Divide and Conquer: Efficient Multi-path Validation with ProMPV

Anxiao He(✉), Yubai Xie, Wensen Mao, and Tienpei Yeh

Zhejiang University, Hangzhou 310007, ZJ, China
{zjuhax,xiaoxiaobai,wsmao,3130001051}@zju.edu.cn

Abstract. Path validation has long be explored toward forwarding reliability of Internet traffic. Adding cryptographic primitives in packet headers, path validation enables routers to enforce which path a packet should follow and to verify whether the packet has followed the path. How to implement path validation for multi-path routing is yet to be investigated. We find that it leads to an impractically low efficiency when simply applying existing single-path validation to multi-path routing.

In this paper, we present ProMPV as an initiative to explore efficient multi-path validation for multi-path routing. We segment the forwarding path into segments of three routers following a sliding window with size one. Based on this observation, we design ProMPV as a proactive multi-path validation protocol in that it requires a router to proactively leave to its second next hop with proofs that cannot be tampered by its next hop. In multi-path routing, this greatly optimizes the computation and packet size. A packet no longer needs to carry all proofs of routers along all paths. Instead, it iteratively updates its carried proofs that correspond to only three hops. We validate the security and performance of ProMPV through security analysis and experiment results, respectively.

Keywords: Path validation · Multi-path routing · Source authentication · Routing strategy

1 Introduction

Path validation has long been explored to secure the forwarding process of Internet traffic [2]. In the current Internet, both the source and the destination have no control over the forwarding of their communication traffic. This leaves various forwarding anomalies unnoticeble. For example, the source and the destination may have signed up for a premium service (e.g., high bandwidth) for their communication. However, the service provider may direct their traffic along a path with inferior performance. Such mis-forwarding can also be exploited to breach security. Consider, for example, when the destination requires that all traffic toward it be examined via a security middlebox. If an attack packet toward it circumvents the security middlebox yet cannot be detected, the destination may be attacked. To address these concerns, path validation enables routers to perform additional cryptographic primitives on packets. Specifically, it introduces

X. Chu et al. (Eds.): QShine 2019, LNICST 300, pp. 219–234, 2020.
https://doi.org/10.1007/978-3-030-38819-5_14

both enforcement and verification over the packet forwarding process. Forwarding enforcement aims to regulate routers of how to forward a specific packet. Forwarding verification aims to enable routers to check whether a packet has been forwarded as required. Both operations are implemented through packet-carried cryptographic proofs. The key idea is that routers add their proofs in packet headers toward an unforgeable forwarding history of packets. Current path validation solutions focus mainly on the single-path routing scenario and has an $O(n)$ overhead given an n-hop path.

As network develops rapidly, single-path routing cannot satisfy the demands of speed and security, thus multipath technology appeals. Multipath routing combines efficiency and robustness, allowing packets travel through multiple paths with same source to the same destination [3]. The balance between efficiency and security is critical for the deployment of Internet protocols [10]. However, simply use single-path validation scheme in multipath situation would cost an unacceptable overhead. First, in multipath situation, every node has a group of downstream node to choose, and with the path length increases, the number of nodes would grow in exponential form, making the validation cost too big to store. According to the measurement study in [11], for 300 million AS pairs, more than 50% of them can find at least tens of alternate paths while 25% of them have more than 100 alternate paths. Second, single-path schemes request the whole path's information to build a verification chain, but it is impossible in multipath situation since source may not know the entire path followed by a packet. With specific routing strategy, routers can only decide its next hop according to real-time network status [12]. Therefore, it is meaningless to pre-build a path as packet would change directions in the middle way unless we pre-compute all the possible ways, however, the cost is too large.

In this paper, we present ProMPV that achieves efficient multi-path validation [7,9] in a divide-and-conquer way. To investigate useful efficiency techniques, we start with analyzing the characteristics of existing single-path validation solutions. We find that a packet may not have to carry a router's proof to all the router's downstream routers for verification. The enforcement and verification of a forwarding path can be divided and each division conquers a segment of the forwarding path. Specifically, we segment the forwarding path into segments of three routers following a sliding window with size one. Consider, for example, an n-hop forwarding path $(r_1, r_2, ..., r_n)$. The segments ProMPV handles are (r_1, r_2, r_3), (r_2, r_3, r_4), (r_3, r_4, r_5), and so on. The key idea is that we can require r_1 to leave a proof to its next hop's next hop, that is, r_3. If r_3 can successfully verify the expected proof from r_1, it demonstrates that the intermediary r_2 forwards packets correctly without any misbehavior. Based on this observation, we design ProMPV as a proactive multi-path validation protocol in that it requires a router to proactively leave to its second next hop with proofs that cannot be tampered by its next hop. In multi-path routing, this greatly optimizes the computation and packet size. A packet no longer needs to carry all proofs of routers along all paths. Instead, it iteratively updates its carried proofs that correspond to only three hops.

In summary, we make the following major contributions to efficient multi-path validation.

- Identify the challenges of path validation in multi-path routing. We analyze the infeasibility of directly applying single-path validation solutions to multi-path validation.
- Propose a proactive multi-path validation technique that achieves efficient multi-path validation in a divide-and-conquer fashion.
- Design ProMPV based on the proactive divide-and-conquer technique.
- Prove the security properties of ProMPV.
- Implement ProMPV using OpenSSL and evaluate its performance. The time and space overhead of ProMPV is insensitive to the path length. Different key lengths can be selected under different security requirements to balance time cost and security level.

The rest of the paper is organized as follows. Section 2 reviews existing path validation solutions and underline their infeasibility to multi-path routing. Section 3 proposes ProMPV toward efficient multi-path validation using a divide-and-conquer technique. Section 4 details the ProMPV design. Section 5 proves the security of ProMPV. Sects. 6 and 7 prototype ProMPV and evaluate its performance. Finally, Sect. 8 concludes the paper.

2 Problem

In this section, we review existing single-path validation solutions and identify their infeasibility in multi-path routing. Nowadays, the number of Internet users and their requirement on bandwidth and quality are increasing more rapidly than ever. Single-path routing can no longer keep up with this pace. In comparison with single-path routing, multi-path routing can effectively avoid the situations like router failures and link congestion as more than one forwarding paths are available. Besides, idle network bandwidth can be fully utilized; the end-to-end delay can be reduced. From the security point of view, packets transmitted by multi-path routing are more difficult to be attacked because it is more challenging for the attacker to simultaneously control several alternate forwarding paths. Albeit multi-path routing is more advantageous than single-path routing in terms of fault tolerance, routing reliability, bandwidth utilization, and security, various challenges lie in multi-path validation.

2.1 Single-Path Validation

The initiative work on singe-path validaiton is ICING [5]. ICING introduces two types of proofs, Proof of Consent (PoC) and Proof of Provenance (PoP). PoCs are used for each router to demonstrate that is has the permission to forward certain packets. PoPs are used to prove that a router has processed certain packets. Both of these proofs are added to the corresponding verification fields in packet headers. When the packet propagates through the network, each node

verifies whether the packet has followed its approved path. ICING performs path validation as follows. First, it check whether the single routing path is approved. The path can be confirmed by checking whether the PoC is consistent. Then each node confirms the previous routers by verifying its verification domain. Finally, the PoP is used to update its verification domain for all subsequent verification nodes. The PoP takes the first 8 bytes of the AES-CBC-MAC hash value. So every node can perform source authentication and path verification. In OPT, a PoP alike field called PVF is used. The source authentication and path verification are verified by PVF. PVF is a set of nested MAC values, and the source generates all PVFs in advance. The value of each path node is calculated using PVF. Source authentication and path verification only need to compare the calculation results given by the source. OPT lets the source to take over a large portion of computation that otherwise is performed by intermediate routers as in ICING. OPT is therefore much faster than ICING.

2.2 Infeasiblity to Multi-path Validation

As mentioned before, ICING and OPT are both single-path validation solutions. They need to determine the entire path before the session begins, including the path length and routers that the packet passes through. However, in multi-path routing, there are many paths connecting the source and the destination. These paths may pass through different routers and their path lengths are not fixed. The choice is depending on the routing strategy. So if we simply use single-path validation scheme in multi-path situation without modifying, multi-path would be considered as a set of many single paths. In this way, the overhead of time and space might be explosive with the number of possible forwarding paths.

Waste of Packet Size and Bandwidth. Traditional schemes need to build the complete path at first to help verify the packet. It is feasible in single path since there may be at most no more than 30 hops in a path [6] and every router knows exactly who is the next hop. But in multi-path situation, things are different. Routers do not know their next hops at the beginning, they need to choose them in real time according to network status such as link availability. This way, each router may have several choices for forwarding a packet to the next hop. If the source uses the normal way, the size complexity would be an exponential function related to the number of hops. It is a huge pressure on packet size. Also, it would cost much of the bandwidth used for transferring package data.

Low Efficiency. Path validation needs to compute many validation fields before the packet transfer to fast the validation process. The source can pre-compute some of these fields to accelerate the processing speed. However, in the multi-path situation, to verify each other, every two routers have to share a pair of symmetric keys and this would cost many calculation resources. Moreover, the source has to pre-compute many fields used for verification. All these computation lead to a low validation efficiency when many possible forwarding paths enforce a large amount of computation.

3 Overview

In this section, we construct a symmetric-key encryption scheme to address the aforementioned validation inefficiencies. It motivates out efficient multi-path validation ProMPV to be presented in Sect. 4.

3.1 Motivation

To address the limitations of the former schemes, we propose a new scheme with the following properties: symmetry and divide-and-conquer. First, we use symmetric encryption to minimize the key size at the same security level compared with when asymmetric encryption is used. This decreases the cost of key storage and increases the speed of computation. Second, since storing and validating an entire path lead to a high overhead, our scheme divides the path into several segments.

Symmetry. Our scheme uses symmetric-key encryption to minimize the amount of calculation and accelerate the processing speed. Asymmetric-key encryption can decrease the number of key pairs between every pair of routers. Although this can save some cost, its large key size and group size would cost more space. For example, given a 128-bit security level, symmetric-key encryption like AES only needs 128 bits to store a key while asymmetric-key encryption like DH needs 256 bits for key size and another $3,072$ bits for group size, and even ECC will needs other 256 more bits to store. Therefore, using symmetric-key encryption can save a lot of space and we can use it to store more validation parts. Besides, symmetric-key encryption is faster than asymmetric-key encryption. Since path validation is a part of packet, even a little delay per packet would cumulatively cause a large end-to-end delay. Using symmetric-key can simplify lots of computation like modulo or power, which both need a large amount of computation ability.

A major design challenge raised by symmetric-key encryption is the number of key pairs. Unlike asymmetric-key encryption using public key, to identify each other, every two routers have to share a pair of secret key. A number n of routers require a number $O(n^2)$ of keys. In multi-path validation, there might be more routers and now calculating and deriving keys would be much more complex. Our solution is to pre-compute as much proof related information as possible. Before packets start to transmit, routers have already identified each other and there is no need to derive the keys twice. Routers can thus store them locally rather than put them in the header.

Segmentation. As aforementioned, if we directly use a single-path validation algorithm in multi-path situation, we would face a challenge that the source does not know the entire path to the destination before hand. Since there exists lots of possibility, calculating all of them is impossible. Thus, we use a method of segmentation. Specifically, we segment the forwarding path into segments of

three routers following a sliding window with size one. Consider, for example, an n-hop forwarding path $(r_1, r_2, ..., r_n)$. The segments ProvMPV handles are (r_1, r_2, r_3), (r_2, r_3, r_4), (r_3, r_4, r_5), and so on. The key idea is that we can require r_1 to leave a proof to its next hop's next hop, that is, r_3. If r_3 can successfully verify the expected proof from r_1, it demonstrates that the intermediary r_2 forwards packets correctly without any misbehavior. In multi-path routing, this greatly optimizes the computation and packet size. A packet no longer needs to carry all proofs of routers along all paths. Instead, it iteratively updates its carried proofs that correspond to only three hops.

How to choose the length of each segment is the key. It is obvious that the more routers a segment contains, the higher security level we get. The space complexity and difficulty of calculation, however, grow with the length. So what we should consider is its lower bound. Suppose that each segment only has two routers, then a collusion attack is easy to launch as adjacent malicious routers can fake the verification and transfer it to the next. If there are three nodes in a segment, even if a collusion attack occurs, the next segment would easily spot the attack and discard the packet. Our scheme would therefore use three as a balance of efficiency and security.

3.2 Encryption Construction

We now present our encryption scheme. Using AES-256, routers encrypt and decrypt the proofs to accomplish the verification with high security-level guaranteed. Using specific strategy, which we would introduce in Sect. 4.4, routers can determine its next hop, but it cannot control the second hop. Our solution is simple but effective: we use aggregate MAC to all the possibilities together so that the next hop can compute its next hop's signature and search it in the set to verify whether it can match.

We propose a new symmetric-key encryption scheme. It has three parts. The first part is used to record the sequence of identifier. The second part stores the information of the previous node, the current node and the next node. The third part stores the information of the node before last, the last node and the current node. Once decrypting the first part, router can use the sequence to verify the path order. And the portion of it would be used to decrypt the third part to accomplish path validation. If verification is passed, the second part would be updated to contain the downstream node's information. In this way, we guarantee the upstream nodes' verification and make sure the next hop is in the right path.

4 Design

In this section, we detail the design of ProMPV.

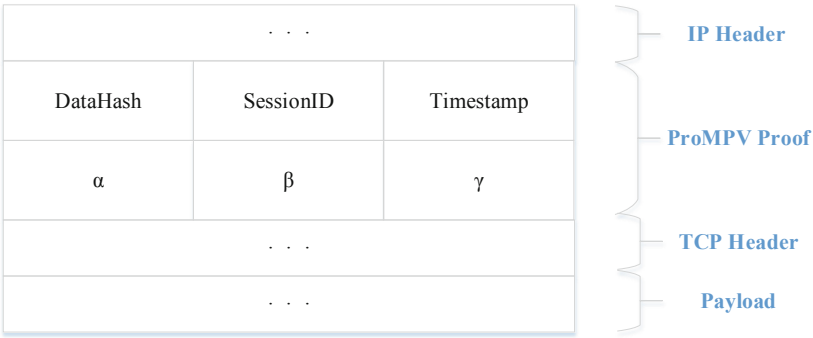

Fig. 1. ProMPV header architecture. DataHash, SessionID and Timestamp are traditional fields while ProMPV proof is designed uniquely. All the four parts are initialized by the source and only the fourth part need to be updated by the downstream routers.

4.1 Header Architecture

As shown in Fig. 1, ProMPV introduces its proof fields between the IP header and the TCP header. It contains four parts: DataHash, SessionID, Timestamp and ProMPV proof. The first three parts are regular parts of a verification scheme and the last one is unique as it is the core of our scheme. All these four fields are initialized by the source. The following routers only need to update the ProMPV proof.

DataHash. DataHash is the hash value of a given package's payload, written as $H(P)$. It is used to guarantee the integrity of packet. Hash function is shared with all the nodes. If attackers do not get the right function, they cannot forge a fake $H(P)$ which can pass the verification. However, if attacker is a malicious router in the path, then it can compute a valid $H(P)$ with changed payload, so simply put $H(P)$ in the packet header is not enough, DataHash must take part in the verification computation. When path validation begins to work, nodes calculate $H(P)$ independently and compare the result with DataHash stored in the header. If the result fits, packet's integrity is confirmed, otherwise, the packet is altered and router refuse and drop the packet.

SessionID. For every session, the source and the destination choose a SessionID to identify. It contains the hash value of nodes' ID and session initialization time. Routers generate symmetric keys according to it. And SessionID should be put into the computation. Otherwise we may suffer from path deviation attack. Suppose there are two node sequences: (r_1, r_2, r_3) and (r_1, r_2, r_4). r_2 is a malicious router and it transfer packet to r_4 rather than r_3. If we do not contain SessionID in the computation, r_4 can forge the proof and deceive others. With calculating SessionID, we guarantee only specific routers can take part in the process.

Timestamp. Timestamp records the time when the packet is created. As a part of computation, it is used to defend against replay attack. Without it, malicious router can copy an out-of-style message and send it to other routers to bypass the validation.

Table 1. Symbols and notations used in the section

P	The packet's payload
t_σ	The session initialization time
DataHash	The hash value of payload
SessionID	The hash value of nodes' ID and session initialization time
Timestamp	The time when packet is created
R_i	The ith router
ID_i	The ith router's ID
$K_{i,j}(= K_{j,i})$	The symmetric key shared between router R_i and R_i
Queue	Used to store the id of three routers
$Enc_{K_{i,j}}$	Encrypt the content d using symmetric key $K_{i,j}$
$Dec_{K_{i,j}}$	Decrypt the content d using symmetric key $K_{i,j}$
$H(d)$	Compute the hash value of content d using SHA-256

ProMPV Proof. As shown in Fig. 1, ProMPV proof consists of three parts: proof α, proof β and proof γ. We compute them according to the method proposed in Sect. 3.2. The computation needs the information of packet itself like DataHash, SessionID and router itself like router's ID and symmetric keys. Algorithm 1 describes the main working process of our scheme. And details are put in Sects. 4.2 and 4.3. Table 1 recaps the preceding fields and defines symbols to be used in later design details.

4.2 Proof Construction

Initialization. In this phase, routers generate and exchange symmetric keys with each other. Nodes store corresponding keys in a local table. And every router records its adjacent routers to form a routing table. In this way, they can confirm who is their next hop. Meanwhile, source generates DataHash, SessionID and Timestamp and enclose it into the header.

Construction on the Source. When a packet is created, the ProMPV proof field is empty, and source need to fill all three parts: proof α, proof β and proof γ.

Proof α: It is designed to store the information of a group of three continuous nodes: the one before last R_{i-2}, the previous one R_{i-1} and the current one R_i. Routers encrypt their ID using shared symmetric key between current router and its next router. Since source doesn't have a previous router, this field only contains the source itself.

Proof β: It is a set of all the possible path's information. The interval of it is the previous node R_{i-1}, current node R_i and next node R_{i+1}. When the previous node decide the current node according to the specific strategy, then it would traverse all possible next nodes, calculate their signature according to

Algorithm 1 and store them together. Since source do not have a previous node, the key used for encryption is shared between source and its next node.

Proof γ: Its content is similar with proof β, the only difference is that the interval of it is the node before last R_{i-2}, the last node R_{i-1} and the current node R_i. What's more, it is not computed but inherits Proof β before it updates. Thus, when source creates a packet, the content of proof γ is empty.

Algorithm 1: ProMPV Path Validation

Function `Source Initialization`
  DataHash ← H(P)
  SessionID ← $H(t_\sigma||ID)$
  Timestamp ← current time
Function `Update for General` R_i `in Path`
  HASH_Values ← empty **if** $ID_i == ID_{Destination}$ **then**
    End function
  Queue.dequeue()
  Queue.enqueue(ID_i)
  Proof $\alpha \leftarrow Enc_{K_{i,i+1}}(Queue)$
  Proof $\beta \leftarrow$ proof γ
  if $ID_i == ID_{Second-to-LastRouter}$ **then**
    Forward the packet to Destination
    End function
  for *each next Router j=i+2* ***from*** R_i ***to*** $R_{Destination}$ **do**
    HASH_Value $=H(DataHash||SessionID||Timestamp||ID_j)$
    Proof γ += cipher_$Set(Enc_{K_{i,j}}$(HASH_Values))
  Forward the packet to next Router
  Accept the packet and update Proofs
Function `Verification for General` R_i `in Path`
  if $ID_i == ID_{Source}$ **then**
    End function
  Queue ← $Dec_{K_{i-1,i}}$(Proof α) **if** $Queue[R_{i-1}] \neq ID_{i-1}$ **then**
    Drop the packet
  if $ID_i \neq ID_{SecondRouter}$ **then**
    Accept the packet and update Proofs
    End function
  if $HASH\_Values \neq Dec_{K_{i-2,i}}(Proof\ \beta)$ **then**
    Drop the packet
  if $H(DataHash||SessionID||Timestamp||ID_i)$ *NOT in HASH_Values* **then**
    Drop the packet.
  Accept the packet and update Proofs

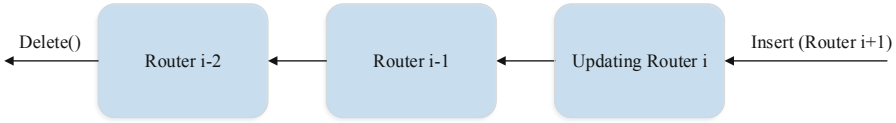

Fig. 2. Update process of proof α

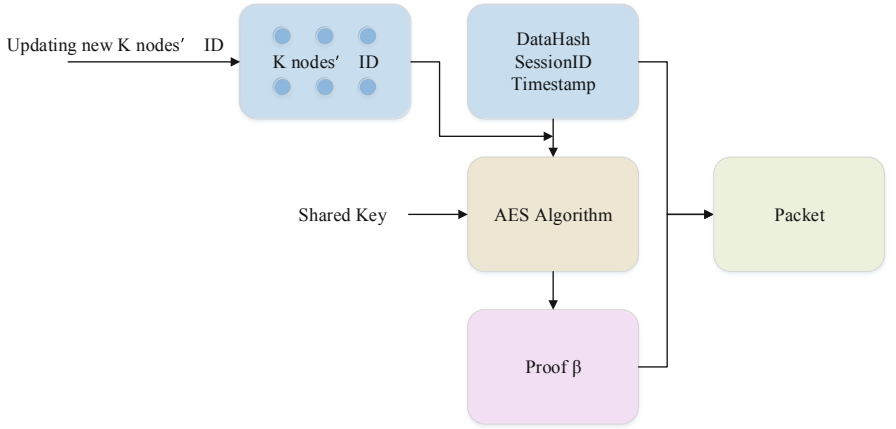

Fig. 3. Update process of proof β

Construction on Intermediate Routers. Intermediate routers only need to update ProMPV proof field. The other three field as DataHash, SessionID and Timestamp are remained, they are used to verify packet's confidentiality, integrity, authentication and non-repudiation.

Proof α: We use a queue to maintain the update of routers' id as Fig. 2 shows. The characteristic of queue is "first in first out", just like the traveling order. Current node push its next router into the queue and pop the node before last. Then current router encrypts the message with key shared with the next router. In this way, the update is guaranteed.

Proof β: The current router first clears the field. Then it travels the routing table and gets the possible nodes after next. Suppose its number is k, hash function would be used to compute k groups of hash values, after that they would be encrypted with corresponding keys each by each. Figure 3 shows the entire process.

Proof γ: When the current router R_i determines its next router R_{i+1}, proof γ inherits proof β before it updates.

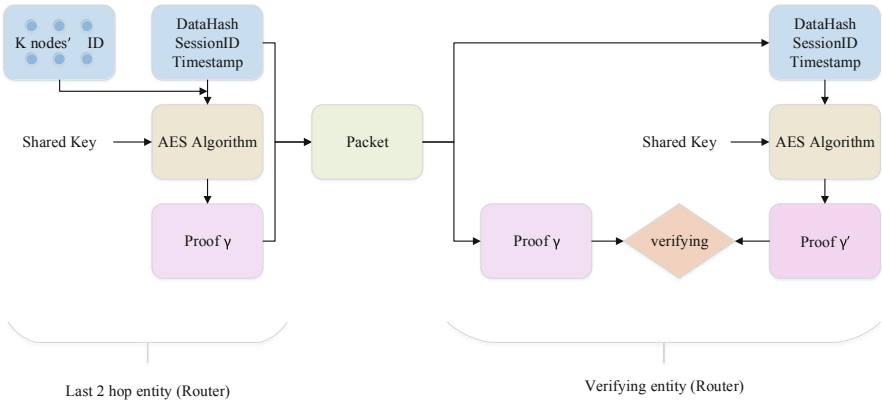

Fig. 4. Verification process of proof γ

4.3 Proof Verification

As aforementioned, DataHash, SessionID and Timestamp are used to defend attack, not directly take part in the validation. So the main task is to verify the ProMPV proof field. If all things can match, then router accepts the packet and transfers it to the next hop, otherwise, it drops the packet.

Proof α: As all the symmetric keys are stored in the local area, router R_i can decrypt the message using the key $K_{i-1,i}$. The decrypted message is a sequence of three nodes: R_{i-2}, R_{i-1} and R_i. Routers can judge whether the packet passes through the right path by this way. And the router id R_{i-2} would take part in the verification of proof γ.

Proof β: Because proof β is a kind of message authentication code provided to the next hop from the previous hop, it doesn't need to be checked when the current hop is being verified.

Proof γ*:* As Fig. 4 shows, after verifying the proof α, router R_i has already known the router before last R_{i-2}, so that it can find the corresponding symmetric key $K_{i-2,i}$ from local search. Router would link DataHash, SessionID, Timestamp and router id, calculate its hash value using the shared hash function and encrypt it to get a signal. Since proof γ is set of such signals, the only thing router needs to do is search from it to see whether there exists a signal fits each other. If there exists one, the verification is done, or it fails.

4.4 Routing Strategy

From preamble, we can find that the another core design skill of our scheme is the choose of routing strategy, since proof α and proof γ both depend on selecting a next hop first. Besides, a good strategy can greatly improve the efficiency.

According to our ProMPV algorithm, the routing strategy we need to use should be a dynamic, distributed router selection algorithm. Multi-path routing can be controlled by a specific algorithm to ensure that multi-path routing is generated without closed loop [8], and the next hop path node is selected through various parameters, such as multi-path acyclic algorithm. Each network node distinguishes other network nodes into a forward set and a backward set. The element in the forward set indicates that the element is one of the potential routing nodes. If one node B is a forward set's element of the other node A, the node A is also a backward set's element of the node B. As long as the correctness of the forward set and backward set elements is maintained, it is ensured that each time the data packet is forwarded to the elements in the forward set, the routing path is guaranteed to be loop-free. In addition, through specific parameters such as delay, bandwidth utilization, throughput, etc., or a certain policy given by the Internet service provider, such as bypassing certain regions or countries, forcing routing through specific nodes, etc., it can select specific output node in the forward set to meet the advantages of reliability and throughput.

5 Security

It is proved here that when both the source and the receiver are trusted, this multi-path route has the attributes of source authenticity and path verification. This attribute applies to any network configuration, including the configurations where networks contain malicious entity nodes.

Packet Alteration. When the malicious node R_i changes the content of data packet, such as modifying the data packet payload and DataHash, the malicious node does not know the shared key between the previous node R_{i-1} and the next node R_{i+1}. So the forgery of Proof β is not achievable and attacker can only forge Proof γ. When the packet is forwarded to the next entity node R_{i+1}, R_{i+1} uses the forged DataHash to verify the Proof β which cannot be forged. Then it can detect the packet exception, thus resisting Packet Alteration attack.

Packet Injection. Data injection attacks for general positions are easily to be detected due to the presence of hash operation and shared key encryption. So it is difficult to forge or attack. A valid packet injection operation can be replaying a previously packet header and injecting it into the current data packet. However, this replay attack can be protected by determining if the current Timestamp is correct [4].

Packet Deviation. In multi-path segmentation routing, the path deviation attack is actually equivalent to attacking the segment route. Since the next hop(R_{i+1}) is determined, the current router(R_i) will calculate a set of routers for the one after the next hop(R_{i+2}) based on the information of the next hop router(R_{i+1}). After a path offset attack, the next router that arrives at the packet will tell if the updated next hop router is in the pre-computed router set. Therefore, under the condition that each segment route is correct, the validation

of validator Proof α, Proof β of each packet can effectively resist path offset attacks.

DoS Attack. In this segmented multi-path routing, each node only needs to keep a fixed number of keys, and it has strong defense against memory attacks. However, the defense capability in computing is weak. Due to the use of a large quantity of encryption and decryption operations, DoS attacks can exhaust the computation power of the victim node by sending a large number of packets.

Collusion. In this segmented multi-path routing, the source authentication and path verification depend on whether the router works as it is, that is, the data packet is normally forwarded according to the protocol. When there are no two consecutive malicious routers in the path, the segmented multi-path route can detect attack and thus resists collusion attacks. However, when there are two consecutive malicious routers on the path, they can forge at the same time, so that the data such as DataHash, SessionID, Timestamp, Proof β, and Proof γ are malicious but consistent. At this time, the downstream routers cannot know that the data packet has been modified, then continue to forward the packets. This situation can be improved by increasing the length of the segment path, but at the meantime it increases the time and space overhead.

6 Implementation

We use OpenSSL [1], which can be concluded by three main functions: SSL protocol library, application, and cryptographic algorithm library. In this experiment, we call correlated function in the cryptographic algorithm library, including the symmetric encryption algorithm AES with 128-bits, 192-bits, 256-bits key and the information digest hash algorithm SHA.

The key sequence and network topology are randomly generated at initialization phase and the relevant hash values are initialized using EVP_Digest with EVP_sha256 being the parameter. Operations for each node (including the source and destination) are divided into two functions: Verification and Update. In the Verification, the decryption function EVP_DecryptInit_ex, EVP_DecryptUpdate, EVP_DecryptFinal_ex in the OpenSSL library are used to decrypt the AES keys of different digits. Similarly, in the Update, the symmetric encryption function EVP_EncryptInit_ex, EVP_EncryptUpdate and EVP_EncryptFinal_ex are able to handle AES encryption operations under different security level.

7 Evaluation

In this section, we compare ProMPV with ICING. Due to the calculation requirement, the experiment run on a 4 core cloud server with Intel Xeon Skylake 6146 (3.2 Hz) and 8 GB memory provided by Tencent. In order to obtain reliable data, all reported statics are average over 10000 runs.

The two main parameters that influence the experiment are total length of the path and security level. According to Figs. 5 and 6, the choice of path length does

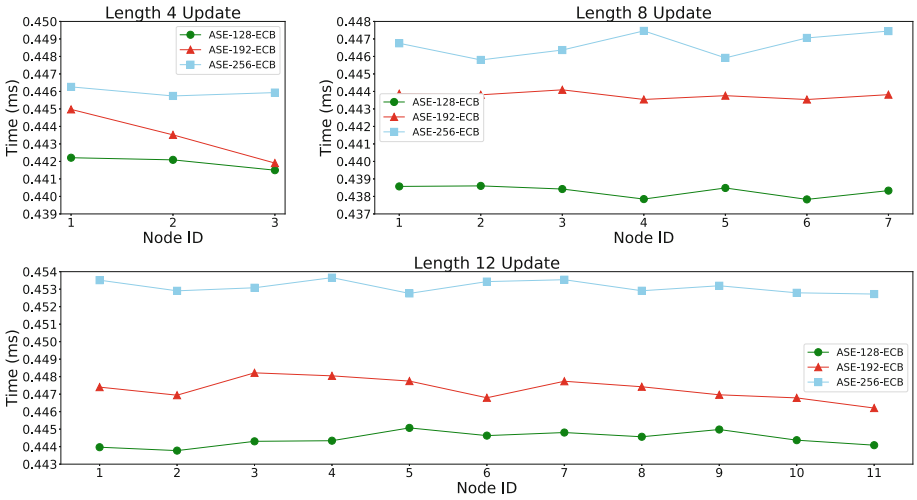

Fig. 5. Proof update time with different security level and path length.

not have much effect on time overhead, because ProMPV cares mainly about segment length. On the other hand, the higher the security level of the encryption algorithm is, the more the method costs. Therefore, ProMPV is flexible for users to determine segment length and security level basing on their own needs. To balance safeness and cost, users can use higher security level or enhance segment length to improve security, or use lower security level or reduce segment length to cut down cost.

As is shown, the time required for the update is much greater than the verification, since the verification only needs to verify Proof α, Proof γ, and the update requires Proof α, Proof β and Proof γ. In addition, Fig. 5 has only $n-1$ nodes, because the last node's updating time, which only needs to update the Proof α and Proof γ, is much smaller than others and doesn't have a representative meaning.

7.1 Time Overhead

As expected, the time overhead is not related to total length of the path. Also, time cost and node ID are not positively correlated. The controllable factor that influence time overhead is the security level of encryption algorithm, higher security level means more calculation in the process which costs more time.

In the segmented multi-path environment, the update of proof is only related to the number of connected nodes of each node so lines are smooth, while ICING's processing time is proportional to the path length. Since ICING uses cache, the processing time improves dramatically, for creating is $2.6x + 40.1$ µs, and for verifying is $2.6x + 24.4$ µs, where x is the path length. Given a 12 nodes path, the creation time and verification time are 71.3 µs and 55.6 µs respectively. Our

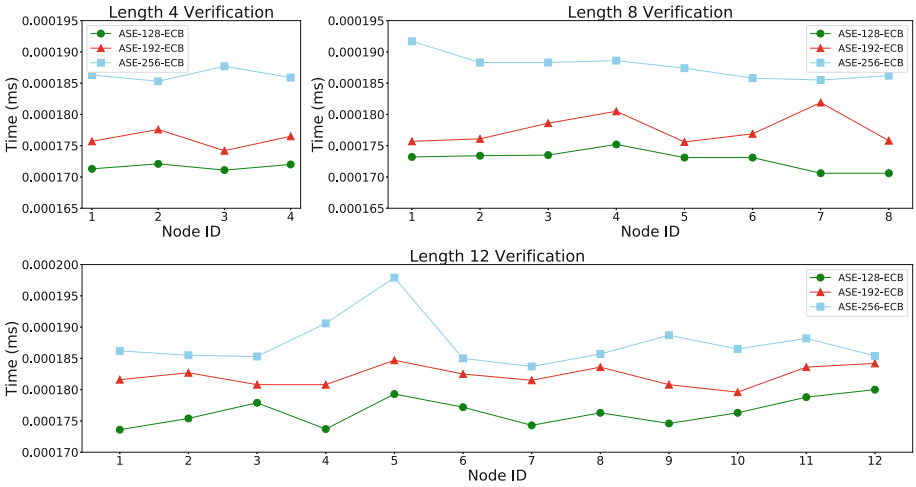

Fig. 6. Proof verify time with different security level and path length.

Table 2. Proof β (or γ) size in 80-bit security level.

Node number	1	2	3	4	5	6	7	8	9	10
Proof β(or γ) size (byte)	10	20	30	40	50	60	70	80	90	100

solution uses the AES algorithm with 256-bit key. It is tested in fully-connected case that each node is connected to 15 other nodes. The time of creating and verifying are about 4.3×10^{-5} s and 5.12×10^{-3} s. And ProMPV can also use cache to accelerate speed and improve performance.

7.2 Space Overhead

As shown in Table 2, the size of proof β (or γ) is positively correlated to the number of connected nodes of each segment's next node, because ProMPV requires information out of the segment to verify the path. Therefore, total length of the path is hardly a factor of space overhead. Also, the security level decides space that each node takes.

Under same security level, the ProMPV header space increases slowly by a much smaller constant compared with ICING as the path grows. For instance, under an 80-bit security level, 10 bytes would add to the proof size of ProMPV for each node while the proof size of ICING increases by 42 bytes [5]. Therefore, our scheme saves header space and makes multi-path validation more feasible.

8 Conclusion

We design and achieve a new multi-path routing scheme, satisfying path validation and source authentication. By cutting entire path to small segments,

we overcome the weakness of single-path routing. If we can guarantee every segment's safety, the reliability of path validation scheme can be guaranteed. Besides, the working time doesn't relate to path length as segment method is used but relates to the number of nodes its next node connects with. With specific routing strategy, we can limit the proof size and accelerate speed. And by changing the security level or the segment length, we can sacrifice security to get speed or sacrifice speed to get security. Both theory and experiment tell our scheme's superiority. For future work, we plan to achieve hardware acceleration on it to improve the computation speed without using more computation power.

Acknowledgement. This work is supported by The Natural Science Foundation of Zhejiang Province under Grant No. LY19F020050. We would also like to thank Professor Kai Bu for mentoring us on the project.

References

1. OpenSSL: Cryptography and SSL/TLS Toolkit. https://www.openssl.org/
2. Bu, K., Yang, Y., Laird, A., Luo, J., Li, Y., Ren, K.: What's (not) validating network paths: a survey. arXiv:1804.03385 (2018)
3. He, J., Rexford, J.: Toward internet-wide multipath routing. IEEE Netw. **22**(2), 16–21 (2008)
4. Lee, T., Pappas, C., Perrig, A., Gligor, V., Hu, Y.C.: The case for in-network replay suppression. In: ACM AsiaCCS, pp. 862–873 (2017)
5. Naous, J., Walfish, M., Nicolosi, A., Mazières, D., Miller, M., Seehra, A.: Verifying and enforcing network paths with ICING. In: CoNEXT (2011)
6. Paxson, V.: End-to-end routing behavior in the internet. In: ACM SIGCOMM, pp. 25–38 (1996)
7. Segall, A.: Optimal distributed routing for virtual line-switched data networks. IEEE Trans. Commun. **27**, 201–209 (1979)
8. Singh, R., Singh, Y., Yadav, A.: Loop free multipath routing algorithm, January 2016
9. Villamizar, C.: OSPF optimized multipath (OSPF-OMP), September 2019
10. Wu, B., et al.: Enabling efficient source and path verification via probabilistic packet marking. In: IWQoS (2018)
11. Xu, W., Rexford, J.: Miro: multi-path interdomain routing. In: ACM SIGCOMM (2006)
12. Yang, X., Wetherall, D.: Source selectable path diversity via routing deflections. ACM SIGCOMM **36**, 159–170 (2006)