Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 5327949, 13 pages
http://dx.doi.org/10.1155/2016/5327949

Research Article

A Security Authentication Protocol for Trusted Domains in an Autonomous Decentralized System

Ruikang Zhou,[1,2,3,4] Yingxu Lai,[2,3,4] Zenghui Liu,[5] Yinong Chen,[6] Xiangzhen Yao,[1] and Jiezhong Gong[1]

[1]*China Electronics Standardization Institute, Beijing 100007, China*
[2]*College of Computer Science, Beijing University of Technology, Beijing 100124, China*
[3]*Beijing Key Laboratory of Trusted Computing, Beijing 100124, China*
[4]*National Engineering Laboratory for Critical Technologies of Information Security Classified Protection, Beijing 100124, China*
[5]*Automation Engineering Institute, Beijing Polytechnic, Beijing 100176, China*
[6]*School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA*

Correspondence should be addressed to Yingxu Lai; laiyingxu@bjut.edu.cn

Received 8 October 2015; Revised 24 December 2015; Accepted 30 December 2015

Academic Editor: Luis Javier Garcia Villalba

Copyright © 2016 Ruikang Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Software Defined Network (SDN) architecture has been widely used in various application domains. Aiming at the authentication and security issues of SDN architecture in autonomous decentralized system (ADS) applications, securing the mutual trust among the autonomous controllers, we combine trusted technology and SDN architecture, and we introduce an authentication protocol based on SDN architecture without any trusted third party between trusted domains in autonomous systems. By applying BAN predicate logic and AVISPA security analysis tool of network interaction protocol, we can guarantee protocol security and provide complete safety tests. Our work fills the gap of mutual trust between different trusted domains and provides security foundation for interaction between different trusted domains.

1. Introduction

Autonomous decentralized systems (ADS) have been extended and applied to a variety of domains [1–6]. The main extension in ADS is to implement the dynamic management and to optimize the allocation of virtualized computing, storage, device controllers, and cyber resources. The current major platforms are mature in management and optimization of traditional computing and storage resources. However, the dynamic management and the allocation of ADS devices as cyber resources are a new problem which has not been studied thoroughly. The problem is mainly from network architecture design. The traditional TCP/IP architecture cannot meet the increasing demands, especially the virtualized network and ADS services. Thus, ADS needs new network architecture and techniques with flexibility, robustness, security, and credibility to solve the problem.

In the next generation network architecture design, many experts and scholars have completed a series of remarkable research work. Specifically, OpenFlow [7], introduced by Nick McKeown, gave researchers a way to run experimental protocols in the networks. Based on Software Defined Network (SDN) and OpenFlow protocol, researchers implemented network management and security functions mainly in the aspects of control, traffic forwarding, and load balancing. But the security issue in OpenFlow design should be thoroughly concerned, especially the security interaction function between different controllers' nodes. The problem of security certification issues between the different single controllers limits the interaction range of a SDN architecture, which is only used in a single domain with a single controller, such as the Data Center.

In order to solve secure interaction issues, experts and scholars have completed a series of research work. Reference [7] proposed the SANE network architecture in which logic controllers could provide secure authentication for device access and loading strategy. Moreover, [8] further extended SANE network architecture with data forwarding strategy

by using core controller method. The improved architecture implemented with data forwarding function, in a certain extent, could solve the security threats between controller layer and data forwarding layer.

Reference [9] implemented a new network architecture GENI and recognized that a large number of malicious attacks and flood attacks can be easily spread through logic control layers and data forwarding layers and lead to corresponding secure issues, for instance, DDoS attack.

Reference [10] evaluated the weakness of OpenFlow protocol. The author thought OpenFlow had an accurate secure certification method between controllers and switches, but this method did not provide security mechanisms under transport layer.

Based on SDN architecture, [11] did a comprehensive analysis for evaluating NOX, POX, Ryu, and Beacon [12–15] controllers in compatibility, reliability, security, and processing capacity. Focusing on the aspect of security, the authors pointed out that the reason to most controllers' security problems was forged stream message length, protocol version, or wrong type of message flow.

Besides, in [16], the authors analyzed the entire SDN security issues and pointed out that the new network architecture with a rapid response and antithreatening security mechanism was needed to be reestablished, especially in the differences of security threats between controllers and traditional network.

Based on the above issues, one of the key features of ADS is to allow content-based message broadcasting. In order to ensure the real-time system, message package could not be encrypted. The receiver node does not check whether the source address is valid or not, and whether the message package has been tampered or not. It means that an attacker can set up the entire attack in essentially one operation. While the security for ADS is drawing attention, it is an indisputable fact that there is a lack of effective methods to support those frameworks. The contributions of our paper are as follows:

(1) Our paper introduces a new architecture to increase the probability of guaranteeing the credibility for future network architecture by applying the ideas of trusted network to ADS. The new architecture with trusted function modules can measure sensitive information and provide a security interaction function between different SDN domains.

(2) Based on the new architecture with trusted function modules, we propose a trusted domain authentication protocol that protects controllers' credibility among entire network architecture when communicating with a nontrusted third party. Trusted function modules, such as Trusted Measurement Module (TMM) and TCG Software Stack (TSS), could provide a set of services to ensure authentication protocol's security, for example, encryption, decryption, digital sign, or key management. The protocol gets trust certifications among different trusted domains and provides secure base in domain session.

(3) We demonstrate security of our trusted domain authentication protocol by BAN logic and AVISPA security analysis system and perform a comparative analysis of our trusted domain authentication protocol against some prevalent protocols, namely, IKEv2 [17], IDAKE-MA, and SKAP [18], in performance analysis. We believe that the protocol performance, including calculating consumption and storage consumption, is an index which can give us obviously evidence to prove advantage of our protocol.

This paper is organized as follows. Section 2 introduces a SDN trusted domain network architecture and describes our trusted domain authentication protocol. In Section 3, we demonstrate security of our protocol by the BAN logic [19]. In Section 4, we analyze security of our protocol with different malicious test scenarios by AVISPA security analysis system [20]. Section 5 performs an analysis between our protocol and some authentication protocols.

2. Security Authentication Protocol for ADS Applications

As shown in Figure 1, we propose a SDN trusted domain network architecture which combines SDN and trusted computing. This architecture contains a single controller and a number of network devices. For solving the trust certification problem among trusted domains, we design some modules in SDN controller. TMM, based on TSS [21], is used to measure the sensitive information of SDN controller and connect network devices. Sensitive information includes controller platform hardware information, controller platform OS information, controller software information, and trusted function modules information. Controller Flow Rule (CFR) is trusted policy, including SDN data forwarding strategy and trusted measurement strategy, which was measured by TMM. Controller Communication Module (CCM) ensures controller authentication process between individual trusted domains.

In SDN trusted domain network architecture, we propose a domain secure certificated protocol between SDN trusted domains, and, in particular, our protocol can be utilized on nontrusted third party circumstance. From the viewpoint of the trusted chain [22], we can implement a consistency test of measuring information integrity about controller hardware, operating system, controller software, and CCM for trusted negotiation. After finishing trusted negotiation, we can adopt the method of multilevel authentication to guarantee security. More specific, controller authentication can be divided into two parts, controller platform authentication and controller software authentication, which protects the release of sensitive information and prevents unauthorized users from capturing sensitive information. The combination of sensitive information and random numbers, based on the strong protection of sensitive information, ensures that sensitive information is not being hijacked and not subject to replay attacks by unauthorized users, thereby avoiding the security of sensitive information by refusing the connection of those illegal or security risks.

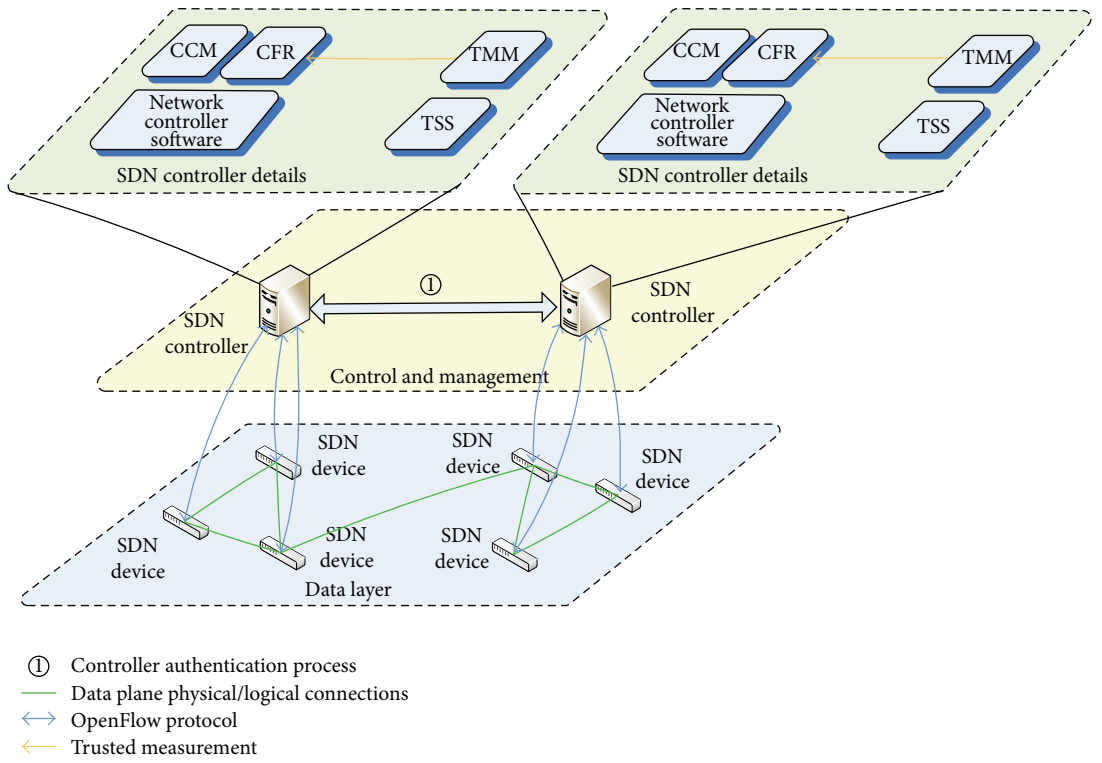

Figure 1: SDN trusted domain network architecture.

2.1. Protocol Certification Process Overview. Before we design our security certification protocol, we present the three major aspects of our protocol certification process.

First, in accordance with the trust chain transfer rules in trusted computing, the TMM is based on TSS and will follow the orders by measuring sensitive information of SDN controller hardware, operating system, CCM, and storing measurement results into RTS (Root Trusted Storage) of the PCR register.

Second, the controller platform certification: when implementing different trusted domain controller authentication, authentication requester sends the HMAC calculation results of hardware information, operating system, and random numbers to the receiver, respectively, and expects the identical HMAC result with the receivers. If the controller platform's HMAC result of the receiver is consistent with that of the requester, then we complete the certification process of *controller platform*.

Finally, the controller software certification: in terms of implementing the trusted authentication of sensitive information in controller software, the requester sends HMAC calculation results, including controller software metrics, core controller module metrics, and random numbers, to the receiver. As shown in the controller platform certification in Figure 1, the receiver will compare the controller software HMAC results with that of the requester. If their results are consistent, we complete the certification of the *controller software*.

2.2. Specific Certification Process. First, we define the related symbols used in the certification process:

(1) R: authentication requester,

(2) N: authentication receiver,

(3) N_{PUB}: authentication requester public key,

(4) R_{PUB}: authentication receiver public key,

(5) N^{-1}: authentication requester private key,

(6) R^{-1}: authentication receiver private key,

(7) Plat_ID_R: authentication requester platform ID,

(8) Plat_ID_N: authentication receiver platform ID,

(9) Nonce_R: random number of authentication requester, used to measure HMAC and prevent replay attack,

(10) Nonce_N: random number of authentication receiver, used to measure HMAC and prevent replay attack,

(11) AK_R: random number of authentication requester, used to make a session key by authentication receiver,

(12) AK: session key,

(13) ACK: authentication successful symbol,

(14) CS_ID_R: controller software ID of authentication requester,

(15) CS_ID_N: controller software ID of authentication receiver,

(16) R_PCR: controller platform hardware/controller platform OS/controller software/controller application module measurement of authentication requester,

(17) N_PCR: controller platform hardware/controller platform OS/controller software/controller application module measurement of authentication receiver,

(18) HMAC(): hash function based on HMAC_SHA-1 of TPM.

Figure 2 shows the specific certification process. More details will be introduced in following two subsections.

2.2.1. Controller Platform Certification Process. Controller platform certification process consists of the following steps.

Step 1 (R (the requesting controller) sends an authentication request to N (the receiving controller)). First, R creates a digital signature for its own controller platform identity information Plat_ID_R, random numbers Nonce_R, and hardware sensitive information $R\text{\_PCR}_1$ by its own private key. Then, the public key of N is expected to encrypt the HMAC value, and finally R sends the encrypted message to N:

$$R \longrightarrow N : \left\{\left\{\text{HMAC}\left(R\text{\_PCR}_1, \text{Nonce}_R\right) \parallel \text{Nonce}_R \parallel \text{Plat\_ID}_R\right\} R^{-1}\right\} N_{\text{PUB}}. \quad (1)$$

Step 2 (N receives the authorized information from R). First, N receives the encrypted message including R's ID and N_{PUB} from Step 1. Then, N will implement a negotiated registration by using R's platform identity information. And then, aiming at the hardware sensitive information in the R's platform, N implements a credible verification which uses PCR values of N's hardware sensitive information to make HMAC with Nonce_R. If the HMAC result is consistent with N received information in Step 1, R's hardware is trusted. Finally, if the verification is completed, N will create a digital signature for signing controller platform identity information Plat_ID_N, random numbers Nonce_N, and hardware sensitive information $N\text{\_PCR}_1$ and correspondingly use the public key of R to encrypt them and send the result to R. On the contrary, the negotiation fails:

$$N \longrightarrow R : \left\{\left\{\text{HMAC}\left(N\text{\_PCR}_1, \text{Nonce}_N\right) \parallel \text{Nonce}_N \parallel \text{Plat\_ID}_N\right\} N^{-1}\right\} R_{\text{PUB}}. \quad (2)$$

Step 3 (R receives the authorized information from N). First, R receives the verification feedback from N, which is encrypted by the public key of R, indicating that there is a trusted negotiation between R and N. Second, Plat_ID_N could similarly implement its negotiated registration. Third, as shown in Step 2, R begins the credible verification with the hardware sensitive information of N; if the results are consistent, N can be trusted in hardware. Finally, if the verification passes, R will generate trust negotiation session random numbers; meanwhile N creates the negotiated private key and implements the next round trust negotiation. R will sign sensitive information of controller platform operating system HMAC value and encrypt the sensitive information by session key. The negotiation fails when the verification is not successfully passed:

$$R \longrightarrow N : \left\{\left\{\text{HMAC}\left(R\text{\_PCR}_2, \text{Nonce}_R\right) \parallel \text{AK}_R\right\} R^{-1}\right\} N_{\text{PUB}}. \quad (3)$$

Step 4 (N receives the HMAC sensitive information of the controller platform operating system $R\text{\_PCR}_2$ and key random numbers of R, AK_{R}). First, N receives $R\text{\_PCR}_2$ and implements a credible verification, as shown in Step 2, with the operating system measurement result $N\text{\_PCR}_2$ of its own controller. Second, if the verification passes, N will produce a negotiated private key random number AK_N. Then, using AK_R, AK_N, the pseudo random numbers, and function PRGF, N will create a controller software verification session private key AK, which is bound with the platform information of R. Finally, N's platform sends the sensitive information HMAC value of N to R. The negotiation fails if the credible verification is not successfully passed:

$$N \longrightarrow R : \left\{\left\{\text{HMAC}\left(N\text{\_PCR}_2, \text{Nonce}_N\right) \parallel \text{ACK} \parallel \text{AK}\right\} N^{-1}\right\} R_{\text{PUB}}. \quad (4)$$

2.2.2. Controller Software Certification Process. In this subsection, we continue to explain the steps in certification process. These steps are related to the controller software certification process.

Step 5 (R receives the sensitive information HMAC value from the receiving controller platform operating system). First, R receives the operating system sensitive information from the receiving controller platform and implements a

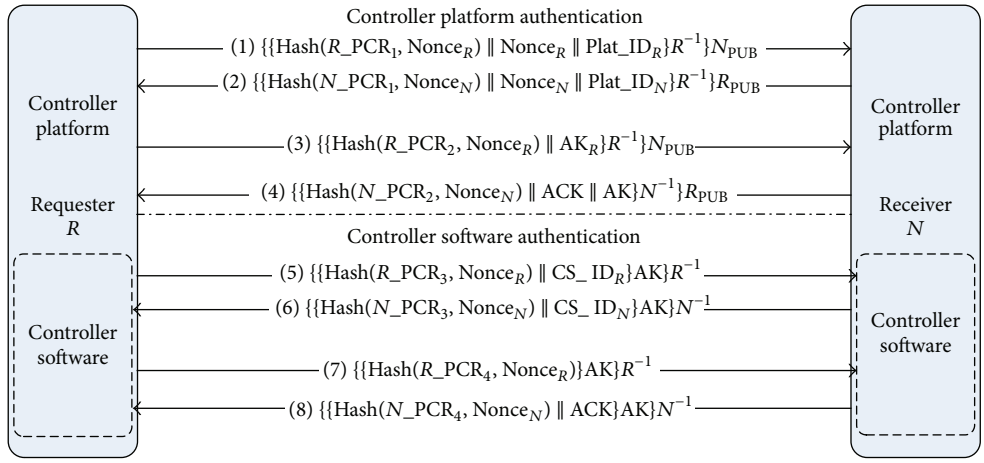

Figure 2: Trusted domain authentication process.

credible verification, as shown in Step 2, with the operating system measurement result R_PCR_2 of its own controller platform. Then, if the credible verification passes, the trust negotiation will enter into the controller software verification process, and, therefore, session private key AK will bind the information of N and also sign the encrypted information through R's private key. Finally, R returns the signed information to N. The negotiation fails if the credible verification fails:

$$R \longrightarrow N : \{\{\text{HMAC}(R\_PCR_3, \text{Nonce}_R) \parallel \text{CS\_ID}_R\}\, \text{AK}\}\, R^{-1}. \quad (5)$$

Step 6 (N receives the controller software verification request from R). First, N decrypts the information by R's public key, proving the existing base of the trust negotiation between R and N. Then, using the decryption of the sensitive information HMAC value of the controller software, N implements a credible verification as shown in Step 2. If the credible verification passes, N will proceed a negotiated registration using the controller software information CS_ID_R of R, encrypt the sensitive information HMAC value of controller software and the controller software ID through session private key AK, and sign the encrypted information through R's private key. Finally, N returns the signed information to R. The negotiation fails if the credible verification is not successfully passed:

$$N \longrightarrow R : \{\{\text{HMAC}(N\_PCR_3, \text{Nonce}_N) \parallel \text{CS\_ID}_N\}\, \text{AK}\}\, N^{-1}. \quad (6)$$

Step 7 (R receives the verification information from N). First, as shown in Step 6, using the sensitive information HMAC value of the controller software, the receiving controller implements a credible verification as shown in Step 2. Then, if the credible verification passes, R will implement a negotiated registration using the controller software information of N. R encrypts the sensitive information HMAC value of R_PCR_4 and Nonce_R through session private key AK and signs the encrypted information by R's private key. Finally, N returns the signed information to R. The negotiation fails if the credible verification is not successfully passed:

$$R \longrightarrow N : \{\{\text{HMAC}(R\_PCR_4, \text{Nonce}_R)\}\, \text{AK}\}\, R^{-1}. \quad (7)$$

Step 8. First, receiving controller software application modules HMAC value of R, N implements a credible verification as shown in Step 2. Then, if the credible verification passes, N will encrypt the sensitive information HMAC value of N_PCR_4 and Nonce_N by session private key AK and sign the encrypted information by R's private key. Finally, R implements a credible verification on application module sensitive information of N, shown in Step 2. If the verification passes, the verification of the controller software will be completed and the controller verification will pass. Otherwise, the negotiation fails:

$$N \longrightarrow R : \{\{\text{HMAC}(N\_PCR_4, \text{Nonce}_N) \parallel \text{ACK}\}\, \text{AK}\}\, N^{-1}. \quad (8)$$

3. BAN Logic Security Analysis

According to the BAN logic security analysis, this section analyzes whether our authentication protocol is secure or not.

3.1. BAN Logic Security Analysis. BAN predicate logic is composed of 10 basic syntax and semantic clauses:

(1) $Q \mid\equiv A$ Q believes that A is credible.

(2) $Q \triangleleft A$ Q receives message A.

(3) $Q \mid\sim A$ Q has sent message A.

(4) $Q \mid\Rightarrow A$ Q has jurisdiction for message A.

(5) $\#(A)$: message A is fresh, which means that A is temporary value and not the first to be sent as the information.

(6) $P \overset{\text{Key}}{\longleftrightarrow} Q$: the key is shared key between P and Q.

(7) $\overset{\text{Key}}{\longrightarrow} Q$ or $\mid\overset{\text{Key}}{\longrightarrow} Q$: key is Q's public key.

(8) $P \overset{X}{\rightleftarrows} Q$ X events are shared secrets between P and Q.

(9) $\{X\}_{\text{Key}}$: message X is encrypted by key.

(10) $\langle X \rangle_Y$ is the cascade of information of X and Y.

3.2. BAN Logic Inference Rules. This section describes the four main specific rules of BAN logic to provide a theoretical basis for SDN trusted domains security authentication protocol. $\vdash$ is primitive symbol, such as $P \vdash Q$ representing that P derives conclusions Q.

(1) Message Meaning Rule

Theorem 1. *Consider*

$$\begin{aligned} P \mid\equiv \stackrel{K}{\longrightarrow} Q, \\ P \lhd \{X\}_{K^{-1}} \vdash P \mid\equiv Q \mid\sim X. \end{aligned} \tag{9}$$

Explanation. If P believes that Q's public key is K, and P has received the message $\{X\}_{K^{-1}}$ which is encrypted by K^{-1}, we can infer that P believes message X which was sent by Q.

(2) Temporary Value Validation Rules

Theorem 2. *Consider*

$$\begin{aligned} P \mid\equiv \#(X), \\ P \mid\equiv Q \mid\sim X \vdash P \mid\equiv Q \mid\equiv X. \end{aligned} \tag{10}$$

Explanation. If P believes that message X is fresh, and P believes that Q has sent message X, we can infer that P believes that Q believes the message X.

(3) Jurisdiction Rules

Theorem 3. *Consider*

$$\begin{aligned} P \mid\equiv Q \mid\Rightarrow X, \\ P \mid\equiv Q \mid\equiv X \vdash P \mid\equiv X. \end{aligned} \tag{11}$$

Explanation. If P believes that Q has jurisdiction for message X, and P believes that Q believes message X, we can infer that P believes the message X.

(4) Receive Messages Rules

Theorem 4. *Consider*

$$P \lhd (X, Y) \vdash P \lhd X. \tag{12}$$

Theorem 5. *Consider*

$$\begin{aligned} P \mid\equiv \stackrel{K}{\longrightarrow} Q, \\ P \lhd \{X\}_{K^{-1}} \vdash P \lhd X. \end{aligned} \tag{13}$$

Theorem 6. *Consider*

$$\begin{aligned} P \mid\equiv P \stackrel{K}{\longleftrightarrow} Q, \\ P \lhd \{X\}_{K} \vdash P \lhd X. \end{aligned} \tag{14}$$

Explanation. The above theorem states that if a subject has received a formula, and the main part knows the relevant secret key, we can infer that the body has received part of the formula.

(5) Belief Rules

Theorem 7. *Consider*

$$P \mid\equiv Q \mid\sim (X, Y) \vdash P \mid\equiv Q \mid\sim X. \tag{15}$$

Explanation. If P believes that Q has sent a (X, Y), then P believes that Q has sent message X.

3.3. SDN Trusted Domain Security Authentication Protocol's Formal Presentation. In this section, we formalize SDN trusted domain security authentication protocol:

$$\begin{aligned} &R \longrightarrow N : \left\{\{\mathrm{HMAC}(R\_\mathrm{PCR}_1, \mathrm{Nonce}_R) \parallel \mathrm{Nonce}_R \parallel \mathrm{Plat\_ID}_R\} R^{-1}\right\} N_{\mathrm{PUB}}, \\ &N \longrightarrow R : \left\{\{\mathrm{HMAC}(N\_\mathrm{PCR}_1, \mathrm{Nonce}_N) \parallel \mathrm{Nonce}_N \parallel \mathrm{Plat\_ID}_N\} N^{-1}\right\} R_{\mathrm{PUB}}, \\ &R \longrightarrow N : \left\{\{\mathrm{HMAC}(R\_\mathrm{PCR}_2, \mathrm{Nonce}_R) \parallel \mathrm{AK}_R\} R^{-1}\right\} N_{\mathrm{PUB}}, \\ &N \longrightarrow R : \left\{\{\mathrm{HMAC}(N\_\mathrm{PCR}_2, \mathrm{Nonce}_N) \parallel \mathrm{ACK} \parallel \mathrm{AK}\} N^{-1}\right\} R_{\mathrm{PUB}}, \\ &R \longrightarrow N : \{\{\mathrm{HMAC}(R\_\mathrm{PCR}_3, \mathrm{Nonce}_R) \parallel \mathrm{CS\_ID}_R\} \mathrm{AK}\} R^{-1}, \\ &N \longrightarrow R : \{\{\mathrm{HMAC}(N\_\mathrm{PCR}_3, \mathrm{Nonce}_N) \parallel \mathrm{CS\_ID}_N\} \mathrm{AK}\} N^{-1}, \\ &R \longrightarrow N : \{\{\mathrm{HMAC}(R\_\mathrm{PCR}_4, \mathrm{Nonce}_R)\} \mathrm{AK}\} R^{-1}, \\ &N \longrightarrow R : \{\{\mathrm{HMAC}(N\_\mathrm{PCR}_4, \mathrm{Nonce}_N) \parallel \mathrm{ACK}\} \mathrm{AK}\} N^{-1}. \end{aligned} \tag{16}$$

3.4. BAN Logic Security Goals. For security requirements of SDN trusted domains security authentication protocol, this paper sets up multiple security goals:

(1) $N \mid\equiv R \mid\equiv R\_\mathrm{PCR}$: N believes that R is credible to believe R_PCR.

(2) $R \mid\equiv N \mid\equiv N\_\mathrm{PCR}$: R believes that N is credible to believe N_PCR.

(3) $N \mid\equiv R \mid\equiv \mathrm{Nonce}_R$: N believes that R is credible to believe Nonce_R.

(4) $R \mid\equiv N \mid\equiv \mathrm{Nonce}_N$: R believes that N is credible to believe Nonce_N.

(5) $N \mid\equiv R \mid\equiv \mathrm{Plat\_ID}_R$: N believes that R is credible to believe $\mathrm{Plat\_ID}_R$.

(6) $R \mid\equiv N \mid\equiv \mathrm{Plat\_ID}_N$: R believes that N is credible to believe $\mathrm{Plat\_ID}_N$.

(7) $N \mid\equiv \mathrm{AK}_R$: N believes that AK_R is credible.

(8) $R \mid\equiv \mathrm{AK}$: R believes that AK is credible.

(9) $N \mid\equiv R \mid\equiv \mathrm{CS\_ID}_R$: N believes that R is credible to believe $\mathrm{CS\_ID}_R$.

(10) $R \mid\equiv N \mid\equiv \mathrm{CS\_ID}_N$, $R \mid\equiv N \mid\equiv \mathrm{ACK}$: R believes that N is credible to believe $\mathrm{CS\_ID}_N$.

(11) R believes that N is credible to believe ACK.

3.5. BAN Logic Initialization Assumptions. In order to verify the authentication protocol compliance with BAN logic, we make the initialization assumptions of BAN logic:

$$\begin{gathered} R \mid\equiv \xrightarrow{N_{\mathrm{PUB}}} N, \\ N \mid\equiv \xrightarrow{R_{\mathrm{PUB}}} R, \\ N \mid\equiv R \mid\Rightarrow \mathrm{AK}_R, \\ N \mid\equiv \#\left(R\_\mathrm{PCR}\right), \\ N \mid\equiv \#\left(\mathrm{Nonce}_R\right), \\ N \mid\equiv \#\left(\mathrm{AK}_R\right), \\ R \mid\equiv \#\left(N\_\mathrm{PCR}\right), \\ R \mid\equiv \#\left(\mathrm{Nonce}_N\right). \end{gathered} \tag{17}$$

3.6. BAN Logic Secure Analysis. This section will use the BAN predicate logic inference to verify the authentication protocol's security goals, which is based on the BAN logic initialization assumptions of Section 3.5.

(1) Inference 1: From Step 1

$$R \longrightarrow N : \left\{\left\{\mathrm{HMAC}\left(R\_\mathrm{PCR}_1, \mathrm{Nonce}_R\right) \parallel \mathrm{Nonce}_R \parallel \mathrm{Plat\_ID}_R\right\} R^{-1}\right\} N_{\mathrm{PUB}}. \tag{18}$$

⟨1⟩ $\because N \mid\equiv \xrightarrow{R_{\mathrm{PUB}}} R$,

$$N \lhd \left\{\mathrm{Hash}\left(R\_\mathrm{PCR}_1, \mathrm{Nonce}_R\right) \parallel \mathrm{Nonce}_R \parallel \mathrm{Plat\_ID}_R \parallel \mathrm{Plat\_ID}_R\right\} R^{-1}. \tag{19}$$

⟨2⟩ $\therefore$ From Theorem 1, we can infer

$$N \mid\equiv R \mid\sim \left\{\mathrm{Hash}\left(R\_\mathrm{PCR}_1, \mathrm{Nonce}_R\right) \parallel \mathrm{Nonce}_R \parallel \mathrm{Plat\_ID}_R\right\} R^{-1}. \tag{20}$$

⟨3⟩ $\because$ From Theorems 5 and 7 $\therefore$ we can infer

$$\begin{gathered} N \mid\equiv R \mid\sim R\_\mathrm{PCR}_1, \\ N \mid\equiv R \mid\sim \mathrm{Nonce}_R, \\ N \mid\equiv R \mid\sim \mathrm{Plat\_ID}_R. \end{gathered} \tag{21}$$

⟨4⟩ $\because N \mid\equiv \#\left(R\_\mathrm{PCR}\right)$ and $N \mid\equiv \#\left(\mathrm{Nonce}_R\right)$ and $N \mid\equiv \#\left(\mathrm{Plat\_ID}_R\right)$.

⟨5⟩ $\therefore$ From Theorem 2, we can get the conclusions

$$\begin{gathered} N \mid\equiv R \mid\equiv R\_\mathrm{PCR}, \\ N \mid\equiv R \mid\equiv \mathrm{Nonce}_R, \\ N \mid\equiv R \mid\equiv \mathrm{Plat\_ID}_R. \end{gathered} \tag{22}$$

(2) Inference 2: From Step 2

$$N \longrightarrow R : \left\{\left\{\mathrm{HMAC}\left(N\_\mathrm{PCR}_1, \mathrm{Nonce}_N\right) \parallel \mathrm{Nonce}_N \parallel \mathrm{Plat\_ID}_N\right\} N^{-1}\right\} R_{\mathrm{PUB}}. \tag{23}$$

⟨1⟩ Same as inference 1, we can get the conclusions

$$R \mid\equiv N \mid\equiv N\_\mathrm{PCR},$$
$$R \mid\equiv N \mid\equiv \mathrm{Nonce}_N, \quad (24)$$
$$R \mid\equiv N \mid\equiv \mathrm{Plat\_ID}_N.$$

(3) Inference 3: From Step 3

$$R \longrightarrow N : \left\{\left\{\mathrm{HMAC}\left(R\_\mathrm{PCR}_2, \mathrm{Nonce}_R\right) \parallel \mathrm{AK}_R\right\} R^{-1}\right\} N_{\mathrm{PUB}}. \quad (25)$$

⟨1⟩ $\because N \mid\equiv \xrightarrow{R_{\mathrm{PUB}}} R$,

$$N \mid\sim \left\{\mathrm{Hash}\left(R\_\mathrm{PCR}_2, \mathrm{Nonce}_R\right) \parallel \mathrm{AK}_R\right\} R^{-1}. \quad (26)$$

⟨2⟩ $\therefore$ From Theorem 1, we can infer

$$N \mid\equiv R \mid\sim \left\{\mathrm{Hash}\left(R\_\mathrm{PCR}_2, \mathrm{Nonce}_R\right) \parallel \mathrm{AK}_R\right\} R^{-1}. \quad (27)$$

⟨3⟩ $\because$ From Theorems 5 and 7

$\therefore$ we can infer $N \mid\equiv R \mid\sim \mathrm{AK}_R$.

⟨4⟩ $\because N \mid\equiv \#(\mathrm{AK}_R)$.

$\therefore$ From Theorem 2, we can infer

$$N \mid\equiv R \mid\equiv \mathrm{AK}_R. \quad (28)$$

⟨5⟩ $\because N \mid\equiv R \mid\Rightarrow \mathrm{AK}_R$ and $N \mid\equiv R \mid\equiv \mathrm{AK}_R$.

From Theorem 3, we can infer $N \mid\equiv \mathrm{AK}_R$.

(4) Inference 4: From Step 4

$$N \longrightarrow R : \left\{\left\{\mathrm{HMAC}\left(N\_\mathrm{PCR}_2, \mathrm{Nonce}_N\right) \parallel \mathrm{ACK} \parallel \mathrm{AK}\right\} N^{-1}\right\} R_{\mathrm{PUB}}. \quad (29)$$

⟨1⟩ Same as inference 3, we can get the conclusions

$$R \mid\equiv \mathrm{AK}. \quad (30)$$

(5) Inference 5: From Steps 5 and 6

$$\begin{aligned} R &\longrightarrow N : \left\{\left\{\mathrm{HMAC}\left(R\_\mathrm{PCR}_3, \mathrm{Nonce}_R\right) \parallel \mathrm{CS\_ID}_R\right\} \mathrm{AK}\right\} R^{-1}, \\ N &\longrightarrow R : \left\{\left\{\mathrm{HMAC}\left(N\_\mathrm{PCR}_3, \mathrm{Nonce}_N\right) \parallel \mathrm{CS\_ID}_N\right\} \mathrm{AK}\right\} N^{-1}. \end{aligned} \quad (31)$$

⟨1⟩ Same as inference 1, we can get the conclusions

$$N \mid\equiv R \mid\equiv \mathrm{CS\_ID}_R,$$
$$R \mid\equiv N \mid\equiv \mathrm{CS\_ID}_N. \quad (32)$$

(6) Inference 6: From Step 8

$$N \longrightarrow R : \left\{\left\{\mathrm{HMAC}\left(N\_\mathrm{PCR}_4, \mathrm{Nonce}_N\right) \parallel \mathrm{ACK}\right\} \mathrm{AK}\right\} N^{-1}. \quad (33)$$

⟨1⟩ Same as inference 1, we can get the conclusions

$$R \mid\equiv N \mid\equiv \mathrm{ACK}. \quad (34)$$

From the above BAN predicate logic inference, we can get that the conclusions conform to the BAN logic security goals which are defined in Section 3.4. Obviously, PCR, random numbers, and session key are important to the authentication protocol. From the BAN predicate logic inference, we have proved that sensitive information can meet the demands of network security.

4. Protocol Security Testing

This section will use the Dolev-Yao (DY) attack model [23] for actual security testing. The DY attack model could have the following variety of knowledge:

(1) Attackers are familiar with encryption, decryption, hashing, and other cryptographic operations, and they have the public key and private key of themselves.

(2) Attackers hold the network identity and public key of each subject.

(3) Attackers have basic password analysis ability.

Attackers could perform a variety of attacks, such as replay attack.

4.1. Security Goals. The DY attack model can control the entire network and catch the data for tampering attack and replay attack. For ensuring the protocol security, the authentication protocol must be detected by DY attack model. To this end, this paper sets up multiple security objections [24] for the DY attack model:

(1) N_PCR/R_PCR value is confidential during transmission.

(2) Random number, Nonce_R/Nonce_N, is confidential during transmission.

(3) AK_R is confidential during transmission.

(4) Controller platform ID/controller software ID is confidential during transmission.

(5) Controller software authentication session key is confidential.

(6) Successful certification logo (ACK) is completed.

4.1.1. Security Goals Formalization. In response to these security goals, this paper will test the safety of the authentication protocol using AVISPA network analysis tool. For testing protocol security, the AVISPA network interaction protocol security analysis system makes a formalized definition about test method of protocol security goals. Description is shown below.

(1) Information Confidential Test

Secret (E, id, S). This is a statement which means that subject S shares information E, and this secret is named an unchanged id which will be used in the goals definition.

(2) Information Verification Test

Witness (A, B, id, E). This is a weak authentication attribute, which means that A declares that A has sent a message E to B, and this statement is named an unchanged id which will be used in the goals definition.

Request (B, A, id, E). This is a strong authentication attribute, which means that B declares that B has received a message E from A, and this statement is named an unchanged id which will be used in the goals definition.

WRequest (B, A, id, E). This is a weak authentication attribute, which is similar to Request (B, A, id, E), but it does not verify replay attacks.

4.1.2. Modeling and Simulating Security Goals. In this section we will use HLPSL to build security goals modeling, which is based on formal definition of security objectives.

(1) For testing the security goals in Section 4.1.1, the authentication requesting platform needs to verify PCR, random numbers, and platform id of the authentication received platform. The HLPSL code appears as shown in Box 1.

(2) Obviously, the authentication receiving platform also needs to verify PCR, random numbers, and platform id of the authentication requesting platform. The HLPSL code appears as shown in Box 2.

4.2. Security Test

4.2.1. Test Scenarios. For the security goals, this paper sets up three test scenarios to verify whether the protocol satisfies the security goals or not. As shown in Table 1, in Scenario 1,

```
Role sdnTNap_Init(A,B : agent,
Kab :symmetric_key,
H :hash_func,
Ap_Start, Ap_Init, Ap_aSuccess : text,
                    SND,RCV: channel(dy)
)
played_by A
init State :=0
transition
∧request(A,B,b_a_K1ab,K1ab')
∧request(A,B,b_a_SIGKpubKb,SIGKpubKb')
∧secret(K1ab',k1ab,{A,B})
∧request(A,B,b_a_bNonce,BNonce')
∧request(A,B,bpcr1,H(BPcr1'.BNonce'))
∧request(A,B,bpcr2,H(BPcr2'.BNonce'))
∧request(A,B,bpcr3,H(BPcr3'.BNonce'))
∧request(A,B,bpcr4,H(BPcr4'.BNonce'))
end role
```

Box 1

```
role sdnTNap_reply(A,B : agent,
                   Kab : symmetric_key,
                   H : hash_func,
Ap_Start,Ap_Init,Ap_aSuccess,Ap_bSuccess :
text,
                   SND,RCV:
channel(dy)
)
played_by B
init State :=1
transition
∧request(B,A,a_b_SIGKpubKa,SIGKpubKa')
∧request(B,A,a_b_aNonce,ANonce')
∧request(B,A,apcr1,H(APcr1'.ANonce'))
∧request(B,A,apcr2,H(APcr2'.ANonce'))
∧request(B,A,apcr3,H(APcr3'.ANonce'))
∧request(B,A,apcr4,H(APcr4'.ANonce'))
15.State = 15∧RCV(Ap_aSuccess') = |>
State':=17∧SND(Ap_bSuccess)
end role
```

Box 2

we implemented a single session with all the roles played by legitimate agents. In Scenario 2 and Scenario 3 we tested the situations, in which the intruder would impersonate each of the legitimate agents: the authentication requesting controller platform (Scenario 2) and the authentication receiving controller platform (Scenario 3).

4.2.2. Test Results. As shown in Boxes 3 and 4, the authentication protocol passed OFMC security test and ATSE security test, and the authentication protocol is not attacked by DY model, so we can conclude that the authentication protocol is secure.

From the summary details of Boxes 3 and 4, we can see that our authentication protocol is tested by CL-AtSe

Table 1: Scenario description of security test.

Scenario	Scenario description
(1)	session(a,b,kab,h, ap_start,ap_init,ap_Asuccess,ap_Bsuccess)
(2)	session(a,i,kai,h, ap_start,ap_init,ap_Asuccess,ap_Bsuccess)
(3)	session(i,b,kib,h, ap_start,ap_init,ap_Asuccess,ap_Bsuccess)

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  D:\...\NPLAB\temp\130476350162500000.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.04s
  visitedNodes: 1 nodes
  depth: 0 plies
```

Box 3: Security test result of OFMC system.

```
TYPED_MODEL
PROTOCOL
  D:\...\NPLAB\temp\130476352862187500.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed    : 4 states
  Reachable   : 0 states
  Translation: 0.14 seconds
  Computation: 0.00 seconds
```

Box 4: Security test result of ATSE system.

model and OFMC system, based on constraint logic, and our protocol was analyzed by 4 states. So, the conclusion of our test is quite convincing.

5. Performance Analysis

5.1. Calculation Consumption. This section uses the authentication response time defined in [25] to evaluate the calculation overhead of the protocol. The authentication response time (T) is mainly composed of three parts: the authentication requesting platform computing time (T_R), the authentication receiving platform computing time (T_N), and the protocol transmission time (T_T). The following relationship can be obtained through the above three variables:

$$T = T_R + T_N + T_T. \tag{35}$$

Thus, we define the authentication requesting platform computing time including operation overhead of HMAC, digital sign, and encryption and decryption.

Similarly, we define the authentication receiving platform computing time including operation overhead of HMAC, digital sign, and encryption and decryption.

In particular, we consider that the ideal transport network excludes the impact of network latency.

According to the above definition, the calculation overhead of authentication protocol is mainly composed of HMAC, digital sign, message encryption and decryption, and the network transmission. As shown in Table 2, we make a detailed comparison between IKEv2, IDAKE-MA [26], and SKAP in this paper.

It can be seen in Table 2 that the authentication protocol of this paper has higher calculation consumption compared with other domain protocols. In particular, we propose a no third party certification method. So our approach has advantages in communication frequency, which effectively reduces the total number of communications and network overhead. Furthermore, the authentication protocol is based on trusted computing, which effectively protects the credibility of network domains, and the trusted authentication protocol could make more advantages in security. Last but not the least, the authentication protocol does not have index calculation. Compared with other protocols in Table 2, our approach could significantly improve computational efficiency and reduce the cost of computing.

5.2. Storage Consumption. For the authentication protocol of this paper, we assume that the average frequency of the requester connecting to the receiver under random conditions is $1/T_{\text{access}}$, and the process of receiving request can be modeled as a Poisson process, where the average frequency is $1/T_{\text{access}}$. Using the receiver as an example, when the receiver receives a request message, the receiver needs to store the public key of the requester (R_{public}), the random number of the requester (Nonce_R), and the session key (AK) until the session is completed or the timer is over. We assumed the

Table 2: Comparison in the performance of protocols.

Protocol	Communication	Communication between domains	Encryption/decryption	Digital sign	Hash/index calculation
IKEv2	12	5	0	14	0/0
IDAKE-MA	14	6	6	5	0/2
SKAP	10	4	4	3	0/2
Our method	8	8	8	8	8/0

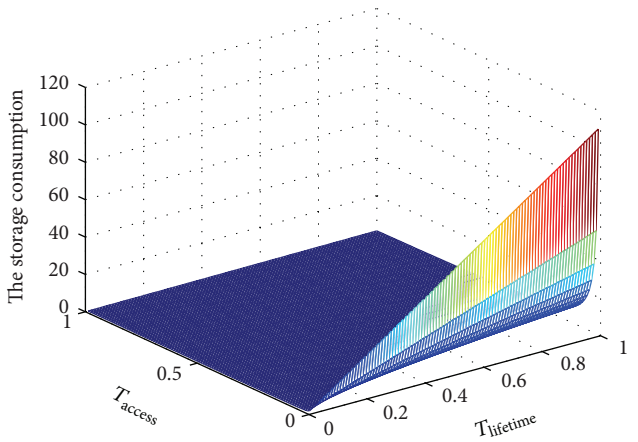

Figure 3: The storage consumption of our method.

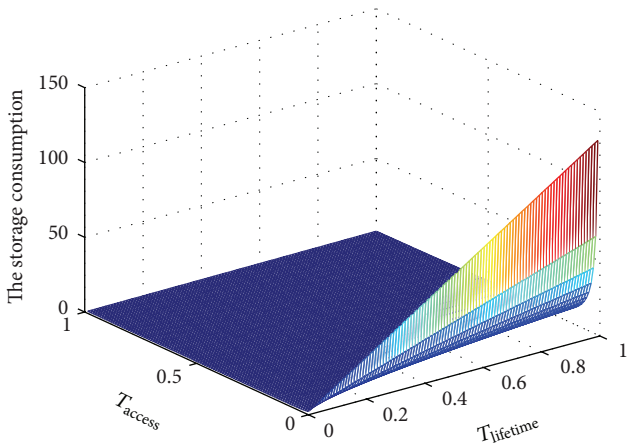

Figure 4: The storage consumption of SKAP.

time of wait timer is T_{lifetime}, the response time of receiving the message is T_{req}, and the packet loss rate of experiment network is P_{loss}, and we derived that the storage average amount of the receiver is

$$2 \times \left[\frac{1}{T_{\text{access}}} \times T_{\text{lifetime}} \times P_{\text{loss}} + \frac{1}{T_{\text{access}}} \times T_{\text{req}} \times \left(1 - P_{\text{loss}}\right) \right]. \tag{36}$$

Assuming the time of the latency timer is two times that of the response message time, we can further simplify that the storage average amount of the receiver is $T_{\text{lifetime}}/T_{\text{access}} \times (P_{\text{loss}} + 1)$.

According to the above formula, on the one hand, if T_{lifetime} and T_{access} remain unchanged, the storage consumption is proportional to the network packet loss rate. If the network is stable and packet loss rate is not high, the authentication protocol of proposed in this paper does not require a low storage space to store data, and the storage consumption is in the ideal range.

On the other hand, as shown in Figure 3, if P_{loss} remains unchanged, the storage consumption is proportional to T_{lifetime} and T_{access}. It means that if controllers' processing time is too long, the controller will continue to receive request, and thus the storage consumption will increase sharply.

So far we assumed that the P_{loss} rate remains unchanged. We further perform a comparative analysis with IKEv2, IDAKE-MA, and SKAP. The results are shown in Figures 4, 5, and 6; our method has more advantages in terms of storage consumption.

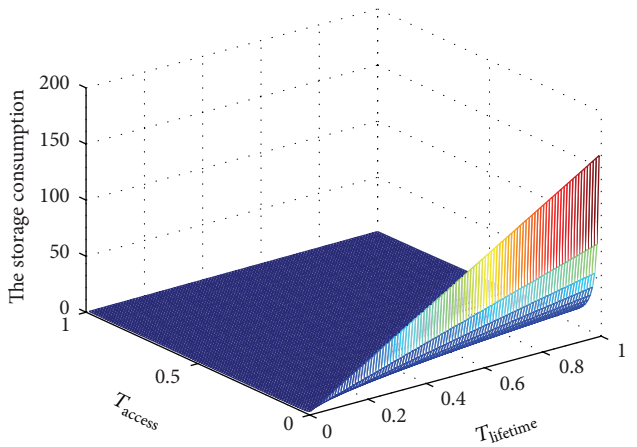

Figure 5: The storage consumption of IDAKE-MA.

6. Conclusion

In this paper, we introduced and demonstrated a security authentication protocol of SDN trusted domain in ADS applications and designed the trusted domain network architecture to solve the credential problem of SDN architecture. The trust negotiation concept with nontrusted third party is a prerequisite for communication between different SDN trusted domains in ADS applications.

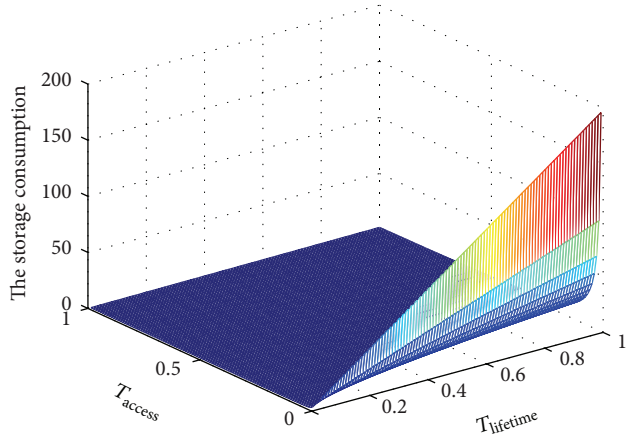

Figure 6: The storage consumption of IKEv2.

The main contributions are as follows: (1) Our protocol does not contain any unnecessary information. We analyzed the redundant information by BAN logical system. The results show that the protocol is a concise protocol. (2) The protocol is secure in theory. The BAN logic security analysis has proved that our protocol is secure. (3) The protocol is secure in experiment. We demonstrate the security of our trusted domain authentication protocol by BAN logic and AVISPA security analysis tool, and we compared our trusted domain authentication protocol with other prevalent protocols in performance analysis. Our work fills the gap of mutual trust between different trusted domains and provides security foundation for interaction between different trusted domains.

In the paper we considered replay attacks and intermediator attacks. In the future, we plan to consider other attacker behavior models including opportunistic collusion attacks, random attacks, and insidious attacks to further demonstrate superiority of our protocol.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by Funding Project for Beijing Key Laboratory of Trusted Computing, National Engineering Laboratory for Critical Technologies of Information Security Classified Protection, Open Research Fund of Beijing Key Laboratory of Trusted Computing, and 2015 Intelligent Manufacturing Special Project: The Comprehensive Standardized Test. The paper is also supported by a Beijing Natural Science Foundation project (no. 4162006).

References

[1] K. Mori, "Assured service-oriented system engineering technologies and applications," in *Proceedings of the Fifth International Symposium on Service Oriented System Engineering (SOSE '10)*, 4 pages, Nanjing, China, June 2010.

[2] H. Takahashi, K. Mori, and H. F. Ahmad, "Efficient I/O intensive multi tenant SaaS system using L4 level cache," in *Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering (SOSE '10)*, pp. 222–228, IEEE, Nanjing, China, June 2010.

[3] H. Takahashi, K. Mori, and H. F. Ahmad, "Autonomous short latency system for web application layer firewall," in *Proceedings of the 6th World Congress on Services (SERVICES '10)*, pp. 447–452, Miami, Fla, USA, July 2010.

[4] Q. Zuo, M. Xie, and W.-T. Tsai, "Autonomous decentralized tenant access control model for sub-tenancy architecture in software-as-a-service (SaaS)," in *Proceedings of the 12th IEEE International Symposium on Autonomous Decentralized Systems (ISADS '15)*, pp. 211–216, IEEE, Taichung, Taiwan, March 2015.

[5] Y. Chen and Y. Kakuda, "Autonomous decentralised systems in web computing environment," *International Journal of Critical Computer-Based Systems*, vol. 2, no. 1, pp. 1–5, 2011.

[6] Y. Chen and W. T. Tsai, *Service-Oriented Computing and Web Software Integration*, Kendall Hunt, 5th edition, 2015.

[7] M. Casado, T. Garfinkel, A. Akella et al., "SANE: a protection architecture for enterprise networks," in *Proceedings of the 15th USENIX Security Symposium*, Vancouver, Canada, July-August 2006.

[8] M. Casado, M. J. Freedman, J. Pettit et al., "Ethane: taking control of the enterprise," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 1–12, 2007.

[9] D. Li, X. Hong, and J. Bowman, "Evaluation of security vulnerabilities by using ProtoGENI as a launchpad," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '11)*, pp. 1–6, IEEE, Houston, Tex, USA, December 2011.

[10] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 151–152, ACM, Hong Kong, China, August 2013.

[11] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '13)*, vol. 1, ACM, Moscow, Russia, October 2013.

[12] N. Gude, T. Koponen, J. Pettit et al., "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[13] L. Rodrigues Prete, C. M. Schweitzer, A. A. Shinoda, and R. L. Santos de Oliveira, "Simulation in an SDN network scenario using the POX controller," in *Proceedings of the IEEE Colombian Conference on Communications and Computing (COLCOM '14)*, pp. 1–6, IEEE, Bogotá, Colombia, June 2014.

[14] Ryu documentation, http://osrg.github.com/ryu/.

[15] D. Erickson, "The beacon openflow controller," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 13–18, ACM, Hong Kong, August 2013.

[16] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 55–60, Hong Kong, China, August 2013.

[17] C. Kaufman, "Internet key exchange (IKEv2) protocol," RFC 4306, 2005, http://tools.ietf.org/html/rfc4306.

[18] L. Chen and M. Ryan, "Attack, solution and verification for shared authorisation data in TCG TPM," in *Formal Aspects in Security and Trust*, vol. 5983 of *Lecture Notes in Computer Science*, pp. 201–216, Springer, Berlin, Germany, 2010.

[19] K. Fan, H. Li, and Y. Wang, "Security analysis of the kerberos protocol using BAN logic," in *Proceedings of the 5th International Conference on Information Assurance and Security (IAS '09)*, pp. 467–470, Xi'an, China, September 2009.

[20] F. Dadeau, P.-C. Héam, and R. Kheddam, "Mutation-based test generation from security protocols in HLPSL," in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification, and Validation (ICST '11)*, pp. 240–248, IEEE, Berlin, Germany, March 2011.

[21] Y. Yang, H. Zhang, M. Pan, J. Yang, F. He, and Z. Li, "A model-based fuzz framework to the security testing of TCG software stack implementations," in *Proceedings of the 1st International Conference on Multimedia Information Networking and Security (MINES '09)*, pp. 149–152, IEEE, Hubei, China, November 2009.

[22] TCG TPM, *Main Part 1: Design Principles*, Specification Version, 1, 2003.

[23] P. N. Mahalle, B. Anggorojati, N. R. Prasad, and R. Prasad, "Identity establishment and capability based access control (IECAC) scheme for internet of things," in *Proceedings of the 15th International Symposium on Wireless Personal Multimedia Communications (WPMC '12)*, pp. 187–191, IEEE, Taipei, Taiwan, September 2012.

[24] N. Toledo, M. Higuero, J. Astorga, M. Aguado, and J. M. Bonnin, "Design and formal security evaluation of NeMHIP: a new secure and efficient network mobility management protocol based on the Host Identity Protocol," *Computers & Security*, vol. 32, pp. 1–18, 2013.

[25] J. Liu, J. Liao, X. Zhu et al., "Password authentication scheme for mobile computing environment," *Journal on Communications*, vol. 5, article 005, 2007.

[26] P. Huaxi, "An identity-based authentication model for multi-domain," *Journal of Computers*, vol. 29, no. 8, pp. 1271–1281, 2006.