium.org

Sharing or, for short, MSSS) because it was shown to have perfect information theoretic security. MSSS relies for its security on the assumptions that paths are atomic and packets travel on such paths with the same delay. In essence, what this means is that the attacker gets one chance to capture a packet (a.k.a. a share) on a path. That is not the case in real networks where paths have multiple hops, and each hop (and path) can have different delays. These conditions create an environment for data to linger in the network, creating a side-channel vulnerability that can potentially be exploited by an attacker to collect enough shares and break the confidentiality guarantees of secret-sharing schemes. Note that this side-channel will be present not only in MSSS, but in all schemes whose security relies on the assumptions that all paths are atomic and that messages travel with same delay on all paths. We refer to this side-channel as Network Data Remanence (NDR) inspired by data remanence side-channels defined in the NSA/NCSC Rainbow Series as "the residual physical representation of data that has been in some way erased" [15]. While data remanence has been studied extensively in the context of storage media (e.g., hard disk drives, flash drives, RAM, *etc.*), it has received very little attention in the context of networking. To our knowledge, this is the first time that a data remanence side-channel has been considered outside storage systems.

We study how an attacker can exploit the vulnerability introduced by NDR. We identify two new attacks, one where the attacker blindly tries to exploit the side-channel unaware of the topology of the paths in the network (referred to as NDR Blind), and the other where the attacker is aware of the paths that might be used by shares in the network and exploits this information (referred to as NDR Planned). We propose a model that captures the multi-hop nature of paths and analyze the two attacks against MSSS. Our analysis shows that while the NDR Blind attacker is not very effective, the NDR Planned attacker can successfully exploit the side-channel to reconstruct messages sent using MSSS. Specifically, the NDR Planned attacker is able to recover the message sent in a network with 7 paths of length 6 with a probability of more than 0.5. If the paths are longer, the probability is even higher.

To show the impact of these attacks in practical settings, we then implement MSSS in an SDN setting, using the ONOS controller. SDN is a natural choice to implement multi-path protocols because of the ability of communicants to control the paths that are used by data packets. Our results collected from the testbed show that many shares do linger in the network long enough to open a side-channel, and our Mininet experiments corroborated the analysis results about the effectiveness of the NDR Planned attacker. Namely, in a network with 10 disjoint paths, the attacker is able to recover up to 30% of the data with different numbers of shares and path lengths, both when paths all have the same delay and when path delay varies.

This work does not argue the merits of multipath schemes or SDN, nor focus on securing SDN (the security of SDN itself is orthogonal to this work). Instead, we argue that with the widespread usage of SDNs in data centers and cloud computing platforms, it is imperative to understand the practical limitations of the newly proposed multipath schemes (as was the case with other schemes such as TLS). We emphasize that NDR does not depend on SDN, the essence of this side-channel is that paths are not atomic and that they have different delays.

Equipped with this new system model and attack strategies, we design a countermeasure that specifically targets NDR attacks. The main idea behind our countermeasure is to create shares and distribute them across not only *space* (changing the K paths used), but also *time* (different shares for a single message are sent at different times). Our countermeasure introduces a new parameter H, that controls how many additional shares are sent, and thus trades-off security for overhead. We demonstrate, through analysis and Mininet experiments, that our proposed countermeasure mitigates the attacks. In a network with 10 disjoint paths and 3 intermediate nodes the countermeasure was able to drop all recovery to below 2%, where recovery in a similar setting without our countermeasure reached up to approximately 30%.

We summarize our contributions as follows:

- We implement and evaluate the MSSS scheme in an SDN, using the ONOS controller.
- We uncover NDR, a network data remanence side-channel, general to all secret sharing and multi-path switching schemes whose security relies on the assumption that packet transmission is an atomic event. We show two attack strategies, NDR Blind and NDR Planned, that exploit this side-channel that break the security of MSSS.
- We evaluate multiple strategies in a simulated environment to study the probability of success by an attacker. We find that the chance of success for a NDR Planned attacker can be as high as 30% in Mininet experiments.
- We discuss a countermeasure and possible improvements to these schemes to mitigate our attack. We show effectiveness of the proposed countermeasure using theoretical analysis and experiments conducted using Mininet.

The rest of the paper is organized as follows. In Section II, we provide background on the main building blocks behind MSSS, secret sharing and path switching, and then describe MSSS and our SDN implementation of it. In Section III, we describe the side-channel and attack strategies to exploit it. Section IV is dedicated to showing the existence of the side channel in a real SDN testbed. In Section V, we present a new model for systems like MSSS and an analysis of the attacks. In Section VI, we provide an evaluation of the attacks for our SDN implementation of MSSS using Mininet. In Section VII, we present a countermeasure for the attacks and show its effectiveness. Section VIII discusses the generalizability of the uncovered side channel and our proposed countermeasure. Finally, we present related work in Section IX, and conclude our paper in Section X.

II. Multi-path Switching with Secret-Sharing

Message confidentiality in network communication is traditionally achieved by using cryptographic primitives that assume a computationally bounded adversary. For such schemes, security relies on secret keys that are established before secure communication starts.

There are, however, a number of approaches that use network properties or cryptographic primitives that do not rely on computational assumptions about the adversary. These approaches primarily use the following mechanisms: (i) secret sharing where a secret is broken into multiple shares (ii) disjoint multi-path routing, where data is sent on multiple node-disjoint paths between the sender and the receiver, and (iii) path switching where the sender changes the path(s) that are used for communication over time. Each of these primitives achieves certain security properties under different assumptions; for secret sharing, security relies on the attacker not being able to obtain enough shares; in multi-path routing, security relies on attacker not being able to control all the paths; and in path switching, security relies on the attacker not knowing the pattern with which paths are switched. Below we provide an overview of these mechanisms, and describe Multi-path Switching with Secret Sharing (MSSS) a scheme that combines these techniques to withstand strong adversaries. We also present a design for MSSS using SDN.

A. Secret Sharing

First proposed by Shamir [30] and independently by Blakely [6], secret sharing is a fundamental building block in secure multiparty computation [9], distributed storage [11], and side channel protection [25].

A (t, n) *threshold secret sharing scheme* uses a randomized share generation algorithm that takes a message m and generates n shares, and a deterministic reconstruction algorithm that takes any t shares and reconstructs the message m. The security property of the algorithm is that any $t-1$ shares do not reveal any information about the message. That is, the message will be *perfectly (information theoretically) secure* if the adversary can have access to at most $t-1$ shares. In Shamir secret sharing, the share generation and message reconstruction algorithms correspond to polynomial evaluation and interpolation over finite fields, both with efficient algorithms.

A special case of threshold secret sharing is a (k, k) scheme where all shares are needed for reconstruction. Unlike (t, n) schemes that are constructed over finite fields, they can be constructed over any (mathematical) group such as groups of integers modulo M, denoted by $\mathbb{Z}_M$, where M is a composite number. Let the message $m \in \mathbb{Z}_M$. The share generation for a (k, k) secret sharing is done by selecting $k-1$ random values, $s_1, s_2, \cdots, s_{k-1} \in \mathbb{Z}_M$, and computing $s_k = m - \sum_{i=1}^{k-1} s_i$. Reconstruction is done by simply adding all the shares.

B. Multi-path Routing and Path Switching

Secret sharing and multi-path routing are a natural match to provide confidentiality. Such a scheme constructs shares of a message using a (t, n) secret sharing scheme and sends each share on a distinct node-disjoint path. The message remains perfectly secret as long as the adversary can access at most $t-1$ paths (for example, by compromising a node on the path). Additionally, the system provides reliability for communication since, as long as t of the paths can deliver their corresponding shares, the secret can be correctly reconstructed. In schemes such as [36], [35], [22], [13], there are n paths that connect the sender and the receiver, and a set of paths will be chosen by the sender and receiver to transmit the shares. For schemes that use a fixed set of paths, the adversary can infer the set of paths used for long flows by monitoring network activity, enabling them to break the security of the communication.

Another approach used to provide confidentiality is path switching. In schemes such as [34] a random path is chosen for each message and used for transmission of that whole message. For these schemes, a single path carries the whole message, providing no additional reliability.

One approach to address the limitations of the above schemes is to combine path switching with multi-path routing and secret sharing, here time is divided into intervals and the sender and receiver *switch* to a randomly selected set of paths in each time interval and send message shares on these paths.

C. Multi-path Switching with Secret Sharing

In this paper we consider the scheme in [29] and refer to it as *Multi-path Switching with Secret Sharing (MSSS)*. The scheme allows a sender to send a message to a receiver with perfect information theoretic security without the need to use a secret shared key. The scheme assumes that the sender and the receiver are connected by N node-disjoint paths, K of which can be observed by the adversary at any given time. The sender uses (K, K)-secret sharing to generate K shares for the message, randomly selects a subset of K paths (from the total N paths), and sends each share on a distinct path. The adversary can monitor K paths of their choice. Time is divided into intervals where in each interval the sender and the receiver have the opportunity to change the set of paths that they use for transmission. The adversary is mobile, and can also change the set of paths that they are using in each time interval. The system is analyzed as a Markov game between the attacker and the sender and receiver, with the goal of the attacker being to find the K paths that are used in each interval (to recover the message). Using (K, K) secret sharing implies that all shares must be available for reconstruction of the message, although the scheme can be generalized to (t, K) secret sharing where only t of the K chosen paths are needed for data recovery. Note that MSSS was proposed for the explicit purpose of enhancing secrecy when faced with strong attackers or when traditional protocols may have weaknesses. While this scheme provides information-theoretic security and remains secure against an adversary with access to a quantum computer, it increases the required bandwidth. Thus, it is not suitable when optimization of bandwidth utilization is the main concern.

D. SDN-based Design for MSSS

MSSS requires that a set of N node-disjoint paths exist between the sender and the receiver and that the paths on which packets travel are switched periodically. These requirements are challenging to obtain in traditional networks because the sender and receiver do not typically control the paths that are taken by their packets. There are, however, several possible approaches to ensuring a disjoint set of paths, including overlay networks, source routing, and SDN.

SDN is the most appealing as it also provides the ability to dynamically control the paths in the network. Specifically, SDNs separate the network into a control-plane and a data-plane. The data-plane consists of the *switches* in the network

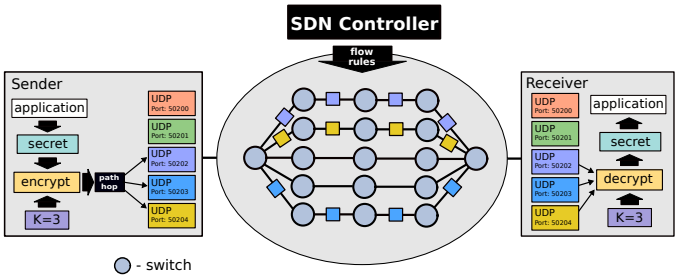


Fig. 1: An SDN-based MSSS

and its sole job is to move packets in the network as directed by the control-plane. The control-plane consists of a logically centralized *controller* that has a global view of the network and provides forwarding instructions to the data-plane. This controller typically runs one or more apps that implement policy for the network. One such policy can be path switching on disjoint paths as required by MSSS.

Figure 1 shows our design of MSSS using SDN. Here we use an SDN app that allows a sender to request a set of N node disjoint paths to a receiver from the controller. The SDN controller computes a set of disjoint paths between the sender, and receiver and configures the data-plane to provide them. At each time interval, the sender randomly picks K paths from the set of N paths to use for sending message shares, thereby switching within the total set of N paths. The receiver listens to all N paths so it receives all shares of the message.

An important question for any design of MSSS 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. Otherwise, some part of the 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 are limited to matching fields in Ethernet, ARP, IP, IPv6, TCP, and UDP.

III. Side-Channel Attacks in MSSS

In this section, we identify a new side-channel created by implementations of MSSS in real networks and show several attacks that exploit the side-channel to break confidentiality. Below we first describe the side-channel, and then describe attacks that can exploit it.

A. Network Data Remanence Side-Channel

In the design of MSSS, the network is abstracted as a set of *wires* [12], each corresponding to a direct path connecting the sender to the receiver, over which all packets travel instantaneously. The security analysis of such secret sharing schemes considers a strong mobile adversary, but is based on the above network abstraction model.

In a real network, each path consists of a sequence of links and switches. In other words, the propagation of shares along network paths is not an atomic operation, rather shares traverse each link and switch sequentially following the store-and-forward design of TCP/IP networks. To eavesdrop on a path, an attacker can probe any of the links and switches that constitute the path. Or, perhaps, an attacker could even probe multiple links or switches on a path, depending on their probing capabilities. In particular, while probing links requires physical access, probing switches can be done remotely. For example, in an SDN network, by installing appropriate forwarding rules on chosen switches, an attacker could receive a copy of any packet that matches the IP addresses of the sender and receiver. Because network paths are not atomic wires and paths do not all have the same delays (two of the main assumptions in the 'idealized' network model in secret-sharing schemes), residual shares from previous messages still exist in the network when a new message is being sent. Note that this behavior is general to any real network and is not specific to SDN-based implementations.

Implementations of MSSS in real networks, therefore, introduce a side-channel that we call *Network Data Remanence (NDR)*, because of its similarity to the known data remanence in storage systems. This side-channel allows the attacker to effectively break the boundaries of the abstract model (*i.e.*, the attacker is limited to capturing on at most K paths in each time interval in [29]), and recover the message.

B. Threat Model

We assume that the attacker captures packets at nodes/hops. In a typical network attack, switches are compromised when they are vulnerable, maybe because a weak password was used or a software vulnerability is present in their implementation. Switches in an enterprise-level network often have rather homogeneous models, so we assume that the attacker has access to all switches and can redirect a copy of the traffic to their machine. It is important to note that, while the attacker has access to all of the switches, they cannot possibly capture traffic from all of them at all times because that would require an unreasonably fast machine with significant resources and bandwidth (even RAM write speeds become an issue in that scenario), and such an attack is also easily identifiable. Therefore, the attacker can only realistically capture a fraction of traffic from each switch (say 10%) at a fraction of switches at each time (say 10%).

We assume that the attacker is able to listen to at most K switches simultaneously, where K is equal to the number of paths used to send shares of a message in MSSS. Based on its resources, the attacker can switch what paths they are listening to and at what intermediate nodes.

C. Network Data Remanence Attacks

We define a *Network Data Remanence (NDR)* attack as an attack that leverages residual network packets left in network links or devices (including switches, hubs, access points, routers, and network interface cards) when they are assumed to be gone.

We consider a number of possible attacks that exploit the NDR side-channel to break MSSS. Table 1 summarizes the attacks considered in this paper. Below we explain the details. Note that we refer to the number of edges between two nodes as the distance between the nodes. Also, *hop* and *intermediate node* are used interchangeably.

Name	Abrv.	Exploits NDR	Knows Switching Time	Switches Nodes	Knowledge of Path Composition
Fixed	FIX	No	Yes	No	Partial
Independent	IND	No	No	Yes	Partial
Sychronized	SYN	No	Yes	Yes	Partial
NDR Blind	BLD	Yes	Yes	Yes	Nothing
NDR Planned	PLN	Yes	Yes	Yes	Complete
NDR Planned Opt	OPT	Yes	Yes	Yes	Complete

TABLE I: NDR Attacks.

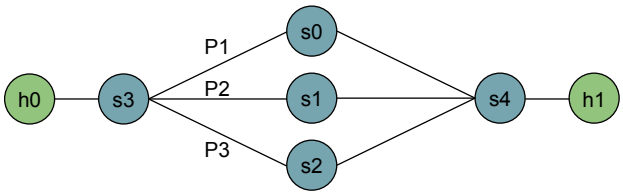


Fig. 2: A multi-hop network topology. The hosts $h0$ and $h1$ are connected to the network via the ingress/egress switches $s3$ and $s4$, which are assumed to be trusted. There are $N = 3$ paths between $s3$ and $s4$, each of length $L = 2$ hops.

Basic Attacks. The most basic attacker, which we call *Fixed* is an attacker that selects a set of K paths and one hop along each of those paths, and listens to those hops. A slightly smarter attacker changes the set of K paths they are listening to at the same time as the sender changes paths, but they listen to a fixed hop of any selected path. Depending on whether or not the attacker's switching is synchronized with that of the sender, we refer to such attackers as *Synchronized* and *Independent*, respectively. An independent attacker can switch paths faster or slower than the sender, as they do not know when the sender switches path. These three attackers get one chance at capturing each share on a path, thus, without loss of generality, we assume that the attackers can listen to hops placed at distance 1 from the sender. These attacks may benefit from the NDR side-channel, but they do not deliberately exploit it.

To see how the unrealistic assumptions made in the models affect message security, consider the following toy example. The sender uses a (2, 2) secret sharing scheme to send a message securely to the receiver over $N = 3$ paths. Time is divided into time slots of length δ. In each time slot, the attacker is able to listen to at most $K = 2$ paths. The network topology is depicted in Fig. 2, in which the sender and receiver are connected together via 3 paths P_1, P_2, and P_3, where each path has two hops. We assume that the ingress/egress switches that connect the sender and receiver to the network are trusted. As such, we focus on message transmission between these boundary switches. Assume that it takes exactly δ for a share to traverse one hop in the network. Thus, it takes 2δ for each share to reach the receiver. Under the model from [29], the message shares are sent and received in a single clock tick, δ, and the attacker, therefore, gets only one chance at capturing shares. However, in our multi-hop network example, the attacker will get 2 chances because a share takes 2δ to traverse a path.

NDR Attacks. The most basic attacker which aims to exploit the side-channel, randomly selects K intermediate nodes to monitor from the set of all intermediate nodes every δ time interval. In addition, we assume that this attacker is synchronized with the sender (*i.e.*, knows when the sender switches its paths). We refer to this attacker as NDR Blind.

A smarter attacker follows shares as they travel along the paths in the network (this attacker is also synchronized with the sender). Initially, the attacker listens to K random intermediate nodes of distance 1 from the sender. In order to capture the shares it missed in the first switching interval, the attacker then probes K random intermediate nodes of distance 2 from the sender during the second switching interval. The attacker goes one link further at each switching interval until all shares of the first message are delivered to the receiver. At the next switching interval, the attacker then selects K random intermediate nodes of distance 1 from the sender to capture the shares of another message, and so on. We refer to this attacker as NDR Planned. An optimization of the NDR Planned attacker is possible where the attacker checks at each step to see if all shares needed to reconstruct a message are captured, and if so, immediately starts listening at distance 1 again, instead of continuing to listen to the next hop. We refer to this attack as NDR Planned Opt. All these attacks, NDR Blind, NDR Planned, and NDR Planned Opt are attacks that aim to explicitly exploit the NDR side-channel.

IV. Initial Evidence From Testbed

The objective of this section is to experimentally demonstrate the presence of the NDR side-channel when using MSSS in a real network. To this end, we implemented a small-scale SDN testbed to experiment with MSSS, and derived some initial results which indicate the possibility of an NDR side-channel.

A. Testbed Setup

We first describe the configuration of the SDN testbed that we built to experiment with MSSS. The testbed is designed to allow the collection of full payload packet traces as well as statistical counters (*e.g.*, number of bytes exiting a switch port) from switches for further offline analysis and attack emulation.

Physical Topology. We used four Aruba 2930F switches [24] 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. Each of the OpenFlow agent instances hosted by a particular switch is assigned a subset of the physical ports present on the switch. Each Aruba 2930F switch includes 24 ports, each at 1 *Gbps*. From the perspective of the SDN controller, each OpenFlow agent instance appears as a distinct OpenFlow enabled switch in the substrate network. This scheme, in which multiple OpenFlow

agent instances are co-located at the same physical switch, 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.

Orchestration Framework. In order to conduct our experiments, we have implemented a generic orchestration framework that allows configuration state to be generated programmatically via a Python3 API. The orchestration framework can derive forwarding plane and traffic generation configuration from experiments specified as Python3 objects and subsequently configure both the forwarding plane and the traffic generation tools. Our testbed uses the ONOS SDN controller to interface with the physical switches via OpenFlow 1.3.

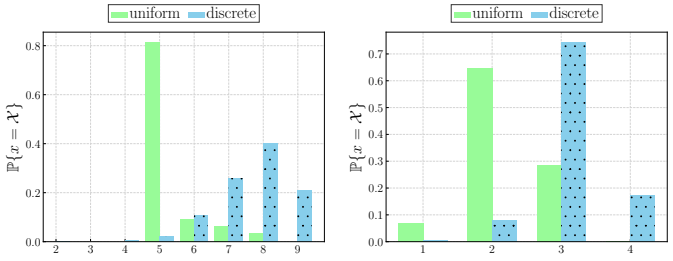
B. Experimental Results

The security guarantees provided by MSSS rely on a synchronous transmission model in which all paths have the same end-to-end delay and negligible path delays (*i.e.*, transmissions are instantaneous). In this experiment, we study the implications of these assumptions for MSSS. We show that the delay diversity of network paths, which is a natural characteristic of real-world networks, opens up a data remanence side-channel for an attacker to break MSSS.

Experiment Setup. Each experiment consists of a single path switching flow between a source and destination node connected by $N = 9$ disjoint paths. The flow was a bulk data transfer of size 20 *MB*. The bulk data is divided to messages of size 256 *B* for transmission. The multi-path switching parameter K is set to 5 and the switching interval δ is set to to 100 ms. Two scenarios are considered where path delays are drawn uniformly from the range $[0, 250]$ ms and the set $\{0, 100, 200\}$ ms. Each of the paths also incurred 50 ms of jitter meaning that the latency on an individual path could randomly vary by up to 50 ms for each packet transmitted over the path. These latency values were selected as we believe they represent a reasonable upper bound on the amount of latency experienced by typical Internet traffic.

Results and Discussion. Fig. 3a shows the empirical Probability Density Function (PDF) of the number of active paths in each switching interval. A path is considered active in a given interval if at least one share (of any packet) is traversing the path in that interval. Ideally, in any interval, exactly $K = 5$ paths should be active. However, due to the diversity of path latency, we observe that the number of active paths in this experiment was as high as 9 (recall that $N = 9$). Notice that security of MSSS relies on the assumption that there is a non-zero probability (as a function of N and K) that the attacker misses one of the shares of a message. This experiment clearly demonstrates that, in this network setting, choosing any subset of paths to eavesdrop on frequently results in capturing a share on each of the chosen paths, even though the captured shares may belong to different messages.

Similarly, in Fig. 3b, we have plotted the empirical PDF of the number of switching intervals in which the shares of the same message were present. Ideally, the shares of a message should be present in the network in only one interval. However, as can be seen from the figure, in this experiment, the shares of a packet were present in the network for up to 4 intervals, which significantly increases the window of opportunity for the attacker to capture these shares. Specifically, the attacker could eavesdrop on faster paths in one switching interval, and then hop to slower paths which are still active in the following interval. With a careful strategy that accounts for the delay diversity of paths relative to the switching interval, an attacker may be able to capture all shares of a message.



(a) PDF of the number of active paths per switching interval where shares of any packet were present.

(b) PDF of the number of switching intervals where shares of the same packet were present.

Fig. 3: Distribution of shares in the network ($N = 9$, $K = 5$).

In both figures, experiments with discrete path delays exacerbate the problem. The reason is that, in the discrete delay experiments there is substantially more divergence across path delays compared to the continuous delay case.

V. Attack Analysis

In this section, our focus is on understanding the impact of the multi-hop nature of paths on the effectiveness of NDR attacks. We first develop a model to investigate the effectiveness of different attacks, including NDR Attacks, in a network with multi-hop paths. Then, we derive analytical results for each attack, and assess whether they can benefit from the NDR side-channel.

A. System Model

We consider a network where nodes are connected through disjoint multi-hop paths, and the number of total disjoint paths between a sender and receiver is N. We denote the number of links on a path as the length of the path, L. We assume that all disjoint paths have the same path length, L, $L > 1$.

We consider time as clock ticks, and assume that it takes one clock tick for each share to traverse each link of the network. The sender breaks the message into $K < N$ shares according to (K, K) secret sharing. At clock tick $t = 0$, the sender sends shares of the information along K random paths, and then at each subsequent tick it selects a new set of K paths.

B. Analysis of Attacks

In order to analyze the effectiveness of the attacks, we compute the probability that each of the Fixed, Synchronized,

NDR Blind, and NDR Planned attackers, described in Section III, can recover a message being sent with MSSS.

Fixed and Synchronized. The probability of capturing K shares at tick 1 by the Synchronized and Fixed attackers can be computed as $1/\binom{N}{K}$.

NDR Blind. Let $P_{bln}(m,t)$ denote the probability of capturing m shares until tick t by the NDR Blind attacker. $P_{bln}(m,1)$ can be computed as follows. Note that if $2K > N$ and $L = 2$, then the NDR Blind attacker captures at least $2K - N$ shares at $t = 1$.

$$P_{bln}(m,1) = \begin{cases} \frac{\binom{K}{m}\times\binom{(L-1)N-K}{K-m}}{\binom{(L-1)N}{K}}, & (L=2 \text{ and } 0<2K-N\leq m\leq K) \\ & \text{or } 0\leq m\leq K\leq \frac{N}{2} \\ 0, & otherwise \end{cases} \tag{1}$$

If $L = 2$, the probability of data recovery equals $P_{bln}(K,1)$; otherwise, when $t > 1$, $P_{bln}(m,t)$ can be computed recursively as follows.

$$P_{bln}(m,t) = \sum_{x=0}^{K} P_{bln}(m-x,t-1)\times D_{bln}(m,x), \tag{2}$$

where $D_{bln}(m,x)$ denotes the probability of capturing x new shares by the NDR Blind attacker at tick t provided that $m-x$ shares were captured before tick t. This probability is given by

$$D_{bln}(m,x) = \frac{\binom{K-m+x}{x}\times\binom{(L-1)N-K+m-x}{K-x}}{\binom{(L-1)N}{K}}. \tag{3}$$

The probability of data recovery of the NDR Blind attacker is equal to $P_{bln}(K, L-1)$ which can be computed recursively.

NDR Planned. Let $P_{pln}(m,t)$ denote the probability that the NDR Planned attacker has captured exactly m shares by tick t. Thus, the probability of recovering a message by the attacker is $P_{pln}(K, L-1)$, i.e. the attacker captures all the shares within the duration of transferring a message, which is from tick 1 to tick $L-1$. It is obtained that,

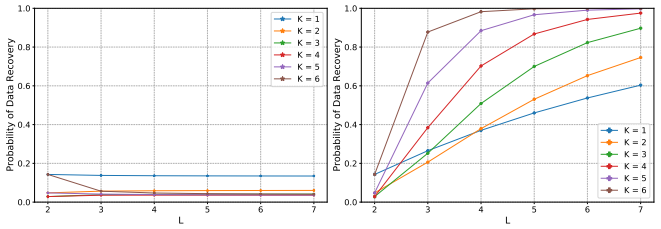
$$P_{pln}(m,1) = \begin{cases} \frac{\binom{K}{m}\times\binom{N-K}{K-m}}{\binom{N}{K}}, & 0\leq m\leq K\leq \frac{N}{2} \text{ or} \\ & 0<2K-N\leq m\leq K \\ 0, & otherwise \end{cases}. \tag{4}$$

For $1 < t < L$, $P_{pln}(m,t)$ can be computed using the following recursive formula,

$$P_{pln}(m,t) = \sum_{x=0}^{f(m)} P_{pln}(m-x,t-1)\times\frac{\binom{K-m+x}{x}\binom{N-K+m-x}{K-x}}{\binom{N}{K}}, \tag{5}$$

where

$$f(m) = \begin{cases} \min(m-(2K-N),K), & 2K>N \\ \min(m,K), & 2K\leq N \end{cases}. \tag{6}$$



(a) NDR Blind (b) NDR Planned

Fig. 4: Data recovery by the NDR Blind and NDR Planned attackers for a network with seven disjoint paths considering different values of path length L. K denotes the multi-path switching parameter.

C. Numerical Examples

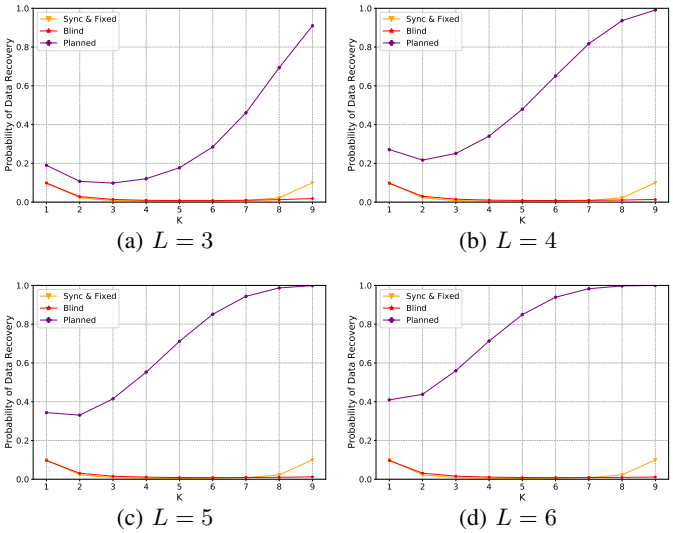
In this subsection, we consider a network with $N = 7$ disjoint paths and show the impact of various system parameters on the effectiveness of each attacker.

Fig. 4 represents the probability of data recovery by each of the NDR Blind and NDR Planned attackers for different values of L and K. When $L = 2$, the NDR Blind and NDR Planned attackers have the same performance. Thus, as observed in Fig. 4, they recover the message with the same probability. Focusing in on the NDR Blind attacker first, Fig. 4a shows that when secret sharing is not used (i.e., when $K = 1$ and the message is sent on a single path), the NDR Blind attacker has a probability of around 0.14 of recovering the message. When the path length, L, is greater than two and the sender uses secret sharing (i.e., $K > 1$), the NDR Blind attacker still has a non-zero probability of recovering the message, but this probability is very low.

The NDR Planned attacker is much more effective as can be seen in Fig. 4b. As path length L increases, the probability of data recovery by the NDR Planned attacker increases as well, such that it approaches one when L is sufficiently large. Moreover, it can be observed that when L is large enough, i.e., $L > 3$ for a network with seven paths, the NDR Planned attacker becomes more successful in capturing all shares as K increases.

Fig. 5 shows the effect of number of shares on the probability of recovering a single message with varying path length. Since the Sync and Fixed attackers probe only the nodes at distance one from the sender, their probability of recovery does not change with path length. Moreover, as can be seen in both Figs. 4a and 5, when the path length is greater than two, increasing the path length does not have a significant impact on the probability of recovery by the NDR Blind attacker.

In summary, our analysis shows an attacker that does not intelligently attempt to exploit the side-channel is not very effective. However, an NDR Planned attacker that strategically exploits the side-channel is increasingly effective at capturing enough shares to reconstruct the message as the path length increases.



(a) $L = 3$ (b) $L = 4$

(c) $L = 5$ (d) $L = 6$

Fig. 5: Effect of number of shares on the probability of recovering a single message with varying path length in a network with 10 paths ($N = 10$).

VI. Experimental Results

In this section, we evaluate the impact of our attacks in practical settings using the SDN-based implementation of MSSS described in Section II.

A. Methodology

We use the network topology depicted in Fig. 6 in which the source and destination nodes are connected by $N = 10$ disjoint paths, each of length $L = 4$, unless otherwise noted. This presented the most natural topology for assessing how system parameters affected goodput and percentage of recovered data. The capacity of each of the emulated links in the network was not restricted. Each experiment was conducted on a CentOS VM in QEMU with 6 Cores and 8 *GB* of RAM. These VMs are spawned from a server with Intel Xeon Silver 4114 CPUs running at 2.20 *GHz*. ONOS version 1.14.0-SNAPSHOT and Open vSwitch 2.9.2 supporting OpenFlow 1.4 are used. In each experiment, the sender sends a file to the receiver, and the metrics are averaged over 10 runs. The default experimental parameters, unless otherwise noted, are file size = 10 *MB*, $\delta = 100$ *ms*, message size $M = 512$ *B*, and $K = 3$. Our implementation uses sequence numbers to identify shares belonging to the same message.

We evaluate the Fixed, Sync, Independent, NDR Blind, and NDR Planned attackers described in Section III. The NDR Planned Opt attacker is not included in the figures as we did not see a real benefit over the normal NDR Planned attacker, likely a result of real networks not entirely mirroring the attacker analysis and model. The Independent attacker was using $\delta = 200$ *ms* for its switching interval. The NDR Planned and NDR Blind attackers use $\delta = 100$ *ms* as their switching interval. The attackers are implemented by adding proxy nodes before each intermediate hop on each path of the network. Shares passing through the proxies are captured, and attacker statistics are computed post-run.

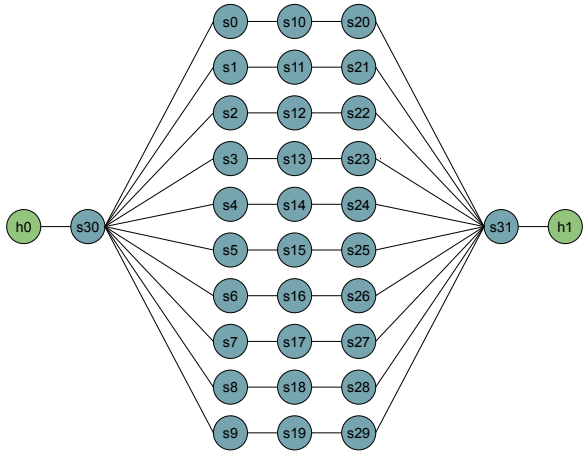
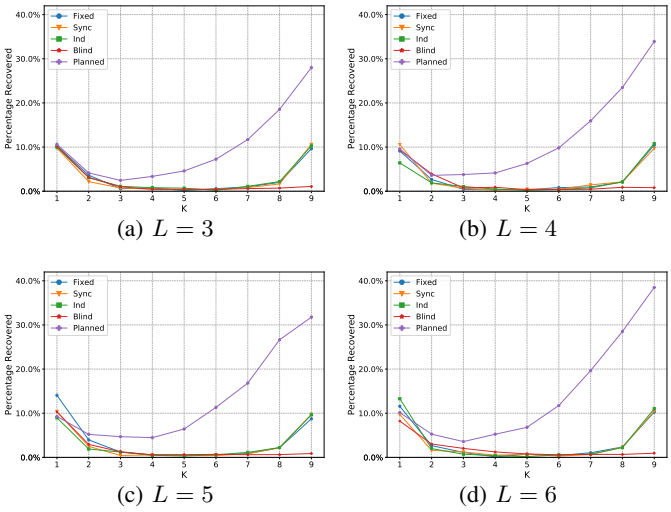


Fig. 6: Depiction of the network topology used for our Mininet evaluation with number of paths $N = 10$ and path length $L = 4$. h nodes are the end hosts and s nodes are switches.



(a) $L = 3$ (b) $L = 4$

(c) $L = 5$ (d) $L = 6$

Fig. 7: Effect of the number of shares on the percentage of recovered data with varying path length when each link has the same 50 *ms* delay.

We measure the effectiveness of the NDR attacks with the metric *Percentage Recovered*, defined as the fraction of total messages that were completely reconstructed by the attacker from the captured shares.

B. Impact of Path Length on NDR Attacks

We first study the impact of path length, L, on the attacks. In order to isolate the path length behavior, we configure each link to have the same constant delay of 50 *ms*, which corresponds to a round trip time between USA and Europe.

Fig. 7 shows the percentage of data recovered by an attacker employing each of the attack strategies we consider, as a function of the number of shares, K, for different path lengths, L. As can be seen in all 4 graphs, the Fixed, Sync,

and Independent attackers are very ineffective at recovering messages, as expected. With $K = 1$, these basic attackers are able to recover approximately 10% of the data, as a single intercepted share is enough to fully recover the message. A similar phenomena happens at $K = 9$, where an attacker picking 9 of 10 nodes at distance 1 from the sender is likely to pick correctly some of the time. Intermediate K values show a very low recovery rate. The NDR Blind attacker is not doing very well at leveraging the side-channel, regardless of the path length. In contrast, the NDR Planned attacker is very effective, managing to reconstruct approximately 5% to 40% of the messages sent by the sender across the range of K values. It is worth noting that we experimented with a broad range of latencies and found that the NDR Planned attack is effective regardless of link latency.

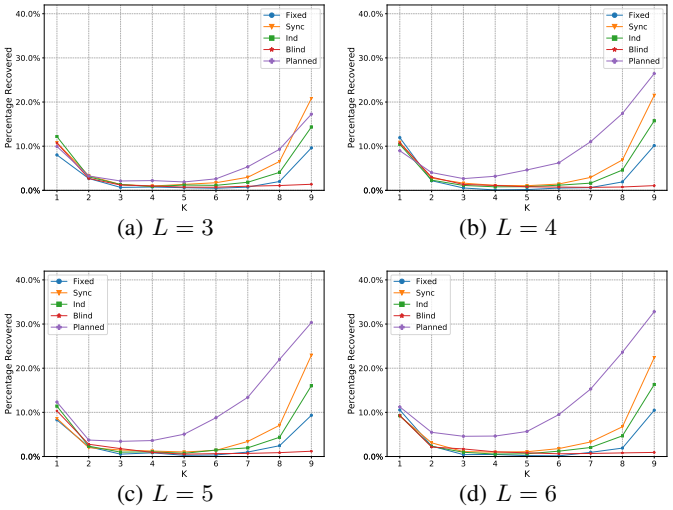
Note that the recovery rate for our NDR Planned attacker is significantly less than predicted by our analysis in Fig. 5. This is because our analysis only considers probability of recovery for a single message while our experiments analyze a stream of messages and our NDR Planned attacker follows the shares of a single message through the network. As a result, our NDR Planned attacker misses some messages while it is focused on collecting shares from other messages.

C. Impact of Path Delay on NDR Attacks

Our analytical model for the attackers considers only path length and assumes all links have the same delay, which is exactly equal to one clock tick. In real networks each link, and, in turn, each path, has a different delay. In order to simulate more realistic network latency, a delay is added to the first link of the network to emulate real end-to-end delays. In addition, jitter is applied to each message to emulate the small variations in delay common in real networks. The delay was randomly sampled from a continuous interval for each path. The jitter was randomly sampled from a continuous interval for each share sent.

In Fig. 8, we show the percentage of recovered messages for all considered attacks, with delay introduced as explained above. The first observation we make is that the Fixed, Sync, and Independent attackers seem to benefit from the variability of path delay. For example, Fig. 7b with fixed delay shows a recovery at $K = 8$ of less than 5%, and at $K = 9$ of about 10% for the three basic attackers, each recovering a comparable amount of data. However, examining the equivalent plot with varying delay, Fig. 8b, at $K = 8$ shows a Sync recovery rate slightly over 5%, and at $K = 9$ this jumps to over 20%. This shows a significant increase in recovery over the fixed-delay scenario present in Fig. 7. In addition, there is also a separation between the attackers. The Sync attacker performs best, and the Fixed attacker performs worst among the basic attackers. Each of these attackers can clearly leverage the side-channel to their benefit, but because the Sync attacker is switching paths at a higher rate than the Independent attacker, it has more of a chance to capture shares left in the network. The Fixed attacker has no ability to choose new nodes, so is at the mercy of the shares that travel across its set of nodes.

As in the fixed-delay scenario, the NDR Planned attacker performs significantly better than the basic Sync, Independent, and Fixed attackers in the majority of scenarios. For



(a) $L = 3$ (b) $L = 4$

(c) $L = 5$ (d) $L = 6$

Fig. 8: Effect of the number of shares on the percentage of recovered data with varying path length. In addition to a 50 *ms* delay on each link, a randomly sampled delay from the continuous interval [0 *ms*, 100 *ms*] is applied before the first node and a randomly sampled jitter from the continuous interval $[0\ \mu s, 100\ \mu s]$ is applied on each share sent.

example, in Fig. 7c with fixed delay when $K = 9$ the NDR Planned attacker recovers 20% more messages than even the best basic attacker. When comparing this to the varying delay scenario, where the basic attackers perform better, Fig. 8c shows that the NDR Planned attacker still recovers 10% more messages than the best basic attacker, even though the basic attacker's recovery rates have more than doubled. Despite this, the NDR Planned attacker does worse with varying delay than it does with fixed delay. This is likely because while with fixed delay shares do not travel through the network in total synchrony, they do travel in close temporal proximity to each other, especially in comparison to the varying delay case. This leads to a slight drop in the recovery rate of the NDR Planned attacker under varying delay. However, there is still an approximate 5% to 30% recovery rate for this attacker, as shown in Fig. 8. The exception is Fig. 8a, which indicates that if the paths in the network are short, then the recovery rate of the NDR Planned attacker approaches that of basic attackers.

The results for the NDR Blind attacker are very similar in both the fixed-delay and the varying delay scenarios, as well as across K and L values. When $K = 1$ the NDR Blind attacker performs similarly to the other attackers, likely because at this point any share recovered is a fully recovered message. However, when K begins to increase, the recovery quickly drops and does not recover. This attacker relies on choosing randomly from a large set of nodes, making the probability of intercepting unique shares of the same message low.

VII. Countermeasures

In this section, we propose a mitigation technique for NDR attacks. Afterwards, we analyze the effectiveness and overhead of the proposed technique. Finally, we implement this technique and present experimental results using Mininet.

A. Countermeasure Description

We first describe the key idea behind our approach to mitigate NDR attacks. To keep information theoretic security the only possibility is to break the message into more shares. However, sending more shares is challenging. A naive approach would require more disjoint paths, or will use the same K paths repeatedly which could result in reduced protection. Increasing the number of paths without increasing the attacker capacity (i.e., the amount of traffic they can capture) would not be fair. Also, increasing K can be problematic as larger values of K may increase the recovery probability. Thus, we propose distributing shares over both *time* and *space* instead of just space using a random set of paths to send a K-sized set of shares. In this section, we apply this idea to MSSS and show how it reduces the probability of data recovery.

Recall that in MSSS the sender generates K shares and spreads them across K disjoint paths. Given that the sender and the attacker cannot use more than K paths at the same time, we propose to generate more shares and spread them across both space and time. Let H be an integer greater than one. We refer to H as the *resilience factor*, a system parameter that can be configured by the sender. Instead of (K, K) secret sharing, the sender uses (HK, HK) secret sharing and divides the shares into H sets of K shares, and then gradually sends these sets of shares, one at each at consecutive clock tick. Specifically, at $t = 0, 1, \ldots, H-1$, the sender node chooses K paths uniformly at random, and then sends a share along each chosen path. By applying the proposed countermeasure, we increase T and a by $H-1$ and $(H-1)K$, respectively, which increases the attacker's effort dramatically. This countermeasure will have an impact on throughput; thus, the resilience factor, H, should be chosen to balance performance and attack resilience.

B. Countermeasure Analysis

We compute the probability of data recovery by the NDR Planned attacker and an upper bound for the probability of data recovery by the NDR Blind attacker. The probability of data recovery by the NDR Planned attacker is computed assuming that the attacker continues until all K shares of a set are captured as described in Section III-C. If all shares are captured at time t, the attacker remains at the same distance at time $t+1$ to capture shares of the next set; otherwise, it probes K random nodes which are located one link further away.

An attacker needs all shares in order to be able to recover the message. As the last set of shares is sent at time $t = H-1$, after time $t = L+H-2$ there will no shares available on any intermediate node. We first compute an upper bound for the probability of data recovery by the NDR Blind attacker. Assume that $N(L-1) \geq HK$. If all sets of shares could be available in the intermediate nodes at all ticks from 1 to $L+H-2$, then the NDR Blind attacker has a higher chance of capturing a share. Therefore, the probability of capturing m shares by tick t for the NDR Blind attacker under this assumption provides an upper bound. Let $U_{bln}(m,t)$ be the upper bound for the probability of data recovery by the NDR Blind attacker given that $N(L-1) \geq HK$. $U_{bln}(m,t)$ can be computed as follows,

$$U_{bln}(m,1) = \begin{cases} \frac{\binom{HK}{m} \times \binom{(L-1)N-HK}{K-m}}{\binom{(L-1)N}{K}}, & e \leq m \leq K \\ 0, & otherwise \end{cases}, \quad (7)$$

where e denotes an upper bound for the minimum number of shares that the NDR Blind attacker captures at $t = 1$. Thus, e is equal to the maximum of $(H+1)K - (L-1)N$ and 0. It is obtained that,

$$U_{bln}(m,t) = \sum_{x=0}^{K} U_{bln}(m-x, t-1) \times E_{bln}(m,x), \quad (8)$$

where $1 < t < L+H-1$, and $E_{bln}(m,x)$ denotes an upper bound for the probability of capturing x new shares by the NDR Blind attacker at tick t provided that $m-x$ shares were captured before tick t. This upper bound is given by,

$$E_{bln}(m,x) = \frac{\binom{HK-m+x}{x} \times \binom{(L-1)N-HK+m-x}{K-x}}{\binom{(L-1)N}{K}}. \quad (9)$$

Next, we turn our attention to computing the probability of data recovery for the NDR Planned attacker. Using the notation $P_{pln}(m,t)$, defined in Section V-B, the probability of data recovery for the NDR Planned attacker is $P_{pln}(KH, L+H-2)$. $P_{pln}(m,1)$ is computed according to (4), as in the case that no countermeasure is applied. If $1 < t < L+H-1$ and $m < HK$, then $P_{pln}(m,t)$ can be computed as follows,

$$P_{pln}(m,t) = \sum_{x=0}^{\min(K,m)} P_{pln}(m-x, t-1) \times D_{pln}(m,x), \quad (10)$$

where $D_{pln}(m,x)$ denotes the probability of capturing x new shares by the NDR Planned attacker at tick t provided that $m-x$ shares were captured before tick t. If one of the following conditions holds, $D_{pln}(m,x)$ is given by (12); otherwise, it is zero:

$$\begin{aligned} &C1: 2K - N \leq x \text{ and } (m-x)\%K = 0 \text{ and } 2K > N \\ &C2: x \leq K - (m-x)\%K \leq N - K \text{ and } 2K > N \\ &C3: x \leq m\%K \text{ and } m\%K > 0 \text{ and } 2K \leq N \end{aligned} \quad (11)$$

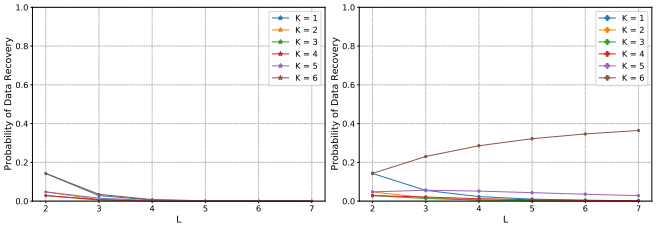
$$D_{pln}(m,x) = \begin{cases} \frac{\binom{N-x}{K-x}}{\binom{N}{K}}, & m\%K = 0 \\ \frac{\binom{K-(m\%K-x)}{x} \times \binom{N-K+(m\%K-x)}{K-x}}{\binom{N}{K}}, & m\%K > 0 \end{cases} \quad (12)$$

Moreover, $P_{pln}(HK,t)$ can be computed as follows,

$$P_{pln}(HK,t) = \begin{cases} \frac{P_{pln}((H-1)K, H-1)}{\binom{N}{K}}, & t = H \\ P_{pln}(HK, t-1) + \frac{P_{pln}((H-1)K, t-1)}{\binom{N}{K}} + \sum_{x=1}^{N-K} \frac{P_{pln}(HK-x, t-1)\binom{N-x}{K-x}}{\binom{N}{K}}, & t > H, 2K > N \\ P_{pln}(HK, t-1) + \sum_{x=1}^{K} \frac{P_{pln}(HK-x, t-1)\binom{N-x}{K-x}}{\binom{N}{K}}, & t > H, 2K \leq N \end{cases} \quad (13)$$

C. Numerical Examples

Fig. 9b represents an upper bound for the probability of data recovery by the NDR Blind attacker and the probability of data recovery by the NDR Planned attacker for different values of K and L when there are seven disjoint paths from the sender to the receiver and the proposed countermeasure is



(a) NDR Blind (b) NDR Planned

Fig. 9: The probability of data recovery by the NDR Blind and NDR Planned attackers for a network with seven disjoint paths and different values of path length L when the countermeasure is applied for $L > 2$ (resilience factor $H = L - 1$).

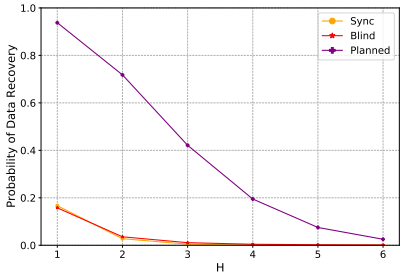


Fig. 10: Effectiveness of the proposed countermeasure for different values of H against different attackers when $N = 4$, $L = 6$, and $K = 2$. Curves corresponding to the NDR Planned and Synchronized attackers represent the probabilities of data recovery while the curve corresponding to the NDR Blind attacker represents an upper bound for its data recovery.

applied with $H = L - 1$. Comparison of Figs. 4 and 9 shows that the proposed countermeasure is successful in reducing the probability of data recovery. Specifically, comparison of Figs. 4b and 9b indicate that the proposed countermeasure significantly reduces the probability of data recovery by the NDR Planned attacker. Excluding the case $K = N - 1 = 6$ (discussed below), Fig. 9b shows that as L increases, the proposed countermeasure becomes more effective. Fig. 9b suggests that in order to achieve a specific level of protection as L increases, the required value for parameter H needs to increase more slowly than L.

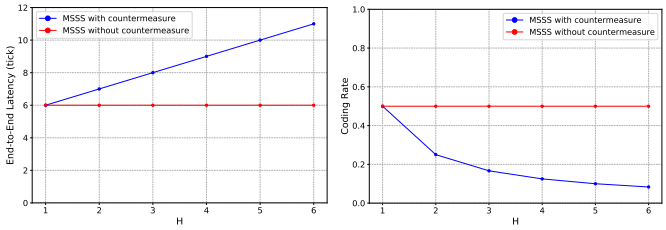
For the case $K = N - 1 = 6$, Fig. 9b shows different behavior. Here, the recovered data grows with increasing H unlike the other cases. What happens here is that the NDR Planned attacker will listen on $K = 6$ paths at the first hop. If this attacker fails to capture all K shares in its first attempt, then the remaining share must have been sent on the one unprobed path. Probing this new path at the next tick is enough to capture the remaining share. Note that the probability that the NDR Planned attacker will not probe the same set of paths at the next tick is $(N-1)/N$ which is quite high. As a result, for the case where $K = N - 1$, H should grow faster than L.

For the Synchronized attacker, all shares belonging to each set of shares are captured with probability $1/\binom{N}{K}$. Thus, the probability of data recovery can be computed as follows.

$$P_{syn}(HK, H) = \frac{1}{\binom{N}{K}^H} \tag{14}$$

Fig. 10 represents an upper bound for the probability of data recovery by the NDR Blind attacker and the probabilities of data recovery for the NDR Planned and Synchronized attacker for a network with parameters $N = 4$, $L = 6$, $K = 2$, and different values of H. The value 1 for H corresponds to the case of where countermeasure is applied and only K shares are generated. As can be seen in Fig. 10, applying the proposed countermeasure significantly reduces the probability of data recovery for all three attackers. For example, the probability of data recovery for the NDR Planned attacker is close to one when the countermeasure is not applied. However, if it is applied with resilience factor $H = 5$, the probability of data recovery becomes more than 10 times smaller.

While the proposed countermeasure decreases the probability of data recovery, it does impose some costs/overheads



(a) Impact on latency (b) Impact on coding rate

Fig. 11: The overhead of the proposed countermeasure with varying resilience factor, H, in terms of end-to-end latency and coding rate when the length of each path is 6 ($L = 6$) and $K = 2$ compared to the case that no countermeasure is applied ($H = 1$).

on the performance of the network. In order to quantify this, we define the following metrics and examine the impact of the proposed countermeasure.

End-to-End Latency: This is the time from when the sender sends the first share until all shares are delivered to the receiver. With no countermeasure, the latency is L. This value increases by one for any additional K-sized set of shares.

Coding Rate: Coding rate is a well-known measure in the context of coding theory. It is equal to the proportion of information over the total data generated by an encoder. The goodput achieved by each secret sharing scheme is directly proportional to its coding rate. Each share generated by the secret sharing scheme has the same size as the message. When the countermeasure is not applied, K shares are required for a coding rate of $1/K$. When the proposed countermeasure is applied, HK shares are sent so the coding rate is $1/(HK)$.

Fig. 11 represents the end-to-end latency and coding rate for MSSS considering different values of H. Note that $H = 1$ corresponds to the no countermeasure case. As observed in Fig. 11, the end-to-end latency increases with increasing H, making the communication more secure, but increasing overhead. Moreover, the coding rate decreases with increasing H, which leads to a decrease in goodput for increased protection.

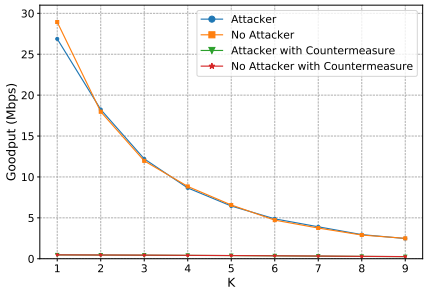


Fig. 12: Effect of the attacker and number of shares on goodput when $L = 3$, with and without the proposed countermeasure ($H = 2$), in a fixed scenario with 2 *ms* delay between each node. The sender's $\delta = 4$ *ms* and the attacker's $\delta = 8$ *ms*. File size is 1 *MB*.

D. Implementation of Countermeasure

To evaluate our countermeasure experimentally, we added it to our SDN-based implementation of MSSS described in Section VI. We first examined the impact the countermeasure has on performance. Fig. 12 shows the goodput when varying the number of shares, K, both with and without the countermeasure. As can be seen, the goodput without the countermeasure is significantly higher than with the countermeasure. This is a consequence of having (HK) shares per message, and having to wait δ time for each message. We also observe that the attacker does not significantly impact goodput.

The goal of the countermeasure is to limit the attacker's ability to capture enough shares to recover messages. Fig. 13 shows the recovery of data across L values, with and without the countermeasure implemented. What should immediately be apparent is that recovery is significantly lower with the countermeasure implemented. For instance, without the countermeasure at $L = 3$, Fig. 13a shows a recovery rate for the basic Fixed, Sync, and Independent attackers of approximately 10% at $K = 1$ and $K = 9$. The NDR Planned attacker shows a recovery rate across the different K values of about 5% to 30%. In comparison, when looking at the same experimental parameters, but with a countermeasure of $H = 2$ in Fig. 13b, all attackers show a recovery rate of, at most, 2%.

Next we see how recovery rate varies with path length, L. Without our countermeasure, the recovery rate at $L = 6$ is similar to that at $L = 3$. However, with the countermeasure, the NDR Planned attacker shows a higher recovery rate (by about 15%) at $L = 6$ compared to $L = 3$. In either case, the basic attackers have almost no ability to recover messages. What this demonstrates is that the countermeasure parameter, H, must be adjusted with the path length, L, in order to minimize the ability of the attacker to recover messages. Note, however, that the defender still benefits from implementing a countermeasure against all attackers, even with the lowest H value of 2.

E. Discussion

Here we discuss additional considerations for our countermeasure. Although the proposed countermeasure provides improved protection, as our results show, there are challenges to its implementation in real networks. In a real network, links may have different latencies and other traffic will be traversing nodes in the paths, causing latency variations. A resourceful attacker could try to overload nodes by generating heavy traffic and thus delay shares in the network longer and nullify the impact of our countermeasure. If such a situation occurs, the buffer of an affected node may contain multiple shares belonging to different sets, which can be captured together by probing that node.

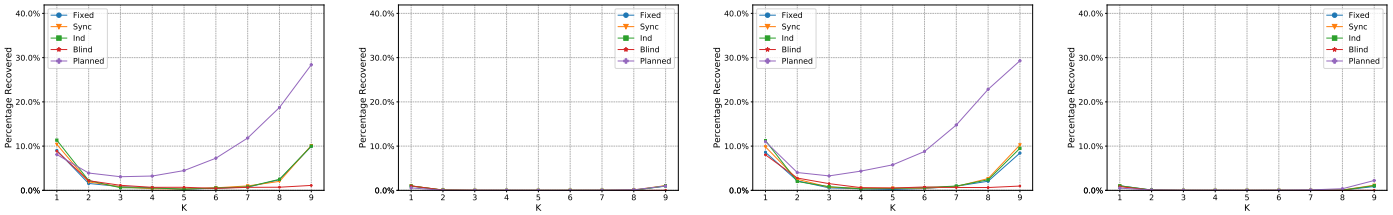
The proposed countermeasure also has negative impact on performance measures like end-to-end latency and goodput. Therefore, choosing an appropriate value for the parameter H is important to provide a suitable trade-off between the probability of data recovery and the performance of network. This raises the question: what is the minimum value for H needed to achieve a desired level of protection? Parameter H should scale with L since each network hop provides an opportunity for attacker to capture shares. Thus, the heuristic approach would be to set the value of H to L. As can be seen in Figs. 4 and 9, applying the proposed countermeasure with $H = L - 1$ to a network with $N = 7$ decreases the probability of data recovery for all attackers significantly. For example, the probability of data recovery by the NDR Blind attacker becomes negligible, less than 0.01 for $L > 3$. Using Eqs. (8), (13), and (14), we can compute an upper bound for the probability of recovery by the NDR Blind attacker and the exact probabilities of recovery by the NDR Planned and Synchronized attackers in the ideal system model. Thus, a more rigorous guideline is to compute the exact recovery probability or its upper bound, depending on the target attacker, for different values of H starting from $H = 2$ and increasing H until the recovery probability becomes as low as desired.

In real networks, links usually have different latencies. Moreover, queuing and processing delays occur at each node. Although all of these factors affect the protection provided by the countermeasure and, thus, the appropriate value of H in real networks, we can select H based on the numerical results for the ideal system model where values of parameters N, K, and L are the same as those in real network. If paths do not have the same length, a conservative strategy would be to set H to the number of intermediate nodes in the longest path.

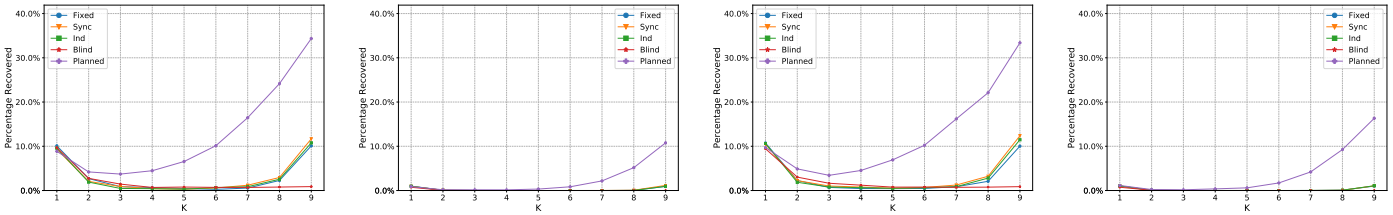
VIII. Generality

In this work, we focused on a specific scheme to be able to provide detailed experiments and analytical results. However, the new class of vulnerabilities that we discussed in this paper is general and attacks similar to ours apply to other schemes that assume transfer atomicity. Since the threat models of these schemes are not all the same, the precise interpretation of the attack may vary. Having said that, because the MSSS is the most featureful of such schemes, the attacks presented naturally apply to other schemes.

Previous schemes can be categorized in two dimensions: whether or not the scheme uses secret sharing (*i.e.*, a message is broken into multiple shares), and whether or not the schemes dynamically switches the path(s) used for communication. Since it does not make sense to send multiple shares on the same path, all secret-sharing based schemes also use multi-path routing, and all schemes without secret sharing use a single path for each message. Thus, there are four possibilities, depicted in Fig. 14. Our analysis so far has been focused on the top-left quadrant. In this section, we discuss the generality



(a) $L = 3$, No Countermeasure (b) $L = 3$, With Countermeasure (c) $L = 4$, No Countermeasure (d) $L = 4$, With Countermeasure



(e) $L = 5$, No Countermeasure (f) $L = 5$, With Countermeasure (g) $L = 6$, No Countermeasure (h) $L = 6$, With Countermeasure

Fig. 13: Effect of the countermeasure and number of shares on percentage of recovered data with varying path length. Fixed scenario with 2 *ms* delay between each node. The sender's $\delta = 4$ *ms* and the attacker's $\delta = 8$ *ms*. File size and the resilience factor, H, are set to 1 *MB* and 2, respectively.

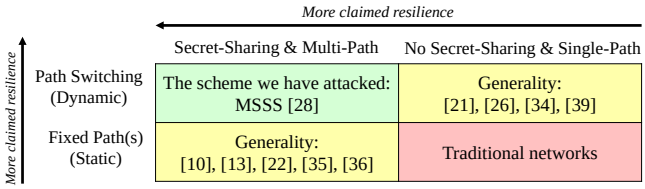


Fig. 14: Previous work and the generality of our attack.

of our work for the top-right and bottom-left quadrants. The bottom-right quadrant would just be a traditional network.

Secret Sharing & Multi-Path with Fixed Paths: Schemes proposed in [10], [13], [22], [35], [36] leverage the secret sharing scheme and multi-path routing, but only on a fixed set of paths (static). They are vulnerable to our attacks because in essence they are a weak form of MSSS in which the switching interval is ∞. In fact, their vulnerability is more severe because as soon as the set of paths is revealed to an attacker, the attacker do not need to switch nodes anymore and can capture the rest of the flow with probability 1.

No Secret Sharing & Single-Path but Path Switching: Schemes proposed in [21], [26], [34], [39] do not use secret sharing and send each message via a single path, but they use path switching for resilience. They thus correspond to the special case of $K = 1$ in our experiments/analyses. They are thus vulnerable to the NDR side-channel we uncovered herein.

Finally, while we presented our attacks on an SDN implementation for convenience, neither our attacks, nor our countermeasure, is limited to SDNs. In fact, any network that can implement the schemes proposed in the literature can indeed implement our countermeasure as well, because it only relies on increased shares and timing of share transmission.

IX. Related Work

We discussed the most relevant previous work in Section VIII. Here, we briefly mention other, tangentially-related previous work for the sake of completeness.

Switching Other Network Identifiers: A great variety of work [37], [8], [38], [41], [31], [32], [40], [3], [20] has looked at switching among different IP addresses, ports, and other identifiers in an attempt to prevent attacks which require network reconnaissance [3], [20], reduce the amount of information an attacker gains about the network [37], [8], [31], [32], [40] or make it hard to target a particular flow [38], [41]. However, none of these approaches attempt to protect the data itself, and are thus out of scope for our work.

Data Remanence Attacks: Data remanence in the context of storage devices has been studied extensively in the literature. We refer the reader to the vast body of literature on that topic [18], [19], [4], [5]. Networks are also vulnerable to memory data remanence attacks as memory is a main component of switches. Cao et al. [7] show that a malicious application can exploit the lack of consistency checking for buffer IDs, which allows an attacker to hijack the buffered packets, implementing a memory data remanence attack, in essence. Our work introduces another kind of data remanence attack that is not based on the memory data remanence property. We exploit the data that is being transferred along network paths.

X. Conclusion

In this paper, we uncovered a new side-channel vulnerability, called Network Data Remanence (NDR), that can be used to attack secret sharing-based schemes. We focused on the most featureful of such schemes, which provides secret sharing, multi-path routing, and path switching, and illustrated, via testbed experiments, Mininet experiments, and analytical results, that their guarantees can indeed be broken by the new NDR side-channel. Moreover, we proposed a countermeasure to spread shares over both time and space, and implemented and evaluated our countermeasure. Finally, we discussed the generality of our attack and proposed countermeasures for other schemes that provide fewer features.

REFERENCES

[1] H. Ahmadi and R. Safavi-Naini, "Multipath private communication: An information theoretic approach," *arXiv preprint arXiv:1401.3659*, 2014.

[2] ——, "Private message transmission using disjoint paths," in *International Conference on Applied Cryptography and Network Security*, 2014, pp. 116–133.

[3] E. Al-Shaer, Q. Duan, and J. H. Jafarian, "Random host mutation for moving target defense," in *Security and Privacy in Communication Networks*, A. D. Keromytis and R. Di Pietro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 310–327.

[4] B. Albelooshi, K. Salah, T. Martin, and E. Damiani, "Experimental proof: Data remanence in cloud vms," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 1017–1020.

[5] X. Bellekens, G. Paul, J. M. Irvine, C. Tachtatzis, R. C. Atkinson, T. Kirkham, and C. Renfrew, "Data remanence and digital forensic investigation for cuda graphics processing units," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1345–1350.

[6] G. R. BLAKLEY, "Safeguarding cryptographic keys," in *1979 International Workshop on Managing Requirements Knowledge (MARK)*, 1979, pp. 313–318.

[7] J. Cao, R. Xie, K. Sun, Q. Li, G. Gu, and M. Xu, "When match fields do not need to match: Buffered packet hijacking in sdn," in *NDSS*, 2020.

[8] S.-Y. Chang, Y. Park, and A. Muralidharan, "Fast address hopping at the switches: Securing access for packet forwarding in sdn," in *IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 454–460.

[9] R. Cramer, I. Damgård, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 316–334.

[10] R. A. Dilruba, "Quantum-safe switch-controller communication in software-defined network," Master's thesis, Science, 2017.

[11] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[12] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *Journal of the ACM (JACM)*, vol. 40, no. 1, pp. 17–47, 1993.

[13] S. Dolev and S. Tzur-David, "A method for establishing a secure private interconnection over a multipath network," European Patent EP3146668A1, Mar. 2017.

[14] O. N. Foundation. (2012) Openflow switch specification. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[15] P. R. Gallagher, "A guide to understanding data remanence in automated information systems," 1991.

[16] D. Genkin, L. Pachmanov, E. Tromer, and Y. Yarom, "Drive-by key-extraction cache attacks from portable code," in *International Conference on Applied Cryptography and Network Security*. Springer, 2018, pp. 83–102.

[17] J. Granjal, E. Monteiro, and J. Sá Silva, "Security for the internet of things: A survey of existing protocols and open research issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[18] P. Gutmann, "Data remanence in semiconductor devices." in *USENIX Security Symposium*, 2001, pp. 39–54.

[19] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: Cold-boot attacks on encryption keys," *Commun. ACM*, vol. 52, no. 5, p. 91–98, 2009.

[20] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012, p. 127–132.

[21] ——, "Formal approach for route agility against persistent attackers," in *Computer Security – ESORICS 2013*, J. Crampton, S. Jajodia, and K. Mayes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 237–254.

[22] W. Lou and Y. Kwon, "H-spread: A hybrid multipath scheme for secure and reliable data collection in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 55, pp. 1320 – 1330, 2006.

[23] B. Möller, T. Duong, and K. Kotowicz, "This poodle bites: exploiting the ssl 3.0 fallback," *Security Advisory*, 2014.

[24] A. Networks. Aruba 2930f switch series. [Online]. Available: https://www.arubanetworks.com/assets/ds/DS_2930FSwitchSeries.pdf

[25] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *Information and Communications Security*, P. Ning, S. Qing, and N. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 529–545.

[26] Qi Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *IEEE Conference on Communications and Network Security (CNS)*, Oct 2013, pp. 260–268.

[27] E. Rescorla and T. Dierks, "The transport layer security (tls) protocol version 1.3," 2018.

[28] R. Safavi-Naini, A. Poostindouz, and V. Lisy, "Path hopping: An mtd strategy for quantum-safe communication," in *ACM Workshop on Moving Target Defense*, 2017, pp. 111–114.

[29] ——, "Path hopping: An mtd strategy for quantum-safe communication," in *ACM Workshop on Moving Target Defense*, 2017, pp. 111–114.

[30] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[31] L. Shi, C. Jia, S. Lü, and Z. Liu, "Port and address hopping for active cyber-defense," in *Pacific-Asia Workshop on Intelligence and Security Informatics*, 2007, pp. 295–300.

[32] M. Sifalakis, S. Schmid, and D. Hutchison, "Network address hopping: a mechanism to enhance data protection for packet communications," in *IEEE International Conference on Communications*, 2005, pp. 1518–1523.

[33] T. Simonite, "Nsa says it "must act now" against the quantum computing threat," 2016.

[34] E. Talipov, D. Jin, J. Jung, I. Ha, Y. Choi, and C. Kim, "Path hopping based on reverse aodv for security," in *Asia-Pacific Network Operations and Management Symposium*, 2006, pp. 574–577.

[35] Wenjing Lou, Wei Liu, and Yuguang Fang, "Spread: enhancing data confidentiality in mobile ad hoc networks," in *IEEE INFOCOM 2004*, vol. 4, 2004, pp. 2404–2413 vol.4.

[36] Wenjing Lou and Yuguang Fang, "A multipath routing approach for secure data delivery," in *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, vol. 2, Oct 2001, pp. 1467–1473 vol.2.

[37] L. Zhang, Y. Guo, H. Yuwen, and Y. Wang, "A port hopping based dos mitigation scheme in sdn network," in *IEEE International Conference on Computational Intelligence and Security (CIS)*, 2016, pp. 314–317.

[38] L. Zhang, Z. Wang, K. Gu, F. Miao, and Y. Guo, "Transparent synchronization based port mutation scheme in sdn network," in *The 5th International Conference on Computer Science and Network Technology (ICCSNT)*, 2016, pp. 581–585.

[39] L. Zhang, Q. Wei, K. Gu, and H. Yuwen, "Path hopping based sdn network defense technology," in *IEEE International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2016, pp. 2058–2063.

[40] Z. Zhao, D. Gong, B. Lu, F. Liu, and C. Zhang, "Sdn-based double hopping communication against sniffer attack," *Mathematical Problems in Engineering*, 2016.

[41] Z. Zhao, F. Liu, D. Gong, L. Chen, F. Xiang, and Y. Li, "An sdn-based ip hopping communication scheme against scanning attack," in *IEEE International Conference on Communication Software and Networks (ICCSN)*, 2017, pp. 559–564.