P^3V: Privacy-Preserving Path Validation System for Multi-Authority Sliced Networks

Weizhao Jin
weizhaoj@usc.edu
Information Sciences Institute
University of Southern California

Erik Kline
kline@isi.edu
Information Sciences Institute
University of Southern California

T. K. Satish Kumar
tkskwork@gmail.com
Information Sciences Institute
University of Southern California

Lincoln Thurlow
lincoln@isi.edu
Information Sciences Institute
University of Southern California

Srivatsan Ravi
sravi@isi.edu
Information Sciences Institute
University of Southern California

Abstract

In practical operational networks, it is essential to validate path integrity, especially when untrusted intermediate nodes are from numerous network infrastructures operated by several network authorities. Current solutions often reveal the entire path to all parties involved, which may potentially expose the network structures to malicious intermediate attackers. Additionally, there is no prior work done to provide a systematic approach combining the complete lifecycle of packet delivery, i.e., path slicing, path validation and path rerouting, leaving these highly-intertwined modules completely separated. In this work, we present a decentralized privacy-preserving path validation system P^3V that integrates our novel path validation protocol with an efficient path slicing algorithm and a malice-resilient path rerouting mechanism. Specifically, leveraging Non-Interactive Zero-Knowledge proofs, our path validation protocol XOR-Hash-NIZK protects the packet delivery tasks against information leakage about multi-hop paths and potentially the underlying network infrastructures. We implemented and evaluated our system on a state-of-the-art 5G Dispersed Computing Testbed simulating a multi-authority network. Our results show that while preserving the privacy of paths and nodes and enhancing the security of network service, our system optimizes the performance trade-off between network service quality and security/privacy.

1 Introduction

To satisfy the various requirements for Quality of Service (QoS) from users, network slicing in 5G networks [38] is proposed to allocate utilizable resources across different Internet infrastructures. The most common application is to provide dynamic and intelligent cross-party network paths for content delivery with different service levels specified by users. In an ideal world where trust can be universally placed, network slicing solutions with proper consensus mechanisms alone are sound enough to fulfill dynamic network services requested by various use cases. However, in a mutually-untrusted collaborative operation network environment at large, it is catastrophic to blindly assume that packets will follow the advertised cross-party multi-hop paths especially through a multi-authority network without a path validation solution in place. The void of sufficient path validation posed an obvious threat to service quality [4, 19, 22]. Therefore, to deliver the promised QoS of network services and enforce the correct order of network operations, designing and implementing path validation solutions that adapt to the evolving network technology standards is a constant challenge for corresponding network operators. One research direction is to use PKI-based signatures with third-party authorities (BGPSec [25] introduced 23 years after the BGP adoption) to validate network paths, which unfortunately fails to scale with multiple independent network operators. Current scalable path validation solutions [5–7, 22, 28] usually inform all nodes en route with information about the path (sometimes even the entire path), which exposes to all nodes not only the path but also potentially the underlying network structure. With rare exceptions, information about substrate network infrastructures is generally regarded as valuable yet sensitive corporate assets which the network operators will be reluctant to share openly and fully. More importantly, such critical exposure enables an adversary to easily observe and correlate anonymous or private network traffics [9, 12, 21, 29], launch accurate targeted attacks on specific network bottlenecks when vulnerable network structures are pinpointed [2, 13, 14, 33], and intrude sensitive data centers where critical data is aggregated [1, 30, 32, 40].

Another challenge is to construct a systematic network path solution that integrates different lifecycle

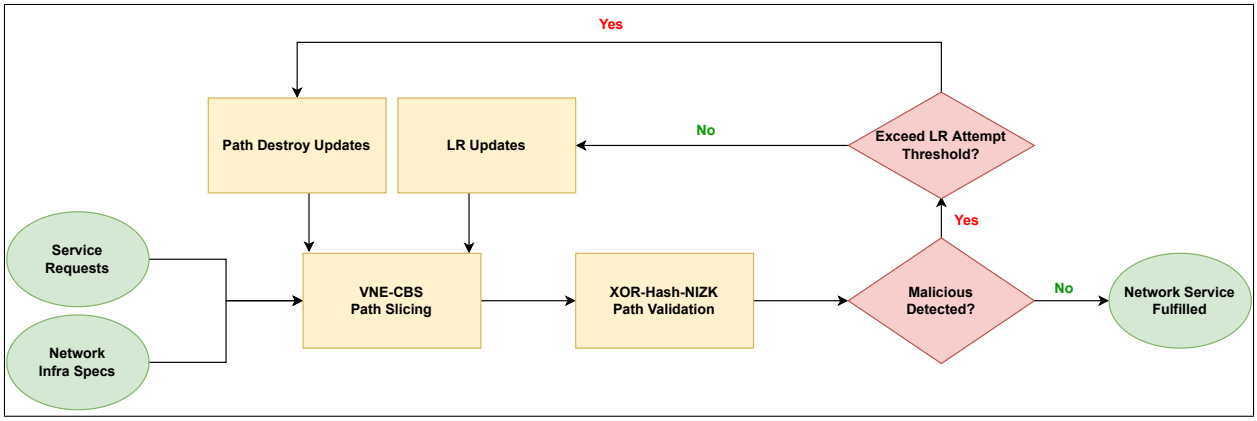


Figure 1. The 3-Stage Modular System Structure of P^3V: in the **Path Slicing** module, the slicing oracle takes in service requests from users and network infrastructure specifications from infrastructure providers, then runs the VNE-CBS algorithm to generate a viable path; in the **Path Validation** module, nodes en route execute the XOR-Hash NIZK protocol to validate the path while forwarding packets; in the **Path Rerouting** module, if any malicious behavior is detected on the path, the system initializes the malice-resilient rerouting procedure by rerunning the first modules with updated substrate networks.

stages of the packet forwarding task, namely path slicing, path validation and path rerouting. However, to the best of our knowledge, there is no prior work that provides a modular system that thoroughly builds linkages between each path-related module within the framework. Previous work focused heavily on designing solutions for each individual component for each specific sub-tasks [5–7, 10, 11, 15, 16, 22, 28, 35, 39, 42]. Simply assembling these sub-solutions fail to realize a functioning network path resolution system in practice. For example, most existing path rerouting [16, 35, 39] cannot be applied to mitigate the malicious intermediate node behaviors, especially in the context of the multi-authority 5G environment we defined above, because these rerouting solutions simply ignore the mechanisms of path slicing utilized in the first stage, making the path recovery pragmatically impossible under their schemes.

This paper proposes a decentralized privacy-preserving path validation system, P^3V, that provides a systemic and modular network slicing and validation solution for multi-authority networks.

Contributions

1. We propose a systematic approach to the path validation problem in a secure and privacy-preserving fashion integrating modules like path slicing, path validation and path rerouting in the system design, thoroughly covering different stages of the packet forwarding lifecycle (shown in Figure 1) while performing path validation without revealing neither the overall path information nor node identities to the intermediate nodes beyond their predecessor and successor in the path.
2. We present the design of a decentralized privacy-preserving path validation protocol XOR-Hash-NIZK using Non-Interactive Zero-Knowledge (NIZK) proofs, which provides path/node privacy guarantees. The NIZK-based pairwise validation design also optimizes the network performance and mitigates impacts on end-user service quality.
3. We implemented and evaluated our path validation system both in a simulated 5G environment and on a state-of-the-art 5G Dispersed Computing Testbed, where the results substantiate our optimized trade-off between security, privacy and performance.

Paper roadmap The rest of the paper is structured as follows: in §2 and §3 we provide an overview of the background knowledge needed and related work; in §4 we define the path validation problem; in §5 and §6 we introduce our privacy-preserving path validation protocol XOR-Hash-NIZK and the malicious rerouting design respectively; in §7 and §8, we construct the system and evaluate our system both in a simulated multi-node 5G network environment and on a large-scale 5G Dispersed Computing Testbed.

2 Background

2.1 Network Slicing

Network slicing is used in 5G networks to efficiently fit dynamic needs of different Service Level Requirements (SLR) by slicing and configuring the actual monolithic network infrastructures into multiple logical parts across parties [41]. For example, data transmission for first responder service requires low latency and low loss but low bandwidth whereas home media storage devices require high bandwidth but do not prioritize loss or latency. With proper network slicing in place, such various types of services can be efficiently satisfied.

Per Service Level Agreements (SLA), the slicing authorities will provide a suitable network path from the physical infrastructures for various requests.

Virtual Network Embedding The Virtual Network Embedding (VNE) problem is the combinatorial problem of embedding Virtual Network Requests (VNRs) into a Substrate Network (SN) while satisfying constraints such as bandwidth capacities on the SN edges, CPU capacities on the SN vertices, and geographical constraints on the VNR vertices. The embedding maps each VNR vertex to a unique SN vertex and each VNR edge to a path in the SN between the corresponding vertices. A solution to the VNE problem is essentially a set of multi-hop paths cognizant of network resources.

VNE-CBS The VNE-CBS algorithm [42] is an efficient algorithm for the VNE problem that minimizes the amount of physical network resources spent on embedding VNRs. It is a bi-level heuristic search algorithm that carries out searching in the conflict-resolution space. As shown in Figure 2, the high-level search detects violations of constraints and resolves them by branching. During the high-level searching, the low-level search performs path planning for VNR vertices under the restrictions imposed by the high-level search.

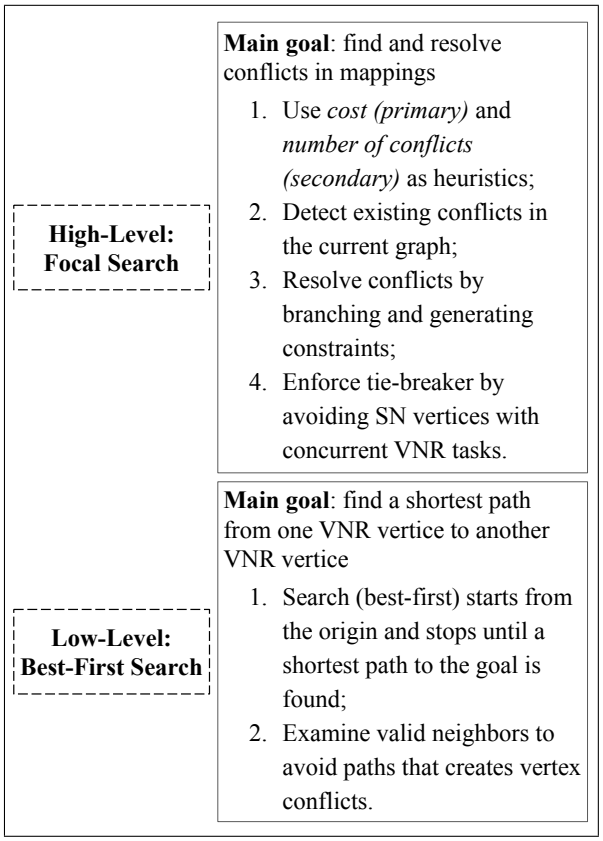


Figure 2. VNE-CBS Algorithm (Bi-Level Heuristic Search): the high-level search detects and resolves conflicts in mappings; the low-level search finds the shortest path (constrained by the high-level search) for a VNR vertex to its goal.

Compared to other proposed solutions for the VNE problem [10, 11, 15], the advantages of the VNE-CBS algorithm include: (a) the ability to incorporate other security-related constraints in the high-level and low-level searches; (b) the ability to exploit independence and loose interactions between the paths; (c) the ability to incorporate advances of heuristic search methods into the high-level and low-level searches; (d) the ability to utilize a small leeway in the optimality of the solution towards a significant improvement in the running time; and (e) the completeness and optimality properties.

Another potential advantage of the VNE-CBS framework is that the high-level conflict-resolution search tree can be stored and reused. Thus, when a new constraint is added, the incremental computation can be enabled to update the solution. Such a framework is beneficial when certain segments of multi-hop paths cannot be validated, requiring the paths to be updated under the added restrictions. For the above reasons, we choose the VNE-CBS algorithm as the underlying solution for path slicing module in our system.

2.2 Path Validation

Network path validation [4] is used to enforce and verify cross-party multi-hop paths agreed to satisfy certain service requirements. Deviating from agreed paths not only downgrades network service quality but also disrupts network orchestration at large. The general objectives of path validation can be defined as follows [22]:

- **Enforcement** Path validation enforces a packet is forwarded on the agreed path over each node en route in the correct order.
- **Verification** The sender node, intermediate nodes or the receiver node is able to verify that the packet forwarding follows the correct path.

It is worth mentioning that the verification objective enhances the enforcement objective in a sense that to successfully pass path verification, each node tends to follow the protocol thus enforcing the path.

General procedure of path validation can be described as that, after correctly executing its portion of a given packet forwarding task, a node en route is required to prove that it indeed follows certain paths/protocols to either its neighboring nodes or some third-party trusted authorities/APIs.

2.3 Non-Interactive Zero-Knowledge

Non-Interactive Zero-Knowledge proofs [3, 27] are zero-knowledge proofs that does not require excessive interaction (only one exchange) between a prover $\mathcal{P}$ to a verifier $\mathcal{V}$. During the process, $\mathcal{P}$ computes a proof π to convince $\mathcal{V}$ that a statement $x \in \mathcal{L}$ is true. $\mathcal{V}$ verifies π then decides to either accept or reject. NIZK proofs have certain properties:

- **Completeness** The $\mathcal{V}$ always accepts correct π for a statement $x \in \mathcal{L}$.

Protocol	Transparent Validation [5–7, 22, 28]	Semi-PP Validation [34]	XOR-HASH-NIZK
Privacy Guarantee	All nodes en route learn the info about the whole path	Intermediate nodes learn some info about the path such as number of nodes en route	Each intermediate node only learns its neighboring nodes
Integration with Network Slicing	No	No	Yes
Malicious Rerouting Resolution	No	No	Yes
Cryptographic Overhead	Lightweight Encryption/Decryption	Overhead mainly from Signature Signing and Encryption/Decryption	Overhead mainly from NIZK

Table 1. Comparison of Different Path Validation Protocols: we refer to path validation protocols without much privacy-preserving design as Transparent Validation and path validation solutions with some privacy-preserving design as Semi-PP (privacy-preserving) Validation. As stated in §1, protocols from both categories are stand-alone path validation solutions without integration with network slicing algorithms and path rerouting approaches, which are essential and challenging in practice. Protocol overheads listed are qualitative evaluations based on cryptographic primitives utilized.

- **Soundness** $\mathcal{V}$ always rejects any π for all $x \notin \mathcal{L}$, except with negligible probability.
- **Zero-knowledge** A NIZK proof should guarantee that $\mathcal{V}$ learns nothing from π beyond the fact that $x \in \mathcal{L}$.

In our protocol, each party obtains the validation token from its successor after finishing its promised task and uses NIZK to prove to its predecessor that it correctly delivers the packet with a statement that this party possesses the token. Note that this NIZK proof can be verified by any related party in the system. To be more specific, the authorities can also verify the proof for malicious node detection (will be discussed in detail in §5). The Algorithms 1 and 2 demonstrate the general process of NIZK with hashing.

Algorithm 1 Prover

1: $secret \leftarrow random$
2: $provK \leftarrow Generator.KeyGen()$
3: Primary input $pI :=$ SHA256$(secret)$
4: Auxiliary input $aI := secret$
5: $pf := PfGen(provK, pI, aI)$

Algorithm 2 Verifier

1: $veriK \leftarrow Generator.KeyGen()$
2: $pI \leftarrow$ public values
3: $bool := PfVerify(verK, pf, pI)$

3 Related Work

Several path validation protocols have been proposed over the years. A path validation protocol by Kim et al. enables a node to validate all its predecessors en route before it moves on to delivering the packets [22]. However, like other works on path validation [5–7, 28], it fails to protect the privacy of nodes and the path. To validate the path, their protocol reveals the information of the entire path to every intermediate node as well as the identities of the end users. Sengupta et al. proposed a privacy-preserving path validation to tackle this privacy issue [34], but their work also fails to provide a completely secure system due to the following major reasons: (a) each intermediate node still learns the number of nodes en route, which can still potentially leak more information, even the entire path information under certain attacks; (b) there is no sound malicious node report mechanism; (c) there is no system-level solution once certain malicious nodes are detected, such as re-routing and backup paths. With better security and privacy properties, our protocol guarantees that intermediate nodes would learn nothing beyond their predecessor and successor. The summarized comparison can be found in Table 1

NIZK-based path validation has been adopted and used in the bitcoin payment channels by Malavolta et al [26]. Their work uses NIZK to enforce bitcoin payment-channel transaction orders. However, we leverage the idea of NIZK in the content of content delivery in 5G networks and propose novel solutions like anonymous sender verification and pairwise validation to improve efficiency and compatibility with 5G networks path validation in a more systematic but also thorough fashion. Note that the use case and detailed design of our protocol is different from their protocol despite that both protocols use NIZK.

Several network rerouting solutions have been proposed [16, 35, 39], but none of them consider the rerouting scenario where malicious actors are present and detected. In our work, we propose a complete pipeline from detecting malicious nodes on packet forwarding paths to resolving rerouting according to dynamic malicious behaviors.

4 Problem Definition

We model $\mathcal{A}$ as a probabilistic polynomial-time machine and the nodes on the path as interactive Turing machines.

Definition 1 (Adversary). *An adversary $\mathcal{A}$ is capable of corrupting a subset of nodes in the network. A compromised node will divert to $\mathcal{A}$ all the received messages and act as $\mathcal{A}$ requests.*

As shown in Figure 3, the adversary controlling a subset of compromised nodes may have the following malicious behaviors [4]:

- **Skipping** A malicious node skips its successor and forwards the packet to another malicious node later en route.
- **Detour** A malicious node forwards the packet to other malicious nodes that are not en route but eventually returns back to the agreed path.
- **Out-of-order** A group of malicious nodes forwards the packet but not in the agreed order.

All parties can interact with the ideal world functionality $\mathcal{F}$ via secure channels and the interaction is described in Figure 4.

$\mathcal{F}$ guarantees such security and privacy properties:

- **Path verification** Each intermediate node has to forward the packact received from its predecessor to its successor per the assigned path order.
- **Endhost anonymity** The identity of neither the sender or the receiver will be revealed to intermediate nodes during packet forwarding. Only $\mathcal{F}$ learns the identities.
- **Path privacy** (1) Even if $\mathcal{A}$ compromises intermediate nodes, $\mathcal{A}$ is still uncertain about the sender-receiver pair identity; (2) If there are at least two adjacent honest nodes between two compromised intermediate nodes, $\mathcal{A}$ is not certain whether there is more nodes between this two honest nodes, which means malicious nodes will not learn anything beyond its predecessors and successors. Note that violation of path privacy also provides attack surface for tracking down endhosts which impacts endhost anonymity.
- **Malicious node detection** If any node deviates from its fair share of the agreed path, it cannot provide the valid token to $\mathcal{F}$ hence the malicious action is detectable.

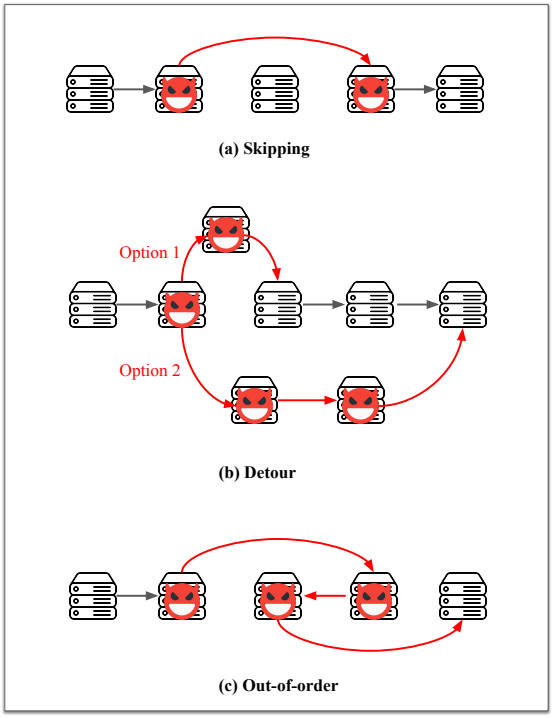


Figure 3. Examples of Typical Malicious Behaviors En Route: compromised nodes (red) can conduct there major malicious behaviors, namely skipping (skips certain honest nodes between compromised nodes), detour(reroutes packets via other compromised nodes that are not on the path) and out-of-order (disrupts the assigned node order).

Initialization

- $\mathcal{F}$ generates sends validation tokens for each nodes en route. $\mathcal{F}$ forwards x_i to each intermediate node N_i.

Secret Release

- Upon receiving the packet, the node N_{i+1} sends the token x_{i+1} to its predecessor.

Secret Submission

- The intermediate N_i receives the token x_{i+1} from its successor and sends it back to $\mathcal{F}$ except for the receiver node who does not need to send a token to $\mathcal{F}$.

Secret Verification

- $\mathcal{F}$ validates x_{i+1} received from N_i.

Figure 4. Ideal World Functionality: $\mathcal{F}$ is computed in the ideal world by a trusted party. In the ideal world version of P^3V, $\mathcal{F}$ guarantees security of path validation.

Definition 2 (UC-Security). *A protocol π is secure if there exists a simulator $\mathcal{S}$ in the ideal world such that, in the presence of $\mathcal{A}$, for all inputs, probability distributions of the ideal world and the real world are computationally indistinguishable.*

In our protocol, we guarantee properties of the ideal world functionality using XOR, hashing and NIZK.

5 XOR-Hash-NIZK Path Validation

In this section, we introduce a novel path validation protocol using hashing, XOR and NIZK to address issues of previous attempts on the privacy-preserving path validation problem. Our protocol aims to provide security and privacy without a huge performance compromise.

5.1 Oblivious Anonymous Sender Verification

Traditional path validation methods require the sender node to reveal its identity to intermediate nodes [22]. This means that certain authenticity verification procedures need to take place. To avoid identity leakage during this process, we propose to use anonymous channels [8] for the sender-node communication along with a digital signature lightweight anonymous sender verification method for other nodes to authenticate the sender for each session but without knowing its actual identity. This is essential to our protocol and it is executed in the initialization stage.

To prove its anonymous identity to other nodes, the sender would first generate a pair of the private key and public key (which will be distributed to each node via authorities). Then the sender uses its private key to sign the message containing validation tokens.

5.2 XOR-Hash Approach

A base approach has been introduced using XOR operations and hashing for path validation [20]. In this approach, XOR operators mask the paths from the current intermediate nodes to the receiver node and hashing is used to validate the XORed results of tokens en route. The protocol interacts as follows:

With a delivery task involving n hops, the sender samples n independent strings $(x_1, \ldots, x_n)$ correspondingly with $y_i = H\left(\bigoplus_{j=i}^{n} x_j\right)$ using a hash function H and the XOR operator $\bigoplus$. Each intermediate node i received from the sender over an anonymous channel a tuple (y_{i+1}, x_i). with the exception where the receiver node receives (y_n, x_n).

The validation procedure starts from the receiver by the receiver node sending the XOR bit string to its predecessor, which should match the hash digest that the predecessor obtained in the initialization stage. Moving on from here, the predecessor uses the XOR bit string with its token to generate a new bit string for validation at its own predecessor. This procedure propagates until the sender node and the sender node will then infer a binary validation result.

Several issues exist in this base approach: (a) each node needs all the nodes after it on the path to finish their validation such that it can execute its own validation procedure, such a serial execution fashion incurs the excessive overall waiting period from a task perspective; (b) failures at any point of the path can fail the validation of all predecessor nodes in the serial execution; (c) it is hard to pinpoint the locations of malicious nodes, i.e., the protocol yields a binary result on whether one or more malicious nodes exist on the path but does not provide the accurate location of the malicious node(s).

5.3 Improved Design: XOR-Hash-NIZK

To solve the issues mentioned above, atop the vanilla protocol, we construct a protocol that utilizes the XOR combiner and the hash function in the form of $y_i = H\left(\bigoplus_{j=i}^{n} x_j\right)$. However, instead of a pure hash function, the validation uses NIZK with SHA256. After completing the content delivery task and receiving its successor's secret, each node N_i generates a NIZK proof as the prover, proving that $y_i = H\left(\bigoplus_{j=i}^{n} x_j\right)$ without revealing $\bigoplus_{j=i}^{n} x_j$ to its predecessor. The predecessor receives the proof and verifies it. This validation process starts from the receiver and propagates back to the sender. Unfortunately, such an easy alternative using NIZK does not provide a sound solution and the listed issues still remain unsolved. Additionally, considering the cost of NIZK operations, the sequential validation execution introduces a dramatically increasing overhead, which makes the protocol impractical in a real-time network. However, the zero-knowledge property of NIZK provides us with the possibility to perform local validation without revealing extra information about the path.

Algorithm 3 $Init_{N_0}(\{N_0, N_i, \ldots, N_n\})$:

1: $(s_0, s_1) \leftarrow random$
2: $(pubK, privK) := KeyGen(s_0)$
3: $Send(OA_0, pubK)$
4: **for** i from n to 1 **do**
5: **if** i == n **then**
6: $(x_n, x_{n+1}) \leftarrow s_1$
7: **else**
8: $x_i \leftarrow s_1$
9: **end if**
10: $y_i := x_i \oplus x_{i+1}$
11: $m_i := Sign(privK, (x_i, y_i, y_{i+1}, provK, veriK))$
12: $Send(N_i, m_i)$
13: **end for**

To completely overcome these issues, we propose a pairwise validation version that validates nodes only using neighboring nodes' secrets, i.e., each node proves to its predecessor that it has delivered the content using its secret and its successor's secret. The intuition behind this design is that if the delivery between each pair is

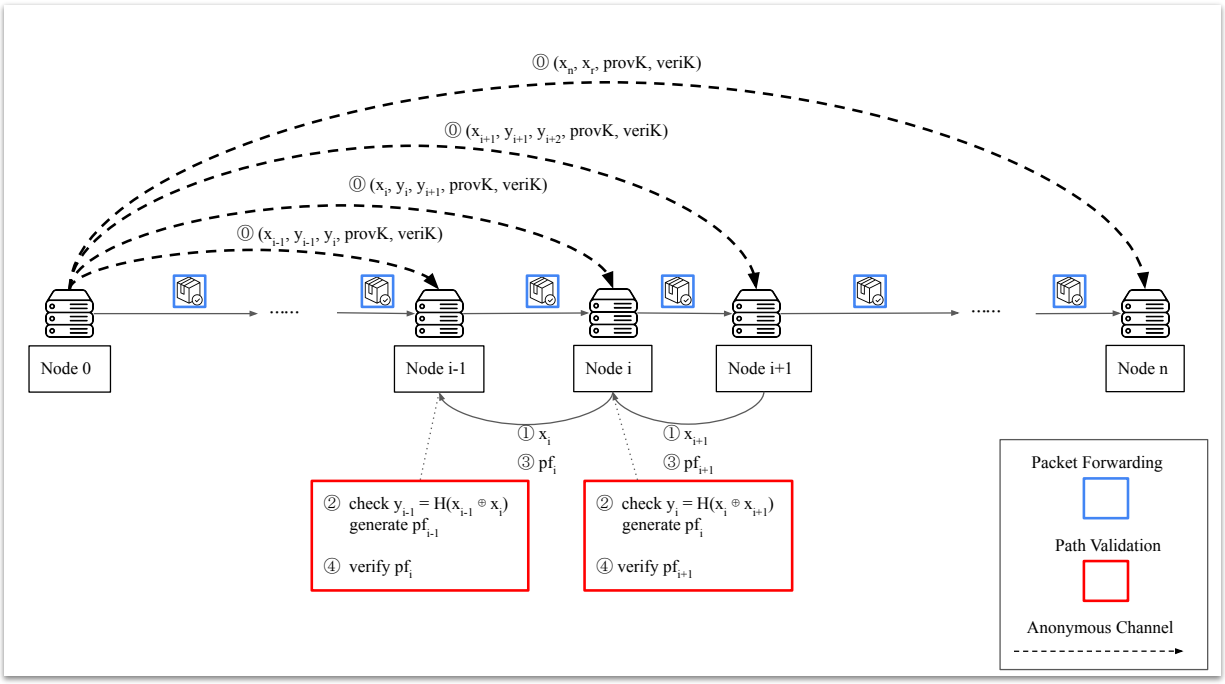


Figure 5. XOR-Hash-NIZK Protocol Workflow (Anonymous Sender Verification Omitted In The Figure): blue depicts the packet forwarding process and red indicates the backward path validation process.

Algorithm 4 $Prove_{N_i}(x_{i+1}, provK, m_i, pubK)$:

1: **if** $(y_i == x_i \oplus x_{i+1}$ **and** $VerifySign(m_i, pubK))$ is not true **then**
2: **return** error
3: **end if**
4: $pf_i := PfGen(provK, y_i, x_i \oplus x_{i+1})$
5: $Send(N_{i-1}, pf_i)$

Algorithm 5 $Validate_{N_i}(pf_{i+1}, verK)$:

1: **if** $PfVerify(verK, pf_{i+1}, y_{i+1})$ is not True **then** $Send(OA_i, \text{rptMsg})$
2: **end if**

proven to be valid, the entire path should be valid as well. We refer to this type of validation as pairwise validation.

Figure 5 and Figure 6 show how the protocol works. The sender node N_0 first sends to each intermediate node N_i along the agreed path its secret x_i and the hash value y_{i+1} for validating its successor. Each hash value $y_i = H(x_i \oplus x_{i+1})$, except the receiver's hash value $y_n = H(x_n)$. If a packet is delivered to the next node and the content of the packet is untampered, the next node reveals its secret to its predecessor. After the delivery task is fulfilled, the intermediate N_i receives the secret x_{i+1} from its successor and generates its proof to prove that it obtains x_{i+1}. N_i sends the proof to its predecessor N_{i-1}. N_{i-1} then verifies the proof. The entire path is validated only when all nodes en route complete their pairwise validation. Note that the validation process runs across nodes in a near-parallel fashion and from the perspective of the entire system it is not a sequential execution of path validation per node with NIZK, which to some degree circumvent the overhead constraint introduced by the use of NIZK.

5.4 Malicious Node Detection

NIZK does not only make a privacy-preserving pairwise validation solution possible. Additionally, NIZK proofs can be proved by anyone in the system, which makes local path validation and distributed malicious node detection viable. The key idea behind our malicious node detection function is that authorities can also learn the hash digests of their managed nodes without revealing any critical information, which means the authority can step in and also be able to verify potential malicious nodes using NIZK proofs.

As shown in Figure 7, whenever a potential malicious node is reported, the corresponding authority will request and then verify the NIZK proof. If the proof fails or is not received, the authority further investigates the node for potential infrastructure compromise or malicious attacks like DDoS and reacts according to its policy; if the proof passes, the authority will mark the reporting node as suspicious and also conducts an investigation. Authorities can form certain malicious behavior policies at their will.

Initialization

- The sender node N_0 generates secrets x and hash digests y for each node along the path. The sender node N_0 sends via an anonymous channel to each intermediate node N_i along the agreed path directed by the network slicing authority its secret x_i (except that the receiver also receives an additional value x_r), its hash value y_i and the hash value y_{i+1} for validating its successor. Each hash value $y_i = H(x_i \oplus x_{i+1})$, i.e., y_i is the hash value of XORing x_i and x_{i+1}, except the receiver's hash value $y_n = H(x_n \oplus x_r)$. To prove its anonymous identity to other nodes, the sender also generates a pair of private key and public key (which will be distributed to each node via authorities). Then the sender uses its private key to sign the message containing validation tokens.
- The authority of each node acts as NIZK Generator to produce a proving key pk and a verification key vk (which will be distributed to the node's predecessor via other authorities).
- Upon receiving messages on the anonymous channel, each node first verifies the anonymous sender's identity by checking the signature using the received public key and only proceeds if the identity verification passes.

Secret Release

- After the delivery contract is fulfilled, the node N_{i+1} sends the secret x_{i+1} to its predecessor.

Proof Generation

- The intermediate N_i receives the secret x_{i+1} from its successor and generates its NIZK proof to prove that it obtains x_{i+1}.

Proof Verification

- N_i sends the proof to its predecessor N_{i-1}. N_{i-1} verifies the proof $y_i = H(x_i \oplus x_{i+1})$.
- If N_i's proof does not pass, N_{i-1} reports the potential malicious behavior to its authority. The authority requests the NIZK proof from N_i and verifies it. If the proof fails or is not received, the authority punishes the node according to its policy; if the proof passes, the authority will conduct an investigation on the reporting node per policy.

Figure 6. XOR-Hash-NIZK Path Validation Protocol

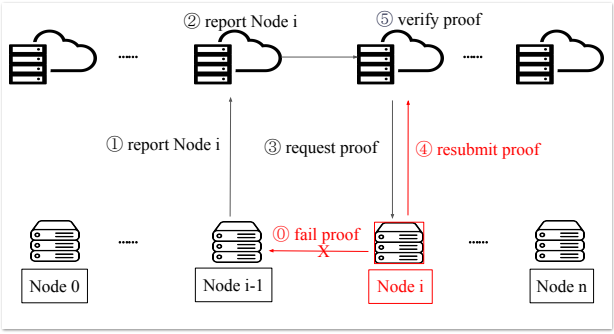


Figure 7. Malicious Node Detection

5.5 Security Analysis

Theorem 1. *(UC-Security) Let H be a random-oracle hash function and $(\mathcal{P}, \mathcal{V})$ be a NIZK proof system, our protocol π securely realizes the ideal functionality $\mathcal{F}$ in the presence of an adversary $\mathcal{A}$.*

We use a simulator $\mathcal{S}$ to simulate the real-world executions and interact with the ideal world functionality $\mathcal{F}$. Communications among parties are through secure channels.

In our protocol, the sender is assumed to be an honest party that obtains the knowledge of the entire path. The operations of our protocol are redescribed here from the perspectives of $\mathcal{F}$, $\mathcal{S}$ and $\mathcal{A}$. Each honest node u_i receives a message parsed as $(< u_{i-1}, u_i >, < u_i, u_{i+1} >, x_i, y_i, y_{i+1})$ with the special case of the honest receiver as $(< u_{n-1}, u_n >, x_n, x_r, y_n)$. For each u_i, $\mathcal{S}$ checks if $\mathcal{V}(y_i = H(x_i \oplus x_{i+1})) = 1$ (for the receiver, it is $\mathcal{V}(y_i = H(x_n \oplus x_r)) = 1$) and sends $\mathcal{F}$ a correct message otherwise aborts. For each corrupted intermediate node, $\mathcal{S}$ is also notified with $(< u_{i-1}, u_i >, < u_i, u_{i+1} >, x_i, y_i, y_{i+1})$ from $\mathcal{F}$. $\mathcal{S}$ samples an $x \in \{0,1\}^\lambda$ and $x' \in \{0,1\}^\lambda$. On input $(x, x', H(x \oplus x'))$, $\mathcal{S}$ runs the simulation of NIZK to obtain the proof as $\mathcal{P}(y = H(x \oplus x')) = 1$. $\mathcal{A}$ sends $\top$ to $\mathcal{F}$ if $\mathcal{A}$ sends x'' such that $x'' = x \oplus x'$ to $\mathcal{S}$, otherwise the simulation is aborted.

We then prove that the view of the environment in such a polynomial-efficient simulation is indistinguishable from the view of the execution of the real-world protocol. Our NIZK-based protocol breaks the multi-hop validation into a single-hop interaction mode. $\mathcal{F}$ always receives $\top$ returned from each honest node that completes the delivery and passes the validation, which means the validation is not interrupted at honest nodes. $\mathcal{S}$ aborts if $\mathcal{A}$ fails to output $x_i \oplus x_{i+1}$ for corrupted node u_i. Additionally, $\mathcal{S}$ learns nothing more from $\mathcal{F}$ than identities of the corrupted node's predecessor and successor, which is the same in the real-world execution. The distributions of real-world protocol executions and simulated results are probabilistically indifferentiable.

6 Malice-Resilient Rerouting

Once malicious nodes en route are detected, it is critical to restore the content delivery path per service promised. The principle of service rerouting is to search for a suboptimal path that excludes the detected malicious nodes and their related entities. Excluding certain nodes shifts the topology of substrate networks as the underlying network infrastructures, which in general should be dynamically adjusted according to the status of malicious behaviors within the network.

However, it is challenging to apply a rerouting strategy while minimizing impacts on service quality guarantee when malicious path-forwarding behaviors are present. We first study the two fundamental strategies for path rerouting and then propose an efficient rerouting design integrated with VNE-CBS.

6.1 Base Strategies

There are two basic rerouting strategies for resolving compromised paths, namely the pre-generated path backup and the on-the-fly path regenerating:

- **Pre-generated path backup** When generating a path for a certain content delivery task, the authorities also produce one or more backup paths by selectively removing some or even all nodes from the substrate network.
- **On-the-fly (OTF) path regenerating** Once a set of malicious nodes are detected, the slicing authorities work in real-time to generate a new path that excludes the malicious nodes and their potentially-related nodes.

In practice, compared to the OTF path regenerating, the pre-generated path backup approach has several problems. First, to obtain decent coverages for potential malicious nodes, the amount of backup paths needed increases dramatically with more hops en route. For example, a 4-hop (3 intermediate nodes) path requires in total 7 different backup paths and a 6-hop path (5 intermediate nodes) path requires in total 31 backup paths. Additionally, reserving such a large amount of resources on the network (from a business perspective, it does not make sense to service providers to put all these resources on hold just for failures caused by potentially malicious actors) and storing all the backup path information and also related tokens on the responding party are not practical in reality.

However, the OTF path regenerating approach does not have the issues discussed above by dynamically recomputing a new path after detecting malicious nodes. This guarantees that detected malicious nodes will not be on the new path. Apparently, this OTF approach requires additional communication among parties in the system, but the extra communicational overhead is also inevitable using the pre-generated backup paths.

For these reasons, our malicious-node-resilient rerouting mainly adopts the OTF path regenerating strategy.

6.2 Efficient and Effective Rerouting

Algorithm 6 $Rerouting(G, N_{mal})$:

```
1: Remove N_mal from the network G
2: Scan G for edge changes
3: for all egde changes do
4:     Update edges and vertices
5: end for
6: Run VNE-CBS focusing on changes within G
```

When regenerating OTF paths, the path slicing works on an updated substrate network with detected malicious nodes removed. This means that the major part of the network topology remains unchanged. Using this observation, we leverage the idea from D^* Lite [23] to construct an efficient path regenerating algorithm for malicious node resolution. The main idea of D^* Lite is that changes in the graph topology (e.g., a new blockage that was unknown before) only change a small amount of cells' estimated goal distances while most of the cells status stays the same, which means recalculating a path only involves cells with changed status.

As shown in Algorithm 6, for a network path finding task, detected malicious nodes can be viewed as new blockages in the network. The removal of these malicious nodes updates the topology but only on vertices and edges that are changed and relevant. Since the algorithm does not expand unchanged vertices, it is efficient compared to rerunning the entire VNE-CBS from scratch. We refer to this rerouting strategy in our system as Local Repairing.

In the extreme case that there is no successful suboptimal path searched with a focus on only changes after a certain threshold of attempts, we destroy the path by replacing all nodes and rerouting searching for a completely new path instead of forcing Local Repairing.

7 System Construction

7.1 Substrate Network

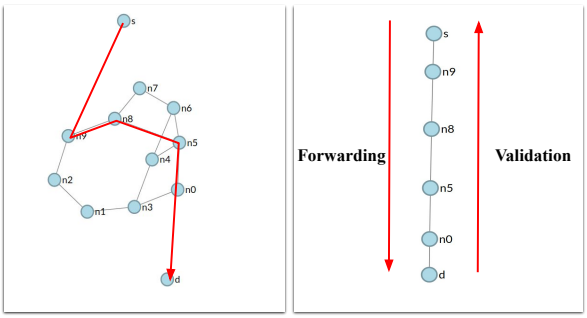
We build our system on a 5G Dispersed Computing Testbed with up to 1200 available computing nodes (Ubuntu 18.04.5 LTS with Dual-core Intel E3826 and 2 GB of RAM), which are dynamically configured to be the substrate network per request. Figure 8 shows an example of our network system using such a testbed as the substrate network. Ansible [17] is used to orchestrate the nodes within the system.

7.2 Path Slicing

The first module of our system is VNE path slicing/finding. Given a set of network nodes with various

Setup	Accept Ratio	Avg Revenue	Avg Cost	Revenue/Cost Ratio	Avg Runtime (s)
10-20-50	1.00	66.05	66.05	1.00	0.009
10-20-80	0.98	419.81	598.63	0.70	0.106
10-50-50	0.99	114.33	114.33	1.00	0.001
10-50-80	0.96	122.71	122.71	1.00	0.002
20-20-50	0.76	126.12	186.34	0.68	1.001
20-20-80	0.71	159.34	159.72	1.00	1.012
20-50-50	0.74	444.99	512.43	0.87	1.351
20-50-80	0.69	561.44	837.86	0.67	2.211

Table 2. Microbenchmark of VNE-CBS: setup format is {NumberOfVNRs}-{BoundOfCPU}-{BoundOfBandwidth}. Each setup has been executed 100 times and the results are averaged.



(a) Path Slicing (b) Path Forwarding + Validation

Figure 8. Testbed Example: with a substrate network of 12 nodes and 15 edges, an optimal path with 5 hops is found.

available resources of CPU and bandwidth at different locations, the path finding algorithm will be executed to generate a suitable path that satisfies QoS/SLA for a specific task. From an input/output perspective, the inputs are VNR with the specific sender-receiver relationship and substrate network specifications; the output is information about nodes selected in order.

We adopt an assumption that each authority submits information about its available infrastructures to a trusted party (an oracle) to run the VNE-CBS algorithm and the oracle returns the path back to slicing authorities and then all associated nodes. This is because, to the best of our knowledge, there is currently no sound secure multi-party path finding algorithms that fit our needs. Secure multi-party path finding is not in the research scope of this paper but our system can be easily modified with a secure path finding algorithm.

7.3 Path Validation

After path slicing, authorities agree on a certain path for the packet delivery task and then inform each node en route of its predecessor and successor.

NIZK implementation We use a concatenation SHA256 hashing version of NIZK based on libsnark library [24]'s compression hashing gadget by adding finalization steps like padding. In the initialization stage upon the sender's service request, the sender's authority (acting as Generator) is responsible for generating a key pair of proving key and verifying key. The key pair then will be distributed to all nodes via authority-to-authority communication. Keys and proofs are serialized/deserialized using basic ifstream/ofstream.

7.4 Path Rerouting

After malicious nodes are detected, the system removes these nodes from the substrate network by releasing their resources. In our implementation, malicious nodes are simulated randomly at a given rate. The removal of nodes will update the substrate network, which will be shared with the path slicing oracle via orchestration. The oracle will attempt VNE-CBS reruns to find a malice-free suboptimal path with focusing on edge changes in the underlying substrate network topology as discussed in §6. A successful Local Repairing will yield a semi-new designated path that restarts the system pipeline, otherwise a destroy path solution for a completely new path will be adopted. In our current system implementation, we manually set an attempt threshold for path rerouting, i.e., the packet delivery task in our implementation will be aborted after a fixed number of attempts.

8 Evaluation

In this section, we present our experimental setup and a detailed evaluation of the P^3V system. We seek to answer the following key questions:

- The effectiveness and efficiency of the VNE-CBS algorithm with specified service requests on restricted substrate network resources
- The extra overhead introduced by our privacy-preserving path validation protocol XOR-Hash-NIZK in both simulation and 5G testbed implemenation as well as the potential impact on service quality from enhanced security and privacy

8.1 Experimental Setup

We implement our system in C++ and JavaScript with NZIK library libsnark and OpenSSL [36] for hashing. We

Operation (s)	Sender	Intermediate	Receiver
Token Generation + Hashing	n: 8.32	0	0
Signature Generation (RSA 1024)	n: 48.17	0	0
Signature Verification (RSA 1024)	0	1:0.02	1:0.02
NIZK Proof Generation	0	1:5988.46	1:5988.46
NIZK Proof Verification	1:90.52	1:90.52	0

Table 3. Step Cost of XOR-Hash-NIZK Protocol in Docker Compose Simulation: here shows an example of 100 nodes; in $(i : j)$, i means iterations required across parties and j means total runtime (in ms).

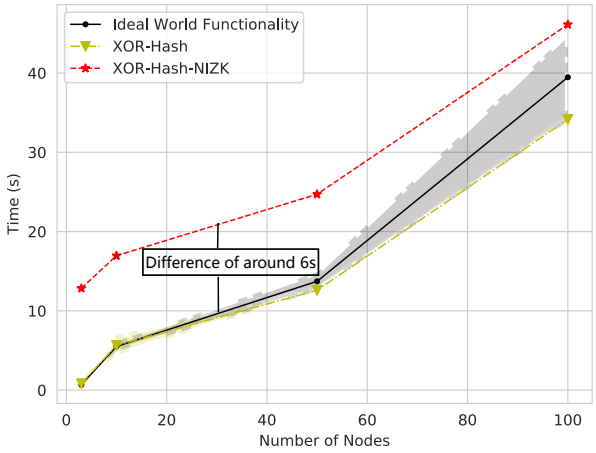


Figure 9. Small Scale Comparison in Docker Compose Simulation: three protocols are ideal world functionality (a trusted central server), the XOR-Hash protocol and our improved protocol (XOR-Hash-NIZK); the cost difference of around 6s between XOR-Hash-NIZK and Ideal/XOR-Hash mainly comes from the execution of one round of NIZK proof generation and verification.

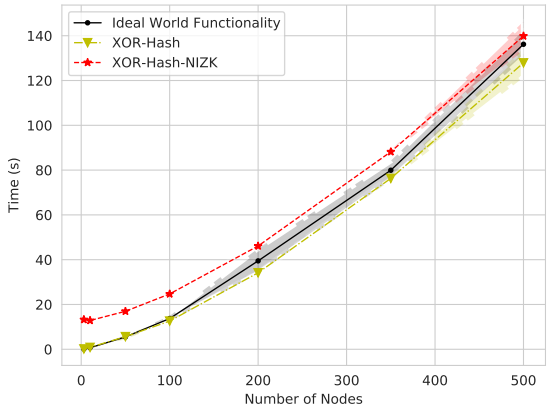


Figure 10. Large Scale Comparison in Docker Compose Simulation: the cost comparison of three versions of path validation at a larger scale with similar setups

evaluate our system in two different setups: a Docker Compose [18] simulation and an actual multi-node 5G Dispersed Computing Testbed as described above in §7.1. Specifically, our simulation runs on a machine (Intel® Core™ i7-7700 CPU @ 3.60GHz × 8 and 16 GB of RAM) using Docker Compose to simulate a network with multiple distributed nodes.

To generate substrate network topologies and random VNR requests in our VNE experiments, we utilize the Waxman graphs [37]. The Waxman graph is a popular algorithm to model random but realistic geometric graphs. In our experiments, we generate testing communication networks and network services requests with Waxman parameters α as 0.5 and β as 0.2 [31]. In the generated substrate networks, the node CPU resource values are bounded within $[20, 50]$ and the edge bandwidth values are bounded within $[50, 100]$. For VNRs, we have a set of different request configurations, with the number of VNRs being either 10 or 20; the bound of node CPU resources being 20 or 50; the bound of edge bandwidth being 50 or 80. These setups are used in microbenchmarking the VNE-CBS path slicing functionality.

8.2 Evaluation Results

VNE-CBS evaluation We first evaluate the performance of the VNE-CBS algorithm under our experimental setting. As shown in Table 2, we test the slicing algorithm using 3 different parameters for configuration with 2 boundary values for each parameter (i.e., in total 8 setups). In each setup, there are 100 VNR requests and their results are averaged. Our VNE-CBS evaluation metrics consist of 4 categories, namely accept ratio (success rate of finding a solution before timeout), revenue (satisfied VNR resources), cost (utilized SN resources), revenue/cost ratio and runtime (execution time to find a solution that satisfies the requirements).

Takeaways The evaluation results show that with more strict service requirements in VNR requests, there are not always feasible solutions guaranteed in the current substrate networks and the accept ratio drops. The more strict requests also result in a longer average runtime to find satisfied solutions. As for revenues and costs, it is easier overall to find VNE mappings with high revenue/cost ratios from requests with loose constraints. In our experiment, we purposefully set generally strict requirements for testing and the VNE-CBS algorithm is shown to be efficient as a path slicing solution.

Path validation benchmark evaluation We then benchmark the cost of our path validation protocol in the simulation environment. In the Docker Compose simulation, we first break down the overhead of the protocol by looking at different parties in the system, i.e., the sender, the intermediate nodes and the receiver shown in Table 3. The sender needs to generate tokens with hash digests and also sign the messages. These operations have to be repeated for all nodes but are relatively low-cost. The intermediate nodes (and the receiver) are mainly responsible for verifying the message signature received. Additionally for the backward path validation, they need to generate their own NIZK proof and also verify their predecessor's proof (but not at the receiver). Although each node at most is only required to perform one iteration of NIZK operation, NIZK proofs are in general computation-demanding cryptographic primitives where the majority share of the cost resides in the proof generation phase (with our pairwise optimization in place, the nearly parallel execution of proof verification still yields around 6 seconds). The effect of this additional overhead, compared to path validation protocols that are not based on NIZK (the trusted centralized server protocol and the XOR-Hash protocol), can be seen in Figure 9 and Figure 10. In these two figures, costs involve stages of initialization, validation token distribution, secret release and validation execution for each protocol. For the ideal world functionality, the cost includes an extra step of secret submission; for the XOR-Hash and XOR-Hash-NIZK protocols, the cost includes anonymous sender verification, XORing and hashing tokens, and NIZK proof generation/verification (only in XOR-Hash-NIZK). Especially in the small-scale evaluation, it is clearly demonstrated that our NIZK-based path validation protocol brings in a noticeable performance gap which although becomes less obvious with more nodes on the path.

Takeaways Intuitively, this delay of around 6 seconds will inevitably compromise the service quality for end users. However, this apparent network performance compromise can be justified. Recall that our path validation is carried out in a backward fashion and is also executed in a pair-wise validation mode. With these two characteristics, as we focus on the service quality in terms of user experience, our path validation process is rather separated from the forward packet delivery process as a backend service compared to traditional validate-before-forward approaches and will not be noticed from a user standpoint in normal cases with no malicious behaviors. With the presence of malicious parties, the forwarding service is mainly destructed by malicious behaviors already. For example, in the case of detouring, the user experience might also not encounter any huge difference caused by the validation module but only from the detouring forced by the adversary.

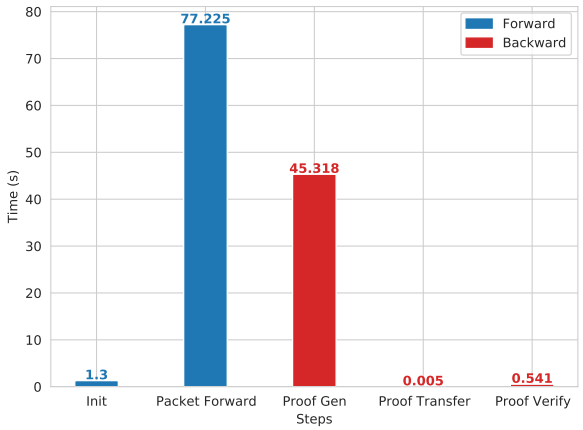


Figure 11. Testbed Step Evaluation (Example of 20 Nodes): an end-user perspective where forward steps (blue) directly impact network service quality (e.g., the content delivery time cost per communication constraints) and backward steps (red) are related to the path validation process running in the backend (in normal cases which will not be noticed by users)

XOR-Hash-NIZK testbed evaluation In addition to the simulation analysis, we also build and test a path validation system on our 5G Dispersed Computing Testbed. Our results indicate that our system demonstrates a similar performance tendency for the path validation module on the multi-machine testbed compared to the simulation results shown above. In our testbed system implementation, we also focus on evaluating the trade-offs from the end-user perspective. In Figure 11, we use an example of a path with 20 nodes en route on the testbed and the sender has requested a forward task of a 5GB file (the typical file size of an HD movie). The tested average bandwidth from Node 1 to Node 20 is around 530.4 Mbps (i.e., 66.3 MB/s).

Takeaways According the mechanism of our validation process, the backward pairwise validation starts right after the packet reaches the first hop, which means that a large portion of the nodes have finished the validation at the time when forward process completes. As we discussed in the simulation result part, similarly, from the end-user experience, the potential performance compromise from additional privacy-preserving secure path validation is inevitable, but to some extent, can be justifiable regarding service quality guarantee.

9 Concluding Remarks

In this work, we propose a decentralized privacy-preserving path validation system P^3V, which guarantees the privacy of paths and nodes while further enhancing network security during packet delivery tasks against information leakage about multi-hop paths and

potentially the underlying network infrastructures. Additionally, our system integrates our path validation protocol with an efficient path slicing algorithm and a malice-resilient path rerouting mechanism, which is built and evaluated in a simulation as well as a testbed implementation. As for future work, it is crucial to further improve the trade-offs between security/privacy and service quality and provide more complex functionalities (e.g., privacy-preserving multi-authority path slicing and agreement) in our path validation system.

References

[1] Abdallah Mustafa Abdelrahman, Joel JPC Rodrigues, Mukhtar ME Mahmoud, Kashif Saleem, Ashok Kumar Das, Valery Korotaev, and Sergei A Kozlov. Software-defined networking security for private data center networks and clouds: vulnerabilities, attacks, countermeasures, and solutions. *International Journal of Communication Systems*, 34(4):e4706, 2021.

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[3] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349, 2012.

[4] Kai Bu, Avery Laird, Yutian Yang, Linfeng Cheng, Jiaqing Luo, Yingjiu Li, and Kui Ren. Unveiling the mystery of internet packet forwarding: A survey of network path validation. *ACM Computing Surveys (CSUR)*, 53(5):1–34, 2020.

[5] Hao Cai and Tilman Wolf. Source authentication and path validation with orthogonal network capabilities. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 111–112. IEEE, 2015.

[6] Hao Cai and Tilman Wolf. Source authentication and path validation in networks using orthogonal sequences. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–10. IEEE, 2016.

[7] Kenneth L Calvert, James Griffioen, and Leonid Poutievski. Separating routing and forwarding: A clean-slate network layer design. In *2007 Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS'07)*, pages 261–270. IEEE, 2007.

[8] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *Annual International Cryptology Conference*, pages 169–187. Springer, 2005.

[9] Abdelberi Chaabane, Pere Manils, and Mohamed Ali Kaafar. Digging into anonymous traffic: A deep analysis of the tor anonymizing network. In *2010 fourth international conference on network and system security*, pages 167–174. IEEE, 2010.

[10] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, 41(2):38–47, 2011.

[11] NM Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM 2009*, pages 783–791. IEEE, 2009.

[12] Liangdong Deng, Yuzhou Feng, Dong Chen, and Naphtali Rishe. Iotspot: Identifying the iot devices using their anonymous network traffic data. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2019.

[13] Sergey N Dorogovtsev and José Fernando F Mendes. Scaling behaviour of developing and decaying networks. *EPL (Europhysics Letters)*, 52(1):33, 2000.

[14] Ernesto Estrada. Network robustness to targeted attacks. the interplay of expansibility and degree distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 52(4):563–574, 2006.

[15] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[16] Masoumeh Gholami and Behzad Akbari. Congestion control in software defined data center networks through flow rerouting. In *2015 23rd Iranian conference on electrical engineering*, pages 654–657. IEEE, 2015.

[17] Red Hat. Ansible is simple it automation, 2022. https://www.ansible.com/.

[18] Docker Inc. Overview of docker compose, 2022.

[19] Jian Jiang, Wei Li, Junzhou Luo, and Jing Tan. A network accountability based verification mechanism for detecting inter-domain routing path inconsistency. *Journal of Network and Computer Applications*, 36(6):1671–1683, 2013.

[20] Weizhao Jin, Srivatsan Ravi, and Erik Kline. Decentralized privacy-preserving path validation for multi-slicing-authority 5g networks. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 31–36. IEEE, 2022.

[21] Ishan Karunanayake, Nadeem Ahmed, Robert Malaney, Rafiqul Islam, and Sanjay Jha. Anonymity with tor: A survey on tor attacks. *arXiv preprint arXiv:2009.13018*, 2020.

[22] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, pages 271–282, 2014.

[23] Sven Koenig and Maxim Likhachev. D^* lite. *Aaai/iaai*, 15:476–483, 2002.

[24] SCIPR Lab. libsnark: a c++ library for zksnark proofs, 2021. https://github.com/scipr-lab/libsnark.

[25] Matt Lepinski and Kotikalapudi Sriram. BGPsec Protocol Specification. RFC 8205, September 2017.

[26] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 455–471, 2017.

[27] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[28] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazieres, Michael Miller, and Arun Seehra. Verifying and enforcing network paths with icing. In *Proceedings of the Seventh Conference on Emerging Networking Experiments and Technologies*, pages 1–12, 2011.

[29] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1962–1976, 2018.

[30] Carlos Natalino, Amaro de Sousa, Lena Wosinska, and Marija Furdek. Content placement in 5g-enabled edge/core data center networks resilient to link cut attacks. *Networks*, 75(4):392–404, 2020.

[31] NetworkX. waxman graph, 2022. https://networkx.org.

[32] David R Raymond and Scott F Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7(1):74–81, 2008.

[33] Nadav Schweitzer, Ariel Stulman, Tirza Hirst, Roy David Margalit, and Asaf Shabtai. Network bottlenecks in olsr based ad-hoc networks. *Ad Hoc Networks*, 88:36–54, 2019.

[34] Binanda Sengupta, Yingjiu Li, Kai Bu, and Robert H Deng. Privacy-preserving network path validation. *ACM Transactions on Internet Technology (TOIT)*, 20(1):1–27, 2020.

[35] Akash Srikanth, P Varalakshmi, Vignesh Somasundaram, and Pavithran Ravichandiran. Congestion control mechanism in software defined networking by traffic rerouting. In *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, pages 55–58. IEEE, 2018.

[36] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS. www.openssl.org, April 2003.

[37] Bernard M Waxman. Routing of multipoint connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.

[38] Wikipedia. 5g network slicing, 2022. https://en.wikipedia.org/wiki/5G_network_slicing.

[39] Eric WM Wong, Andy KM Chan, and T-SP Yum. A taxonomy of rerouting in circuit-switched networks. *IEEE Communications Magazine*, 37(11):116–122, 1999.

[40] Peng Xiao, Wenyu Qu, Heng Qi, and Zhiyang Li. Detecting ddos attacks against data center with correlation analysis. *Computer Communications*, 67:66–74, 2015.

[41] Shunliang Zhang. An overview of network slicing for 5g. *IEEE Wireless Communications*, 26(3):111–117, 2019.

[42] Yi Zheng, Srivatsan Ravi, Erik Kline, Sven Koenig, and T. K. Satish Kumar. Conflict-based search for the virtual network embedding problem. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32(1):423–433, Jun. 2022.