Hindawi
Security and Communication Networks
Volume 2018, Article ID 9845426, 8 pages
https://doi.org/10.1155/2018/9845426

Research Article

Cloud Based Data Protection in Anonymously Controlled SDN

Jian Shen,[1,2] Jun Shen,[1] Chin-Feng Lai,[3] Qi Liu,[1] and Tianqi Zhou[1]

[1] *Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, China*
[2] *State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*
[3] *Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan*

Correspondence should be addressed to Chin-Feng Lai; cinfon@ieee.org

Received 28 July 2017; Accepted 3 January 2018; Published 29 January 2018

Academic Editor: Iván García-Magariño

Nowadays, Software Defined Network (SDN) develops rapidly for its novel structure which separates the control plane and the data plane of network devices. Many researchers devoted themselves to the study of such a special network. However, some limitations restrict the development of SDN. On the one hand, the single controller in the conventional model bears all threats, and the corruption of it will result in network paralysis. On the other hand, the data will be increasing more in SDN switches in the data plane, while the storage space of these switches is limited. In order to solve the mentioned issues, we propose two corresponding protocols in this paper. Specifically, one is an anonymous protocol in the control plane, and the other is a verifiable outsourcing protocol in the data plane. The evaluation indicates that our protocol is correct, secure, and efficient.

1. Introduction

Nowadays, as cloud computing develops rapidly [1–6], an increasing number of applications and services are transplanted to the cloud to satisfy the requirements of cloud clients. As time goes by, the burden on the cloud and the management of services and applications become heavier. In addition, the cloud is not flexible enough for the reconfiguration of the underlying networking when facing different applications. Therefore, SDN is proposed, which has caught the attention of a large number of researchers [7, 8].

SDN originates from the Clean Slate research project of Stanford University in 2006. Three years later, professor Nick McKeown proposed the specific definition of SDN. Since the data plane and the control plane of network are separated, which differs from the cloud, SDN is equipped with unprecedented flexibility [9–12]. As a breakthrough, SDN will never suffer from the reconfiguration of the underlying networking when facing different applications. The network devices named "SDN switches" form the data plane, which are responsible for the transmission of data packets merely. Since data transmission is an uncomplicated work and these switches should carry nothing else, the data packets on transmission can be dealt with substantially quickly, which is more suitable for the popularized network. The control plane consists of the controller, and this plane holds the global information of the whole network and controls SDN switches in the data plane. There is a unified interface named southbound API between the control plane and the data plane, which realizes the interaction between the two planes. In a word, the management of the network can be well-bedded and convenient with the proposal of SDN.

Although SDN brings benefits to network, there are still some problems impeding its development. On the one hand, there is only one controller in the control plane of the conventional SDN model, and the single controller is the core of the whole network. Once the only controller is attacked or corrupted, the network will be imploded. In this paper, we aim to involve more than one controller in SDN, and the data transmission commands will be sent from a random and anonymous controller. On the other hand, when more and more applications are deployed in SDN, the information in the data plane will be more. Since the storage space of SDN switches is limited, the burden on these switches will be heavier as time goes by. Compared with the limited storage space in SDN switches, the space in the cloud is almost endless. Moreover, cloud storage is equipped with on-demand outsourcing function, ubiquitous network access, and location-independent resources. Hence, we aim to employ the cloud for data storage in this paper.

The main contributions of the paper are summarized as follows:

(1) We propose an anonymous control protocol in the control plane. The threats on a certain controller can be relieved and SDN will be substantially more secure by the design of this protocol.

(2) We design a verifiable outsourcing protocol in the data plane. Through the protocol, SDN switches with limited space can employ the cloud for data storage. In addition, the outsourced data can be verified for security.

The remainder of the paper is organized as follows. In Section 2, we present the system and security model. In Section 3, we introduce some preliminary works to allow the reader to obtain better understanding of the topic. In Section 4, we describe the two protocols in this paper in detail, including the anonymous control protocol and the verifiable outsourcing protocol. In Section 5, we present the correctness analysis, security analysis, and cost analysis of our protocols. The conclusion is drawn in Section 6.

2. Problem Statement

2.1. The System Model. In the application plane, there are many software applications, and we define the number of them as n. As time goes by, the applications will be more for the popularity of SDN. In the control plane, there is traditionally one controller in the system merely. However, the network will be paralyzed when the only controller is attacked or corrupted. Therefore, we aim to deploy more than one controller in our system, the number of which is not confined to "4" in Figure 1. With several controllers, the data transmission instructions will be commanded by a random controller anonymously and the threats on a certain controller will be relieved. Suppose that there are one hundred controllers in the system; then the possibility of attacks on a certain controller can be relieved from 100% to 1%. There is a north-bound API between the application plane and the control plane, which is used for information interactions. Through north-bound API, the network can be customized by the actual requirements of each application. In the data plane, there are a large number of SDN switches, which are responsible for data transmission. Controllers command the process of transmission through the interface between the control plane and the data plane called south-bound API. Generally, the functions of south-bound API are realized through OpenFlow, which is a standardized protocol. Moreover, since we consider the limited storage in SDN switches, we introduce an additional plane—the storage plane consisting of one or more cloud service providers (CSPs). CSPs provide SDN switches with storage services in the cloud.

2.2. The Security Model. It is assumed that the introduced key generation center (KGC) is trustworthy enough and will never attack the system. Although CSPs are curious about the data in the data plane which are outsourced in the cloud, they will never collude with any switch. We say that protocols proposed in this paper are secure if the following conditions hold:

(1) *Unforgeability of controllers*: the possibility that an adversary can pretend to be any one valid controller in the control plane and hold the public and private key pair ($SK_i = P_i^{x_i},\ PK_i = Q_i^{x_i}$) of the controller is negligible.

(2) *Unforgeability of BLS-based homomorphic verifiable authenticator*: CSPs will never succeed in forging the data tag $\tau = (h \cdot u^m)^k$ of any outsourced data m to deceive the challenger.

(3) *Resistance of replay attack*: it is impossible for CSPs to generate the proof matched with the challenge using the previous proofs ($T_{\text{pre}}, D_{\text{pre}}$), without retrieving the actual cloud data m.

(4) *Resistance of replacement attack*: CSPs can never find out another valid proof ($T_{\text{oth}}, D_{\text{oth}}$) of data to replace the challenged data proof, when they already discarded or damaged cloud data m.

3. Preliminaries

This section presents some preliminaries to facilitate the reader's understanding, including bilinear pairing [13–16], ring signature, BLS-based homomorphic verifiable authenticator, and computational Diffie–Hellman (CDH) problem.

Definition 1 (bilinear pairing). Suppose that G_1, G_2, and G_T are multiplicative cyclic groups of a large prime p, and a map function $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear pairing only if the following properties are satisfied:

(1) Bilinearity: for all the generators $g_1 \in G_1$ and $g_2 \in G_2$, there always exists $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$, where $a, b \in {Z_p}^*$.

(2) Nondegeneracy: there exist $g_1 \in G_1$ and $g_2 \in G_2$ satisfying the equation $e(g_1, g_2) \neq 1$.

(3) Computability: for all the generators $g_1 \in G_1$ and $g_2 \in G_2$, there exists an efficient algorithm to compute $e(g_1, g_2)$.

Definition 2 (ring signature). Given a group of n entities each having public and private key pair ranging from (P_1, S_1) to (P_n, S_n), suppose that the ith entity is the signer, and the message is m. Then, the signature σ on m can be computed when $(m, S_i, P_1, P_2, \ldots, P_n)$ is inputted. Anyone can check the validity of a ring signature when σ, m, and P_1 to P_n are given. All the members in the ring can sign a message with their own private keys and the others' public keys without the agreement of them [17–23]. With ring signature, the message can be transmitted securely and the identity of the actual signer is protected as well.

Definition 3 (BLS-based homomorphic verifiable authenticator). Given two multiplicative cyclic groups G and G_T and

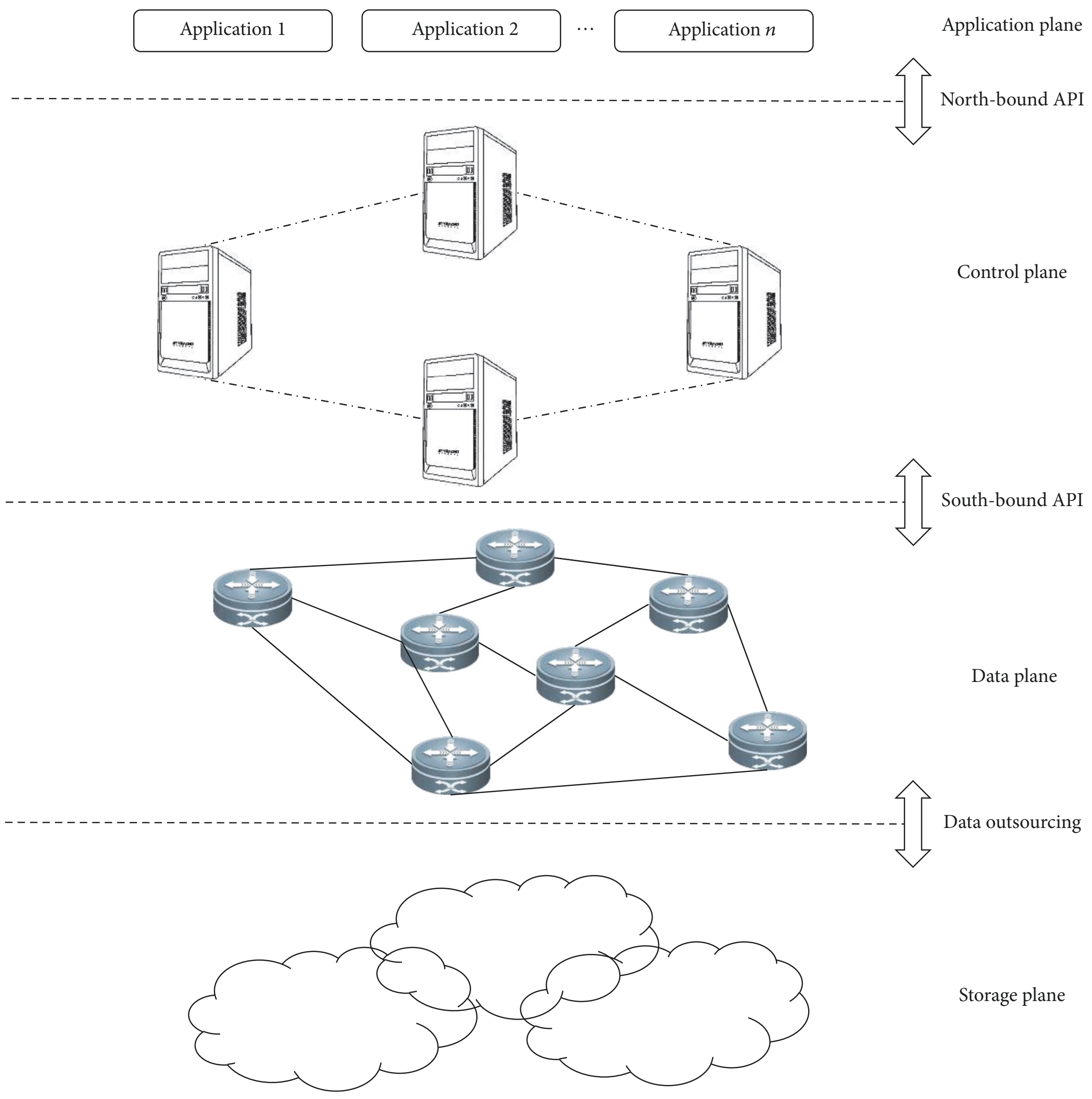


FIGURE 1: The system model.

a bilinear pairing $e : G \times G \rightarrow G_T$, select random $a \in Z_p{}^*$ as the private key. The public key is $\{g, v = g^a\}$. The BLS-based homomorphic authenticator for some data m is computed as $\sigma = (h(m))^a$, where $h(\cdot)$ is a one-way hash function. For verification, the verifier will simply check the equality of the equation $e(h(m), v) \overset{?}{=} e(\sigma, g)$ [24–29]. If the equation holds, the verifier will believe that the BLS-based homomorphic authenticator is valid; otherwise, the validity of the authenticator cannot be proved.

Definition 4 (CDH problem). Given g^a and g^b, where g is a generator of G_1 and $a, b \in_R Z_p{}^*$, compute g^{ab}.

4. The Proposed Protocol

In this section, the core part of this paper will be introduced, which consists of two protocols: the anonymous protocol in the control plane for controllers and the verifiable outsourcing protocol in the data plane for data on SDN switches, which are in Sections 4.2 and 4.3 separately. Some primary notations are introduced as follows.

4.1. Notations. As is shown in Notations section, some primary notations used in protocols are listed. G, G_T, G_1, and G_2 are four multiplicative cyclic groups of a large prime order p. g is the generator of G, and u, g_1 are generators of G_1. $Z_p{}^*$ is a set of nonnegative integers less than p. $e : G \times G \rightarrow G_T$ and $e : G_1 \times G_1 \rightarrow G_2$ are bilinear pairings. $H_1 : \{0,1\}^* \rightarrow Z_p{}^*$, $H_2 : \{0,1\}^* \rightarrow G$, and $H_3 : \{0,1\}^* \rightarrow G_1$ are cryptographic collision-resistant hash functions from strings to elements in $Z_p{}^*$, G, and G_1.

4.2. The Anonymous Protocol in the Control Plane. In this section, the proposed anonymous protocol in the control plane will be described in detail, which aims to decentralize the power of and the attacks on a certain controller. Specifically, the protocol can be divided into 4 phases: setup phase, key generation phase, sign phase, and verification phase. Note that the second phase consists of 4 algorithms: *partial-private-key extraction*, *secret-value selection*, *private-key generation*, and *public-key generation*. The details of the protocol are shown as follows.

4.2.1. Setup Phase. In this phase, some preparation works are made for system setup. First, the whole network selects a trustworthy third party as a KGC, which is responsible for extracting the partial private keys in key generation phase. Then, the KGC will finish the remaining preparations.

Firstly, the KGC selects random $s \in Z_p^*$ as the master key of the network, which is kept secret by the KGC itself. Secondly, the KGC picks random generator $g \in G$ and computes $pk = g^s$. Consequently, the secret and public key pair of the KGC is $(sk = s,\ pk = g^s)$. Thirdly, the identities of all the controllers are aggregated together as a set $\mathrm{ID} = \{\mathrm{ID}_i \mid i = 1, 2, \ldots, c\}$. Finally, all the public parameters of the system are $\{G, G_T, p, g, e, H_1, H_2, pk\}$.

4.2.2. Key Generation Phase. This phase performs the generation of private and public key pairs of all the controllers, and we define that there are totally c controllers in our system. In order to decentralize the power of the KGC, we aim to employ the KGC to extract the partial private keys of controllers merely. Therefore, even though the KGC is attacked by hackers or leaks out some secrets, the data and information of the system will keep secure. The detailed design of the four algorithms is shown as follows:

(i) *Partial-private-key extraction*: this algorithm is performed by the KGC, which takes the public parameters ID and sk as input. The KGC computes $Q_i = H_2(\mathrm{ID}_i)$ and $P_i = Q_i^{sk}$ for all the c controllers as the partial private keys, where i ranges from 1 to c. After that, the KGC sends the pair of Q_i and P_i to each controller separately.

(ii) *Secret-value selection*: for the generation of the complete private and public key pair, each controller selects random number $x_i \in Z_p^*$ as the secret value, which is kept secret by the corresponding controller itself.

(iii) *Private-key generation*: since each controller has selected a secret value and received the partial key pair from the KGC, they will compute $SK_i = P_i^{x_i}$ as the complete private key separately.

(iv) *Public-key generation*: each controller computes $PK_i = Q_i^{x_i}$ as the public key, where i ranges from 1 to c.

Finally, the private and public key pairs $(SK_i = P_i^{x_i}, PK_i = Q_i^{x_i})$ for all the controllers are generated.

4.2.3. Sign Phase. Inspired by [30, 31], a certain controller will sign the message sent to SDN switches using the aux information of other controllers. We define that the message is M and the identity of the randomly chosen actual signer is of the identity ID_s. In order to reduce the risk of attacks on a single controller, the signer will employ the public keys of all the other controllers to sign the message, which realizes the anonymity.

Specifically, the singer picks random $r_i \in Z_p^*$ $(1 \le i \le c)$ and computes $U_i = PK_i^{r_i}$ $(i \ne s)$. If there are collisions among U_i, the signer reselects $r_i \in Z_p^*$ randomly. Then, the signer computes $h_i = H_1(M \parallel \mathrm{ID} \parallel U_i)$ for $i \ne s$. Through h_i, U_s can be calculated by $U_s = PK_s^{r_s}/PK_i^{\sum_{i \ne s}(r_i + h_i)}$. After that, ID_s computes $h_s = H_1(M \parallel \mathrm{ID} \parallel U_s)$ and $S = SK_s^{h_s + r_s}$. Finally, the signature on the message is $\sigma = (M, U_1, U_2, \ldots, U_c, S)$ and it will be sent by the controller which generates the signature to SDN switches through secure channels.

4.2.4. Verification Phase. This phase is designed to check the validity of the signature. First, the verifier computes $h_i = H_1(M \parallel \mathrm{ID} \parallel U_i)$ for $1 \le i \le c$. Then, it checks the equality of the equation

$$e\left(pk, \prod_{1 \le i \le c} \left(U_i \cdot PK_i^{h_i}\right)\right) \overset{?}{=} e\left(g, S\right). \tag{1}$$

If (1) holds, the signature is valid; otherwise, it is not.

4.3. Verifiable Outsourcing Protocol in the Data Plane. The second protocol proposed in this paper is designed for the limited storage space in SDN switches. We employ the cloud for data storage in the data plane, which is of great convenience. As for the attacks in the cloud threatening the security of these data, our protocol is designed to be verifiable, so that it can ensure the correctness and integrity of the outsourced data. Specifically, the second protocol can be divided into 2 phases: setup phase and verification phase. The following are the details.

4.3.1. Setup Phase. Setup phase is for some preparations of the protocol, which consists of two algorithms: *key generation* and *tag generation*. The public and private parameters are generated in *key generation*, and the tag for the outsourced data is computed in *tag generation*. The designs of the two algorithms are shown below.

(i) *Key generation*: the SDN switch which wants to outsource data in cloud selects random $a \in Z_p^*$ as the private key k and keeps it secret. Then, it generates $v = g_1^a$. Finally, the outputs of the algorithm *key generation* are the private and public key pair $\{(k = a), (g_1, u, v = g_1^a)\}$.

(ii) *Tag generation*: it is obvious that the data outsourced in the cloud should be encrypted. We define that the time point to outsource the data m is t and the identity of the switch is the concatenation of its latitude lat and longitude lon, which is unique enough. First, the switch computes $h = H_3(t \parallel \mathrm{lat} \parallel \mathrm{lon})$. Then, the tag is calculated as $\tau = (h \cdot u^m)^k$. Finally, the switch uploads the data and the corresponding tags to the cloud for storage.

4.3.2. Verification Phase. The second phase performs data verification, which enables switches that outsource their data to ensure the correctness and completeness of these data. Specifically, the switch asks the cloud for the proof which indicates the integrity of the outsourced data, and the cloud should respond with the corresponding valid proof. If the proof is invalid, it will never pass the verification equation. This phase includes two algorithms: *proof generation* and *verification proof*. The details are as follows:

(i) *Proof generation*: upon receiving the challenge request from a certain switch, the cloud should generate the corresponding data proof to claim that these data are correctly and completely stored in cloud services. Specifically, the cloud generates the tag proof $T = \tau^r$ and the data proof $D = m \cdot r$, where r is randomly picked from Z_p^*. Then, the cloud sends the proof (T, D) to the switch (challenger).

(ii) *Verification proof*: when the switch receives the proof, it will check the validity of the proof. Specifically, the switch checks whether the equation

$$e(T, g_1) \overset{?}{=} e(h \cdot u^D, v) \tag{2}$$

holds. If (2) holds, the outsourced data in the cloud proves to be correct and complete; otherwise, the data is corrupted.

5. Evaluation

In this section, some analysis concerning the proposed protocols is presented, including correctness analysis, security analysis, and computational cost analysis.

5.1. Correctness Analysis. The correctness analysis of the proposed two protocols will be presented. Equation (1) can verify the signatures of controllers and (2) can verify the outsourced data of SDN switches in the cloud. Specifically, the correctness of the two equations will be elaborated separately in the following.

If (1) is correct, only the valid signatures generated by controllers can pass the verification of this equation. Otherwise, the invalid signatures can pass the verification as well. The correctness of (1) is demonstrated as follows:

$$\begin{aligned} & e\left(pk, \prod_{1\le i\le c}\left(U_i \cdot {PK_i}^{h_i}\right)\right) \\ & = e\left(g^{sk}, U_s \cdot {PK_s}^{h_s} \cdot \prod_{i\ne s}\left(U_i \cdot {PK_i}^{h_i}\right)\right) \\ & = e\left(g^{sk}, \frac{{PK_s}^{r_s}}{{PK_i}^{\sum_{i\ne s}(r_i+h_i)}} \cdot {PK_s}^{h_s} \cdot \prod_{i\ne s} {PK_i}^{(h_i+r_i)}\right) \\ & = e\left(g^{sk}, \frac{{PK_s}^{r_s}}{{PK_i}^{\sum_{i\ne s}(r_i+h_i)}} \cdot {PK_s}^{h_s} \cdot {PK_i}^{\sum_{i\ne s}(r_i+h_i)}\right) \\ & = e\left(g^{sk}, {PK_s}^{(h_s+r_s)}\right) = e\left(g^{sk}, {Q_s}^{x_s\cdot(h_s+r_s)}\right) \\ & = e\left(g, {Q_s}^{sk\cdot x_s\cdot(h_s+r_s)}\right) = e\left(g, {P_s}^{x_s\cdot(h_s+r_s)}\right) \\ & = e\left(g, {SK_s}^{(h_s+r_s)}\right) = e(g, S). \end{aligned} \tag{3}$$

If (2) is correct, only the valid proofs sent from the cloud can pass the verification of this equation. Otherwise, either the cloud or the SDN switches will be cheated by the incorrect check results. The correctness of (2) is proved as follows:

$$\begin{aligned} e(T, g_1) & = e(\tau^r, g_1) = e\left(\left(\left(h \cdot u^m\right)^k\right)^r, g_1\right) \\ & = e\left(\left(h \cdot u^m\right)^r, {g_1}^k\right) = e\left(h \cdot u^{m\cdot r}, v\right) \\ & = e\left(h \cdot u^D, v\right). \end{aligned} \tag{4}$$

In summary, (1) and (2) are both correct, which indicates the correctness of our protocols.

5.2. Security Analysis. We will prove the security of our protocols with proofs of the following several theorems.

Theorem 5. *In the anonymous protocol, the adversary can never pretend to be any controller in the control plane to generate a valid signature. In other words, the signature signed by the adversary will never pass verification phase.*

Proof. We define that the adversary has not been broken through KGC and has no knowledge about the partial private keys of controllers. In order to forge a valid signature, the adversary should compute the complete private key for a controller. The private key is computed as $SK_i = P_i^{x_i}$, where $P_i = Q_i^{sk}$. Since the adversary has no knowledge of x_i and sk, the possibility of generating the correct private key of a controller is negligible. In this theorem, although $P_i = Q_i^{sk}$ and $PK_i = Q_i^{x_i}$ are known to the adversary, it is difficult for the adversary to figure out $SK_i = Q_i^{sk\cdot x_i}$, due to the CDH problem in bilinear groups. Therefore, the adversary will fail to forge a valid signature of any controller. □

Theorem 6. *In the verifiable outsourcing protocol, the BLS-based homomorphic verifiable authenticator cannot be forged by any adversary if the computational Differ-Hellman assumption in bilinear pairing holds.*

Proof. This theorem suggests that forging an BLS-HVA is computationally infeasible. As shown in the security analysis of the protocol in Shen et al. [5], the HVA scheme has been proved to be effectively unforgeable, the reason for which is that BLS is secure when the CDH problem is difficult in bilinear groups. The proof can be found in [5] and is omitted here. □

Theorem 7. *In the verifiable outsourcing protocol, replay attacks would be never effective. In other words, the cloud can never pass the verification if it responds to the challenges with some previous information.*

Proof. We define that in the replay attack game, the cloud does not protect the challenged data of the switch. In order to pass the verification, the cloud finds out a previous version of these data and sends the corresponding proof $(T_{\text{pre}}, D_{\text{pre}})$ to the switch. Since the verification equation is $e(T, g_1) = e((H_3(t \parallel \text{lat} \parallel \text{lon}) \cdot u^m)^{kr}, g_1)$ in the *verification proof*, where the time to outsource the data is t, obviously, data of different versions are outsourced at different times. Hence, the replay attacks can never succeed under our protocol. □

Theorem 8. *In the verifiable outsourcing protocol, replacement attacks would never work. In other words, the cloud can never pass verification if it responds the challenges with other switches' completely preserved information.*

Proof. We define that in the replacement attack game, the challenged data in cloud is corrupted for some reasons. In order to pass the verification, the cloud decides to send the perfectly protected data information of some other switches back to the challenger. Consequently, the proof will be $(T_{\text{oth}}, D_{\text{oth}})$. Since the equation is $e(T, g_1) = e((H_3(t \parallel \text{lat} \parallel \text{lon}) \cdot u^m)^{kr}, g_1)$ in the *verification proof*, where (lat $\parallel$ lon) indicates the unique and specific location of the switch, obviously, though the time to outsource the data is exactly the same, the locations of any two different switches differ from each other. Hence, the replacement attacks will never succeed in our design. □

5.3. Computational Cost Analysis. The computational cost analysis of the two protocols is presented in this section. In addition, some simulation experiments have been conducted to prove the low cost of our protocols. The experiments were performed by C programming language with the pairing-based cryptography (PBC) library and the GUN multiple precision arithmetic (GMP) library on a VMware workstation machine with Intel Core i5-2450M processors running at 2.50 GHz and 12 G memory, Ubuntu 12.04 × 64.

First, the brief descriptions of the cost of the two protocols will be introduced. For simplicity, we denote multiplication as M, modular exponentiation as E, and hash function as H. As for the anonymous protocol in the control plane, it costs $1E$ in setup phase, $cH + 3cE$ in key generation phase, and $cE + cM + cH$ in sign phase, where c is the number of controllers in the control plane. As for the second protocol in the data plane, it costs $1E$ in *key generation* algorithm, $1H + 2E + 1M$ in *tag generation* algorithm, and $2E$ in *proof generation* algorithm, when an SDN switch outsources its data.

Then, the experimental results will be analyzed. In Figure 2, the time cost of the system changes with the increase of the number of controllers in the control plane. It is obvious that when there are more controllers, the computational cost of the system will be higher. However, when there are more controllers in the control plane, the system will be more secure in practical terms. Hence, for different applications, there exists a tradeoff between energy and security in the choice of the specific number of controllers. Figure 3 describes the computational cost of each algorithm in the verifiable outsourcing protocol in the data plane. It is clear that outsourcing the data of an SDN switch to the cloud costs about 0.018 seconds. Compared with the limited storage space in these switches, using 0.018 seconds to trade off on-demand and convenient storage service is worthwhile.

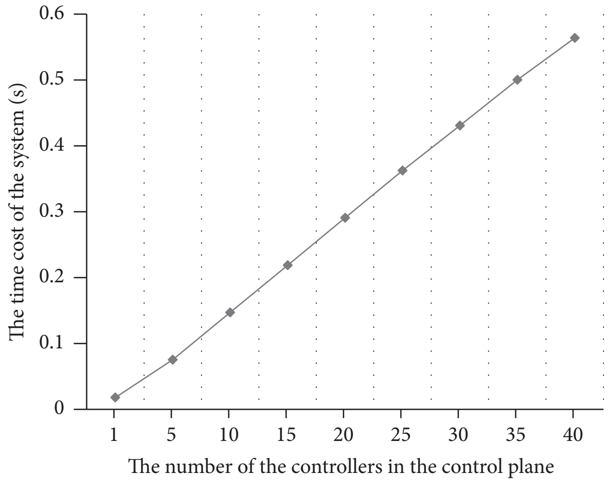


FIGURE 2: The time cost of the anonymous protocol in the control plane.

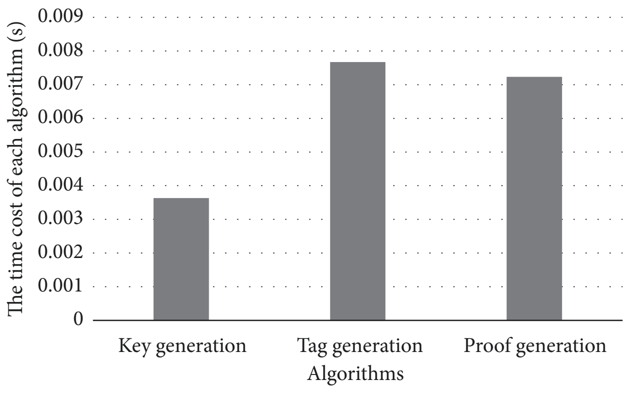


FIGURE 3: The time cost for algorithms of verifiable outsourcing protocol in the data plane.

6. Conclusion

In this paper, two protocols are proposed to realize anonymous control and cloud based data protection in SDN. The first protocol in the control plane employs the ring signature technique, which makes the controller anonymously command the process of data transmission over the SDN switches. The second protocol in the data plane employs the storage services in cloud computing, which extends the limited storage space of SDN switches. Moreover, the data outsourcing is secure enough for its verifiability. Finally, the evaluation demonstrates the correctness, security, and efficiency of our protocols.

Notations

G, G_T, G_1, G_2:	Multiplicative cyclic groups
p:	Prime order of G, G_T, G_1, G_2
g, g_1, u:	Generators of multiplicative cyclic groups
${Z_p}^*$:	Set of nonnegative integers less than p
$e : G \times G \rightarrow G_T$:	A bilinear pairing
$e : G_1 \times G_1 \rightarrow G_2$:	A bilinear pairing
$H_1 : \{0,1\}^* \rightarrow {Z_p}^*$:	Cryptographic collision-resistant hash function
$H_2 : \{0,1\}^* \rightarrow G$:	Cryptographic collision-resistant hash function
$H_3 : \{0,1\}^* \rightarrow G_1$:	Cryptographic collision-resistant hash function.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Science Foundation of China under Grants no. 61672295, no. 61672290, no. 61772280, no. U1405254; the State Key Laboratory of Information Security under Grant no. 2017-MS-10; the 2015 Project of six personnel in Jiangsu Province under Grant no. R2015L06; the CICAEET fund; and the PAPD fund.

References

[1] Y. Cui, J. Song, M. Li, Q. Ren, Y. Zhang, and X. Cai, "SDN-based Big Data Caching in ISP Networks," *IEEE Transactions on Big Data*, 2017.

[2] J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, "A secure cloud-assisted urban data sharing framework for ubiquitous-cities," *Pervasive and Mobile Computing*, 2017.

[3] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[4] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Computing*, pp. 1–10, 2017.

[5] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2401–2415, 2017.

[6] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers and Security*, 2017.

[7] P. Xiao, W. Qu, H. Qi, Z. Li, and Y. Xu, "The SDN controller placement problem for WAN," *IEEE International Conference on Communications in China*, pp. 220–224, 2015.

[8] Z. Hu, M. Wang, X. Yan, Y. Yin, and Z. Luo, "A comprehensive security architecture for SDN," in *Proceedings of the 2015 18th International Conference on Intelligence in Next Generation Networks, ICIN 2015*, pp. 30–37, IEEE, Paris, France, February 2015.

[9] C.-F. Lai, M. Chen, J.-S. Pan, C.-H. Youn, and H.-C. Chao, "A collaborative computing framework of cloud network and WBSN applied to fall detection and 3-D motion reconstruction," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 2, pp. 457–466, 2014.

[10] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," *IEEE Telecommunications Forum Telfor*, pp. 111–114, 2015.

[11] J. Shen, S. Chang, J. Shen, Q. Liu, and X. Sun, "A lightweight multi-layer authentication protocol for wireless body area networks," *Future Generation Computer Systems*, 2016.

[12] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE Network*, vol. 30, no. 1, pp. 58–65, 2016.

[13] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," *Lecture Notes in Computer Science*, vol. 2947, no. 39, pp. 277–290, 2004.

[14] H. Lung, Y. Chuang, and C. Kuo, "A novel remote user authentication scheme from bilinear pairings via internet," *Wireless Personal Communications*, vol. 83, no. 1, pp. 163–174, 2015.

[15] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 65, no. 10, pp. 3184–3195, 2016.

[16] J. Shen, S. Moh, and I. Chung, "Identity-Based key agreement protocol employing a symmetric balanced incomplete block design," *Journal of Communications and Networks*, vol. 14, no. 6, pp. 682–691, 2012.

[17] J. Herranz and G. Sáez, "Forking lemmas for ring signature schemes," *Journal of Management Studies*, vol. 2904, no. 2, pp. 266–279, 2003.

[18] G. Han, L. Liu, S. Chan, R. Yu, and Y. Yang, "HySense: a hybrid mobile crowd sensing framework for sensing opportunities compensation under dynamic coverage constraint," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 93–99, 2017.

[19] F. Zhang and K. Kim, "ID-based blind signature and ring signature from pairings," *Asiacrpt Lncs*, vol. 2501, pp. 533–547, 2002.

[20] J. Herranz and G. Sáez, "New Identity-Based Ring Signature Schemes," in *Information and Communications Security*, vol. 3269 of *Lecture Notes in Computer Science*, pp. 27–39, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[21] S. S. Chow, S. Yiu, and L. C. Hui, "Efficient Identity Based Ring Signature," *International Conference on Applied Cryptography and Network Security*, vol. 3531, pp. 499–512, 2005.

[22] G. Jia, G. Han, J. Jiang, and L. Liu, "Dynamic adaptive replacement policy in shared last-level cache of DRAM/PCM hybrid memory for big data storage," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1951–1960, 2017.

[23] J. Shen, H. Tan, S. Moh, I. Chung, Q. Liu, and X. Sun, "Enhanced secure sensor association and key management in wireless body area networks," *Journal of Communications and Networks*, vol. 17, no. 5, pp. 453–462, 2015.

[24] D. Boneh, "BLS Short Digital Signatures," *Encyclopedia of Cryptography and Security*, pp. 49-50, 2011.

[25] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, 2017.

[26] G. Wei, F. Li, and B. Xu, "An abuse-free optimistic fair exchange protocol based on BLS signature," in *Proceedings of the 2008 International Conference on Computational Intelligence and Security, CIS 2008*, pp. 278–282, Suzhou, China, December 2008.

[27] G. Han, W. Que, G. Jia, and W. Zhang, "Resource-utilization-aware energy efficient server consolidation algorithm for green computing in IIOT," *Journal of Network and Computer Applications*, 2017.

[28] T. Ma, Y. Zhang, J. Cao et al., "KDVEM: a k-degree anonymity with vertex and edge modification algorithm," *Computing*, vol. 97, no. 12, pp. 1165–1184, 2015.

[29] L. I. Xin, H. Y. Liu, M. A. Hui-Ran, and X. G. Cheng, "Group signature scheme based on BLS short signature," *Journal of Qingdao University*, 2011.

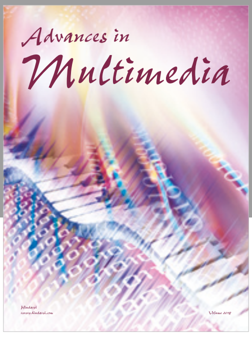
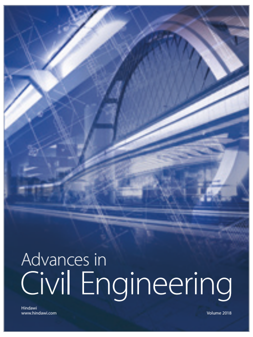
[30] J. Zhang, Y. He, and X. Li, "Cryptanalysis and improvement of two verifiable sing signature schemes," *Computer Engineering and Applications*, vol. 52, no. 8, pp. 115–119, 2016.

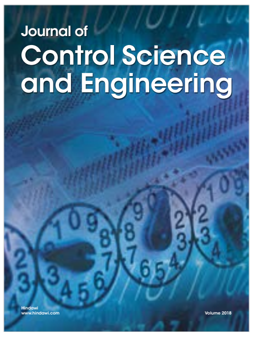
[31] X. Li, X. Liang, K. Liu, and S. Pan, "Analysis and improvement of verifiable ring signature schemes," *Journal of Computer Applications*, vol. 32, no. 12, pp. 3466–3469, 2012.

Journal of Engineering

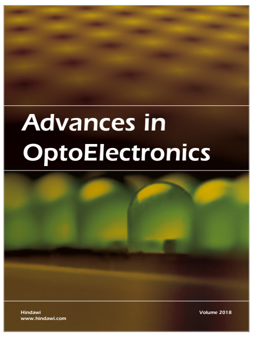
The Scientific World Journal

International Journal of Rotating Machinery

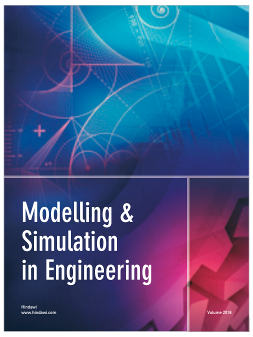
Journal of Sensors

Advances in Multimedia

Advances in Civil Engineering

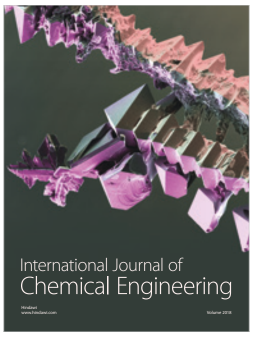
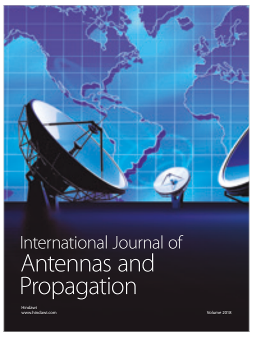

Journal of Control Science and Engineering

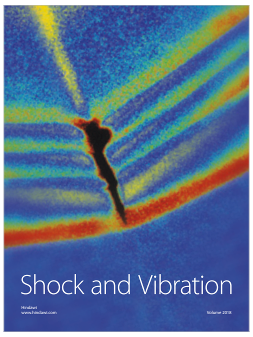
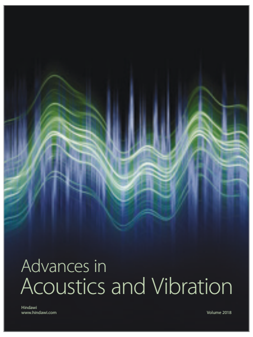
Journal of Robotics

Journal of Electrical and Computer Engineering

Advances in OptoElectronics

VLSI Design

International Journal of Navigation and Observation

Modelling & Simulation in Engineering

International Journal of Aerospace Engineering

International Journal of Chemical Engineering

International Journal of Antennas and Propagation

Active and Passive Electronic Components

Shock and Vibration

Advances in Acoustics and Vibration