# Preventing "Bad" Content Dispersal in Named Data Networking

Yi Wang
Huawei Future Network Theory Lab
Hong Kong
wy@ieee.org

Zhuyun Qi
Shenzhen Key Lab for Cloud Computing Technology & Applications (SPCCTA), School of Electronics and Computer Engineering, Peking University, Shenzhen, China 518055
qizy@pkusz.edu.cn

Kai Lei*
Shenzhen Key Lab for Cloud Computing Technology & Applications (SPCCTA), School of Electronics and Computer Engineering, Peking University, Shenzhen, China 518055
leik@pkusz.edu.cn

Bin Liu
Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University
Beijing, China 100084
liub@tsinghua.edu.cn

Chen Tian
State Key Laboratory for Novel Software Technology, Nanjing University
Najing, China 210023
tianchen@nju.edu.cn

## ABSTRACT

Named Data Networking (NDN) improves the data delivery efficiency by caching contents in routers. To prevent corrupted and faked contents be spread in the network, NDN routers should verify the digital signature of each published content. Since the verification scheme in NDN applies the asymmetric encryption algorithm to sign contents, the speed of content verification is too slow to satisfy the high speed requirement. In this paper, we propose two schemes to improve the verification performance of NDN routers to prevent content poisoning. The first content verification scheme, called "user-assisted", leads to the best performance, but can be bypassed if the clients and the content producer collude. To prevent the aforementioned collusion attack, we improve the user-assisted content verification scheme and propose the second content verification scheme, named "Router-Cooperation", in which the edge routers verify the contents independently without the assistance of users and the core routers no longer verify the contents. The Router-Cooperation verification scheme reduces the computing complexity of cryptographic operation by replacing the asymmetric encryption algorithm with symmetric encryption algorithm. The simulation results demonstrate that this Router-Cooperation scheme can speed up 145.5× (in hardware) and 18.85× (in software) of the original content verification scheme with merely extra transmission overhead.

*Corresponding author: Kai Lei, leik@pkusz.edu.cn.


*ACM TUR-C '17, May 12-14, 2017, Shanghai, China*
 ISBN 978-1-4503-4873-7/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3063955.3063993

## CCS CONCEPTS

•**Networks** → *Packet scheduling; Network experimentation;*

## KEYWORDS

Named Data Networking, Router, Content Verification, Encryption Algorithm

## 1 INTRODUCTION

In the conventional IP network, security is a function of the connection between hosts. By contrast, Named Data Networking [30] (NDN) incorporates security into data itself by forcing the content providers to sign every content with its private key. The signature of a content, implying the data provenance, allows the security of this content to be decoupled from where and how the data is obtained.

In theory, content signatures provide an effective and simple means to detect content poisoning attacks, since "bad" content can be easily identified via signature verification. In other words, NDN should be immune to content poisoning attacks which includes two means: corrupted content, i.e., the content cannot conform its signature; and faked content, i.e., the private key of the signature is forged. However, this assertion might not hold in practice. Though a consumer can afford to verify all content signatures, NDN routers, caching contents to improve network performance, face two challenges: (1) signature verification overhead; and (2) public key management. An effective public key cryptography has been considered by Lixia Zhang et al. [29]. This paper focuses on reducing the overhead of signature verification.

### 1.1 Problem Statement

When a router receives a Data packet, as described in the NDN proposal [30], the router should systematically verify the content signature to avoid content poisoning attacks before forwarding and caching the content. The signature binds the content with its name,

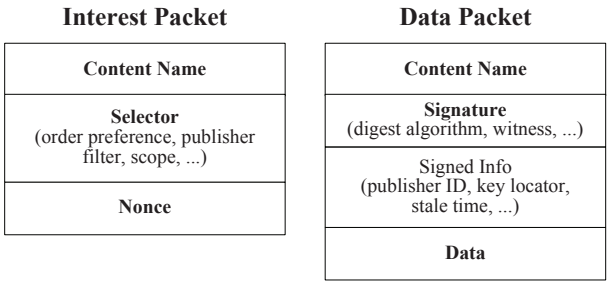


**Figure 1: Two kinds of packet in NDN.**

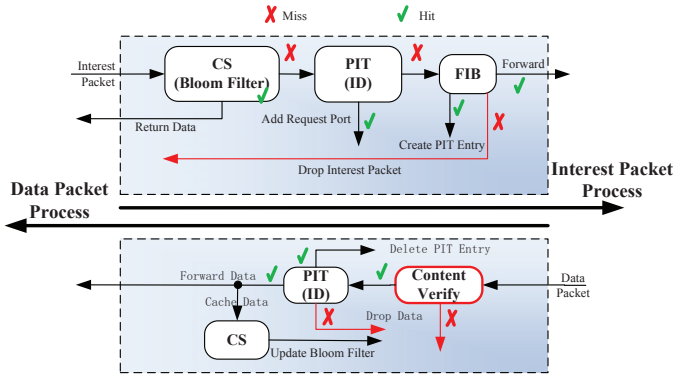


**Figure 2: Packets forwarding processes in NDN.**

and provides original authentication no matter how, when or from where the content is retrieved. However, for NDN routers, the computations of content verification are too expensive to be executed against all the incoming Data packets. Previous work [8] showed that an optimized software implementation of RSA1024 signature verification running on Intel Core 2 Duo 2.53 GHz CPU allows a router to verify about 150 Mbps of traffic, assuming 1,500 Bytes per content packet, or even worse with smaller-sized packets. NDN routers with multiple Gigabit-speed interfaces would need an unrealistic amount of computing power to verify signatures of Data packets at wire speed.

*Participants:* 1) *End users* request content by generating Interests; 2) *Content Providers* produce content by generating data in response to Interests; 3) *Network Devices* forward Interest and Data packets.

*Assumptions:* Routers systematically verify the signatures of the contents that they receive. Routers cache verified contents which can be directly response to the requires from users. Routers in the network are trustworthy and can identify each other. Users are trustless. And some users are the attackers or adversaries.

*Attack:* Adversaries disguise themselves as content providers to produce corrupted and faked contents to routers and users.

*Impact:* By fully exploiting the expensive computations of content verification, an adversary can attack the router and lead it to be out-of-service ultimately [12]. As a result, NDN routers sample a small set of received Data packets to verify their signatures, and leave a loophole which can be employed by adversaries to fill the caches of NDN routers with corrupted or faked contents.

*Countermeasures:* Routers can improve the performance of content verification.

*Risk:* High. Corrupted and faked contents will be spread in the network when the routers cannot verify all contents.

## 1.2 Our Contribution

To prevent content poisoning in NDN, in this paper, we propose two verification schemes to improve the content verification performance:

(1) The user-assisted content verification scheme (described in Section 3) no longer verifies the content itself, instead it verifies the content provider to guarantee the correctness. By bypassing the content verification in NDN routers, this scheme can achieve perfect performance. But this scheme is under the threat of a scenario that customers and attackers collude in requesting and providing poison contents.

(2) To prevent the aforementioned collusion attack, we improve the user-assisted content verification scheme and propose the Router-Cooperation content verification scheme. The Router-Cooperation content verification scheme (described in Section 4) lets the edge routers verify the content provider independently without the assistance of users. By replacing the asymmetric encryption operation with symmetric encryption operation, this Router-Cooperation scheme can effectively reduce the consumption of computing resource for cryptographic operation and improve the content verification performance. The simulation results demonstrate that this Router-Cooperation scheme can speed up 18.85× of the original content verification scheme used in NDN [30].

The rest paper is organized as follows. Section 2 introduces the packet categories and the original content verification scheme in NDN. The user-assisted content verification scheme and the Router-Cooperation content verification scheme are described in Section 3 and Section 4, respectively. Then we present the simulation results in Section 5. After reviewing the related literatures on NDN security in Section 6, we conclude our work in Section 7.

# 2 BACKGROUND: CONTENT VERIFICATION MECHANISM IN NDN

## 2.1 Packet categories in NDN

NDN is a request-driven networking architecture. Different with the sole IP packet in the conventional IP network, packets in NDN have two categories: *Interest* packet and *Data* packet. Interest packets are used to request contents, and Data packets are used to carry contents back. As depicted in Figure 1, an Interest packet consists of the required content name, a selector, and a nonce; a Data packet carries the required data with the name, the signature and the signed information. When a data is returned to the client, the client will verify the signature to confirm that whether this content is the exact one.

## 2.2 Packets forwarding processes in NDN

The forwarding processes in a router's data plane for these two kind of packets are different, as shown in Figure 2.

**Interest Packet Forwarding:** 1) When a router receives an Interest packet, it first searches (exactly match) the content name in the Interest packet header against the Content Store (CS). If hit, this means the router has cached the requested content, then a Data packet with the required content is returned; 2) If not hit, the router looks up the content name in the Pending Interest Table (PIT). If matched, this means this router has received Interest packet(s) requesting the same content, but not responded yet. Then, the arrival interface is appended to the PIT entry, and this Interest packet is discarded and would not be forwarded to upper routers; 3) If not matched in the PIT, the router looks up the content name against the Forwarding Information Base (FIB) complying with the Longest Prefix Matching. If this name is found in the FIB, the router forwards this Interest packet to the corresponding next-hop interface, and creates associative PIT entry; otherwise, discards it.

**Data Packet Forwarding:** 1) When a Data packet arrives at a router, the router must verify the content at first. If the signature in the packet is forged, the packet will be dropped. 2) After verifying the content, the router looks up the content name in the Data packet against PIT based on exact match [9, 10]. If matched, forward Data packet to the interfaces in the matched PIT entry; otherwise, discard this Data packet.

## 2.3 Content verification mechanism

To prevent polluted data from being spread out in the network, NDN routers should verify every content sent to the networks by the content providers or other clients. Figure 3 illustrates the content verification process in NDN.

(1) When the router receives the Data packet, it parses the Data packet and extracts the content $D'$, the signature $K_{pri}(Hash(D))$, and other information;
(2) The router calculates the hash value $Hash(D')$ of the content $D'$ by applying the message-digest algorithm (e.g., MD5, CRC32);
(3) The router uses the content provider's public key to decrypt the signature of this content to obtain the original message-digest $K_{pub}(K_{pri}(Hash(D)))$;
(4) The router compares the message-digest $Hash(D')$ of the sent content with the original message-digest $K_{pub}(K_{pri}(Hash(D)))$ of the required content, if these two message-digests are equal, this Data packet will be sent upstream; otherwise, it will be dropped.

The content verification mechanism in NDN can work well in the low speed networks, but it cannot work in the high speed networks due to its poor verification performance.

# 3 THE USER-ASSISTED CONTENT VERIFICATION SCHEME

As described in Section 2.3, the computational complexity of the content verification is too high for an NDN router to handle wire speed traffic. Therefore, in this section, we propose the user-assisted

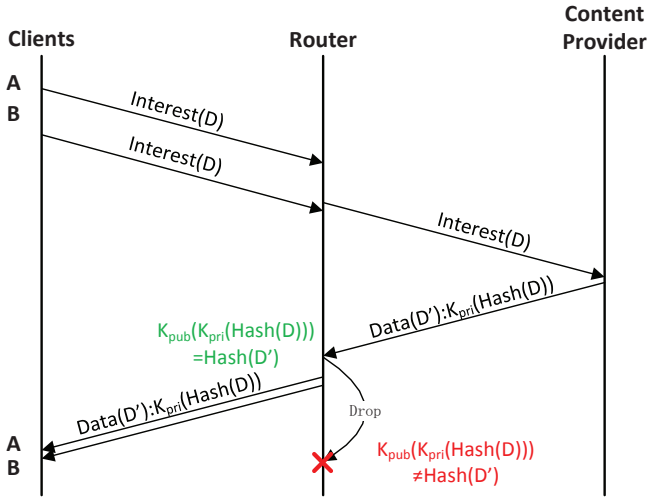


**Figure 3: The content verification process in NDN.**

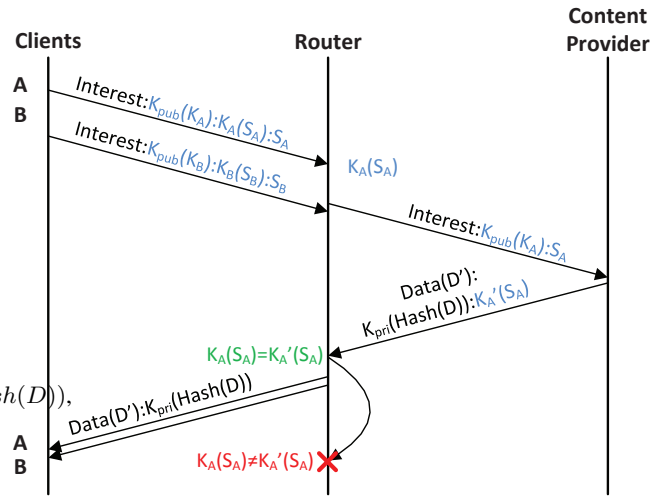


**Figure 4: The content provider verification process.**

content verification scheme to improve the performance by bypassing the content verification in NDN routers. Similar to the client verification in current IP network, the user-assisted content verification scheme no longer verifies the content itself, but it verifies the content provider to guarantee the correctness.

The process of the user-assisted content verification, demonstrated in Figure 4, has 4 steps.

(1) The Interest packet, sent by the user $A$, carries 3 extra data to help routers verify the content provider:
   (a) $K_{pub}(K_A)$. $K_A$ is the temporary symmetric key for the content provider to encrypt the verification message; $K_{pub}$ is the public key of the content provider; and $K_{pub}(K_A)$, the ciphertext of encrypting $K_A$ with $K_{pub}$, can only be decrypted by the content provider with its private key.

(b) $S_A$, a token, is a 64-bit random integer.
(c) $K_A(S_A)$, the ciphertext of encrypting $S_A$ with $K_A$, is used to verify whether the content provider can obtain $K_A$ correctly by decrypting $K_{pub}(K_A)$.

(2) When the edge router receives this Interest packet, it extracts $K_A(S_A)$ from the Interest packet before forwarding it to the content provider. The rest two data $K_{pub}(K_A)$ and $S_A$ will be still forwarded.

(3) The content provider decrypts $K_{pub}(K_A)$ with its private key $K_{pri}$ to obtain $K_A^{'}$=$K_{pri}(K_{pub}(K_A))$ when it receives the Interest packet. Then the content provider encrypts $S_A$ with $K_A^{'}$ and returns the ciphertext $K_A^{'}(S_A)$ in Data packet to the edge router.

(4) After receiving the Data packet, the edge router extracts $K_A^{'}(S_A)$ and compares it with $K_A(S_A)$ to validate the content provider. If $K_A^{'}(S_A)$=$K_A(S_A)$, it means the content provider is the exact one that publishes the required content, and then the edge router will remove $K_A^{'}(S_A)$ from this Data packet and forwards it upgrade according to the reverse path that the Interest packet forwarded; otherwise, the content provider is a fake, hence this Data packet will be dropped.

For the Interest packets with the same required content name, routers in NDN can merge these Interest packets to one, regardless of whether these Interest packets have different tokens and different temporary symmetric keys. For example, in Figure 4, user $A$ and user $B$ send the Interest packets with ($S_A$, $K_A$) and ($S_B$, $K_B$), respectively. These two Interest packet are converged in the edge router, and only one of them is sent to the content provider.

Note that any in-path router can return the Data packet without $K_A^{'}(S_A)$ whenever it caches the required content, since routers trust each other and contents cached in routers have been verified.

Meanwhile, in the user-assisted scheme, routers must identify the roles of the upstream nodes. If the upstream node is a router, the token $K_A(S_A)$ will be forwarded to it; otherwise, the token $K_A(S_A)$ will be removed from the Interest packet.

The correct Data packet is sent back to all the users that required this content. Meanwhile, the edge router caches the content to fast reply the same request and reduce the traffic.

# 4 THE ROUTER-COOPERATION CONTENT VERIFICATION SCHEME

In the process of user-assisted content verification scheme, the calculation of a content, including the hash calculation of the content and the decryption of its signature, is eliminated since routers verify content providers to guarantee the correctness of their contents. Consequently, the router can achieve wire speed content verification with fewer resources and performance overhead.

The user-assisted content verification scheme, however, is vulnerable. The malicious user and the counterfeit content provider can collude on the temporary symmetric key $K_A$ and the token $S_A$ to deceive the edge router and publish polluted contents. First, the malicious client sends an Interest packet with $K_{pub}(K_A)$, $K_A(S_A)$, and $S_A$ as usual. When the counterfeit content provider receives this Interest packet, it sends the Data packet carrying the polluted content and $K_A(S_A)$ to the edge router directly without decrypting $K_{pub}(K_A)$. The edge router cannot detect this polluted content as $K_A(S_A)$ extracted from the Interest packet is equal to $K_A^{'}(S_A)$ extracted from the Data packet. As a result, the counterfeit content provider can publish any polluted content in the network.

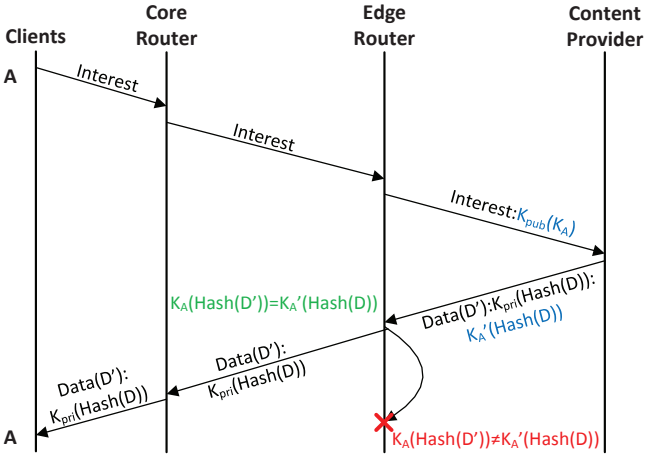


**Figure 5: The process of Router-Cooperation content verification.**

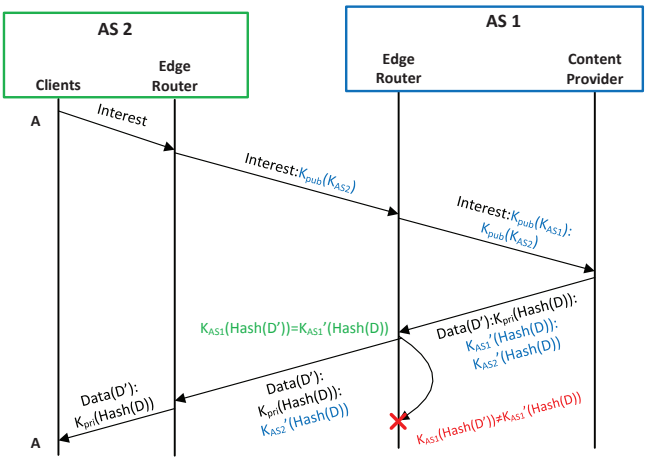


**Figure 6: The process of Router-Cooperation content verification crossing multiple mutual distrust ASes.**

## 4.1 Router-Cooperation Scheme in One AS

To prevent the aforementioned collusion attack, we improve the user-assisted content verification scheme and propose the Router-Cooperation content verification scheme, in which the edge routers verify the contents independently without the assistance of users and the core routers no longer verify the contents. As illustrated in Figure 5, the edge router produces the token $S_A$ and the temporary symmetric key $K_A$. The verification process works as follows:

(1) The client sends out the Interest packet to require the content.

(2) When the edge router receives this Interest packet, the edge router generates the symmetric key $K_A$ and then adds the ciphertext $K_{pub}(K_A)$ to this Interest packet.
(3) After receiving this Interest packet, the content provider first decrypts the ciphertext $K_{pub}(K_A)$ with its private key $K_{pri}$ to obtain $K_A^{'}$; then it hashes the content to get the message-digest $Hash(D)$; finally, it encrypts $Hash(D)$ with $K_A^{'}$, and appends $K_A^{'}(Hash(D))$ to the Data packet.
(4) When the edge router receives this Data packet, it hashes the content to get $Hash(D')$ and encrypts $Hash(D')$ with $K_A$ to get the verification message $K_A(Hash(D'))$. Then the router compares $K_A(Hash(D'))$ with $K_A^{'}(Hash(D))$ to verify the content provider and check the content's integrity. Only the content, passing this verification, will be cached and forwarded to upstream.
(5) The core routers directly forward the Data packets without verifying the contents any more since the all contents have been verified by the edge routers.

## 4.2 Router-Cooperation Scheme crossing multiple mutual distrust ASes

Data packets may pass through multiple ASes. The symmetric key $K_A$ can be shared in a group of mutual trust ASes, therefore one group uses one $K_A$, i.e., one $K_{pub}(K_A)$ is attached in the corresponding Interest packet. In the case of data packets crossing multiple mutual distrust ASes, each group needs one $K_{Ai}$. $N$ $K_{pubi}(K_{Ai})$ need to be added in the Interest packet, where $N$ is the number of groups of mutual distrust ASes. Figure 6 illustrates the packets forwarding process in two mutual distrust ASes.

## 4.3 Performance analysis

Compared to the user-assisted content verification scheme, the Router-Cooperation content verification scheme consumes more resources. In the path of Interest packets, an edge router in the Router-Cooperation content verification scheme has to produce the temporary symmetric key $K_A$ for every Interest packet that is forwarded by this edge router. Besides that, the edge router needs to encrypt $K_A$ with the provider's public key $K_{pub}$ to set up Router-Cooperation communication. Fortunately, $K_A$ and $K_{pub}(K_A)$ can be precalculated offline, and even more the edge router can replace allocating one temporary symmetric key $K_A$ for each Interest packet with allocating one symmetric key $K_A$ for each content provider that directly connects this edge router.

In the path of Data packets, an edge router in the Router-Cooperation content verification scheme needs to hash the content to get the message-digest $Hash(D')$ and encrypts this message-digest with the symmetric key $K_A$ to encrypt the verification message $K_A(Hash(D'))$. The verification message $K_A(Hash(D'))$ should be computed online, since the above calculation process is correlation with the content. In summary, the computing overhead of the Router-Cooperation content verification scheme consists of one hash calculation and one symmetric encryption computation. Compared with the original content verification scheme in NDN, the Router-Cooperation content verification scheme can effectively improve the verification performance since it replaces the asymmetric decryption computation with the symmetric encryption computation. Meanwhile, the verification message is only transferred once, i.e., from the content provider to the edge router. The edge router can directly forward the content to upstream routers without verification information ($K_A(Hash(D'))$) as routers trust each other and contents in the edge router have been verified. In the other cases, a router can directly response the Interest packet and sends back the Data packet without $K_A(Hash(D'))$ whenever the required content is cached in the router.

## 4.4 Preventing symmetric key attacks

In the Router-Cooperation scheme, the edge router can offline generate the temporary symmetric key $K_{pub}(K_A)$ for one content provider and reuse the key $K_{pub}(K_A)$ for all Interest packets that require the same provider's contents. An adversary may collect enough segments of $K_A(Hash(D'))$ to crack the symmetric key $K_A$ by listening the path from the edge router to the content provider. Therefore, we should regenerate the temporary symmetric key $K_B$ for the content provider after using the key $K_A$ after some time. Assuming a node, in-between the edge router and the content provider, caches the requested content with the key $K_A$. If the edge router uses the $K_A$ as the symmetric key, this copy will be valid; otherwise, this copy will be invalid, and the content from the original provider will be sent back. Since one content provider is only assigned one symmetric key $K_A$ over a period of time, the situation that the copy's key cannot match the original content's key happens in the interval of replacing the old key $K_A$ with the new key $K_B$.

# 5 SIMULATION RESULT

In this section, we evaluate the performance of the aforementioned 3 content verification schemes via simulation. The verification speed and the transfer overhead are the major metrics to evaluate the content verification schemes. In brief, the original content verification scheme is named as NDN-Verify; the user-assisted content verification scheme is named as User-Verify; and the Router-Cooperation content verification scheme is named as Router-Verify.

The basic performances [5] of the message-digest algorithms, the symmetric cryptographic algorithms, and the asymmetric cryptographic algorithms are listed in Table 1 and Table 2. All algorithms are coded in C++, compiled with Microsoft Visual C++ 2005 SP1, and run on an Intel Core 2 1.83 GHz processor under Windows Vista in 32-bit mode. X86/MMX/SSE2 assembly language routines are used for integer arithmetic and SHA-256. OpenMP [15] is disabled so that only one core of the CPU is used for this benchmark.

In the message-digest algorithms, CRC32, MD5 and SHA-256 can achieve 253 MB/s, 258 MB/s and 111 MB/s, respectively. Compared to SHA-256, MD5 is 2.32× faster. On the other hand, CRC32 produces a 32-bit hash value, MD5 produces a 128-bit hash value, and SHA-256 produces a 256-bit hash value. After considering both the performance and the application situation, in practice we choose MD5 as the message-digest algorithm.

DES algorithms uniformly use the CTR mode. The encryption speed of DES, DES-XEX3 and DES-EDE3 can achieve 32MB/s, 29MB/s and 13MB/s, respectively. Through overall consideration, we choose DES-XEX3 as the symmetric cryptographic algorithm.

**Table 1: The basic performances of different algorithms.**

| Algorithm | Lookup Speed | |
|---|---|---|
| | MByte per Second | Cycles per Second |
| CRC32 | 253 | 6.9 |
| MD5 | 258 | 6.8 |
| SHA-256 | 111 | 15.8 |
| DES/CTR | 32 | 54.7 |
| DES-XEX3/CTR | 29 | 60.6 |
| DES-EDE3/CTR | 13 | 134.5 |

**Table 2: The basic performances of asymmetric cryptographic algorithms.**

| Algorithm | Lookup Speed | |
|---|---|---|
| | Operation per Second | Cycles per Operation |
| RSA1024 Encryption | 12,500 | 140,000 |
| RSA1024 Decryption | 684.93 | 2,680,000 |
| RSA2048 Encryption | 6,250 | 290,000 |
| RSA2048 Decryption | 164.47 | 11,120,000 |

Asymmetric cryptographic algorithms, e.g. RSA shown in Table 2, is 2∼3 orders of magnitude slower than symmetric cryptographic algorithms. Meanwhile, the processes of encryption and decryption are different in an asymmetric cryptographic algorithm. Generally, the encryption is 1∼2 orders slower than the decryption. For example, as illustrated in Table 2, RSA1024 can encrypt 12,500 keys per second or decrypt 684.93 keys per second, the encryption is 18.25× faster than the decryption. Furthermore, the speed gap between decryption and encryption of RSA increases as the length of ciphertext grows. In our simulation, we apply RSA1024 as our asymmetric cryptographic algorithm.

## 5.1 The speed of content verification

Time, cost by content verification, consists of two parts: the time for calculating hash value and the time for encryption and decryption. The time for calculating hash value of a content will increase as the content itself grows, and the time for encryption and decryption varies as using different cryptographic algorithms. In this section, therefore, we evaluate the verification speeds of different schemes with different content sizes. Among these simulations, we use MD5 as the message-digest algorithm, RSA1024 as the asymmetric cryptographic algorithm, and DES-XEX3/CTR as the symmetric cryptographic algorithm.

Table 3 illustrates the speeds of different schemes to verify a 1000 Bytes content. NDN-Verify costs 0.0839 ms to verify this content, which consists of 0.0039 ms to calculate the hash value of the content, and 0.08 ms to decrypt the digital signature. Since User-Verify assigns all computing tasks to users and content providers, the computing cost of the content verification in the edge router is 0. In online work mode, Router-Verify should encrypt the symmetric key $K_A$ with the content provider's public key, hash the receiving content, and encrypt the hash value to confirm the content's correctness. Hence Router-Verify with online work mode costs 1.46445 ms to verify a 1000 Bytes content. In offline work mode, however, Router-Verify only needs to hash the content and encrypt the hash value to verify the content, it only costs 0.0045 ms. In summary, Router-Verify only needs one content verification in the first edge router that receives the Data packet from the content provider, and Router-Verify with offline work mode can achieve 18.85× speedup of NDN-Verify by replacing the asymmetric cryptographic algorithm with the symmetric cryptographic algorithm in the edge router.

**Table 3: The speeds of different schemes to verify a 1000 Bytes content.**

| Scheme | Hash Time (ms) | Encryption and Decryption Time (ms) | Total Time (ms) |
|---|---|---|---|
| NDN-Verify | 0.0039 | 0.08 | 0.0839 |
| User-Verify | 0 | 0 | 0 |
| Router-Verify (Online) | 0.0039 | 1.46055 | 1.46445 |
| Router-Verify (Offline) | 0.0039 | 0.00055 | 0.00445 |

To verify a content with 1000 Bytes, Router-Verify (offline) costs 0.0039 ms to hash the content, and 0.00055 ms to encrypt the hash value. Therefore, hash calculation occupies 87.64% of the total time and becomes the bottleneck of the content verification. Figure 7 clearly demonstrates the trend of verification speeds of different schemes as the content size grows. From Figure 7, we can conclude that the time of different schemes costing on content verification are linearly increasing as the content size grows. In NDN-Verify, with the content size varying from 64 Bytes to 1472 Bytes, the time of MD5 costs from 0.31% to 0.67% of the total time. Hence RSA1024 is the bottleneck of the system. After replacing RSA1024 with DES-XEX3, in Router-Verify, the time of MD5 costs 31.09% of the total time to verify a content with 64 Bytes, and 91.21% of the total time to verify a content with 1472 Bytes. Therefore, MD5 becomes the system's bottleneck as the content size grows.

However, the verification speed can be improved by applying the hardware module to accelerate the hash calculation in a hardware router. For example, CRC32 (hash calculation) is implemented by hardware in commercial routers. In this case, Router-Verify can improve 145.5× of the original NDN content verification.

## 5.2 The transfer overhead

Different schemes append different data (token, signature, etc.) to implement content verification. The transfer overheads of different schemes are listed in Table 4, where $K$ is the number of hops from the user and the content provider, $M_1$ and $M_2$ are the ciphertexts of symmetric and asymmetric encryption algorithms respectively ($M_1 = 16$, $M_2 = 21$ in our experiments), and the token length is 64 bits in user-assisted content verification scheme.

NDN-Verify appends the signature of the content to the Data packet, and it costs $64K$ Bytes after transferring $K$ hops. To verify the content provider, the Interest packet in User-Verify carries the token $S_A$, the symmetric key $K_A$ and the ciphertext $K_{pub}(K_A)$, and the Data packet in User-Verify carries the identification message $K_A^{'}(S_A)$ in addition to the signature of the content. Since the identification message is removed by the edge router, the total

**Table 4: The transfer overheads of different schemes ($K$ is the number of hops from the user and the content provider, $M_1$ and $M_2$ are the ciphertexts of symmetric and asymmetric encryption algorithms respectively, and the token length is 64 bits in user-assisted content verification scheme.**

| Scheme | Interest Packet (Byte) | Data Packet (Byte) | Total transfer overhead (Byte) |
|---|---|---|---|
| NDN-Verify | 0 | $K \times M_2$ | $K \times M_2$ |
| User-Verify | $K \times (M_1 + M_2 + 8)$ | $K \times (M_1 + M_2)$ | $2K \times (M_1 + M_2 + 4)$ |
| Router-Verify (One AS) | $M_2$ | $K \times M_2 + M_1$ | $K \times M_2 + M_1 + M_2$ |
| Router-Verify (Mutual Distrust ASes) | $\frac{K}{2} M_2 + M_2$ | $K \times M_2 + M_1 + \frac{K}{2} M_1$ | $M_1 + M_2 + \frac{3K}{2} M_2 + \frac{K}{2} M_1$ |

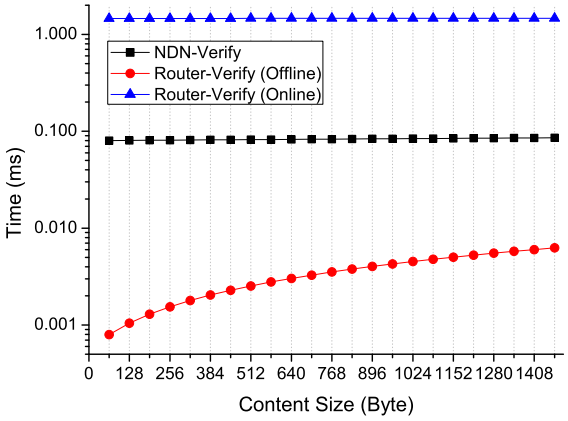


**Figure 7: The trend of verification speeds of different schemes as the content size grows.**

transfer overhead of User-Verify is $2K \times (M_1 + M_2 + 4)$ which is $\frac{2(M_1+M_2)}{M_2}\times$ of NDN-Verify's transfer overhead. In addition to NDN-Verify in one AS, Router-Verify only sends a ciphertext $K_{pub}(K_A)$ and an identification message $K_A^{'}(Hash(D))$, hence the transfer overhead of Router-Verify is $K \times M_2 + M_1 + M_2$. Compared with NDN-Verify, Router-Verify merely costs extra $M_1 + M_2$ Bytes to implement 18.85× speedup whatever the number of hops between the user and the content provider.

## 6 RELATED WORK

Nowaday, the research of NDN in industry and academia mainly focus on the wire speed name lookup [19–24], effective cache strategy [13, 17, 25], fast forwarding mechanism [27, 28], NDN applications [9, 31]. These research has demonstrated the scalability, feasibility and practicality of NDN, and has promoted the development of NDN.

Security mechanism, as an important part of NDN, has attracted widespread attention. Custom private protection [1, 7] and detection of cache missing attack [4, 26] as the two major security mechanisms have been studied. Custom private leakiness is introduced by the cache mechanism in NDN, i.e., the contents cached in routers can leak the privacy of users. Specifically, a malicious cracker can detect a user's privacy through the response time of some specific Interest packets, since the response time of a required content cached in the router is smaller than the fresh content's response time. Gasti et al. [1, 7] propose a few mechanisms to protect custom's private by increasing the response time. The purpose of cache missing attack is to reduce the cache hit ratio in an NDN router. An attacker sends a large amount of unpopular Interest packets to the router to increase the traffic and speed up the frequency of cache replacement. The protection mechanisms, proposed in paper [4, 26], detect the attackers by counting the number of Interest packets and measuring the popularity of Interest packets sent by the user. When a user is deemed as an attacker, the router will drop all Interest packets come from this attacker to prevent cache missing attack. Different from the custom private protection, our work focuses on preventing corrupted and faked contents be spread in the network by improving the content verification speed.

NDN applies content-based [18], rather than connection-based, security mechanism which allows users to authenticate the returned content regardless of where it comes from – the original source, or the cache in a route. Decoupling the content from the content provider, i.e., decoupling "what" from "where" requires security and network primitives that can refer to, and authenticate content itself, rather than the host and file containers where it resides. Similar to prior works [3, 11, 16], NDN authenticate the *linkage* between a name and a content, rather than authenticating the content or its publisher. The signature of both the name and the content represents their linkage. Users or in-path routers can authenticate the name and the content by verifying their signature. The verification process is complex and time-consuming, and is much slower than the line rate [6]. Based on this observation, Giuseppe et al. [2] devise a simple analytical approach which permits to assess performance of an LRU caching strategy storing a randomly sampled subset of requests. This work is orthogonal with our work.

LIVE [14] allows content providers to control content access in NDN nodes by selectively distributing integrity verification tokens to authorized nodes. LIVE is effective and helpful for the authorized nodes to verify the specified contents. However, LIVE is helpless and opaque for the other contents that are free to be cached in routers. Our fast content verification mechanisms complement LIVE by improving the verification performance of all contents.

To our best knowledge, we first propose the Router-Cooperation content verification scheme to improve the performance of content verification in NDN routers. By replacing the asymmetric decryption with symmetric decryption, Router-Verify can speed up 18× of NDN-Verify with negligible extra transfer overhead.

## 7 CONCLUSION

In this paper, we propose two content verification schemes to improve the verification performance of NDN routers to prevent content poisoning. The user-assisted content verification scheme verifies content providers to guarantee the correctness of contents. By bypassing the content verification in NDN routers, the user-assisted scheme can achieve perfect performance. However, this scheme can only be applied in the environment that all consumers can be trusted.

The Router-Cooperation content verification scheme verifies the content provider independently without the assistance of users. By replacing the asymmetric encryption operation with symmetric encryption operation, the Router-Cooperation scheme can effectively reduce the computing resource for cryptographic operation and improve the content verification performance. The simulation results demonstrate that this Router-Cooperation scheme can achieve 18.85× speedups of the original content verification scheme used in NDN.

To verify a content with 1000 Bytes, the Router-Cooperation scheme costs 0.0039 ms on hashing the content, and 0.00055 ms on encrypting the hash value. Therefore, hash calculation occupies 87.64% of the total time and becomes the bottleneck of the content verification. In our further work, we can implement the message-digest algorithm in hardware to improve the total performance of the Router-Cooperation content verification scheme.

## ACKNOWLEDGMENTS

The work is supported by the Shenzhen Key Fundamental Research Projects (No. JCYJ20160330095313861), Huawei Innovation Research Program (HIRP), NSFC (61602271, 61373143, 61432009), China Postdoctoral Science Foundation (No. 2015T80089, 2016M591182), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20130002110084).

## REFERENCES

[1] Gergely Acs, Mauro Conti, Paolo Gasti, Cesar Ghali, and Gene Tsudik. 2013. Cache Privacy in Named-Data Networking. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. 41–51.

[2] Giuseppe Bianchi, Andrea Detti, Alberto Caponi, and Nicola Blefari Melazzi. 2013. Check Before Storing: What is the Performance Price of Content Integrity Verification in LRU Caching? *SIGCOMM Comput. Commun. Rev.* 43, 3 (July 2013), 59–67. DOI: https://doi.org/10.1145/2500098.2500106

[3] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. 2000. *Freenet: A Distributed Anonymous Information Storage and Retrieval System*. 46–66 pages. DOI: https://doi.org/10.1007/3-540-44702-4_4

[4] Mauro Conti, Paolo Gasti, and Marco Teoli. 2013. A lightweight mechanism for detection of cache pollution attacks in Named Data Networking. *Computer Networks* 57, 16 (2013), 3178 – 3191. DOI: https://doi.org/10.1016/j.comnet.2013.07.034

[5] Wei Dai. 2017. Crypto++ 5.6.0 Benchmark. (2017). http://www.cryptopp.com/benchmarks.html

[6] A. Detti, A. Caponi, G. Tropea, G. Bianchi, and N. Blefari-Melazzi. 2013. On the interplay among naming, content validity and caching in Information Centric Networks. In *Global Communications Conference (GLOBECOM), 2013 IEEE*. 2108–2113. DOI: https://doi.org/10.1109/GLOCOM.2013.6831386

[7] Steve DiBenedetto, Paolo Gasti, Gene Tsudik, and Ersin Uzun. 2011. ANDaNA: Anonymous Named Data Networking Application. *CoRR* abs/1112.2205 (2011).

[8] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. 2012. DoS and DDoS in Named-Data Networking. *ArXiv e-prints* (Aug. 2012). arXiv:cs.NI/1208.0952

[9] Van Jacobson, Diana K. Smetters, Nicholas H. Briggs, Michael F. Plass, Paul Stewart, James D. Thornton, and Rebecca L. Braynard. 2009. VoCCN: voice-over content-centric networks. In *Proceedings of the 2009 workshop on Re-architecting the internet (ReArch '09)*. ACM, New York, NY, USA, 1–6.

[10] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. 2009. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09)*. ACM, 1–12.

[11] John Kubiatowicz, David Bindel, Yan Chen, Steven E. Czerwinski, Patrick R. Eaton, Dennis Geels, Ramakrishna Gummadi, Sean C. Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Y. Zhao. 2000. OceanStore: an architecture for global-scale persistent storage. *Sigplan Notices* 35 (2000), 190–201. Issue 11. DOI: https://doi.org/10.1145/356989.357007

[12] Tobias Lauinger. 2010. Security & Scalability of Content-Centric Networking. (September 2010). http://tuprints.ulb.tu-darmstadt.de/2275/

[13] Jun Li, Hao Wu, Bin Liu, Jianyuan Lu, Yi Wang, Xin Wang, YanYong Zhang, and Lijun Dong. 2012. Popularity-driven Coordinated Caching in Named Data Networking. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '12)*. ACM, New York, NY, USA, 15–26. http://doi.acm.org/10.1145/2396556.2396561

[14] Qi Li, Xinwen Zhang, Qingji Zheng, R. Sandhu, and Xiaoming Fu. 2015. LIVE: Lightweight Integrity Verification and Content Access Control for Named Data Networking. *Information Forensics and Security, IEEE Transactions on* 10, 2 (Feb 2015), 308–320. DOI: https://doi.org/10.1109/TIFS.2014.2365742

[15] OpenMP. 2017. The OpenMP API specification for parallel programming. (2017). http://openmp.org

[16] Bogdan C. Popescu, Maarten Van Steen, Bruno Crispo, Andrew S. Tanenbaum, Jan Sacha, and Ihor Kuz. 2005. Securely Replicated Web Documents. In *International Parallel and Distributed Processing Symposium/International Parallel Processing Symposium*. DOI: https://doi.org/10.1109/IPDPS.2005.395

[17] E.J. Rosensweig, D.S. Menasche, and J. Kurose. 2013. On the steady-state of cache networks. In *INFOCOM, 2013 Proceedings IEEE*. 863–871.

[18] Diana Smetters and Van Jacobson. 2009. Securing Network Content. (2009). https://www.parc.com/content/attachments/securing-network-content-tr.pdf

[19] Yi Wang, Huichen Dai, Junchen Jiang, Keqiang He, Wei Meng, and Bin Liu. 2011. Parallel Name Lookup for Named Data Networking. In *IEEE Global Telecommunications Conference (GLOBECOM)*. 1 –5.

[20] Yi Wang, Huichen Dai, Ting Zhang, Wei Meng, Jindou Fan, and Bin Liu. 2013. GPU-accelerated name lookup with component encoding. *Computer Networks* 57, 16 (2013), 3165 – 3177.

[21] Yi Wang, Keqiang He, Huichen Dai, Wei Meng, Junchen Jiang, Bin Liu, and Yan Chen. 2012. Scalable Name Lookup in NDN Using Effective Name Component Encoding. In *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*. 688–697.

[22] Yi Wang, Tian Pan, Zhian Mi, Huichen Dai, Xiaoyu Guo, Ting Zhang, Bin Liu, and Qunfeng Dong. 2013. NameFilter: Achieving fast name lookup with low memory cost via applying two-stage Bloom filters. In *INFOCOM 2013 mini conference, IEEE*.

[23] Yi Wang, Dongzhe Tai, Ting Zhang, Jianyuan Lu, Boyang Xu, Huichen Dai, and Bin Liu. 2013. Greedy Name Lookup for Named Data Networking. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '13)*. ACM, New York, NY, USA, 359–360. http://doi.acm.org/10.1145/2465529.2465741

[24] Yi Wang, Yuan Zu, Ting Zhang, Kunyang Peng, Qunfeng Dong, Bin Liu, Wei Meng, Huichen Dai, Xin Tian, Zhonghu Xu, Hao Wu, and Di Yang. 2013. Wire Speed Name Lookup: A GPU-based Approach. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (nsdi'13)*. USENIX Association, Berkeley, CA, USA, 199–212. http://dl.acm.org/citation.cfm?id=2482626.2482647

[25] Hao Wu, Jun Li, Yi Wang, and Bin Liu. 2013. EMC: The Effective Multi-Path Caching Scheme for Named Data Networking. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*. 1–7.

[26] Mengjun Xie, I. Widjaja, and Haining Wang. 2012. Enhancing cache robustness for content-centric networking. In *INFOCOM, 2012 Proceedings IEEE*. 2426–2434.

[27] Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, Beichuan Zhang, and Lixia Zhang. 2013. A case for stateful forwarding plane. *Computer Communications* 36, 7 (2013), 779 – 791. http://www.sciencedirect.com/science/article/pii/S0140366413000236

[28] Haowei Yuan, Tian Song, and P. Crowley. 2012. Scalable NDN Forwarding: Concepts, Issues and Principles. In *International Conference on Computer Communications and Networks (ICCCN)*. 1–9.

[29] Lixia Zhang, Deborah Estrin, Jeffrey Burkeand, Van Jacobson, James D. Thornton, Ersin Uzun, and Baichuan Zhang. 2013. Named Data Networking (NDN) Project 2011 - 2012 Annual Report. (2013). http://named-data.net/wp-content/uploads/2013/08/ndn-proj-pub.pdf

[30] Lixia Zhang, Deborah Estrin, Van Jacobson, and Baichuan Zhang. 2010. Named Data Networking (NDN) Project. http://www.named-data.net/

[31] Zhenkai Zhu, Sen Wang, Xu Yang, Van Jacobson, and Lixia Zhang. 2011. ACT: Audio Conference Tool over Named Data Networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking (ICN '11)*. ACM, New York, NY, USA, 68–73. http://doi.acm.org/10.1145/2018584.2018601