Securing Communication Against Leaky Switches

Leila Rashidi
Department of Computer Science
University of Calgary, Calgary, Canada
leila.rashidi@ucalgary.ca

Sogand Sadrhaghighi
Department of Computer Science
University of Calgary, Calgary, Canada
sogand.sadrhaghighi@ucalgary.ca

Majid Ghaderi
Department of Computer Science
University of Calgary, Calgary, Canada
mghaderi@ucalgary.ca

Cristina Nita-Rotaru
Khoury College of Computer Science
Northeastern University, Boston, USA
crisn@ccs.neu.edu

Reihaneh Safavi-Naini
Department of Computer Science
University of Calgary, Calgary, Canada
rei@ucalgary.ca

***Abstract*—Internet providers are expected to provide trustable networks for their users. However, this task is becoming increasingly difficult to achieve as network devices become more complex and prone to malicious software and hardware that can be installed on such devices at various stages of the supply chain or after deployment. Herein, we consider the problem of secure communication when an adversary surreptitiously exfiltrates part of transmissions from within the network. We model the leakage as probabilistic sampling of traffic at switches and propose a novel scheme for transmitting user messages that uses secret sharing to disperse shares of a message over multiple network paths such that the leakage probability of the message is bounded by a required leakage threshold, while link bandwidths are respected. To respect the link bandwidth, the proposed scheme dynamically computes the number of shares generated for each message and the number of shares sent on each path while allowing the use of non-disjoint paths and sending more than one share of a message on a path. These features distinguish our scheme from the previous work. The security guarantee of our scheme is information theoretic (hence post-quantum). To validate our theoretical results and show the efficacy of the proposed scheme to reduce the packet drop rate and its impact on communication latency, we evaluate it using both Mininet and discrete-event simulations. The experiments show that message transmission using our scheme achieves higher goodput compared to baseline schemes that (i) exclude leaky switches, and (ii) use only node-disjoint paths.**

***Index Terms*—Untrusted devices, secret sharing**

I. Introduction

A. Motivation

Verifying correct operation of complex networking devices such as routers and switches, is a challenging task. Detecting small leakages in particular is challenging, if possible at all, because a leaky device does not deviate from its normal operational profile. A device can silently copy packets to a secret buffer and gradually send them at a low rate, while remaining undetected. A natural question is, assuming that switches in a networking infrastructure may leak packets, is it possible to counter the effect of leakages without significantly increasing the computation and communication overhead of the network providers. Such a solution must provide long-term security and adhere to the latest requirements in network security, in particular security against an adversary with access to a quantum computer. An immediate solution is to use end-to-end encryption with post-quantum (PQ)-security at the edge routers, i.e., encrypt traffic between a sending edge router and a receiving one. Using PQ-secure key agreement schemes [1]–[3], one can ensure PQ-security for the communication, taking advantage of the PQ-security of symmetric key primitives such as AES [4]. Such an encrypted tunnel would require complex key management at the edge of the network, but more importantly, *will not protect against offline attacks and cannot guarantee future security (future-proofing) of communication*. That is even if one employs PQ-secure key agreement and encryption, the PQ-security guarantee will be for the packets in transit. Leaked packets however can be stored offline and will remain vulnerable to future developments in computing technologies including development of quantum computers and cryptanalysis methods which will directly affect long-term security of communication.

A second solution is to attach trust levels to devices using attributes such as trust in the manufacturers, and remove devices that do not satisfy the required threshold of trustworthiness. This approach however has significant drawbacks. State-of-the-art devices are manufactured by a small number of companies and excluding some of them would result in delays in the deployment of advanced networking services and increased cost. Removing exiting "untrusted" devices from a network can also drastically reduce network transmission capacity and in the worst case can totally break the communication (e.g. if there is at least one untrusted switch on each path). More generally, replacing devices that are already in use in a network requires multi-year planning [5] to ensure network services continue. Finally, trust in a manufacturer may change over time and deployed devices of a currently "trusted" manufacturer may become "untrusted" in the future [6].

A promising approach to provide security in the above setting is to adapt the research on providing security against

L. Rashidi and R. Safavi-Naini were supported by Alberta Innovates Strategic Chair in Information Security grant, and Natural Sciences and Engineering Research Council of Canada Industrial Chair program in partnership with Telus Communications. S. Sadrhaghighi and M. Ghaderi were supported by Natural Sciences and Engineering Research Council of Canada.

978-1-6654-6255-6/22/$31.00 ©2022 IEEE

untrusted switches, in which an untrusted switch discloses all its traffic to the adversary [7]–[10], to the setting of leaky switches. Security is provided by a $(k,\ k)$ secret sharing scheme used to send shares of each message along k node-disjoint paths between the sender and the receiver switches. The critical assumption is that at most $k-1$ switches are untrusted, so that at least one share will remain secure from the adversary. Important attractive properties of this approach are PQ-security that follows from information-theoretic security of the secret sharing, efficient encoding of messages into shares, and efficient decoding at the receiver (reconstruction from received shares). Its drawbacks include an increase in the communication cost which grows linearly with the number of shares, and, more importantly, difficulty of implementing node-disjoint paths in practice, in particular guaranteeing disjointness at the lower layers of the network. *The approach will fail completely if one cannot bound the number of untrusted switches, or cannot find sufficient number of disjoint paths.*

B. Our work

Our main observation is that to stay undetected and below the threshold of network monitoring tools, back-doored switches must limit their leakage and leak a small fraction of the traffic. We use this observation to model the network as a "leaky" communication channel, and represent it by a set of "leaky paths" that may have shared links where links have (limited) known bandwidths. To encode and send messages under this model, we propose a scheme that provides guaranteed security, reminiscent of encoding schemes for wiretap channels [11]. Our scheme is called Adaptive Multipath Secret Sharing (AMSS) and can be seen as the adaptation of secret sharing to our model of a "leaky" network.

We consider an adversary who aims to exfiltrate information from the network over a long period of time, while staying undetected. We assume untrusted switches leak a fraction of the traffic that passes through them and associate a leakage probability to each. A switch independently samples each packet that passes through it according to its associated leakage probability. This probabilistic model is weaker than the adversary who can completely learn the traffic through the switch, or possibly tamper with it. However, it models an undetectable adversary, and it is in line with the probabilistic leakage model of wiretap channels [11], [12] and side-channel leakages [13]. Also, this model can be further strengthened in future works. We assume the network links have assigned bandwidths that must be respected to avoid packet drops at the network switches. We model the network as a set of leaky paths, possibly with common links. The security requirement is given by a *leakage threshold* that is an upper bound on the probability of the leakage of a message to the adversary. The required security is long-term against an adversary that can indefinitely store its (visible) transcript of communication which is the set of sampled packets in our model.

Our scheme employs a number of novel ideas that distinguish our work from previous approaches [7]–[10], [14]. Firstly, we do not require node-disjointness of the paths that is crucial in all previous works that use multiple paths. Our scheme can use any set of paths between the sender and receiver and does not need disjointness of the paths. Secondly, we do not make any assumption on the number of leaky switches. We allow any switch to be leaky although for simplicity, we assume the sender and receiver switches are trusted. Thirdly, we do not fix k, the number of shares that each message is broken into, and the set of paths used to send shares. Instead, for a message m, we determine a corresponding parameter k_m and a $(k_m,\ k_m)$-secret sharing. The number of shares generated for a message depends on the paths used to send them. Thus, each message is sent based on a Share Assignment Vector (SAV (pronounced as one word)), which specifies the number of shares that must be transmitted over each path for a message. The AMSS scheme consists of three components, (i) an algorithm for finding all SAVs, (ii) an optimization problem for finding "secrecy capacity" of the network for the given set of link bandwidths, set of SAVs and the given security requirement, and (iii) an adaptive probabilistic SAV selection strategy.

For a network represented by a set of paths with associated link bandwidths, and a set of SAVs which are computed with respect to a leakage threshold, the secrecy capacity is defined by the maximum number of messages that can be sent over the network in one unit of time, referred to as a *timeslot*, such that link bandwidths are respected. To compute the secrecy capacity, we formulate a constrained Integer Linear Program (ILP), which is solved in an offline manner every time the network topology changes (an infrequent event). The solution of this problem determines the total number of times that each SAV must be used in a timeslot. If buffer capacities at network switches are not sufficient, sending bursty traffic on paths can result in packet loss. To alleviate this problem, our scheme employs an adaptive probabilistic SAV selection strategy to avoid transmitting a large number of subsequent messages using the same SAV.

Experimental Results. We have implemented AMSS in Mininet [15] to show its feasibility in practice. For comparison, we also implemented two baselines including: *i*) Only Trusted Switches (OTS) scheme which uses only paths which do not have any leaky switch, and *ii*) Multipath Secret Sharing (MSS) scheme that uses a fixed secret sharing scheme over multiple node-disjoint paths. We compare the goodput of AMSS and baselines under different traffic rates and show that AMSS can achieve up to 2.5 times higher goodput than the baseline schemes. Mininet experiments were conducted using a small-scale network. To study the behavior and performance of AMSS in large-scale networks, we used a custom-built simulator to implement a realistic ISP topology. Using the ISP simulations, we studied the message drop rate under AMSS and baselines as well as the impact of the number of leaky switches on their respective secrecy capacities. We observed that for the same traffic rate, 0.14% and 70% of messages are dropped under AMSS and the baselines, respectively. Moreover, AMSS was able to send more than 4*Gbps* while OTS failed to send any message.

Our models and assumptions are described in Section II. The AMSS scheme is presented in Section III. Section IV is dedicated to the evaluation of AMSS and baselines using experiments with Mininet and simulation. Related works are reviewed in Section V, and Section VI concludes the paper.

II. Models and Assumptions

A. Network Model

We model the network as a directed graph, where nodes and edges represent network switches and links, respectively. There are two special nodes, called the sender and receiver. A set of M (loop-free) paths, with possibly common links, connects the sender to the receiver. The sender node operates in a time-slotted manner. In each timeslot, a set of user messages arrives at the sender node for transmission to the receiver node. We assume communication between the sender and receiver nodes is through packets with fixed length. A packet has a fixed-length header (metadata) and can carry a message. Let $\{1, 2, \ldots, z\}$ be the set of links along the paths. Each link i has bandwidth $b_i \in \mathbb{N}$ that specifies the maximum number of packets that can be sent over that link in a timeslot. We assume that packets are sent only from the sender to the receiver node. The effect of other traffic in the network can be modelled by reducing the bandwidth of each link.

Each switch has a number of incoming and outgoing links. For each outgoing link, there is a buffer with limited capacity that stores packets for transmission over that link. When a packet arrives at a switch, the outgoing link on which the packet should be sent is determined, and then the packet is written to the corresponding outgoing link buffer. Packets sitting in the buffer are transmitted as soon as their corresponding links become available. If a buffer is full and a new packet arrives at the switch, then it is dropped.

B. Security Model

Threat Model. We assume all switches in the network except the sender and the receiver nodes, are *leaky*. A leaky switch leaks a fraction of packets that pass through it. The leaked packets are visible to an outside adversary who wants to remain undetected. The set of sampled packets forms the *adversary's view of the communication*. We model the leakage by a probabilistic process and attach a *sampling probability* p to each leaky switch. A leaky switch samples each arriving packet with probability p, independently from other packets and other switches. Choosing the same sampling probability is to simplify our exposition. Our model and analysis can be generalized to the case that sampling probabilities are different. We focus on passive eavesdropping adversary, and leave security against an active adversary for future work.

Definition 1 (Path Leakage Probability): Given the probabilistic leakage model above, the *leakage probability* of path i, $1 \leq i \leq M$, denoted by l_i, refers to the probability that a packet is sampled by at least one switch along path i.

If $e_i > 0$ denotes the number of leaky switches along path i, then l_i can be computed as $1 - (1-p)^{e_i}$.

The goal of the system is to provide message confidentiality against an adversary that has access to all the leaked packets. We define a *leakage threshold* τ for the system and require that for any message m, the probability that m is disclosed does not exceed τ. The guarantee is unconditional and holds against a computationally unbounded adversary.

III. Adaptive Multipath Secret Sharing (AMSS)

Overview. The key idea in AMSS is the following: for a given message, use (k, k) secret sharing to encode the message into k shares and send them over multiple paths such that the probability that adversary learns the message is bounded by τ. The adversary needs all shares to reconstruct the message. If the leakage probability of a path does not exceed the leakage threshold τ, a message can be directly sent over that path (as a single share); otherwise, by increasing the number of shares, one can reduce the probability of leaking all shares which equals the leakage probability of the message.

Share Assignment Vectors (SAVs). Let n_i denote the number of shares of message m which are sent over the path i ($1 \leq i \leq M$) after applying (k_m, k_m)-secret sharing where $k_m = \sum_{i=1}^{M} n_i$. In order to meet the security guarantee without generating an excessive number of shares, we require vector $(n_1, n_2, \ldots, n_M)$ to have the following properties.

(1) **Security.** The leakage probability of a message must be upper bounded by τ. When n_i shares are sent on path i ($1 \leq i \leq M$), the probability of recovering a message from the shares in the adversary's view is computed as $\Pi_{j=1}^{M}(l_j)^{n_j}$. Thus, the security guarantee can be stated as,

$$\Pi_{j=1}^{M}(l_j)^{n_j} \leq \tau. \tag{1}$$

(2) **Minimality.** The number of shares on each path must satisfy the following condition which states that sending one fewer share on any of the paths will violate the security guarantee for the message,

$$n_i > 0 \Rightarrow \frac{\Pi_{j=1}^{M}(l_j)^{n_j}}{l_i} > \tau, \forall i : 1 \leq i \leq M \tag{2}$$

Definition 2 (Share Assignment Vector): Given M paths between the sender and receiver nodes, a *share assignment vector* refers to a vector of M non-negative integers $(n_1, n_2, \ldots, n_M)$ that satisfies conditions (1) and (2).

Secure message transmission. To send a message, the sender chooses a SAV $(n_1, n_2, \ldots, n_M)$, generates k $(= \sum_{i=1}^{M} n_i)$ shares using a (k, k) secret sharing, and sends n_i shares over path i. Each share is encapsulated in one packet. Sending too many packets on paths will result in the link buffers to be filled and the packets to be dropped. AMSS controls the number of packets that are sent on a path in each timeslot such that, the total number of packets that arrive at a switch over the link i does not exceed the link bandwidth, b_i.

Definition 3 (Secrecy Capacity): The secrecy capacity of a message transmission scheme (such as AMSS) in an eavesdropped network as defined above, is the maximum number

of messages that can be sent by the sender edge switch in one timeslot such that: i) transmission of each message satisfies the required security, and ii) The total number of packets sent on paths which include link i does not exceed the bandwidth, b_i.

AMSS Operation. AMSS has three phases, defined as follows.

1) **Generating Share Assignment Vectors.** For a leakage threshold τ, the set of SAVs is generated and stored. This phase is done offline for the given network, set of paths, and value of τ, and will only need to be performed again if the set of SAVs changes.

2) **Finding Secrecy Capacity.** Given the computed set of SAVs, we formulate an optimization problem to compute the secrecy capacity of AMSS. The solution of the optimization problem specifies the number of messages that can be sent using each SAV at one timeslot.

3) **Message Encoding and Transmission.** To distribute shares that must be sent over path and reduce the chance of buffer overflows and packet drops, in each timeslot, we use an adaptive probabilistic strategy to select SAVs, where probabilities are obtained using the results of Phase 2.

A. Share Assignment Vector Generation

For a network with a single path and leakage probability l_1, there is only one SAV that assigns $\lceil \log_{l_1} \tau \rceil$ shares to the path. As the number of paths increases, the number of SAVs grows as well, and deriving them becomes challenging. Thus, we devise a recursive algorithm, given in Algorithm 1, which generates the set of SAVs for a network consisting of paths 1 to $m \leq M$ and leakage threshold τ, referred as (m, τ)-SAVs. Variable n_m which represents the value of the last element of an (m, τ)-SAV, is set to 0, 1, ..., and finally $\lceil \log_{l_m} \tau \rceil$. If $n_m < \lceil \log_{l_m} \tau \rceil$, the while-loop is iterated to generate (m, τ)-SAVs, for which their last element is n_m, based on $(m-1, \tau')$-SAVs. Thus, there is a recursive call at step 6.

Algorithm 1: Generation of (m, τ)-SAVs

```
   Input: m, τ, l_1, l_2, ..., l_m
1  X ← {}
2  if m > 1 then
3  |   n_m ← 0
4  |   while l_m^{n_m} > τ do
5  |   |   τ' ← τ / l_m^{n_m}
6  |   |   Y ← Set of (m−1, τ')-SAVs, generated by Algorithm 1
7  |   |   for (n_1, n_2, ..., n_{m−1}) ∈ Y do
8  |   |   |   if Π_{j=1}^{m−1} l_j^{n_j} × l_m^{n_m−1} > τ then
9  |   |   |   |   Add (n_1, n_2, ..., n_{m−1}, n_m) to X
10 |   |   n_m ← n_m + 1
11 |   Add (0, 0, ..., 0, n_m) to X
   else
12 |   X ← {(⌈log_{l_m} τ⌉)}
13 Return X
```

Now, we prove that Algorithm 1 generates only all (m, τ)-SAVs for the network consisting of paths 1 to m and any leakage threshold. To this end, we first prove Lemmas 1 and 2.

Lemma 1: If $(n_1, \ldots, n_{m-1}, n_m)$ is an (m, τ)-SAV, then $(n_1, \ldots, n_{m-1})$ is an $(m-1, \tau')$-SAV, where $\tau' = \tau / {l_m}^{n_m}$.

Proof: If $(n_1, \ldots, n_{m-1}, n_m)$ is an (m, τ)-SAV, it satisfies a security condition as $\Pi_{i=1}^{m} (l_i)^{n_i} \leq \tau$. Using $\tau' = \tau / {l_m}^{n_m}$, this condition can be written as $\Pi_{i=1}^{m-1} {l_i}^{n_i} \leq \tau'$, which represents the security condition for the network with paths 1 to $m-1$ and leakage threshold τ'. Similarly, it can be concluded that if $(n_1, \ldots, n_{m-1}, n_m)$ is an (m, τ)-SAV, then $(n_1, \ldots, n_{m-1})$ satisfies the minimality condition. Thus, proof is complete. ∎

Lemma 2: Let $(n_1, \ldots, n_{m-1})$ be an $(m-1, \tau')$-SAV such that $\tau' = \tau / {l_m}^{n_m}$, where $n_m \in \mathbb{Z}^*$. If (3) holds, then $(n_1, \ldots, n_{m-1}, n_m)$ is an (m, τ)-SAV.

$$\Pi_{i=1}^{m-1} {l_i}^{n_i} \times {l_m}^{n_m - 1} > \tau \tag{3}$$

Proof: The security condition satisfied by $(n_1, \ldots, n_{m-1})$ is $\Pi_{i=1}^{m-1} {l_i}^{n_i} \leq \tau'$, and given that $\tau' = \tau / {l_m}^{n_m}$, we have $\Pi_{i=1}^{m} {l_i}^{n_i} \leq \tau$, which indicates that vector $(n_1, \ldots, n_{m-1}, n_m)$ meets the security guarantee with respect to threshold τ. Moreover, $(n_1, \ldots, n_{m-1})$ satisfies a minimality condition given in (4). When (3) and (4) hold, $(n_1, \ldots, n_{m-1}, n_m)$ satisfies the minimality condition required for the network with paths 1 to m and leakage threshold τ. Thus, proof is complete.

$$\frac{\Pi_{i=1}^{m-1} {l_i}^{n_i}}{l_j} > \tau' = \frac{\tau}{{l_m}^{n_m}}, \quad \forall j : 1 \leq j < m. \tag{4}$$

∎

Theorem 1: For any number of paths, denoted as m, and any leakage threshold, denoted as τ, Algorithm 1 generates all SAVs and no other vector.

Proof: We use induction on the number of paths, m, and prove that Algorithm 1 returns the set of all (m, τ)-SAVs for any leakage threshold τ. As mentioned earlier, when $m = 1$, there exists a single SAV which is a vector of length 1 as $(\lceil \log_{l_1} \tau \rceil)$. This SAV is generated at step 12. Now, assume that Algorithm 1 generates all SAVs of the $(i-1)$-path network for any leakage threshold when $m = i - 1$ $(i > 1)$. We will prove that this algorithm returns only all (i, τ)-SAVs when $m = i$ for any leakage threshold τ. Let $(n_1, \ldots, n_i)$ be a vector returned by Algorithm 1. If $\sum_{j=1}^{i-1} n_j > 0$, then this vector has been generated at step 9, and hence $\Pi_{j=1}^{i-1} {l_j}^{n_j} \times {l_i}^{n_i - 1} > \tau$. Also, based on the induction assumption, $(n_1, \ldots, n_{i-1})$ is an $(i-1, \tau'')$-SAV, where $\tau'' = \tau / {l_i}^{n_i}$. Using this and Lemma 2, $(n_1, \ldots, n_{i-1}, n_i)$ should be an (i, τ)-SAV. If $\sum_{j=1}^{i-1} n_j = 0$, then $(n_1, \ldots, n_{i-1}, n_i)$ has been generated at step 11, and given that $n_i = \lceil \log_{l_i} \tau \rceil$, it is an (i, τ)-SAV. Thus, any vector generated by Algorithm 1 is an (i, τ)-SAV.

Now, assume that $(n_1, \ldots, n_{i-1}, n_i)$ is an (i, τ)-SAV, which is not returned by Algorithm 1. If $\sum_{j=1}^{i-1} n_j = 0$, then $n_i = \lceil \log_{l_i} \tau \rceil$. Vector $(0, \ldots, 0, \lceil \log_{l_i} \tau \rceil)$ is generated at step 11. Thus, $\sum_{j=1}^{i-1} n_j > 0$, and according to Lemma 1, $(n_1, \ldots, n_{i-1})$ is an $(i-1, \tau'')$-SAV, where $\tau'' = \tau / {l_i}^{n_i}$. This vector is added to set $\mathcal{Y}$ at step 6. Since $(n_1, \ldots, n_{i-1}, n_i)$

satisfies the minimality condition, step 9 is run at the iteration of the for-loop which corresponds to $(n_1, \ldots, n_{i-1})$. Thus, $(n_1, \ldots, n_{i-1}, n_i)$ must have been added to set $\mathcal{X}$. Therefore, Algorithm 1 is correct and returns a set including all (i, τ)-SAVs and no other vector when $m = i$. ■

In the rest of this section, we derive the time and memory complexities of Algorithm 1 for $m = M$ and the order of the number of SAVs for a network with M paths and threshold τ.

Theorem 2: The number of SAVs for a network with M paths, I, is $O(\log_{l_{max}} \tau \times (M + \log_{l_{max}} \tau)^{\lceil \log_{l_{max}} \tau \rceil})$, where l_{max} refers to the maximum path leakage probability.

Proof: The sum of elements of a SAV is at most $\lceil \log_{l_{max}} \tau \rceil$ due to (2). Thus, I is at most equal to the number of vectors consisting of M non-negative integers, where the sum of elements is at most $\lceil \log_{l_{max}} \tau \rceil$. There are $\binom{M+i-1}{i}$ vectors in which the sum of elements equals i. Since $\binom{M+i-1}{i} \leq (M+i-1)^i$, we have $I < \sum_{i=1}^{\lceil \log_{l_{max}} \tau \rceil} (M+i)^i$. Thus, $I = O(\log_{l_{max}} \tau \times (M + \log_{l_{max}} \tau)^{\lceil \log_{l_{max}} \tau \rceil})$. ■

Theorem 3: The time complexity of generating all SAVs is $O(M \times (\log_{l_{max}} \tau)^2 \times (\log_{l_{max}} \tau + M)^{2\lceil \log_{l_{max}} \tau \rceil - 1})$.

Proof: The time complexity of Algorithm 1 has the same order with the total number of iterations in the for-loop, including all recursive calls of Algorithm 1. We will find the order of the number of these iterations, denoted by N_{for}. When Algorithm 1 is called with input parameter $m = x$, there are $M - x$ ongoing executions of this algorithm called with values M, $M-1$, …, $x+1$ for input parameter m. Variable n_m in Algorithm 1 represents the number of shares assigned to path m. Let (x, y)-execution refer to an execution of Algorithm 1 with $m = x$ while the sum of values of variable n_m in the $M - x$ ongoing executions is y. The number of (x, y)-executions is at most $\binom{y+M-x-1}{y}$. Due to (2), at most $\lceil \log_{l_{max}} \tau \rceil$ shares are assigned to paths by an (M, τ)-SAV. Thus, y cannot exceed $\lceil \log_{l_{max}} \tau \rceil$. As m decreases or τ increases, the number of (m, τ)-SAVs decreases. Thus, in each execution of Algorithm 1, the for-loop is iterated at most I times. Thus,

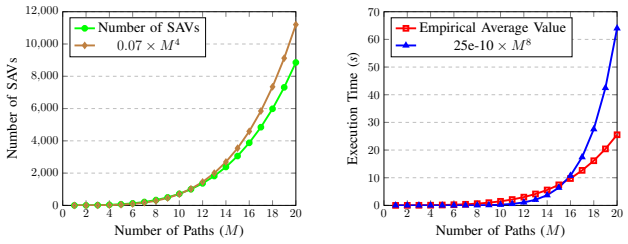
$$\begin{aligned} N_{\text{for}} &\leq I + \sum_{x=1}^{M-1} \sum_{y=0}^{\lceil \log_{l_{max}} \tau \rceil - 1} \binom{y+M-x-1}{y} \times I \\ &\leq M \times \lceil \log_{l_{max}} \tau \rceil \times \binom{\lceil \log_{l_{max}} \tau \rceil + M - 3}{\lceil \log_{l_{max}} \tau \rceil - 1} \times I. \end{aligned} \tag{5}$$

Based on (5) and Theorem 2, we conclude that N_{for} is $O(M \times (\log_{l_{max}} \tau)^2 \times (\log_{l_{max}} \tau + M)^{2\lceil \log_{l_{max}} \tau \rceil - 1})$. ■

Theorem 4: The memory complexity of Algorithm 1 for a network with M paths ($m = M$) and leakage threshold τ is given by $O(M^2 \times \log_{l_{max}} \tau \times (M + \log_{l_{max}} \tau)^{\lceil \log_{l_{max}} \tau \rceil})$.

Proof: As the number of paths (leakage threshold) decreases (increases), the number of SAVs decreases. Thus, the size of sets $\mathcal{X}$ and $\mathcal{Y}$ during each execution of Algorithm 1 is at most I. The maximum number of simultaneous executions is M. Based on this and given that the length of vectors belonging to sets $\mathcal{X}$ and $\mathcal{Y}$ does not exceed M, the memory complexity is $O(I \times M^2)$. Using this and Theorem 2, the proof becomes complete. ■

To verify Theorems 2 and 3, we evaluated the number of SAVs and the execution time of Algorithm 1 for networks with different number of paths, where the leakage probability of each path and the leakage threshold were 0.01 and 10e-7, respectively. Under this setting, according to Theorems 2 and 3, the number of SAVs and the execution time of Algorithm 1 are $O(M^4)$ and $O(M^8)$, respectively. The results presented in Fig. 1 support this. Each empirical result is the average value of 10 runs on a PC with 16 *GB* RAM and Apple M1 (max 3.2 *GHz*). The maximum memory consumption was 21.012 *KB*.



(a) (b)

Fig. 1: Effect of the number of paths on the number of SAVs and the execution time of Algorithm 1 ($l_i = 0.01$, $\tau = 10^{-7}$).

B. Finding Secrecy Capacity

Problem 1 formulates an Integer Linear Program (ILP) that computes the maximum number of messages which can be sent in a timeslot using SAVs, such that the links' bandwidths are not exceeded. Let I denote the number of SAVs and $(n_1^i, n_2^i, \ldots, n_M^i)$ denote SAV i. For each SAV i, a decision variable x_i is defined to represents the number of messages that are sent in a timeslot using SAV i. The objective function is the total number of messages sent using all SAVs. The left-hand side of constraint (6b) represents the number of packets sent over paths which include link k at one timeslot. This constraint ensures that the links' bandwidths are not exceeded.

The secrecy capacity of AMSS can be computed as follows:

$$N_{\text{AMSS}} = \sum_{i=1}^{I} x_i^*, \tag{7}$$

where $(x_1^*, x_2^*, \ldots, x_I^*)$ denotes a solution of Problem 1. To solve the optimization problem, solvers such as Gurobi [16]

Problem 1: Finding the Secrecy Capacity

Input: Set of links which are on path j represented as Γ_j ($1 \leq j \leq M$), Link bandwidths ($b_1, \ldots, b_z$), SAVs represented as $(n_1^i, \ldots, n_M^i)$
Output: $x_1^*, x_2^*, \ldots, x_I^*$

$$\max_{x_1, x_2, \ldots, x_I} \sum_{i=1}^{I} x_i \tag{6a}$$

$$\text{s.t.} \sum_{j=1, k \in \Gamma_j}^{M} \sum_{i=1}^{I} (x_i \cdot n_j^i) \leq b_k, \ \forall k : 1 \leq k \leq z \tag{6b}$$

$$x_1, x_2, \ldots, x_I \in \mathbb{Z}^* \tag{6c}$$

can be used. Solving the optimization problem for the network in Fig. 6, when the leakage probability of each path and the leakage threshold are 0.001 and 10^{-7}, respectively, using Gurobi 9.1.1 takes approximatley 0.02 sec on a low-end PC with an Intel Core(TM) i7-6650U CPU and 8 GB RAM. The number of SAVs in this example was 680.

C. Message Encoding and Transmission

When a message arrives in the sender node, a SAV is selected and used to send it. The chosen SAV determines the number of shares k that must be used for the message, where k is the sum of all elements of the SAV. Then, the message is broken into k shares using a perfect $(k,\ k)$-secret sharing scheme, and on each path as many shares as prescribed by the SAV will be sent. When all the shares of a message are delivered to the receiver node, the message can be reconstructed. More details on SAV selection is given below.

If the number of input messages per timeslot does not exceed N_{AMSS}, AMSS will send at most x_i^* input messages according to SAV i. To enforce this property, one can simply send the first x_1^* messages using SAV 1, the next x_2^* messages according to SAV 2, and so on. In real life application however, if x_i^* is sufficiently larger than the capacity of buffers of the network switches, sending x_i^* consecutive messages over a short period of time, using the same SAV, would likely to result in buffer overflow on some links. To alleviate this problem, we implement an adaptive probabilistic SAV selection strategy for AMSS. For the k^{th} input message in a timeslot, we use the probability distribution $P(i,k)$, given in (8), to select SAV i.

$$P(i,k) = \frac{\lceil \frac{k}{N_{\text{AMSS}}} \rceil \times x_i^* - q_{k-1}^i}{\sum_{j=1}^{I} (\lceil \frac{k}{N_{\text{AMSS}}} \rceil \times x_j^* - q_{k-1}^j)}, \tag{8}$$

where q_{k-1}^j denotes the number of previous messages in the same timeslot which are sent using SAV j.

IV. Experimental Results

In this section, we evaluate the performance of AMSS using Mininet [15] and simulation experiments. Mininet is a network emulator that creates a realistic virtual network composed of virtual hosts, switches, controllers, and links. Experimental results obtained in Mininet are expected to closely resemble those obtained in a physical network. We conducted Mininet experiments to study the performance and behavior of AMSS in a small-scale but realistic network scenario. Simulation experiments, on the other hand, are intended to show the behavior of AMSS in a large ISP network. We aim to answer the following questions:

Q1. What is the impact of the leakage threshold on the number of disclosed messages under AMSS?
Q2. How does the goodput of AMSS compare to the schemes which use only trusted switches or disjoint paths?
Q3. Does the input message rate affect the latency of transmitted messages under AMSS?
Q4. Can AMSS provide perfect message delivery if the input message rate is less than or equal to the secrecy capacity?
Q5. Can sending more than the secrecy capacity put the message delivery, i.e., delivering all shares, at risk?

Alternative Base Schemes. We compare AMSS with the following schemes:

Only Trusted Switches (OTS): It is a multipath routing scheme that only uses fully trusted paths, *i.e.,* paths with no leaky switch. As such, OTS guarantees zero leakage for every message. OTS works in the same way as AMSS when leakage threshold is zero. We assume that the network provider knows which switches are leaky since otherwise, the OTS scheme is not applicable.

Multipath Secret Sharing (MSS): This scheme is a representative of the existing schemes which break each message into a fixed number of shares using secret sharing and send them over a fixed set of disjoint paths [7]–[10]. Each message is divided into as many shares as the maximum number of node-disjoint paths between the sender and receiver, and then one share of the message is sent on each node-disjoint path.

Metrics. We report the following performance metrics:

- *Empirical Leakage Rate:* The ratio of the number of disclosed messages to the total number of messages sent over paths which are not fully trusted.
- *Goodput:* The rate at which message data is delivered by the network to the receiver. A message is delivered if all its shares are received by the receiver.
- *Latency:* The time it takes from processing a message in the sender edge switch to the time all of its shares are delivered to the receiver host.
- *Empirical Secrecy Capacity:* The maximum goodput that can be achieved between the sender and receiver under a message transmission scheme.
- *Percentage of Dropped Messages:* The percentage of messages which cannot be reconstructed at the receiver due to loss of some share(s) in the network.

A. Mininet Experiments

Setup. For this set of experiments, we used the topology depicted in Fig. 2. The switches shown as blue and brown circles are trusted and untrusted leaky, respectively. The switches are interconnected to form a network that includes five paths between the sender and receiver. The sampling probability on each leaky switch is set to 0.01. The port buffer size and propagation delay of each link are set to 50 *KB* and 1 *ms*, respectively. The bandwidth of each link is shown in the figure in units of *Mbps*. Since these experiments are conducted in Mininet, we set link bandwidth to a small value to be able to complete each experiment run in a reasonable amount of time. The duration of timeslot and the leakage threshold are set to 1 *s* and 0.0001, respectively. The packet length is 500 *B*. Each packet has an overhead of 46 *B* for header information. We used the modified `ofsoftswitch` 1.3 [17], [18] to emulate leaky switches. After finding the secrecy capacity, `ofsoftswitch` was extended to implement randomized SAV selection and sending messages using SAVs in

its pipeline, to be used as the sender switch. The experiments were conducted on a VM with 4 CPU cores and 4 *GB* RAM. Additionally, we used *ryu* [19] as our SDN controller to program network switches using OpenFlow. As represented in Fig. 2, we connect two hosts to the sender and receiver edge switches, called *sender host* and *receiver host*. The sender host uses the Scapy utility [20] to generate UDP traffic at a given rate and sends it to the receiver host.

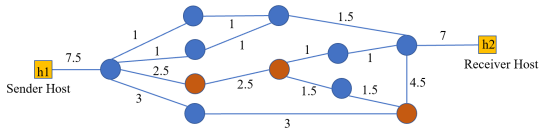
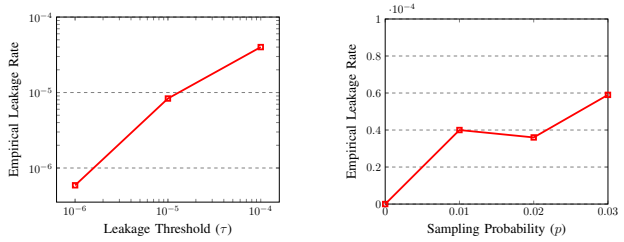


Fig. 2: Network topology for Mininet experiments.

Limiting the Leakage Rate by AMSS. To answer Q1, we measured the leakage rate of AMSS under different values for the sampling probability and the leakage threshold. If the number of messages sent over paths including leaky switch(es) is sufficiently large, we expect that the empirical leakage rate to be less than or equal to the leakage rate. Note that, depending on the sampling probability of leaky switches, network topology and the required leakage threshold, the leakage probability of some messages may be less than the required leakage threshold under AMSS, resulting in the empirical leakage rate being less than the leakage threshold. The results presented in Fig. 3 are averaged over five runs such that in each run, $2,000,000$ messages were sent to the edge switch at the rate of 3 *Mbps*. Please note that the traffic rate was chosen such that no packet was dropped. According to Fig. 3, the empirical leakage rate is always less than the leakage threshold as expected.



(a) $p = 0.01$ (b) $\tau = 0.0001$

Fig. 3: The leakage rate of the AMSS scheme for the network in Fig. 2, obtained from Mininet experiments.

Comparison with Baseline Schemes. To answer Q2, we measured the goodput of each scheme for different traffic rates. In each experiment, the sender host sends traffic to the edge switch for a duration of one minute with a constant traffic rate. The packets delivered to the receiver host were captured and post-processed to determine the goodput. For the network given in Fig. 2, the MSS scheme divides each message into three shares. One of them is sent along a fully trusted path, and the others are sent on the remaining two node-disjoint paths. Table I presents the secrecy capacity achieved with each scheme. In particular, we see that AMSS improves capacity by a factor of 2.5x compared to the OTS scheme. Interestingly, MSS performs poorly, resulting in a capacity that is even less than that of OTS.

Fig. 4 presents the goodput achieved under different algorithms when the traffic rate at the sender host changes from 0.5 *Mbps* to 5.5 *Mbps*. We chose the range of traffic rates such that it includes the secrecy capacity for any scheme. For each scheme, the traffic rate that results in a goodput equal to the secrecy capacity is slightly higher than the secrecy capacity, reported in Table I, due to the overhead added to each packet. As observed in Fig. 4, the goodput of each scheme grows linearly with the traffic rate until the traffic rate exceeds the rate that achieves the secrecy capacity of the scheme. After that point, the goodput achieved by OTS remains constant, while the goodputs of MSS and AMSS experience a downward trend. This observation can be interpreted as a penalty for exceeding the secrecy capacity. In particular, when the number of messages per timeslot exceeds the secrecy capacity of AMSS or MSS, the bandwidth is wasted transmitting shares of those messages that cannot be reconstructed at the receiver due to the loss of some other share(s). *This observation suggests that secret sharing-based schemes such as AMSS and MSS are more vulnerable to packet loss than OTS when all shares are required for reconstruction.*

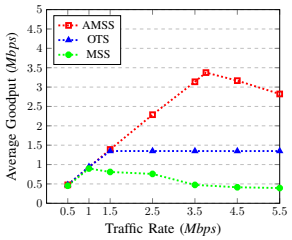


Fig. 4: AMSS comparison with baseline schemes. The goodput of all three schemes initially increases. As the input message rate exceeds the secrecy capacities of AMSS and MSS, the corresponding goodputs drop because the loss of a single share prevents reconstruction of the message.

TABLE I: Secrecy capacities of schemes for topology given in Fig. 2 under $\tau = 0.0001$ and message length 454 *B*.

Scheme	Secrecy Capacity (*Mbps*)	Traffic Rate (*Mbps*)
AMSS	3.446	3.795
MSS	0.908	1.000
OTS	1.362	1.500

Latency Analysis. To answer Q3, we conducted a set of experiments where in each experiment, $100,000$ packets were sent at a constant rate. Fig. 5 represents the Cumulative Distribution Function (CDF) of the latency for different traffic rates. There are at least four links along paths connecting the sender edge switch to the receiver, each of them with propagation delay of 1 *ms*.

Thus, the CDF under all traffic rates is zero in the interval $[0, 4]$ *ms*. Although there are at most six links along paths, the latency of at least 30% of messages is more than 6 *ms* even when the traffic rate is as low as 1 *Mbps* due to processing and queuing delay at switches. As the traffic rate increases, the value of CDF decreases since the queuing delay in switches increases. Thus, Q3 is answered.

B. Simulation Experiments

Setup. To study the performance of AMSS in larger network, we developed a discrete-event simulator in Java. In the simulations, we used a real-world ISP topology, named *rf3967* in dataset [21]. We consider the 15 shortest paths from the sender switch to the receiver switch to transfer traffic. The two links shown as a dashed green arrow have 2.4 *Gbps* bandwidth, while the rest of the links have 10 *Gbps* bandwidth [21]. Additionally, we set the propagation delays of links to the integer values reported in the dataset, considering *ms* as the unit. The processing delay at each switch is 1 *ns*. The default number of leaky switches is 9, shown with brown circles in Fig. 6. We set the leakage threshold, sampling probability, and length of timeslot to 10^{-8}, 0.0001, and 1 *s*, respectively. We connected the sender and receiver hosts to two switches as shown in Fig. 6. The simulation is run for one million messages arrived as the constant bit rate process. The messages are put into packets of size 500 *B*, out of which 46 *B* are used for the header fields. Each input/output port has a buffer which can store up to 50 packets.

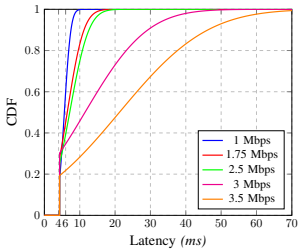


Fig. 5: The CDF of the latency of the AMSS scheme in interval $[0, 100]$ *ms* for different traffic rates.

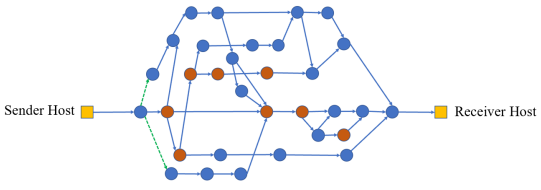
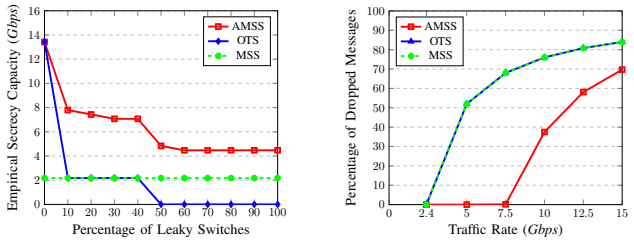


Fig. 6: The topology used for simulation experiments.

To evaluate the effect of the number of leaky switches on the secrecy capacity, we started with an empty set of leaky switches and incrementally expand it by adding new randomly selected switches. Fig. 7a shows the empirical secrecy capacity obtained under various values for the number of leaky switches in the topology given in Fig. 6. When 30% of switches were leaky, their positions were as represented by Fig. 6. Under this setting, the empirical secrecy capacity of each scheme is very close to the secrecy capacity reported in Table II, which indicates that the packet drop rate was negligible. As observed in Fig. 7a, when all switches are trusted, the AMSS and OTS schemes result in the same empirical secrecy capacity since both schemes send only one share per message and use the same set of paths. The secrecy capacity of MSS is much lower than the other schemes when there is no leaky switch.

The secrecy capacities of AMSS and OTS have a downward trend, while the secrecy capacity of MSS stays constant. The reason is that MSS sends three shares for each message over a fixed set of paths regardless of the number of leaky switches. In presence of leaky switches, the secrecy capacity of AMSS is more than two times greater than the secrecy capacity of other schemes. *When at least 50% of switches are leaky, OTS is unusable due to lack of a fully trusted path, but both AMSS and MSS can send at least 2 Gbps data even when all switches are leaky.* The sets of fully trusted paths and SAVs may not change upon increasing the number of leaky switches, which results in some plateaus.



(a) (b)

Fig. 7: The results for the empirical secrecy capacity and the percentage of dropped messages obtained from simulation.

TABLE II: Secrecy capacity for different schemes for topology given in Fig. 6. Traffic rate is computed when the number of messages per timeslot equals the secrecy capacity.

Scheme	Secrecy Capacity (*Gbps*)	Traffic Rate (*Gbps*)
AMSS	7.082	7.500
OTS	2.179	2.400
MSS	2.179	2.400

To answer Q4 and Q5, we measured the percentage of dropped messages under all schemes for different traffic rates, represented in Fig. 7b. According to Table II, the secrecy capacity achieved under both of MSS and OTS is 2.179 *Gbps*, but that of AMSS is around 7 *Gbps*. As shown in Fig. 6, there are two fully trusted paths which have one common link with minimum bandwidth. Thus, both MSS and OTS have the same secrecy capacity, and we expect that the results of OTS and MSS for the same traffic rate to be almost equal as observed in Fig. 7b. When traffic rate is 2.4 *Gbps*, no message is dropped under all three schemes since secrecy capacity is not exceeded.

The traffic rate corresponding to the secrecy capacity of AMSS equals 7.5 *Gbps*. Thus, when the traffic rate is 5 or 7.5 *Gbps*, almost no message is dropped under AMSS, but more than half of messages are dropped under the other schemes, as observed in Fig. 7b. When traffic rate is at least 10 *Gbps*, a considerable fraction of messages is dropped under all three schemes, but a significantly lower percentage is dropped under AMSS.

V. Related Works

Compromised switches exhibit malicious behaviour, *e.g.,* traffic loss, traffic modification. Several works study possible solutions to guarantee the correctness of traffic flows in the presence of such compromised switches in SDNs. [22] ensures the correctness of traffic forwarding, while [23] protects OpenFlow messages. In [24], the authors monitor packet trajectories in a network to detect the malicious switches. All three works target active attack models while our work considers a passive attack model. The works [25]–[27] secure routing in the presence of untrusted hardware by replicating networking devices. The work [26] formulates this approach as a multi-objective optimization problem to maximize the reliability of routing while also minimizing the cost. However, such an approach is costly and not always feasible due to the increased overhead of purchasing and installing extra networking hardware. Our approach is a software-based solution, and, as such, it does not require network infrastructure modifications.

Breaking information into shares as a form of encryption has been used for a variety of purposes, including SDN-based private interconnection [28], switch-controller communication in SDN [29], and secure message transmission in a general network [30]. In [7], [10], and [28], secret sharing and multipath routing are used such that the number of shares going through an individual intermediate node is insufficient to reconstruct the secret. The proposed methods cannot be used for the scenario considered herein since packets sampled on different leaky switches form the adversary's view. The works [8] and [9] use secret sharing and multipath routing to provide confidentiality in mobile ad hoc networks and wireless sensor networks, respectively. Also, the secret sharing and multipath routing can be combined with path switching [14], [31].

VI. Conclusion and Future Work

In this paper, we proposed the Adaptive Multipath Secret Sharing (AMSS) scheme to trade off the network capacity for the security guarantee in a network with untrusted switches which sample the traffic. This scheme guarantees that the probability of disclosing each message does not exceed a threshold. The AMSS scheme maximizes the number of messages that can be sent into the network at one timeslot with respect to the leakage threshold and link bandwidths. We used Mininet and discrete-event simulation to analyze the performance of AMSS and two baselines. One baseline uses only trusted switches while the other allows to use leaky switches by sending one share on each path belonging to the maximal set of node-disjoint paths. According to the experimental results, AMSS achieves the highest goodput. This work can be extended in various ways. We applied the same leakage threshold to all messages. However, in a real network, some controlled messages may require more protection (less leakage threshold) than other messages. Adapting AMSS to this heterogeneous scenario is an interesting future research direction. Another direction is considering a more realistic threat model where attacker targets a specific flow.

References

[1] R. Azarderakhsh, D. Jao, and C. Leonardi, "Post-quantum static-static key agreement using multiple protocol instances," in *Proc. SAC*, 2017.

[2] D. Moldovyan, "Post-quantum public key-agreement scheme based on a new form of the hidden logarithm problem," *CSJM*, 2019.

[3] S. H. Islam, "Provably secure two-party authenticated key agreement protocol for post-quantum environments," *JISA*, vol. 52, 2020.

[4] NIST. (2001) Advanced encryption standard (AES). https://csrc.nist.gov/publications/detail/fips/197/final. Accessed: 2022-04-30.

[5] L. Kelion, "Huawei 5g kit must be removed from uk by 2027," https://www.bbc.com/news/technology-53403793, 2020.

[6] T. Foremski, "Trust in tech companies falls globally," https://www.zdnet.com/article/trust-in-tech-companies-falls-globally/, 2020, accessed: 2021-11-5.

[7] Wenjing Lou and Yuguang Fang, "A multipath routing approach for secure data delivery," in *Proc. MILCOM*, 2001, pp. 1467–1473.

[8] Wenjing Lou, Wei Liu, and Yuguang Fang, "Spread: enhancing data confidentiality in mobile ad hoc networks," in *IEEE INFOCOM*, 2004.

[9] W. Lou and Y. Kwon, "H-SPREAD: A hybrid multipath scheme for secure and reliable data collection in wireless sensor networks," *IEEE TVT*, vol. 55, pp. 1320 – 1330, 2006.

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

[11] A. D. Wyner, "The wire-tap channel," *BSTJ*, vol. 54, no. 8, 1975.

[12] M. Bellare, S. Tessaro, and A. Vardy, "Semantic security for the wiretap channel," in *CRYPTO*, 2012, pp. 294–311.

[13] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *CRYPTO*, 1996, pp. 104–113.

[14] R. Safavi-Naini, A. Poostindouz, and V. Lisy, "Path hopping: An MTD strategy for long-term quantum-safe communication," *Security and Communication Networks*, vol. 2018, pp. 1–15, 2018.

[15] M. P. Contributors. Mininet. http://mininet.org/. Accessed: 2021-06-29.

[16] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2018.

[17] R. Cohen and E. Moroshko, "Sampling-on-demand in SDN," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2612–2622, 2018.

[18] GitHub, "ofsoftswitch13-modified," https://github.com/samplingondemand/ofsoftswitch13-modified, 2018, accessed: 2021-07-07.

[19] "Ryu SDN framework," https://ryu-sdn.org/, 2017, accessed: 2021-07.

[20] P. Biondi and the Scapy community, "Scapy," https://scapy.net/, 2021.

[21] "Declarative and expressive forwarding optimizer," https://sites.uclouvain.be/defo/, 2015, accessed: 2021-04-25.

[22] R. Ghannam and A. Chung, "Handling malicious switches in software defined networks," in *Proc. NOMS*, 2016.

[23] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "Towards resilient in-band control path routing with malicious switch detection in SDN," in *Proc. COMSNETS*, 2018.

[24] K. Thimmaraju, L. Schiff, and S. Schmid, "Software-defined adversarial trajectory sampling," *arXiv preprint arXiv:1705.00370*, 2017.

[25] A. Feldmann et al., "Netco: Reliable routing with unreliable routers," in *Proc. DSN-W*, 2016.

[26] A. Yazdinejad et al., "Cost optimization of secure routing with untrusted devices in software defined networking," *JPDC*, 2020.

[27] D. Senf, H. Shulman, and M. Waidner, "Performance penalties of resilient SDN infrastructures," in *Proc. CoNEXT*, 2020.

[28] S. Dolev and S. T. David, "SDN-based private interconnection," in *NCA*, 2014.

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

[30] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *JACM*, vol. 40, no. 1, pp. 17–47, 1993.

[31] L. Rashidi et al., "More than a fair share: Network data remanence attacks against secret sharing-based schemes," in *Proc. NDSS*, 2021.