StealthPath: Privacy-preserving Path Validation in the Data Plane of Path-Aware Networks

Jiliang Li, Yuan Su, Rongxing Lu, *Fellow, IEEE*, Zhou Su, *Senior Member, IEEE*, Weizhi Meng, *Senior Member, IEEE* and Meng Shen

Abstract—Network path validation aims to give more control over the forwarding path of data packets in a path-aware network, which shields the network from security threats and allows end hosts to receive better services. Therefore, network path validation becomes a vital primitive for secure and reliable Internet services in the next generation networks. The path validation enables end hosts and intermediate router nodes to check whether a packet has followed the intended path. However, the existing solutions fail to protect path privacy and incur significant bandwidth and computation overhead on packet transferring, which degrades packet delivery performance. In this paper, we propose the StealthPath to protect path privacy and improve delivery efficiency. Firstly, StealthPath uses lightweight cryptographic primitives to generate nested proofs and ensures all nodes on the path to check the compliance of the forwarding path efficiently. Secondly, StealthPath hides the forwarding path in the proofs and reduces the proof size from linear to constant, which protects the path information and path length, and decreases the bandwidth consumption. Moreover, StealthPath allows on-path nodes to extract their proofs and the next hop address from proof without leaking on-path node index. Finally, StealthPath is proved to resist various attacks and preserves the path privacy. The experiments show that StealthPath saves nearly 60% header size and bandwidth, and is more efficient than state-of-the-art schemes.

Index Terms—Secure data transmission, Path validation, Path privacy-preserving, Constant proof size.

1 Introduction

With the rapid evolution of the Internet, numerous data packets are being delivered to the diverse application to support various usages. In the Internet, the forwarding path of data packets is beyond the control of end hosts and is entirely determined by the network at its discretion. That is the nodes and end hosts on the path have no way to verify the actual path the packet is traversing on the current Internet [1–4]. As a result, the lack of control over packet delivery may lead to potential data leakage when data is restricted to circulate in a fixed area [5]. Besides, other security threats that waste the Internet resources and expose sensitive information could be suffered, such as source spoofing attacks, packet detouring attacks that cause a significant increase in delay, re-routing attacks launched to obtain illegal certificates, and traffic hijacking attacks that are exploited to de-anonymize users [1, 4, 6–8]. Therefore, the lack of control over packet delivery further compromises the security of online Internet services (e.g., financial, medical and military, etc.) and endangers network security [1, 9–12]. As Path Aware Networking Research Group (PANRG) under the Internet Research Task Force (IRTF) and Internet Engineering Task Force (IETF) advocated [13–15], more control over network paths of packet delivery should be provided to build a robust, fast, and secure network in the next generation networks.

The cryptographic-based path validation is an effective technique, which enables the data packet to be forwarded by a pre-specified network path and allows on-path nodes to verify the path compliance. In existing path validation schemes [16–26], the source node first generates proofs by the cryptographic technique for each on-path node, embeds the proofs to the packet header, and forwards the data packets through the chosen path. Then, each on-path node can check the path compliance of all upstream nodes by the proofs that are embedded in the packet header, and updates the corresponding proofs to allow downstream nodes to validate the routing path.

However, the existing schemes [16–26] (as summarized in Table 1) fail to protect the forwarding path, which makes packet delivery vulnerable to various attacks that affect the reliability, privacy, availability, and service quality offered by the network [1, 4, 6, 7, 27]. Concretely, path validation schemes that lack path privacy protection may expose sensitive information (e.g., physics location) of on-path nodes, and could compromise the security of nodes on the path. Firstly, some key nodes with large degrees could be identified and corrupted if the information of the forwarding path is not concealed, resulting in widespread downtime [28]. Secondly, the intermediate nodes can easily identify the source and destination nodes if the indices of

- *This work is supported in part by National Key R&D Program of China (No. 2022YFB2702800) and NSFC (No. 62102305). (Corresponding author: Yuan Su).*

- *Jiliang Li, Yuan Su, and Zhou Su are with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: jiliang.li@xjtu.edu.cn; suyuan@stu.xjtu.edu.cn; zhousu@ieee.org). Jiliang Li is also with State Key Laboratory of Integrated Services Networks, Xidian University.*
- *Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).*
- *Weizhi Meng is with the Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Copenhagen, Denmark (e-mail: weme@dtu.dk).*
- *Meng Shen is with the School of Cyberspace Security, Beijing Institute of Technology, Beijing 100081, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory (PCL), Shenzhen 518066, China (e-mail: shenmeng@bit.edu.cn).*

TABLE 1: Comparisons of path validation schemes

Scheme	**Method**	**Type**	**Granularity**	**Path privacy protection**	**Proof size**	**Packet.cre**	**Packet.ver**	**Packet.upt**
OSV [16, 17]	orthogonal sequences	deterministic	packet level	✗	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Atomos [18]	asymmetric cryptography	deterministic	packet level	✗	$O(1)$	$O(1)$	$O(1)$	$O(1)$
PPV [19]	symmetric cryptography	probabilistic	flow level	✗	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Hummingbird [20]	symmetric cryptography	probabilistic	packet level	✗	$O(1)$	$O(1)$	$O(1)$	$O(1)$
ICING [21]	symmetric cryptography	deterministic	packet level	✗	$O(n)$	$O(n)$	$O(logn)$	$O(logn)$
OPT [22, 23]	symmetric cryptography	deterministic	packet level	✗	$O(n)$	$O(n)$	$O(1)$	$O(1)$
FSP [24]	symmetric cryptography	deterministic	packet level	✗	$O(n)$	$O(n)$	$O(1)$	–
EPIC [25]	symmetric cryptography	deterministic	packet level	✗	$O(n)$	$O(n)$	$O(1)$	$O(1)$
2PNPV [26]	symmetric cryptography	deterministic	packet level	✓*	$O(n)$	$O(n)$	$O(logn)$	$O(1)$
StealthPath	**symmetric cryptography**	**deterministic**	**packet level**	✓	$\boldsymbol{O(1)}$	$\boldsymbol{O(n)}$	$\boldsymbol{O(1)}$	$\boldsymbol{O(1)}$

Note: n is length of forwarding path; ”✓” means support; ”✗” means not support; ”−”means not mentioned; ”Packet.cre” means that establish a packet in path validation; ”Packet.ver” means that proof verification by nodes on the path in packet delivery phase; ”Packet.upt” means that proof update by nodes on the path in packet delivery phase; ”*” means that it protects path privacy but leaks the length of the forwarding path.

nodes on the path are not protected, which may jeopardize the anonymity of source and destination nodes. Thirdly, obtaining the whole forwarding path is still effortless by only controlling a fraction of the nodes on the path, even if the path information is hidden but the indices of the on-path nodes are not protected. Therefore, the protection of path privacy, including path information and on-path nodes indices, is also indispensable for path validation, which raises a significant question:

How to design a privacy-preserving and efficient path validation scheme with constant proof cost?

It is a challenging task to construct a scheme simultaneously achieving efficient path validation and hiding the path of packet delivery. Firstly, it appears that there is a contradiction between path privacy that requires that the on-path nodes learn nothing about the forwarding path except their neighbors and the path validation which must allow the router nodes to be able to confirm the transfer path. Secondly, both path validation and path hiding require embedding metadata (verification proofs and path hints,which are used for validating path and indicating the next hop) into the packet, which will consume packet space. As a result, the proof size should be as small as possible to transfer as much data in a packet. A way to provide path privacy is adding all proof of on path nodes to the packet in shuffle fashion, allowing the on path node to find the corresponding proof to accomplish the path validation [26]. However, this will incur heavy storage overhead and sacrifice the payload space. In addition, the intermediate node should take as little time as possible to process data packets (that is, verify path compliance and identify the next hop) to improve the transmission efficiency of data packets.

1.1 Contributions

In this paper, we propose StealthPath, a path validation scheme that protects path privacy and makes use of lightweight cryptographic primitives. In the StealthPath, the proofs (including hop proofs and Message Authentication Code (MAC) proofs) are generated using the symmetric encryption and hash function, which allows the nodes on the path to check the path integrity and compliance. The path compliance is guaranteed by verifying the hop proof and MAC proof in a chain manner. That is one invalid hop and MAC proof of a node on the path will cause all subsequent hop and MAC proofs to be invalid since both hop and MAC proofs are computed in a chain manner. To simultaneously achieve path validation and the protection of path privacy, both MAC proofs and the address of the next hop of all nodes on the path are aggregated into a single Proof of Compliance (PoC) by the Chinese Remainder Theorem (CRT) [29]. Then, an on-path node can extract its proof and next hop from the PoC, such that it learns nothing about private information of other nodes on the path and the node index on the path. Our contributions are summarized as follows.

- We present a novel privacy-preserving path validation scheme-StealthPath, which ensures that packet is forwarded following the pre-designed path, protects the path privacy, and improves the efficiency of packet delivery. In the StealthPath, proofs are generated in a chain manner via lightweight symmetric cryptography, which ensures that nodes on the path can efficiently confirm the path compliance of all upstream nodes.
- The StealthPath not only protects the path privacy of nodes on the path but also reduces the proof size from linear to constant, which is achieved through aggregating all MAC proofs and forwarding path into a single PoC. In this way, the intermediate node can extract its proof and the next hop address from the PoC without learning its path index and compromising the path privacy of other nodes.
- Security analysis demonstrates that our StealthPath can achieve the desired security goal. Moreover, compared with state-of-the-art schemes, experimental results show that our StealthPath is efficient and feasible in terms of packet and computation overhead, and saves nearly 60% bandwidth overhead.

This paper is organized as follows. We review the related work in Section 2 and introduce some preliminaries in Section 3. Section 4 presents the system model and security goals of StealthPath. The concrete construction of StealthPath is presented in Section 5 and the security analysis of StealthPath is provided in Section 6. In Section 7, we evaluate the performance of our proposed StealthPath scheme and conclude this work in Section 8.

2 Related Work

Recently, a large number of cryptographic-based path validation schemes have been designed for secure and efficient packet delivery. The path validation can ensure that packet delivery follows the pre-designed path in a verifiable way. That is all downstream nodes can verify whether upstream nodes correctly forward the packets along the pre-designed path. In Table 1, we compare the existing path validation solutions with our proposed StealthPath.

The path validation is initially studied in ICING [21] in 2011, the ICING constructs a validation proof field for each hop node in the packet header to achieve path validation, where the validation proof fields are embedded the proofs computed by symmetric cryptographic techniques (Pseudo-Random Functions (PRFs) and MACs). Then, each intermediate node can verify the proofs generated by upstream nodes and update its proofs in the fields corresponding to its downstream nodes. Each node along the path can thereafter confirm whether a packet was delivered along the pre-designed path. In ICING, each intermediate node's proof size and computation complexity is $O(n)$ on average, where n is the path length. Subsequently, the Origin and Path Trace (OPT) protocols [22, 23] are proposed to shift the corresponding computation to the source node, thus lowering the computation cost of intermediate nodes to $O(1)$ but still with $O(n)$ proof size. OSV [16, 17] attempts to build a more efficient path validation scheme using orthogonal sequences based on Hadamard matrices, which reduces the computations and still maintains $O(n)$ proof size. PPV [19] does not require all intermediate nodes on the path to verify the packet and probabilistically embeds proofs into the packet header to allow a fraction of on-path nodes to validate the packet. PPV reduces both communication and computation overhead while sacrificing some security guarantees. Atomos [18] is presented to design a noncommutative homomorphic asymmetric-key encryption scheme, which reduces the proof size to $O(1)$. However, the asymmetric cryptography makes Atomos expensive, and some security flaws of Atomos are pointed out in [30]. A Flexible Source and Path (FSP) validation scheme is proposed [24] to achieve path validation when the policy of forwarding path changes. The FSP relies on a trusted credible guarantee agent to manage session symmetric keys to validate the packets. The EPIC system is proposed to achieve path authorization, path authentication, and path validation [25], which provides increasingly strong security requirements. It uses PRFs and MACs to achieve path validation but requires $O(n)$ proof size. By the hidden equal-probability sampling technique, Hummingbird [20] is proposed to achieve dynamic path validation.

In summary, the above path validation protocols do not consider protecting path privacy, whereas revealing path privacy could also cause various attacks. 2PNPV [26] is proposed to protect path privacy, which encrypts and shuffles the path information, path index and corresponding proofs. However, 2PNPV incurs $O(n)$ proof size and also leaks the length of the forwarding path to all nodes. Additionally, the nodes on the path require averagely $O(logn)$ complexity to find out their proof since proofs are shuffled. Linear-scale proof size in 2PNPV imposes a high overhead on packet delivery and costs much more network bandwidth, which is inefficient and impracticable. Therefore, this paper focuses on constructing a privacy-preserving path validation scheme for the path-aware network with efficient verification efficiency and constant proof cost.

3 Preliminaries

In this section, the definitions and properties of extended Chebyshev map are first reviewed, and the Chinese Reminder Theorem is next presented. Besides, the following notations are defined: $r \leftarrow_R B$ denotes that r is uniformly and randomly chosen from a set B; $x||y$ denotes the concatenation of bit strings x and y; $b \leftarrow \mathbf{Alg}(a)$ denotes that the output of a randomized algorithm $\mathbf{Alg}$ with input a is b.

3.1 Extended Chebyshev Map

The extended Chebyshev map $T_n(x)$ of degree n is defined as follows, where $x \in (-\infty, +\infty)$ [31].

$$T_n(x) = \begin{cases} 1 & \text{, if n} = 0, \\ x & \text{, if n} = 1, \\ 2xT_{n-1}(x) - T_{n-2}(x) & \text{, if n} \geq 2. \end{cases}$$

Semi-group property. Given an extended Chebyshev map $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \pmod p$, the semi-group property is that for $\forall r, s \geq 2, x \in (-\infty, +\infty)$, $T_s(T_r(x)) \equiv T_{sr}(x) \equiv T_r(T_s(x)) \pmod p$, where p is a large prime.

Definition 1 (Chaotic Map-based Discrete Logarithm (CMDL) Problem). *Given x and y to compute an integer r satisfying $T_r(x) = y$ (mod p).*

Definition 2 (CMDL Assumption). *Suppose that the probability of any Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$ to solve CMDL problem is negligible, i.e., $Pr[\mathcal{A}_{CMDL}(x, y) \rightarrow r : r \in \mathbb{Z}_p^*, y = T_r(x) \text{ (mod } p)] \leq \epsilon$.*

Definition 3 (Chaotic Map-based Decision Diffie-Hellman (CMDDH) Problem). *Given tuple $(x, T_u(x), T_v(x), T_w(x))$ to decide whether $T_w(x) = T_{uv}(x)$ or not, where $u, v, w \in Z_p^*$ are unknown values.*

Definition 4 (CMDDH Assumption). *Suppose that the probability of any PPT adversary $\mathcal{A}$ to solve CMDDH problem is negligible, i.e., $|Pr[\mathcal{A}(x, T_u(x), T_v(x), T_{uv}(x))] - Pr[\mathcal{A}(x, T_u(x), T_v(x), T_z(x))]| \leq \epsilon$.*

Definition 5 (Chaotic Map-based Computational Diffie-Hellman (CMCDH) Problem). *Given tuple $(x, T_r(x), T_s(x))$ to compute $T_{rs}(x)$, where $r, s \in Z_p^*$ are unknown values.*

Definition 6 (CMCDH Assumption). *Suppose that the probability of any PPT adversary $\mathcal{A}$ to solve CMCDH problem is negligible, i.e., $Pr[\mathcal{A}_{CMCDH}(x, T_r(x), T_s(x)) \rightarrow T_{rs}(x) : \forall r, s \in \mathbb{Z}_p^*] \leq \epsilon$.*

3.2 Chinese Reminder Theorem

The Chinese Reminder Theorem is described as follows: for the Equation system (1), if $m_1, \cdots, m_n$ are pairwise coprime numbers, the solution x can be uniquely determined, where $a_1, \cdots, a_n$ can be any positive number. The unique solution

x can be computed by $x = \sum_{i=1}^{n} a_i t_i M_i \bmod M$, where $M = \prod_{i=1}^{n} m_i$, $M_i = \frac{M}{m_i}$ and $t_i = M_i^{-1} \bmod m_i$.

$$\begin{cases} x \equiv a_1 (\bmod\ m_1) \\ x \equiv a_2 (\bmod\ m_2) \\ \vdots \\ x \equiv a_n (\bmod\ m_n) \end{cases} \tag{1}$$

4 System Model and Security Goals

In this section, we present the system model, threat model, definitions, and objectives of our proposed StealthPath.

4.1 System Model

The network consists of the control plane and the data plane, as shown in Fig. 1. The StealthPath focuses on achieving privacy-preservation path validation in the data plane, and consists of four entities: controller lying in the control plane, source node, destination node, and router nodes locating in the data plane, which are described as follows.

- **Controller**. The controller lies in the control plane, and is assumed to be secure. In the StealthPath, the controller is in charge of constructing the network paths and generating keys for nodes in the data plane. The router nodes in the data plane can obtain keys and path information from the controller via a secure channel. The controller may consist of a path server and a key server to help generate forwarding paths and keys. How the controller selects the forwarding path for a source node is independent of the exploration of our StealthPath.
- **Source node** ($\mathcal{S}$). The source node is an entity that wants to send data packets to the destination node in such a way that the data packets traverse a pre-designed network path. Moreover, the forwarding path should be verifiable with path privacy protection, including path information and on-path node index.
- **Destination node** ($\mathcal{D}$). The destination node is the recipient of data packets coming from the source node over a verified and pre-determined network path.
- **Router node**. The router nodes are responsible for (1) transferring data packets along a network path configured by the controller, and (2) confirming the correctness of the forwarding path. The router node follows the longest-prefix match to transfer the data packets in the current Internet network architecture. To achieve path validation, router nodes need to change the forwarding logics of data packets and perform more computations to satisfy the requirements (1) and (2). Concretely, the router node forwards data packets through a pre-defined forwarding path instead of following the longest-prefix match. The router node must be able to check the packet-carried proof of upstream router nodes and update the proof if the check is passed. The above packet processing logic can be updated in the router software [22].

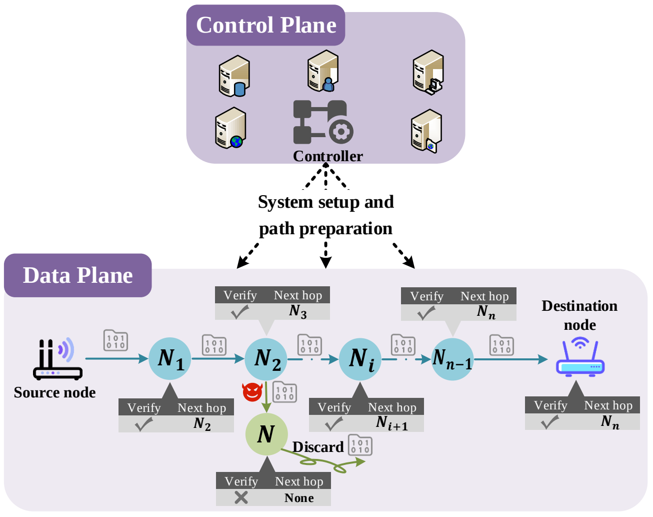


Fig. 1: System model.

4.2 Definitions

The proposed StealthPath consists of five algorithms **Setup**, **KeyGen**, **PathGen**, **PathVerify**, **ProofUpdate**, which are defined as follows.

- SP ←**Setup**(1^λ): on input a security parameter 1^λ, this algorithm outputs a set of system parameter SP.
- (pk, sk) ←**KeyGen**(SP, N): this algorithm takes system parameter SP and node identity N as inputs, and outputs a public-secret key pair (pk, sk).
- $(\mathcal{PVP})$ ←**ProGen**$(SP, path, \{pk_i\}_{i=1}^{n})$: on input system parameter SP, the network path $path$ and public keys of nodes on the path $\{pk_i\}_{i=1}^{n}$, this algorithm outputs the path validation proof $\mathcal{PVP}$.
- $\{0, 1\}$ ←**ProVer**$(SP, \mathcal{PVP}, sk_i)$: on input system parameter SP, the path validation proof $\mathcal{PVP}$ and the secret key sk_i, this algorithm checks the correctness of $\mathcal{PVP}$, and outputs 1 if the verification is passed, 0 otherwise.
- δ_i ←**ProofUpd**(SP, δ_{i-1}, r_i): this algorithm takes the $i-1$-th hop proof δ_{i-1} and session secret key of N_i r_i as inputs, updates the hop proof δ_{i-1} by the session key r_i, and outputs the i-th hop proof δ_i.

4.3 Threat Model

As with existing solutions [16–19, 21–26], we consider an active adversary that succeeds if it can forge a proof to pass path validation and infer the whole forwarding path. Specifically, any router node that does not enforce privacy preserving path validation in Fig. 1 might be a potential adversary. The adversary may attempt to infer the whole forwarding path by launching various attacks, such as path revealing attack, session linking attack, packet modifying attack, and replay attack to identify all addresses and corresponding indices of the nodes on the path. Besides, the following assumptions are required to ensure the security of the proposed StealthPath.

1) While path control that allowing sources to select a forwarding path, it is by itself insufficient to protect

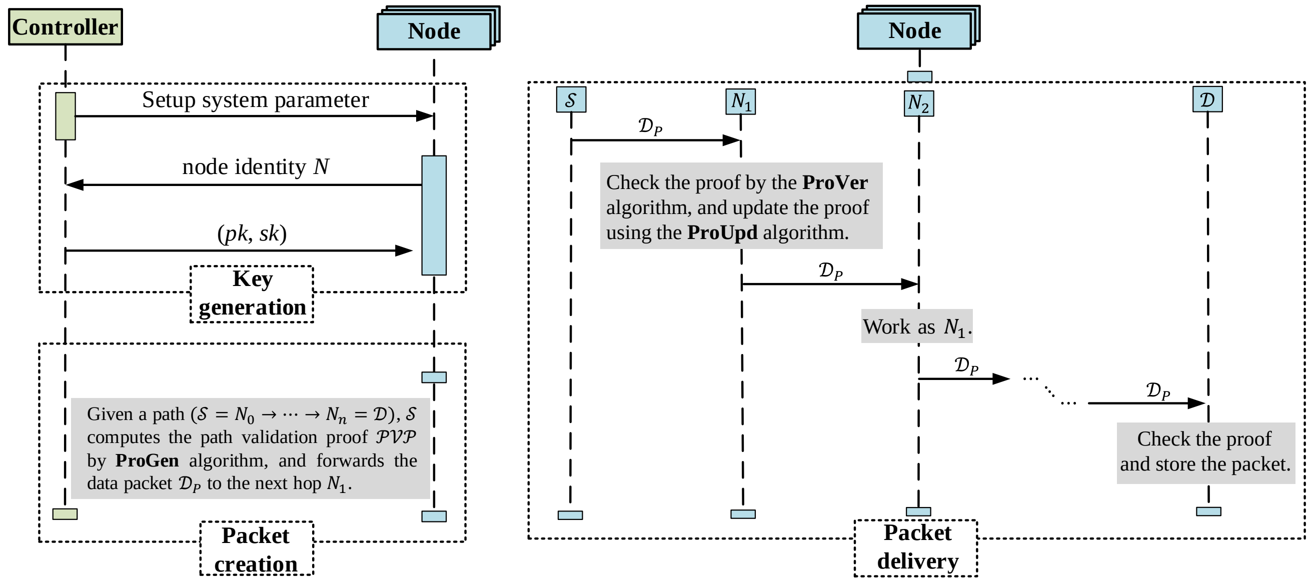

Controller
Node
Setup system parameter
node identity N
(pk, sk)
Key generation
Given a path ($\mathcal{S} = N_0 \to \cdots \to N_n = \mathcal{D}$), $\mathcal{S}$ computes the path validation proof $\mathcal{PVP}$ by **ProGen** algorithm, and forwards the data packet $\mathcal{D}_P$ to the next hop N_1.
Packet creation
Node
$\mathcal{S}$
N_1
N_2
$\mathcal{D}$
$\mathcal{D}_P$
Check the proof by the **ProVer** algorithm, and update the proof using the **ProUpd** algorithm.
$\mathcal{D}_P$
Work as N_1.
$\mathcal{D}_P$
$\mathcal{D}_P$
Check the proof and store the packet.
Packet delivery

Fig. 2: High level overview of StealthPath.

the security and privacy interests of end hosts as it does not provide any guarantees that the directives are actually obeyed. We aim to additionally achieve path validation and protection, i.e., enabling the destination of a packet to verify that the actually traversed path of the packet matches the path intended by the sender, and the intermediate nodes cannot infer the whole forwarding path. The adversary $\mathcal{A}$ does not corrupt the end-hosts, that is, the source node $\mathcal{S}$ and destination node $\mathcal{D}$ are assumed to be honest. Though the corrupted end hosts can easily learn the forwarding path, the forwarding path is various and changes according to different conditions, such as current network traffic, network situation, forward policy, etc [4, 27].

2) The public keys of all nodes in the network are known and can be acquired. In the StealthPath, the controller can obtain the public keys in the key generation phase.

The security model between a challenger $\mathcal{B}$ and a PPT adversary $\mathcal{A}$ is defined as follows.

- **Setup**. $\mathcal{B}$ runs the **KeyGen**(SP, N) to generate keys for all nodes (denote the number as k), that is each node is associated with (sk, pk). Finally, $\mathcal{B}$ sends all public key to $\mathcal{A}$.
- **Query**. In this phase, $\mathcal{A}$ can make polynomial times queries about PoC proof and session keys of forwarding path. When receiving a path $(N_0, \cdots, N_n)$ from $\mathcal{A}$, $\mathcal{B}$ responds $\mathcal{A}$ with session public keys and session secret key of this path.
- **Challenge**. $\mathcal{A}$ runs $\mathcal{A}(\{pk_j\}_{j=0}^n)$ to obtain two different forwarding path $path_0, path_1$ with same length $|path_0| = |path_1| = n + 1$.
- **Forgery**. $\mathcal{B}$ flips a coin $x \in \{0, 1\}$, and computes the PoC proof $\mathcal{P}_x$ for the $path_x$. Otherwise, abort. Given the PoC proof $\mathcal{P}_x$, $\mathcal{A}$ outputs bit x'. If $x' = x$, output 1; otherwise, output 0.

Definition 7 (**Indistinguishability of proof**). *For any PPT adversary, the probability to distinguish the PoC proof is negligible.*

Definition 8 (**Unforgeability of proof**). *Any PPT adversary cannot forge a PoC proof of a forwarding path except knowing the whole forwarding path.*

4.4 Design Objectives

The StealthPath is designed to achieve path validation in a lightweight and privacy-preserving way. The following objectives are designed for the StealthPath scheme.

Objective 1: Lightweight. The path validation must be performed with lower computation and communication costs to accommodate the router nodes with constrained resources.

Objective 2: Path privacy protection. In order to achieve path privacy, the path information and indices of nodes on the path must be protected from all nodes except the neighbors of a node.

Objective 3: Path compliance. All on-path nodes can verify that all upstream nodes have correctly forwarded the packets.

Objective 4: Correctness. The destination node can receive the data packets transferred through the pre-specified path if all on-path nodes honestly forward the data packets following the specification of StealthPath.

Objective 5: Security. The proposed StealthPath scheme should prevent the malicious nodes from cheating in the data forwarding phase, particularly from path revealing attack, session linking attack, packet modifying attack, and replay attack as discussed in Section 4.3.

5 STEALTHPATH

In this section, we first describe the main idea of the StealthPath scheme, and construct the StealthPath in details. Some notations are shown in Table 2.

TABLE 2: Notations

Notation	Description
H_1, H_2, H_{prime}	hash function
$T_x(\cdot)$	chebyshev map
SP	system parameter
(pk, sk)	public/secret key pair
$path$	forwarding path
$path_{\mathcal{E}}$	encrypted forwarding path
$\mathcal{S}, \mathcal{D}$	source/destination node
$\mathcal{PVP}$	path validation proof
$\mathcal{P}$	proof of compliance
s	session master secret key
$T_s(\beta)$	session public key
r	session secret key
δ	hop proof
σ	short digest
MAC	message authentication code
$Time$	timestamp
$D_{\mathcal{P}}$	data packet
$payload$	payload unit of data packet

5.1 Overview

The proposed StealthPath aims to protect the path compliance as well as the privacy of the forwarding paths during packet delivery, and consists of four phases: *system setup, key generation, packet creation*, and *packet delivery*. The workflow of StealthPath is shown in Fig. 2. Specifically, the system setup phase is responsible for establishing the path validation system and generating necessary system parameters. In the key generation phase, the router node registers at the controller and obtains a pair of public and secret keys.

During the packet creation phase, the source node $\mathcal{S}$ initiates the packet delivery along with path validation. $\mathcal{S}$ first obtains a forwarding path from the controller and then generates shared session keys with the nodes on the path. Then, $\mathcal{S}$ generates the path validation proof $\mathcal{PVP}$ consisting session ID, timestamp, session public key, the encrypted path, hop proof and PoC proof, where the PoC proof is computed by aggregating all MAC proofs and address of next hop together. The hop and MAC proofs for each on-path node computed in a chain manner using the symmetric encryption and hash function (e.g., AES and SHA1), and are collaboratively used to verify the path compliance and path integrity. By the way of aggregating fashion, StealthPath not only protects path privacy but also reduces the proof size to the constant, which never has been both achieved in the previous path validation schemes [16–26]. Finally, $\mathcal{S}$ sets up a session to transfer the packet $\mathcal{D}_P$ following the pre-designed path.

In the packet delivery phase, each intermediate node verifies the $\mathcal{PVP}$ to check whether the packets are delivered following the pre-designed path. The chained MAC proofs guarantee that one invalid computation of hop and MAC proof of a node on the path will cause all subsequent hop and MAC proofs to be invalid. If the verifications are passed, the intermediate nodes update the hop proof and continue to forward the packets to the next hop, where the next hop is parsed from $\mathcal{PVP}$. When the destination node $\mathcal{D}$ receives the packet $\mathcal{D}_P$, it verifies the verification and then sends a confirmation to $\mathcal{S}$ if the verifications are passed. Otherwise, $\mathcal{D}$ drops the packet.

5.2 Construction of StealthPath

5.2.1 *System Setup*

Given a security parameter 1^λ, the controller runs the **Setup**(1^λ) to initialize a path validation system. Specifically, a large prime number p is selected, three collusion-resistent hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_2 : \{0,1\}^* \to \{0,1\}^{\frac{l}{2}}$ and $H_{prime} : \{0,1\}^\eta \to \{0,1\}^l$ are defined to compute necessary metadata. The $H_{prime}(u)$ is defined to map u to the next prime after $f(u) = 2(u+2)log_2^{(u+1)^2}$ [32, 33]. The symmetric encryption scheme consisting of three algorithms $\{KeyGen, Enc, Dec\}$ is employed to encrypt message and compute the MAC. Then, a random number $\beta \leftarrow_R \mathbb{Z}_p^*$ is chosen, and a Chebyshev map $T_x(\cdot)$ is selected. Finally, the system parameter SP is set as $SP = \{p, \beta, H_1, H_2, H_{prime}, T_x(\cdot)\}$.

5.2.2 *Key Generation*

In this phase, the router nodes register at the controller, and then the controller computes public and secret key pairs by executing **KeyGen** algorithm. Specifically, for each router node N_i, the controller computes the secret key $sk_i = H_1(N_i||\alpha_i)$ and the public key $pk_i = T_{sk_i}(\beta)\ mod\ p$, where $\alpha_i \leftarrow_R \mathbb{Z}_p^*$.

5.2.3 *Packet Creation*

In this phase, $\mathcal{S}$ selects a forwarding path to communicate with $\mathcal{D}$, where the forwarding path can be obtain from the controller who has a global perspective. Then $\mathcal{S}$ generates the path validation proof $\mathcal{PVP}$ by invoking **ProGen** algorithm, and establishes a session with $\mathcal{D}$ to transfer the data packets with *payload*. Suppose that the path $path = (N_0 \to N_1 \to \cdots \to N_n)$ is the forwarding path, where $\mathcal{S} = N_0$ and $\mathcal{D} = N_n$, and the corresponding public keys of intermediate nodes N_i $(i = 1, \cdots, n-1)$ are known to $\mathcal{S}$. $\mathcal{S}$ generates the $\mathcal{PVP}$ by running the **ProGen** algorithm, which is shown in Algorithm 1 and described as follows.

- Select session master secret key $s \leftarrow_R \mathbb{Z}_p^*$, and compute the session public key $T_s(\beta)$, the session secret key $r_i = T_s(pk_i)$, and $a_i = H_{prime}(N_i||r_i)$ $(i \in [1, n])$.
- Compute the short digest $\sigma = H_2(id||Time||T_s(\beta)||payload)$ and $\delta_0 = Enc_s(\sigma||N_0)$. Then, for each $i \in [1, n]$, compute the hop proof $\delta_i = Enc_{r_i}(\delta_{i-1})$.
- Compute the MAC proof $MAC_i = H_2(\sigma||N_i||N_{i-1}||\delta_{i-1})$ and $b_i = N_{i+1}||MAC_i$ satisfying $b_i < a_i$ $(i \in [1, n])$, where N_{i+1} is the next hop address of N_i and $N_{n+1} = N_n$.

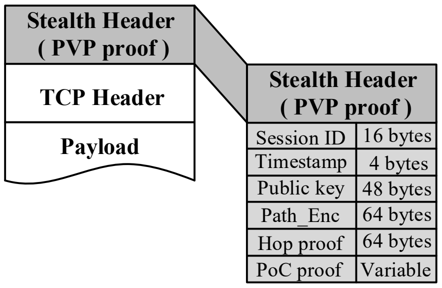


Fig. 3: Header structure of StealthPath.

To protect the privacy of the path information and the indices of nodes on the path, $\mathcal{S}$ constructs a CRT Equation system using $(a_i, b_i)_{i\in[1,n]}$ to hide the path. $\mathcal{S}$ first solves the Equation system (2) to obtain the PoC proof $\mathcal{P}$ by the CRT. Then, $\mathcal{S}$ encrypts the $path$ as $path_{\mathcal{E}} = Enc_{r_n}(n||N_0||\cdots||N_n)$, and computes the session identifier $id = H_2(path_{\mathcal{E}}||Time||T_s(\beta)||\mathcal{P})$, where $Time$ is the current timestamp. When obtaining all ingredients, $\mathcal{S}$ encapsulates the path validation proof $\mathcal{PVP} = id||Time||T_s(\beta)||path_{\mathcal{E}}||\mathcal{P}||\delta_0$ to the stealth header, as shown in Fig. 3. $\mathcal{S}$ can also use the key r_n to encrypt the *payload* to protect the data confidentiality. Finally, $\mathcal{S}$ forwards the data packet $\mathcal{D}_P$ to the next hop N_1.

$$\begin{cases} \mathcal{P} \equiv b_1(\text{mod } a_1) \\ \mathcal{P} \equiv b_2(\text{mod } a_2) \\ \qquad \vdots \\ \mathcal{P} \equiv b_n(\text{mod } a_\text{n}) \end{cases} \tag{2}$$

Algorithm 1 ProGen algorithm

```
Input: system parameter $SP$, $path = (N_0 \to N_1 \to \cdots \to N_n)$, public keys $\{pk_1, \cdots, pk_n\}$ of nodes on the path.
Output: path validation proof $\mathcal{PVP}$.
 1: Select $s \leftarrow_R \mathbb{Z}_p^*$, and compute the session public key $T_s(\beta)$.    ▷ the source node $\mathcal{S}$
 2: for $i = 1$ to $n$ do
 3:     Compute $r_i = T_s(pk_i)$ and $a_i = H_{prime}(N_i||r_i)$.
 4: end for
 5: Compute the short digest $\sigma = H_2(id||Time||T_s(\beta)||payload)$ and $\delta_0 = Enc_s(\sigma||N_0)$.
 6: for $i = 1$ to $n$ do
 7:     Compute the hop proof $\delta_i = Enc_{r_i}(\delta_{i-1})$, the MAC proof $MAC_i = H_2(\sigma||N_i||N_{i-1}||\delta_{i-1})$, and $b_i = N_{i+1}||MAC_i$ satisfying $b_i < a_i$.
 8: end for
 9: Compute the PoC proof $\mathcal{P}$ using $(a_i, b_i)$ $(i = 1, \cdots, n)$, as shown in Equation system (2).
10: Compute the encrypted path $path_{\mathcal{E}} = Enc_{r_n}(n||N_0||\cdots||N_n)$, session identifier $id = H_2(path_{\mathcal{E}}||Time||T_s(\beta)||\mathcal{P})$, where $Time$ is the current timestamp.
11: return the path validation proof $\mathcal{PVP} = id||Time||T_s(\beta)||path_{\mathcal{E}}||\mathcal{P}||\delta_0$.
```

Algorithm 2 ProVer algorithm

```
Input: System parameter $SP$, path validation proof $\mathcal{PVP}$, secret key $sk_i$.    ▷ Perform by node $N_i$ $(i = 1, \cdots, n)$
Output: Path validation proof $\mathcal{PVP}$.
 1: Parse the $\mathcal{PVP}$ as $id||Time||T_s(\beta)||path_{\mathcal{E}}||\mathcal{P}||\delta'_{i-1}$.
 2: if $id \neq H_2(path_{\mathcal{E}}||Time||T_s(\beta))$ then
 3:     return 0
 4: else
 5:     Compute $\sigma = H_2(id||Time||T_s(\beta)||payload)$, $r_i = T_{sk_i}(T_s(\beta))$ and $a_i = H_{prime}(N_i||r_i)$
 6:     Compute $b_i \equiv \mathcal{P}\ mod\ a_i$, and parse $b_i$ as $N_{i+1}||MAC_i$
 7:     if $MAC_i \neq H_2(\sigma||N_i||N'_{i-1}||\delta'_{i-1})$ then
 8:         return 0
 9:     else
10:         return 1
11:     end if
12: end if
```

5.2.4 *Packet Delivery*

In this phase, the data packets are forwarded along the pre-designed path. The intermediate nodes verify the path compliance and update the hop proof, and the destination node also checks the path compliance of all upstream nodes and stores the data packets.

- **Processing at the intermediate node**: when receiving the data packet $\mathcal{D}_P$ from N'_{i-1} $(1 \leq i < n-1)$, N_i performs **ProVer** algorithm to verify the $\mathcal{PVP}$ as follows. N_i first parses the $\mathcal{PVP}$ as $id||Time||T_s(\beta)||path_{\mathcal{E}}||\mathcal{P}||\delta'_{i-1}$ and checks the session identifier by Equation (3). Secondly, N_i computes $\sigma = H_2(id||Time||T_s(\beta)||payload)$, $r_i = T_{sk_i}(T_s(\beta))$ and $a_i = H_{prime}(N_i||r_i)$. Thirdly, N_i computes $b_i \equiv \mathcal{P}\ mod\ a_i$, and parses b_i as the next hop address and the MAC proof $N_{i+1}||MAC_i$. Finally, N_i checks the MAC proof by Equation (4).

$$id \stackrel{?}{=} H_2(path_{\mathcal{E}}||Time||T_s(\beta)||\mathcal{P}) \tag{3}$$

$$MAC_i \stackrel{?}{=} H_2(\sigma||N_i||N'_{i-1}||\delta'_{i-1}) \tag{4}$$

  If any preceding verification fails, **ProVer** algorithm outputs 0, and then N_i drops the data packet. Otherwise, with outputs 1 of **ProVer**, N_i can execute the **ProUpd** to update the hop proof. Concretely, N_i updates the hop proof $\delta_i = Enc_{r_i}(\delta'_{i-1})$, and replaces the hop proof δ'_{i-1} using the δ_i. After that, N_i forwards the data packet to its next hop N_{i+1}, which the address of N_{i+1} is computed from the PoC proof $\mathcal{P}$.

  Note that Equation (4) convinces N_i that all prior on-path nodes have computed the hop proofs $\{\delta_j\}_{j=1}^{i-1}$ correctly, allowing it to complete the verification successfully. The MAC validation can be passed if both N_i and N_{i-1} execute the protocol honestly and correctly. Otherwise, the data packet was not sent over the pre-specified path $path$, on which the next honest node on the path is able to identify. Furthermore, suppose that the node N_i fails to transfer the data packet through the path, then the successor node

N'_{i+1} is unable to determine where the data packet might be sent because resolving a valid next hop from the $\mathcal{P}$ by N'_{i+1} is infeasible without corresponding secret key.

- **Processing at the destination node**: On receiving the data packet $\mathcal{D}_P$ from N'_{n-1}, $\mathcal{D}$ first parses $\mathcal{PVP}$ and then performs **ProVer** algorithm to verify the $\mathcal{PVP}$ as the intermediate node. In addition, it can decrypt $path_{\mathcal{E}}$ using the r_n to obtain forwarding path, and can check $N'_{n-1} \overset{?}{=} N_{n-1}$. When resolving the next hop (and MAC proof) from $\mathcal{P}$, $\mathcal{D}$ will find that the next hop is itself, implying that $\mathcal{D}$ is the destination node. If any check fails, $\mathcal{D}$ discards the data packets. Otherwise, the data packets are stored. Finally, $\mathcal{D}$ sends a confirmation information $Enc_{r_n}(id||Time||T_s(\beta))$ to the source node $\mathcal{S}$. Note that the confirmation does not require reversing the pre-specified path $path$ and does not perform path validation.

Algorithm 3 ProUpd algorithm

```
Input: System parameter SP, hop proof $\delta_{i-1}$ and session
    secret key $r_i$ of $N_i$.
Output: Hop proof $\delta_i$.
1: Compute $\delta_i = Enc_{r_i}(\delta'_{i-1})$.
2: return $\delta_i$
```

6 Security Analysis of StealthPath

In this section, we prove the unforgeability and indistinguishability of PoC proof in the StealthPath, and analyze the security of the proposed StealthPath in terms of path privacy protection, session leakage, security of packet delivery, and node anonymity.

6.1 Security Proof

Theorem 1. *The PoC proofs in the StealthPath scheme is indistinguishable.*

Proof. In the following, we will show that if there exists an adversary $\mathcal{A}$ who can distinguish the PoC proof with a non-negligible probability ε, then we can construct a challenger $\mathcal{B}$ using $\mathcal{A}$ as a sub-routine to solve the CMDDH problem. Given the tuple (β, h_1, h_2, h_3) where $h_1 = T_s(\beta)$, $h_2 = T_r(\beta)$, and h_3 is either $T_{sr}(\beta)$ or $T_z(\beta)$, $\mathcal{B}$ is attempt to determine that where $h_3 = T_{sr}(\beta)$ or $h_3 = T_z(\beta)$. The security game of the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$ proceeds as follows.

- **Setup**. $\mathcal{B}$ runs the **KeyGen**(SP, N) to generate keys for all nodes (assume that the number of nodes is k), that is each node is associated with (sk, pk). Then, $\mathcal{B}$ randomly selects the i-th node N_i, and sets $pk_i = h_1$. Finally, $\mathcal{B}$ sends all public key to $\mathcal{A}$.
- **Query**. In this phase, $\mathcal{A}$ can make polynomial times queries about PoC proof and session keys of forwarding path. When receiving a path $(N_0, \cdots, N_n)$ from $\mathcal{A}$, $\mathcal{B}$ responds $\mathcal{A}$ with $(T_s(\beta), (T_s(pk_1), \cdots, T_s(pk_n)))$, where $s \leftarrow_R \mathbb{Z}_p^*$ is the session master secret key.
- **Challenge**. $\mathcal{A}$ runs $\mathcal{A}(\{pk_j\}_{j=1}^n)$ to obtain two different forwarding path $path_0$ and $path_1$ with the same length $|path_0| = |path_1| = n + 1$.
- **Forgery**. $\mathcal{B}$ flips a coin $x \in \{0, 1\}$ to select the $path_x$, sets the session public key as h_2 and randomly selects session secret keys r_j $(j = 1, \cdots, n)$. If the node N_i is included in the $path_x$, $\mathcal{B}$ substitutes r_j with h_3 and computes the PoC proof $\mathcal{P}_x$ for the $path_x$ as the StealthPath. Otherwise, $\mathcal{B}$ aborts.
  Given the PoC proof $\mathcal{P}_x$ to $\mathcal{A}$, $\mathcal{B}$ obtains an output bit x'. If $x' = x$, output 1; otherwise, output 0.

Let E_1 denote the event that $\mathcal{B}$ does not abort and the chosen path include the node N_i, and E_2 denote the event $\mathcal{A}$ wins the above secure game (i.e., the output is 1). The probability of $\mathcal{B}$ solving the given CMDDH tuple is $\varepsilon' \geq Pr[E_1 \wedge E_2] = Pr[E_1]Pr[E_2|E_1]$. In the above security game, the probability that $\mathcal{B}$ does not abort and the chosen path include the node N_i is $Pr[E_1] = \frac{C_k^n}{2C_k^{n+1}}$, and the probability that $\mathcal{A}$ wins the above secure game is $Pr[E_1] \geq \varepsilon$. Therefore, $\varepsilon' \geq \frac{\varepsilon \cdot C_k^n}{2C_k^{n+1}}$. That is to say, $\mathcal{B}$ can solve the CMDDH problem with probability $\varepsilon' \geq \frac{\varepsilon \cdot C_k^n}{2C_k^{n+1}}$, which contradicts with the CMDDH assumption (Definition 4). Thus, the PoC proofs in the StealthPath scheme is indistinguishable. □

Theorem 2. *Any PPT adversary cannot forge a valid PoC proof $\mathcal{P}$ to pass StealthPath if the adversary cannot compromise the session secret key.*

Proof. StealthPath robustness against the forgeability of $\mathcal{P}$ depends on that the adversary cannot obtain the whole forwarding path. That is any adversary obtaining the forwarding path can easily generate a valid PoC proof. In the following, we will prove that any adversary cannot obtain the forwarding path without compromising the secret key of nodes. For the adversary, it can get the forwarding path if it has the knowledge of the session master secret key s. StealthPath robustness against revealing the knowledge of session master secret key s depends on the difficulty of solving the CMDL problem. The session public key $T_s(\beta)$ is computed by the session master secret key s, and encapsulated into the path validation proof. The adversary may attempt to infer the secret random key s by the public information $(\beta, T_s(\beta))$. However, according to the CMDL assumption, the adversary cannot obtain the session master secret key s since the $(\beta, T_s(\beta))$ is a CMDL tuple and is difficult to break. Thus, our StealthPath protects the knowledge of session master secret key s under the hardness of CMDL assumption. Therefore, any PPT adversary cannot forge a valid PoC proof $\mathcal{P}$ to pass StealthPath. □

6.2 Security Analysis

Path privacy protection. In the StealthPath, the intermediate nodes can only identify their neighbors and do not know other nodes since the path is encrypted. During packet transmission, an honest node computes the address of its successor nodes from the PoC proof by the session secret key, which is computed using the session master secret key. It is infeasible for adversary $\mathcal{A}$ to compute the session secret key of an honest node without the session master secret key. Specifically, the session secret key is generated by extended Chebyshev map in the StealthPath. According to the CMCDH assumption (Definition 6), it is impossible for $\mathcal{A}$ to compute the session secret key. Furthermore, the node index is hidden in the PoC proof $\mathcal{P}$, which is computed by CRT. In summary, $\mathcal{A}$ cannot obtain the path information and node index. Therefore, our proposed StealthPath can resist the path revealing attack and protect path privacy.

In the StealthPath, an intermediate node is capable of discerning the identities of predecessor and successor nodes, as it is responsible for receiving data packets from the predecessor node and transmitting them to the successor node. To infer the whole forwarding path, the adversary without knowledge of on-path nodes indices has to corrupt the minimum number of intermediate nodes such that the number of nodes in between two successive malicious nodes can be at most one, and nodes N_2 and N_{n-1} must be corrupted. Thus, the adversary has to corrupt at least $\lfloor n/2 \rfloor$ nodes. If the indices on path nodes are known, the adversary can infer the whole forwarding path by corrupting the minimum number of intermediate nodes such that every honest node has at least one malicious node as its neighbor. Therefore, the adversary has to corrupt at least $\lfloor n/3 \rfloor$ nodes.

Session linkage. In the StealthPath, the source node selects a random session secret key s to generate path validation proof $\mathcal{PVP}$ for each session. As a result, intermediate nodes on path cannot link packets, even if the packets on two or more forwarding paths of these sessions are generated by the same source node or go to the same destination node. Moreover, each session is created independently from the other sessions because sessions are unrelated to any long-term secret or identifier of the source node. Therefore, two sessions are cryptographically indistinguishable in the StealthPath, implying that two sessions are unlinkable.

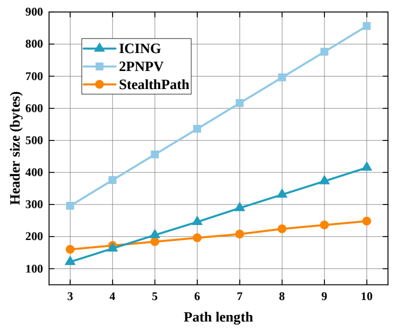
Packet modification. In the packet modification attack, the adversary aims to alter a packet without being detected by the downstream nodes. In the StealthPath scheme, short digest $\sigma = H_2(id||Time||T_s(\beta)||payload)$ is generated to prevent such attacks. Moreover, the short digest is fed as an input to compute the chained MACs. The chained MACs guarantee the integrity of the path and ensure that anyone incorrect MAC computation in the chain invalidates all the subsequent MACs. Thus, our StealthPath thwarts the attacks such as inserting new nodes, splicing two paths, or changing the path order. An intermediate node can check the chained MACs to detect the modification of the packets and path. Then, the intermediate nodes can update the hop proof to help the downstream nodes check the correctness of MACs. Therefore, the StealthPath is protected from packet modification attack.

Replay attacks. The StealthPath achieves freshness and protects the packets from replay attacks by session expiration. In the StealthPath, the combination of unique session master secret key and session public key, PoC proof, and encrypted path serves as a unique packet identifier. The timestamp and session identifier are included in the $\mathcal{PVP}$ and can be validated by all nodes on the path. When observing the packet with an expired timestamp, the intermediate nodes immediately drop the packets. Thus, the StealthPath can prevent replay attacks. The ICING [21] and OPT [22, 23] do not discuss about the resistance to the replay attack. OSV [16, 17] uses the sums of vectors in Hadamard matrix to prevent replay attacks. Sequence numbers and timestamps are not used by PPV [19], which creates a "Packet ID" based on the source, destination, and payload hash. This is insufficient to provide an effective replay-suppression mechanism or to enable packets to be uniquely identified [25]. Similar to our StealthPath, the Atomos [18] and EPIC [25] also employ unique identifiers for replay-suppression.

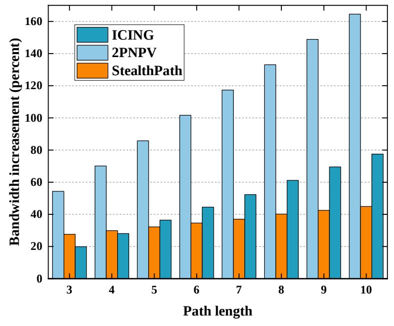
Indistinguishability of end hosts and intermediate nodes. The indistinguishability of end hosts and intermediate nodes indicates that the adversary cannot determine whether an on-path node is the source or destination node when the identity of the on-path node is known to the adversary, as the identity of the on-path node may leak to its predecessor node or successor node during transmitting the data packet. In each session, $\mathcal{S}$ computes the session secret keys for nodes on the path using the random session master secret key s. Moreover, according to the proof generation, the intermediate nodes are unable to determine whether their predecessor node or successor node is $\mathcal{S}$ or $\mathcal{D}$. Therefore, the StealthPath hides source and destination nodes. The OSV [16, 17], Atomos [18], PPV [19], Hummingbird [20], ICING [21], OPT [22, 23], FSP [24], and EPIC [25] do not support the anonymity of source and destination since these schemes do not hide the forwarding path. In 2PNPV [26], path privacy is protected to achieve the anonymity of source and destination nodes.

Limitation. Similar to existing path validation schemes [16-26], StealthPath aims to provide path validation for packet forwarding, but does not address the dynamic path changes and other misbehaviors during packet delivery:

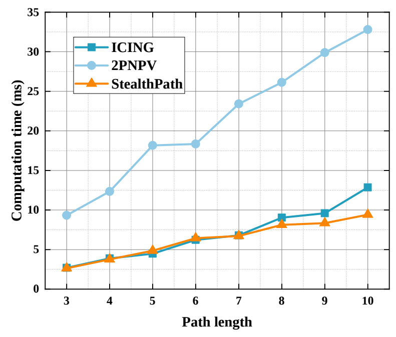
- Dynamic path changes. In the StealthPath technique design, the forwarding path is pre-selected by the source node that needs to compute path validation proof for the selected forwarding path. Then, each on-path node will verify the path validation proof. As the intermediate node cannot modify the path validation proof, our StealthPath does not support dynamic changes.
- Malicious dropping and routing. A malicious node may arbitrarily drop packets to disrupt the packet delivery, which is non-trivial to come up with techniques to address a privacy-preserving path validation scheme like 2PNPV [26] (which incurs linear proof size), and our StealthPath. Appropriate incentive methods [34–36] and trust management of registered nodes [37, 38] may help to mitigate this problem and build a more reliable and efficient privacy-preserving path validation scheme. Moreover, a malicious node is able to relay data packets to the wrong destination by controlling all successor nodes that route to the target destination or acting as a source node to re-route the packet to the wrong destination. However, the destination cannot decrypt the data packet or obtain the whole forwarding path since it cannot compute the valid encryption key.
- Traffic analysis. StealthPath is vulnerable to traffic analysis attack. A global adversary can observe all traffic in the network to gain the forwarding paths.



(a) Comparison of header size.



(b) Comparison of bandwidth overhead.



(c) Comparison of time of packet creation.

Fig. 4: Comparisons of packet overhead and packet creation with state-of-the-art.

Such an attack is hard to circumvent in the path validation schemes.

- Malicious source. A malicious source node may hide the path incorrectly (e.g., insert incorrect next hop to the PoC proof), which makes it possible for the nodes on the path to extract the true address of the next hop, thus disrupting the packet delivery.

7 Performance Evaluation

In this section, we mainly evaluate the performance of ICING [21], 2PNPV [26] and StealthPath. The ICING is the initial study of path validation, but fails to protect path privacy. The 2PNPV focuses on privacy-preserving path validation. For the three schemes, we first evaluate the packet and computation overhead, and then conduct the experiments to show the performance. The experiments are conducted 20 times separately to get the final average result on the desktop with a 1.10 GHz Intel Core i7-10710U CPU and 4 GB memory, where the operating system is Ubuntu 18.04LTS and the programming language is Python.

TABLE 3: Notations

Notations	Descriptions
n	The number of nodes on the path
T_H	Time of computing hash operation
T_{Exp}	Time of computing exponentiation operation
T_{Mul}	Time of computing multiplication operation
T_{Add}	Time of computing addition operation
T_{PRF}	Time of computing pseudo random function operation
T_{XOR}	Time of computing exclusive OR operation
T_{Enc}	Time of encrypting operation of symmetric encryption
T_{Dec}	Time of decrypting operation of symmetric encryption
$T_{\mathcal{T}}$	Time of computing Chebyshev map operation
$T_{\mathcal{CRT}}$	Time of solving a n CRT equation system

7.1 Packet Overhead

Compared to the IP, StealthPath requires a larger packet header (e.g., Stealth Header in Fig. 3) to achieve privacy-preserving path validation. In the following, we will roughly quantify the packet overhead. The stealth header includes 84 bytes that do not depend on the path length: session identifier id of 16 bytes, timestamp $Time$ of 4 bytes, session public key $T_s(\beta)$ of 48 bytes (256 bits security level) and hop proof δ of 16 bytes. Additionally, the length of PoC proof $\mathcal{P}$ and encrypted path $path_{\mathcal{E}}$ depends on the path length (increasing slowly). As shown in Fig. 4(a), the header size with different path lengths is evaluated, and the header size increases with the path length in all three schemes. From a pessimistic estimate of the average provider level in [21] and [39], the length of packet forwarding is often 5. Hence, the header size of ICING, 2PNPV, and StealthPath is 205 bytes, 456 bytes, and 184 bytes. Compared with 2PNPV, StealthPath saves nearly 60% size of the header. Among all three schemes, 2PNPV has the most overhead, and the header size of ICING is larger than StealthPath when the path length is greater than 5.

To measure the increase in bandwidth, we adopt a dataset[1] of real Internet traffic, where the total number of packets observed in 15 minutes on June 6, 2022 was 122, 541, 860 with a total size of 59, 376 MB. The bandwidth increment is shown in Fig. 4(b) with the assumption that path lengths have the same distribution across packet size [21]. Relative to IP, the StealthPath adds $184-20 = 164$ bytes when the path length is 5. Thus, the bandwidth increment for this dataset is $(122, 541, 860 \times 164)/(59, 376 \times 2^{20}) = 32.2\%$. The total increase in bandwidth of ICING and 2PNPV is $(122, 541, 860 \times 185)/(59, 376 \times 2^{20}) = 36.4\%$, $(122, 541, 860 \times 456)/(59, 376 \times 2^{20}) = 85.8\%$ respectively. Compared to 2PNPV, the StealthPath would save nearly 54% bandwidth. The StealthPath has the lowest bandwidth increment among the three schemes.

7.2 Computation Overhead

In this section, we evaluate the computational performance of our proposed StealthPath in comparison to 2PNPV and ICING. Some notations are defined in Table 3.

Theoretical analysis and implementation. We compare the computational cost of our StealthPath with ICING and 2PNPV, and summarize the result in Table 4. In the implementation, the cryptographic primitives are instantiated as follows: AES in CBC mode for symmetric encryption and PRFs, and SHA256 for the hash functions [25, 26].

For the key generation, 2PNPV takes $2T_{Exp}+T_H+T_{Add}+T_{Mul}$ to generate the key for each node, which is higher

1. Packet traces from wide backbone. https://mawi.wide.ad.jp/mawi/samplepoint-F/2022/202206061400.html

TABLE 4: Comparisons of Computational Cost

scheme	Key generation		Packet creation		Packet delivery	
	Anal.	Impl.	Anal.	Impl. ($n=5$)	Anal.	Impl. ($n=10$)
ICING[21]	T_{Exp}	1.11 ms	$(n-1)(3T_{PRF}+T_{XOR}+T_{Exp})$ $+nT_H$	4.48 ms	$(n+2)T_{PRF}+nT_H$ $+(n-1)(T_{Exp}+T_{XOR})$	10.04 ms
2PNPV[26]	$2T_{Exp}+T_H+$ $T_{Add}+T_{Mul}$	3.39 ms	$(4n-1)T_{Enc}+3(n-1)T_{Exp}$ $+nT_H+(n-1)T_{Mul}$	18.16 ms	$(log_2^n+3)T_H+log_2^nT_{Dec}$ $+3T_{Enc}+2T_{Exp}$	2.65 ms
StealthPath	$T_H+T_{\mathcal{T}}$	1.06 ms	$2nT_H+(n+1)T_{Enc}+$ $(n-1)T_{\mathcal{T}}+T_{\mathcal{CRT}}$	4.85 ms	$4T_H+T_{Enc}+T_{\mathcal{T}}$	0.09 ms

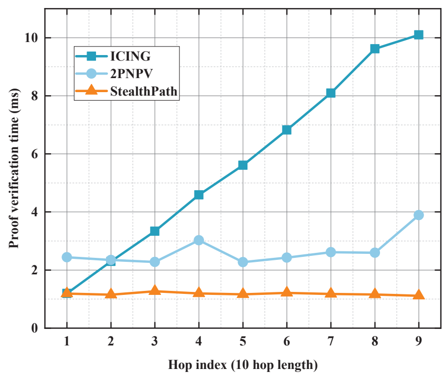


Fig. 5: Comparison of proof verification time.

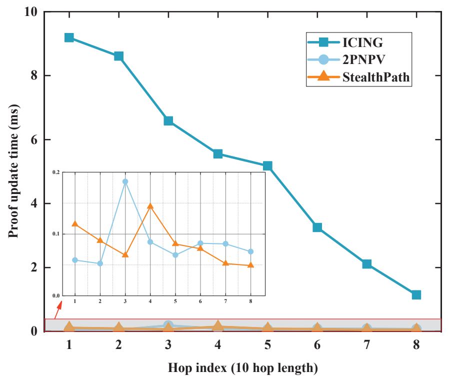


Fig. 6: Comparison of proof update time.

than ICING and StealthPath. Our StealthPath only requires $T_H+T_{\mathcal{T}}$, and has the lowest among the three schemes. In the implementation, ICING, 2PNPV, and StealthPath averagely takes 1.11 ms, 3.39 ms, and 1.06 ms to generate a key for a node, respectively.

In the packet creation phase, all three schemes generate path validation proof $\mathcal{PVP}$ for path validation and are sensitive to path length. It can be seen from Table 4 that the computation cost of all three schemes is in the same order of magnitude and increases linearly with path length. As shown in Fig. 4(c), the average packet creation time with path length 5 is 4.48 ms, 18.16 ms and 4.85 ms of ICING, 2PNPV and StealthPath, respectively. Among these three schemes, StealthPath has the lowest computation time and also ensures path privacy.

In the packet delivery phase, each on-path node verifies the PoC proof and updates the hop proof. In the ICING, each node verifies the proofs of upstream nodes and updates the proofs of downstream nodes. Thus, the time of proof verification increases with the hop index, while the time of proof update decreases with the hop index. As shown in Table 4, Fig. 5 and Fig. 6, the total computation cost of proof verification and update is $(n+2)T_{PRF}+nT_H+(n-1)(T_{Exp}+T_{XOR})$, and the average total time is 10.40 ms for per node with the path length 10. In the 2PNPV, each node has to find its own PoC proof from all PoC proofs one by one, and cost $(log_2^n+3)T_H+log_2^nT_{Dec}+3T_{Enc}+2T_{Exp}$. In the implementation, the average time to verify and update proofs is 2.65 ms and 0.09 ms, respectively. As for our StealthPath, the bulk of path validation work is completed by the source node in packet creation phase, while the path verification and update have constant size/complexity, independent of path length. The average time of verifying proofs is 1.18 ms, and that of updating proofs is 0.084 ms, as shown in Fig. 5 and Fig. 6. Therefore, only our StealthPath has a constant overhead in the packet delivery phase among the three schemes, which shows that our StealthPath is efficient and practical.

Incentives discussion of Deployment. Beyond the setup of the technical StealthPath components, adoption of StealthPath would require coordination between participating Internet services providers (ISPs). In the following, we discuss of efficiency and business incentives for adopting StealthPath.

Efficiency. Our StealthPath uses lightweight cryptographic primitives, which makes it very attractive in practice. These cryptographic primitives can be implemented on hardware, making the StealthPath more efficient. The IP routers use the longest-prefix matching for packet delivery, which requires large amounts of expensive ternary content addressable memory (TCAM). Our StealthPath can be implemented by very little additional hardware. As claimed in [25], hardware implementations of AES are very efficient and only require 13,000 gates while implementations of IP routers requires 8.7 million gates even for very small amounts of TCAM. Thus, these cryptographic computations could be offloaded to multiple dedicated hardware units in network-interface cards to further accelerate the path validation in the future. The lightweight implementation of StealthPath will also help it gain early attention and adoption.

Business incentives. The StealthPath has provided privacy-preserving path control for end hosts, but its lack of direct compatibility with the current Internet may lead to adoption resistance. Nevertheless, the current market trends demonstrate demand for reliable and secure Internet connectivity [15], which could incentivize ISPs to upgrade their router nodes to support StealthPath services. As claimed in [27][2], pre-selected forwarding paths by performance metrics can be translated to better quality of service (such as audio, video, and file transfers) and generally shorter transfer delays. Thus, the ISPs are able to define new business models and sell new services based on the benefits of StealthPath such as path control, path privacy-preserving, and shorter transfer delays. Based on StealthPath, ISPs can create services for customers who demand higher availability than the current Internet can provide, but who cannot afford dedicated leased lines. Furthermore, blockchain can be used to achieve automated incentives for ISPs according to loaded traffic and other critical metrics [34]. However, in this paper, we mainly focus on designing privacy-preserving path control for end hosts, the existing incentive mechanisms [34–36] for routing are also compatible with path control services provided by StealthPath, and other work [16–26].

8 Conclusion

This paper presented StealthPath to simultaneously achieve path validation and path privacy protection. Using symmetric cryptography, the StealthPath generates chained MAC proofs to validate the network path and achieve efficient verification. Based on CRT, the StealthPath reduces proof size of PoC to from linear to the constant and hides the path indices and path length. Additionally, each intermediate node can extract proof and corresponding next hop address from the PoC proof without compromising path privacy. The security analysis showed that StealthPath is secure against various attacks, and sufficient experiment results show that our proposed StealthPath saves nearly 60% header size, and efficiently transfers data in a privacy-preserving way. In future work, we will delve into incentivizing mechanisms and trust management for registered nodes to effectively mitigate malicious dropping attacks, and designing dynamic path changes to achieve efficient and fault-tolerant packet delivery.

References

[1] M. Ambrosin, A. Compagno, M. Conti, C. Ghali, and G. Tsudik, "Security and privacy analysis of national science foundation future internet architectures," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1418–1442, 2018.

[2] M. Shen, Z. Gao, L. Zhu, and K. Xu, "Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021, pp. 1–10.

[3] Y. Su, Y. Li, B. Yang, and Y. Ding, "Decentralized self-auditing scheme with errors localization for multi-cloud storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2838–2850, 2022.

2. SCION is a path validation protocol that does not provide privacy-preserving path protection and has been practical deployed all over the world. Independent testing of round-trip time was conducted from Anapaya's Swiss headquarters to Amazon Web Services' Paris server via both regular internet connection and SCION [40], the SCION's path-aware networking was measured at 12.7 ms - with a 20-25% lower latency and faster speed than the internet today.

[4] C. Krähenbühl, M. Wyss, D. Basin, V. Lenders, A. Perrig, and M. Strohmeier, "FABRID: Flexible Attestation-Based routing for Inter-Domain networks," in *Proceedings of the 32nd USENIX Security Symposium (USENIX)*, 2023, pp. 5755–5772.

[5] C. Kuhn, D. Hofheinz, A. Rupp, and T. Strufe, "Onion routing with replies," in *Proceedings of the Advances in Cryptology (ASIACRYPT)*, 2021, pp. 573–604.

[6] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "RAPTOR: Routing attacks on privacy in tor," in *Proceedings of the 24th USENIX Security Symposium (USENIX)*, 2015, pp. 271–286.

[7] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, "Bamboozling certificate authorities with BGP," in *Proceedings of the 27th USENIX Security Symposium (USENIX)*, 2018, pp. 833–849.

[8] M. Shen, Y. Liu, L. Zhu, K. Xu, X. Du, and N. Guizani, "Optimizing feature selection for efficient encrypted traffic classification: A systematic approach," *IEEE Network*, vol. 34, no. 4, pp. 20–27, 2020.

[9] Y. Su, J. Li, J. Li, Z. Su, W. Meng, H. Yin, and R. Lu, "Robust and lightweight data aggregation with histogram estimation in edge-cloud systems," *IEEE Transactions on Network Science and Engineering*, pp. 1–12, 2024.

[10] X. Qi, J. Li, Z. Wang, and L. Liu, "Probabilistic probe selection algorithm for fault diagnosis in communication networks," *Computer Networks*, vol. 198, p. 108365, 2021.

[11] M. Shen, Y. Liu, L. Zhu, X. Du, and J. Hu, "Fine-grained webpage fingerprinting using only packet length information of encrypted traffic," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2046–2059, 2021.

[12] J. Li, X. Qi, W. Ma, and L. Liu, "Path selection for link failure protection in hybrid SDNs," *Future Generation Computer Systems*, vol. 137, pp. 201–215, 2022.

[13] IRTF, "Path Aware Networking Research Group (PANRG)," https://irtf.org/panrg, 2019.

[14] IETF, "Path Aware Networking RG (panrg)," https://datatracker.ietf.org/rg/panrg/ about/, 2019.

[15] H. Birge-Lee, J. Wanner, G. H. Cimaszewski, J. Kwon, L. Wang, F. Wirz, P. Mittal, A. Perrig, and Y. Sun, "Creating a secure underlay for the internet," in *Proceedings of the 31st USENIX Security Symposium (USENIX)*, 2022, pp. 2601–2618.

[16] H. Cai and T. Wolf, "Source authentication and path validation with orthogonal network capabilities," in *Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2015, pp. 111–112.

[17] H. Cai and T. Wolf, "Source authentication and path validation in networks using orthogonal sequences," in *Proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN)*, 2016, pp. 1–10.

[18] A. He, K. Bu, Y. Li, E. Chida, Q. Gu, and K. Ren, "Atomos: Constant-size path validation proof," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3832–3847, 2020.

[19] B. Wu, K. Xu, Q. Li, Z. Liu, Y.-C. Hu, M. J. Reed, M. Shen, and F. Yang, "Enabling efficient source and

path verification via probabilistic packet marking," in *Proceedings of the 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–10.

[20] A. He, X. Li, J. Fu, H. Hu, K. Bu, C. Miao, and K. Ren, "Hummingbird: Dynamic path validation with hidden equal-probability sampling," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1268–1282, 2023.

[21] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra, "Verifying and enforcing network paths with ICING," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies (CoNEXT)*, 2011.

[22] T. H.-J. Kim, C. Basescu, L. Jia, S. B. Lee, Y.-C. Hu, and A. Perrig, "Lightweight source authentication and path validation," in *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM)*, 2014, pp. 271–282.

[23] F. Zhang, L. Jia, C. Basescu, T. H.-J. Kim, Y.-C. Hu, and A. Perrig, "Mechanized network origin and path authenticity proofs," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 346–357.

[24] F. Yang, K. Xu, Q. Li, R. Lu, B. Wu, T. Zhang, Y. Zhao, and M. Shen, "I know if the journey changes: Flexible source and path validation," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–6.

[25] M. Legner, T. Klenze, M. Wyss, C. Sprenger, and A. Perrig, "EPIC: Every packet is checked in the data plane of a Path-Aware internet," in *Proceedings of the 29th USENIX Security Symposium (USENIX)*, 2020, pp. 541–558.

[26] B. Sengupta, Y. Li, K. Bu, and R. H. Deng, "Privacy-preserving network path validation," *ACM Transactions on Internet Technology*, vol. 20, no. 1, 2020.

[27] L. Chuat, M. Legner, D. Basin, D. Hausheer, S. Hitz, P. Müller, and A. Perrig, *The Complete Guide to SCION*, 2022.

[28] Y. Shen, T. N. Dinh, and M. T. Thai, "Adaptive algorithms for detecting critical links and nodes in dynamic networks," in *Proceedings of the 2012 IEEE Military Communications Conference (MILCOM)*, 2012, pp. 1–6.

[29] D. Pei, A. Salomaa, and C. Ding, *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific, 1996.

[30] Y. Wu, C. Jiang, C. Xu, and K. Chen, "Security analysis of a path validation scheme with constant-size proof," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4246–4248, 2021.

[31] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1338–1351, 2022.

[32] D. Boneh, B. Bünz, and B. Fisch, "Batching techniques for accumulators with applications to iops and stateless blockchains," in *Advances in Cryptology (CRYPTO)*, 2019, pp. 561–586.

[33] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: Efficient, succinct, modular," in *Financial Cryptography and Data Security (FC)*, 2021, pp. 393–414.

[34] C. Machado and C. M. Westphall, "Blockchain incentivized data forwarding in manets: Strategies and challenges," *Ad Hoc Networks*, vol. 110, p. 102321, 2021.

[35] Y. Xu, J. Liu, Y. Shen, J. Liu, X. Jiang, and T. Taleb, "Incentive jamming-based secure routing in decentralized internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 3000–3013, 2021.

[36] C. Ihle, D. Trautwein, M. Schubotz, N. Meuschke, and B. Gipp, "Incentive mechanisms in peer-to-peer networks –a systematic literature review," *ACM Computing Surveys*, vol. 55, no. 14s, 2023.

[37] J.-H. Cho, A. Swami, and I.-R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 562–583, 2011.

[38] Y. Liu, M. Dong, K. Ota, and A. Liu, "Activetrust: Secure and trustable routing in wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2013–2027, 2016.

[39] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable internet protocol (AIP)," in *Proceedings of the 2008 ACM Conference on SIGCOMM (SIGCOMM)*, 2008, pp. 339–350.

[40] L. Bischofberger, "Scion up to 25% faster than today's internet," https://www.anapaya.net/blog/scion-faster-than-todays-internet, 2022.

Jiliang Li received the Dr. rer. nat. degree in computer science from the University of Göettingen, Göettingen, Germany. He is currently a Researcher Professor and PhD Supervisor with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China. His research interests include information security, cryptography, blockchain and IoT security.

Yuan Su received the master's degree from School of Mathematics and Statistics, Shaanxi Normal University, Xi'an, China, in 2021. He is working toward the Ph.D. degree with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China. His research interests include network security and cryptography.

Rongxing Lu (Fellow, IEEE) is Mastercard IoT Research Chair, a University Research Scholar, an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious ”Governor General’s Gold Medal”, when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CISTC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Zhou Su received the Ph.D degree from Waseda University, Tokyo, Japan, in 2003. Prof. Su received the best paper award of IEEE International Conference on Communications 2020, IEEE International Conference on Big Data 2019, IEEE Cyber Science and Technology Congress 2017, and Conference on Wireless Internet 2016. He is an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL, and IEEE Open Journal of Computer Society. He is the Chair of the Multimedia Services and Applications over Emerging Networks Interest Group of the IEEE Comsoc Society, the Multimedia Communications Technical Committee.

Weizhi Meng is currently an Associate Professor in the Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Kongens Lyngby, Denmark. He obtained his PhD degree in Computer Science from the City University of Hong Kong (CityU), Hong Kong. His primary research interests are cyber security and intelligent technology in security, including intrusion detection, smartphone security, biometric authentication, HCI security, malware detection, blockchain in security, cyber-physical security, and IoT security.

Meng Shen is a Professor at Beijing Institute of Technology, Beijing, China. He received the B.Eng degree from Shandong University, Jinan, China in 2009, and the Ph.D. degree from Tsinghua University, Beijing, China in 2014, both in computer science. His research interests include data privacy and security, blockchain applications, and encrypted traffic classification. He has authored over 50 papers in top-level journals and conferences, such as ACM SIGCOMM, IEEE JSAC, and IEEE TIFS. He has guest edited special issues on emerging technologies for data security and privacy in IEEE Network and IEEE Internet-of-Things Journal. He received the Best Paper Runner-Up Award at IEEE IPCCC 2014 and IEEE/ACM IWQoS 2020. Dr. Shen was selected by the Beijing Nova Program 2020 and was the winner of the ACM SIGCOMM China Rising Star Award 2019. He is a member of the IEEE.