

Article

BSSN-SDNs: A Blockchain-Based Security Service Negotiation for the SDN Interdomain

Yingying Ma ^{1,2,*}, Chaowen Chang ¹, Ping Wu ^{3,*}, Jingxu Xiao ¹  and Lu Yuan ¹

¹ Equipment Training Center, PLA Strategic Support Force Information Engineering University, Zhengzhou 450001, China; changchaowen5@163.com (C.C.); xiaojingxu2301@163.com (J.X.); yuanlu_l@163.com (L.Y.)

² Department of Intelligent Science and Technology, Zhengzhou University of Technology, Zhengzhou 450044, China

³ Engineering Division, Huanghe S&T University, Zhengzhou 450052, China

* Correspondence: mayingying2014@139.com (Y.M.); wpieucs@sina.com (P.W.)

Abstract: The security requirements for SDN (Software-Defined Network) cross-domain communication are diverse and dynamically changing; thus, a security service negotiation function is required for the SDN interdomain. However, the SDN interdomain distributed communication environment leads to a lack of trustworthiness and security. Therefore, this paper proposes a blockchain-based SDN interdomain security service negotiation mechanism, BSSN-SDNs, to provide automatic, secure, and trustworthy SDN interdomain security service negotiation. BSSN-SDNs proposes a three-layer reference architecture that enables joint on-chain and off-chain work by extending the security service negotiation module and blockchain client on the controller and deploying security service negotiation smart contracts on the blockchain. It especially adopts non-interactive key exchange and the message authentication code to ensure the confidentiality of the secure service negotiated on-chain. Finally, the timeliness as well as security and trustworthiness of BSSN-SDNs are analyzed, and the FISCO BCOS-based experiment results show that the delay of BSSN-SDNs is acceptable and is positively correlated with the number of policies and the number of SDN domains involved in negotiation.

Keywords: blockchain; software defined network; security service negotiation; interdomain



Citation: Ma, Y.; Chang, C.; Wu, P.; Xiao, J.; Yuan, L. BSSN-SDNs: A Blockchain-Based Security Service Negotiation for the SDN Interdomain. *Electronics* **2024**, *13*, 3120. <https://doi.org/10.3390/electronics13163120>

Academic Editor: Martin Reisslein

Received: 6 July 2024

Revised: 1 August 2024

Accepted: 5 August 2024

Published: 7 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Software-defined network (SDN), as one of the most promising future network technologies, has the characteristics of separation of control and forwarding, logical centralization of control, and being open programmable [1]. At present, SDN has been widely deployed on the Internet of Things (IoT), providing a high data transfer rate and efficient network management [2–4].

1.1. Motivation

Requirements. With the widespread deployment of IoT, a multi-domain IoT architecture with SDN as the infrastructure has emerged, as shown in Figure 1.

Usually, there is a need for cross-domain data exchange in multi-domain IoT [5]. For example, in the field of smart cities, health data, smart car data, smart meter data, surveillance data, and financial transactions of residents are collected, processed, and exchanged among different organizations. However, these user data are of varying degrees of importance and involve varying user privacy, which requires appropriate security services during cross-organizational data sharing [6]. In the context of smart municipalities, services such as smart water management, smart street lighting, and smart waste treatment are provided for urban management and ensure that these systems operate in a coordinated and effective manner. However, one of the main challenges of smart municipal collaborative services is the security and privacy issues of data sharing faced during interoperability

between heterogeneous services [7]. As a further example, in the field of industrial IoT, it is often challenging to have a complete product production process in a single domain. Therefore, it has become a trend to distribute the complete product production process to multiple domains that are closely related. Then, devices distributed in different domains need to share and exchange data using efficient communication mechanisms to collaborate in the production process. However, these data are involved in product parameters, which are usually of some degree of sensitivity, and therefore need to be provided with appropriate security services when shared across domains [8].

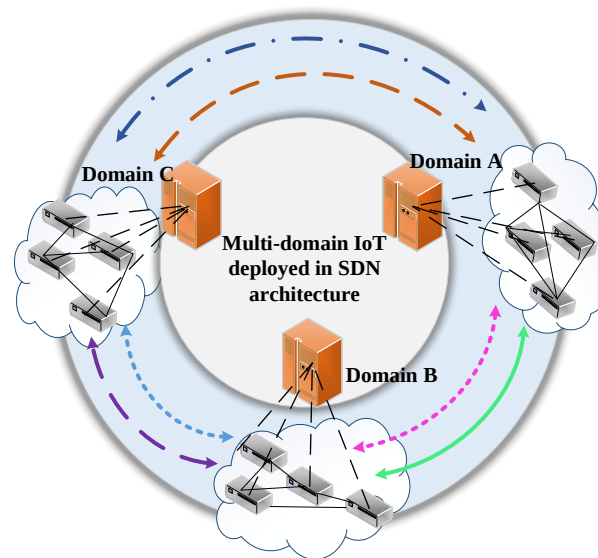


Figure 1. Multi-domain IoT deployed in SDN architecture. Different line types indicate that the data exchange has different security service requirements.

As mentioned above, data shared between different organizations is usually of a certain value and high sensitivity. Therefore, different security services are required between organizations depending on the importance and sensitivity of the interactive information. As a result, it is necessary to conduct SDN interdomain security service negotiations.

Challenges. However, SDN interdomain security service negotiation faces the following challenges:

- (1) Compared with SDN intradomain security service negotiation, since there is no unified authoritative center among SDN domains, SDN interdomain security service negotiation lacks trustworthiness; therefore, ensuring the trustworthiness of SDN interdomain security service negotiation is a challenge.
- (2) SDN interdomain environment communication lacks security, so ensuring the security of SDN interdomain security service negotiation is a second challenge.
- (3) Since security requirements are dynamically changing, how to negotiate security services automatically and in a timely manner to adapt to dynamically changing security requirements is a third challenge.

Objectives. In summary, this paper aims to establish automatic, secure, and trustworthy SDN interdomain security service negotiation to protect cross-domain negotiation from unauthorized access (confidentiality), modification (integrity), and falsification (authentication).

1.2. Proposed Solutions

Blockchain technology, with its decentralized, de-trusted, tamper-proof, traceable, and programmable characteristics, can be used to achieve trusted communication in a distributed, untrustworthy environment; in particular, smart contracts provide a method for automatic and transparent business execution [9]. Currently, blockchain technology is

widely used in the field of information security to provide security for cross-organizational information sharing and business collaboration [10,11].

Therefore, to address the challenges of SDN interdomain security service negotiation, blockchain technology can be applied to realize automatic, transparent, secure, and traceable interdomain security service negotiation. It should be noted that, since the data on the blockchain chain is publicly accessible, it is important to protect the privacy of secure service negotiations [12–14].

In summary, this paper proposes a blockchain-based security service negotiation mechanism for SDN interdomains, BSSN-SDNs, to provide automatic, secure, and trustworthy SDN interdomain security service negotiation.

1.3. Contributions

BSSN-SDNs proposes a three-layer abstract model and enables SDN domains to realize security service negotiation in association with the blockchain by expanding the security service negotiation function and the blockchain client to the SDN controller.

- (1) It utilizes smart contract technology to realize policy publishing, updating, and policy matching to ensure automatic and transparent security service negotiation.
- (2) It adopts a non-interactive key exchange and a message authentication code to ensure the confidentiality of the secure service negotiated on the blockchain.
- (3) It combines the blockchain transaction signature verification mechanism and the simple payment verification function (SPV) for receipt to realize secure and trustworthy security service negotiation in an untrustworthy environment.
- (4) Based on the account state mechanism of blockchain, it realizes automatic updating of policies and ensures the timeliness of policy negotiation; based on the blockchain storage structure, it builds a global and whole-process supervisory view for SDN interdomain security service negotiation.

The rest of this article is organized as follows: Section 2 gives a comprehensive review of related work. Then, Section 3 presents the reference architecture, relevant definitions, theoretical model, functional modules, and workflow of BSSN-SDNs. Section 4 introduces confidentiality protection and smart contracts for secure service negotiation on blockchain. Furthermore, Section 5 analyzes the timeliness and security of BSSN-SDNs and compares them with similar schemes. Finally, the performance evaluation is reported in Section 6, and Section 7 gives the conclusion.

2. Related Work

2.1. SDN-Based IPsec

In traditional networks, IPsec is used to provide on-demand security services at the network layer. However, IPsec in traditional networks is implemented in the kernel of routers or hosts and requires end-to-end manual configuration, which is time-consuming, complicated to operate, and prone to security flaws. Since SDN has a hierarchical structure that separates forwarding and control, some scholars have launched innovative research on how to deploy IPsec in SDN in view of the characteristics of SDN. Gabriel Lopez Millan's team established an SDN-based IPsec management framework and designed two deployment scenarios—with IKE (Internet Key Exchange) negotiation mode and without IKE negotiation mode [15,16]. P4-IPsec implements an IKE-free IPsec based on programmable data plane technology [17,18]. To improve the utilization efficiency of IPsec virtual gateways on the cloud, Protego proposes a distributed IPsec gateway service that separates the control plane and the data plane of IPsec, stores infrequently updated IKE SAs (Security Associations) centrally in the central controller, and the data plane is responsible for datagram encryption and decryption [19]. S-SDS deploys IPsec by extending the OpenFlow message and flow table fields, where the controller manages the Security Policy Depository (SPD) and the gateway is responsible for IKE negotiation [20]. Due to the possible lack of a bidirectional data plane communication channel between VNFs (Virtual Network Functions) in an SFC (Service Function Chaining), the Internet Security Key

Exchange protocol (IPsec-IKE) is not applicable inside a VNF (Virtual Network Function); hence, ref. [21] introduces an alternative to IPsec-IKE that is specifically designed for NFV (Network functions virtualization) environments.

The above scenarios have redesigned the IPsec deployment method and workflow within SDN domains but have not yet been specifically explored for the specific issues of SDN interdomain security service negotiation. In particular, SDN interdomain scenarios across organizations face the following challenges: end-to-end inter-domain security service negotiation faces the security risk of denial and lacks supervision; if the security service negotiation and data forwarding are in the same plane, the encryption and authentication during the negotiation and the forwarding of negotiation messages will occupy the computation and bandwidth resources in the data plane, which will affect the forwarding latency and efficiency of the data plane and is inconsistent with the original intent of the SDN data plane for simple and fast forwarding. Therefore, a security service negotiation scheme needs to be proposed specifically for the SDN interdomain.

2.2. SDN Multi-Domain Organization Architecture

Currently, SDN multidomain organization architectures mainly include vertical and horizontal architectures [22]. For security service negotiation among multiple SDN organizations, if a vertical architecture is adopted and a super central controller manages the security service negotiation, the centralized cross-organizational management will face the management transparency problem and the single-point-of-failure problem; if a horizontally distributed architecture is adopted to negotiate the security services through the east-west interfaces of the controllers, it will be difficult to form a global supervisory view, and thus both of them have root cause deficiencies.

Therefore, a new SDN multi-domain organizational architecture needs to be designed for SDN interdomain secure service negotiation.

2.3. Blockchain Applications in SDN

In recent years, blockchain-based SDN security schemes have also been gradually proposed. For example, the scheme for cross-domain forwarding path control has high real-time requirements, so the scheme deploys the blockchain function directly as a controller extension application, deployed locally in the SDN [23]; the scheme for SDN flow table validation deploys the blockchain network between the control layer and the data layer in order to store and validate the flow table [24]; and the scheme for cross-domain access control implements complex authentication algorithms based on the blockchain in order to authenticate endpoint identify or user identity attributes [25,26]. However, the above schemes have not yet proposed a specialized solution for the blockchain-based SDN interdomain security service negotiation problem.

Given the above, it is necessary to design an SDN interdomain secure service negotiation mechanism to ensure a transparent, secure, and trustworthy negotiation process.

3. Overview of BSSN-SDNs

3.1. Threat Models of Security Service Negotiation in the SDN Interdomain

Since SDN interdomain security service negotiation is in a distributed, untrustworthy communication environment, it faces several security threats. To be able to clearly articulate the security threats that can be mitigated by BSSN-SDNs, the SDN inter-domain security service negotiation threat model is therefore described.

Firstly, to facilitate the illustration of security threats, they are categorized into the following three scenarios, depending on the location of the security service negotiation.

3.1.1. Security Service Negotiation Based on Data Plane

If security services are negotiated through end-to-end interactions in the data plane, then it is possible to face attacks from malicious or compromised endpoints or forwarding

devices in the data plane. The threat model of security service negotiation based on the data plane is shown in Figure 2.

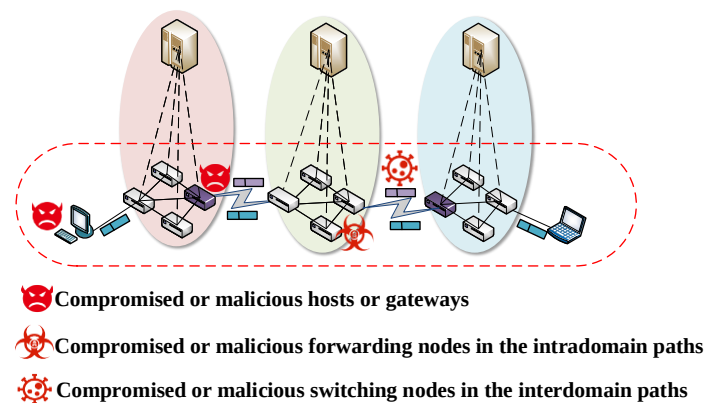


Figure 2. Threat model of security service negotiation based on data plane.

The threat model is as follows:

- (1) Eavesdropping security services: compromised or malicious forwarding nodes in the intradomain paths or switching nodes in the interdomain paths may eavesdrop on the negotiated security services.
- (2) Tampering security services: the compromised or malicious forwarding nodes in the intradomain paths or switching nodes in the interdomain paths are likely to tamper with the negotiated security services.
- (3) Identity forgery: all three types of adversaries may use a spoofed identity to send forged security service negotiation messages.
- (4) Security service negotiation message replay attack: all three types of adversaries are likely to replay security service messages.

The above is based on the following security assumptions.

- (1) Control plane is safe and secure.
- (2) Control plane to data plane southbound interface communication is secure.
- (3) Data plane packets are forwarded without facing malicious drops or path changes.

3.1.2. Security Service Negotiation Based on Control Plane

If the SDN multi-domain organization uses a horizontal architecture for inter-domain management, where each domain controller negotiates security services through the control plane east-west interfaces, then it may be exposed to attacks from malicious or compromised control plane devices. The threat model of security service negotiation based on the control plane is shown in Figure 3.

The threat model is as follows:

- (1) Eavesdropping security services: the compromised or malicious network nodes on the east-west links may eavesdrop on the negotiated security services.
- (2) Tampering security services: the compromised or malicious network nodes on the east-west links are likely to tamper with the negotiated security services.
- (3) Identity forgery: both types of adversaries may use a spoofed identity to send forged security service negotiation messages.
- (4) Security service negotiation message replay attack: both types of adversaries are likely to replay security service messages.

The above is based on the following security assumptions:

- (1) The control plane for each domain is reliable.
- (2) The communication within the control plane of a single domain is secure and trustworthy.

- (3) Control plane to data plane southbound interface communication is secure.

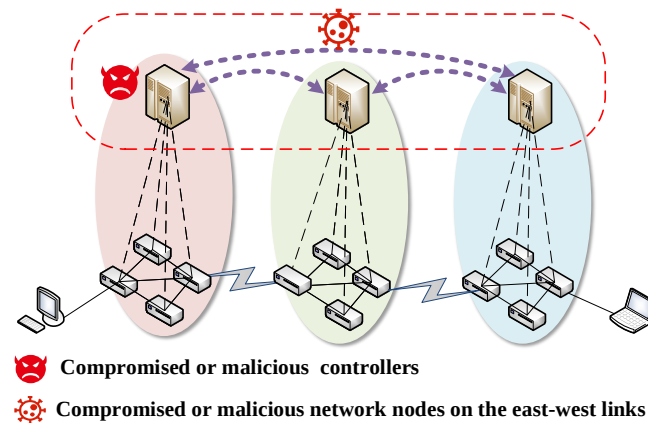


Figure 3. Threat model of security service negotiation based on control plane.

3.1.3. Security Service Negotiation Based on the Super Control Center

If the SDN multi-domain organization uses a vertical architecture for interdomain management, where the domains negotiate centrally by uploading security service negotiation messages to the super control center, then it may face attacks from the compromised super control center or the malicious devices on their links. The threat model of security service negotiation based on the super control center is shown in Figure 4.

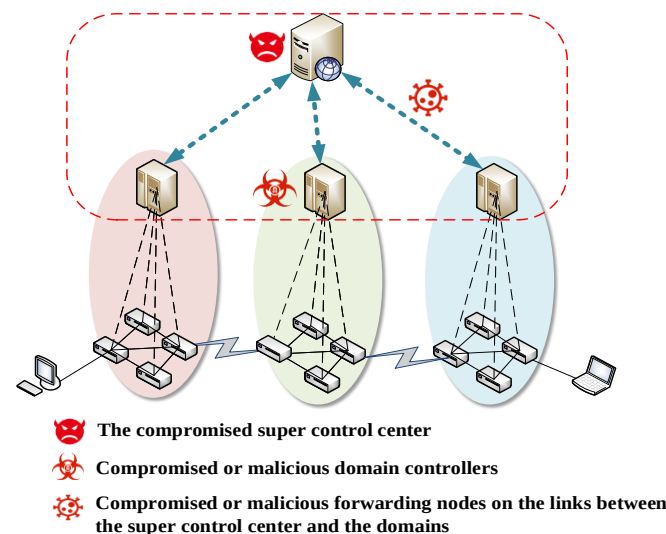


Figure 4. Threat model of security service negotiation based on the super control center.

The threat model is as follows.

- (1) Eavesdropping security services: the network nodes on the links between the super control center and the domains may eavesdrop on the negotiated security services.
- (2) Tampering security services: the network nodes on the links between the super control center and the domains are likely to tamper with the negotiated security services.
- (3) Identity forgery: malicious or compromised domain controllers and network nodes on the links between the center and domain controllers may use a spoofed identity to send forged security service negotiation messages.
- (4) Security service negotiation message replay attack: malicious or compromised domain controllers and network nodes on the links between the center and domain controllers are likely to replay security service messages.
- (5) Management transparency and single points of failure: the compromised control center may confront problems of management transparency and single points of failure.

The above is based on the following security assumptions.

- (1) The control plane for each domain is reliable.
- (2) Communication within the control plane of a single domain is secure and trustworthy.
- (3) Control plane to data plane southbound interface communication is secure.

3.2. Reference Architecture of BSSN-SDNs

In order to address the security threats faced in the above scenarios, BSSN-SDNs were proposed. The theoretical framework of BSSN-SDNs can be represented by a three-layer hierarchical model, which has a blockchain management layer, control logic layer, and data forwarding layer, as shown in Figure 5.

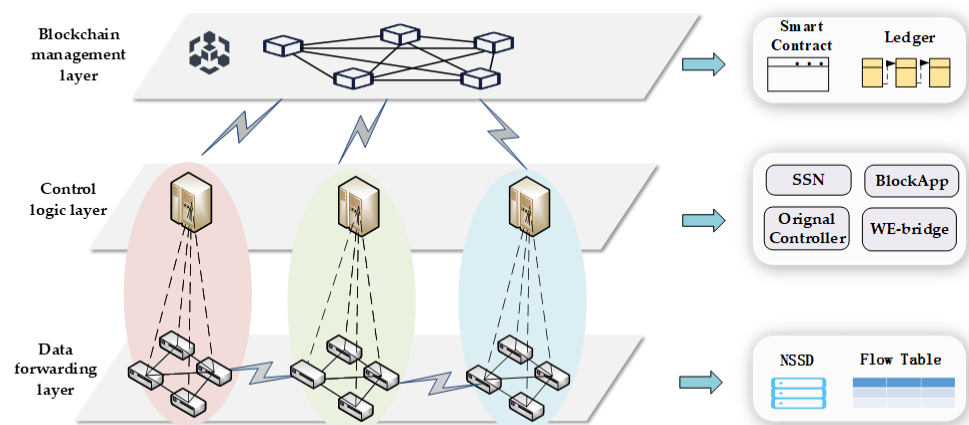


Figure 5. The reference architecture of BSSN-SDNs.

- (1) The blockchain management layer provides the security service negotiation function for the SDN interdomain communication, which mainly includes the security policy publication and update and the security service negotiation. The blockchain management layer is given as a logical schematic of a blockchain whose physical structure is a peer-to-peer network consisting of ledger nodes.
- (2) The control logic layer contains the original SDN controller, the extended security service negotiation functions *SSN* and the blockchain client *BlockApp*. The SDN controller interacts with the blockchain through the blockchain client, including transaction upload and receipt verification. The *WE-bridge* is an east-west interface for SDN controllers to interact with interdomain network topology.
- (3) The data-forwarding layer corresponds to the SDN data plane. The controller issues security services to the border switches in the data forwarding layer through the southbound interface. The border switch performs the corresponding security service for data packets based on the negotiated security service database *NSSD*. Then, the data packets with encapsulated security service identifiers are forwarded downstream by matching the OpenFlow flow table.

Regarding how the controller issues security services down to the data plane, reference can be made to the literature [15,16], where the controller configures *NSSD* the border switches through the YANG model based on the southbound protocol *NETCONF*. Alternatively, reference can be made to the literature [17,18], using the protocol-independent programmable switches *P4* as the border switches and configuring *NSSD* the *P4* switches through *Runtime*.

- (4) The connections between the blockchain client and the blockchain, as well as among the domain border gateway switches, are via traditional networks.
- (5) Within a single SDN domain, to enhance the resilience of the system and mitigate the single point of failure issue, multiple control nodes can be deployed on the control

plane using distributed caches, like Redis, to implement active-standby or even multi-active configurations.

- (6) To guarantee the normal transmission of data within the SDN domain, it is presumed that the interactions within the control plane are secure and trusted and that the communication between the controller plane and the data plane is secure and trusted.

Since the scalability, reliability, consistency, and security of the control plane are important research aspects of SND but not the focus of this paper, further discussion on the control plane can be found in the literature [27–29].

3.3. Definitions Related to BSSN-SDNs

Definition 1. Security Services (SS). It is described using a triple $\langle \text{sec}, \text{enc}, \text{hash} \rangle$, where *sec* denotes a security protocol, *enc* denotes a cryptographic algorithm, and *hash* denotes a hashing algorithm. For example, *sec* can be a security protocol customized within the federation to ensure authentication and confidentiality of communications; *enc* is the encryption algorithm selected in the protocol, for example, it can be DES, 3DES; and *hash* is the hashing algorithm selected in the protocol, for example, MD5, SHA-256, etc.

Definition 2. Security Policy (SP). It is described using $\langle S_{acc}, D_{acc}, \text{Selector}, [SS_i, P_{SS_i}] \rangle$, where S_{acc} and D_{acc} denote the blockchain account addresses of the source domain and the destination domain, respectively; *Selector* describes communication characteristics, such as using $\langle ip_S, ip_D, \text{transProtocol}, port_S, port_D, \text{confLevel} \rangle$ to specify the source IP address, destination IP address, transport layer protocol, source port number, destination port number, and data sensitivity level of communication data; SS_i and P_{SS_i} represent the *i*th optional security service and the corresponding priority accordingly, which have been sorted by priority from highest to lowest by default; and the symbol $[]$ denotes that zero, one, or more contained data units are allowed.

Definition 3. Secret State Security Service (HSS). Since the data on the chain is publicly available, the security services uploaded to the blockchain should be in a secret state in order to ensure the confidentiality of the negotiated security services. $HSS = \text{HMAC}_{\hat{K}_{S,D}}(SS, \hat{K}_{S,D})$, where $\hat{K}_{S,D}$ is the shared key of the source domain and destination domain, and $\text{HMAC}()$ is the message authentication code function.

Definition 4. Secret State Security Policy (HSP). It is described using $\langle S_{acc}, D_{acc}, \text{Selector}, [HSS_i, P_{SS_i}] \rangle$. That is, the security services in the security policy are replaced with the secret state security services.

Definition 5. Negotiated Security Service (NSS). It describes the negotiated plaintext security service. $NSS = \langle S_{acc}, D_{acc}, \text{Selector}, SS \rangle$.

3.4. Theoretical Model of BSSN-SDNs

The theoretical model of BSSN-SDNs can be formalized as a six-tuple, where: $\langle \text{SDN}, \text{BlockNet}, \text{Trans}, \text{SC}, \text{Receipt}, \text{operation} \rangle$, where:

- (1) *SDN* indicates the SDN domains involved in security service negotiation.
- (2) *BlockNet* denotes the federated blockchain, which is jointly maintained by the SDN domains.
- (3) *Trans* denotes the transaction, specifically the security service negotiation transaction *SSNTrans*. It contains $\langle id, tp, S_{acc}, \text{HSPset}_S, \text{sign} \rangle$, where *id* represents the transaction identifier, *tp* represents the timestamp, S_{acc} is the blockchain account address of the domain uploading this transaction, the *sign* is the signature of the account on the transaction, HSPset_S represents the secret state security policy set formulated by the source domain S_{acc} .
- (4) *SC* denotes the smart contract, specifically the security service negotiation smart contract *SSN_SC*. After the successful deployment of the smart contract, once the

- system meets the trigger conditions, *SSN_SC* will be automatically executed, which ensures that the security service negotiation process is automatic and transparent.
- (5) *Receipt*. After the security service negotiation smart contract *SSN_SC* achieves the negotiation result, it issues the secret state security service receipt *HSSReceipt* to the relevant source domain and the destination domain. *HSSReceipt* contains $\langle S_{acc}, D_{acc}, Selector, HSS \rangle$.
 - (6) *operation* indicates the operations involved in *SDN*, *BlockNet* and the interaction between the two. Concretely, this includes uploading the transaction (upload), issuing the receipt (issue), formulating a local security policy (write), encrypting the security policy using a message authentication code (*encrypt*), publishing the secret state security policy (publish), updating the secret state security policy (update), matching the secret state security policy (match), validating the receipt (which can be conducted using a simple payment verification, SPV), and mapping the secret state security services to plaintext security services (map), etc.

3.5. Functional Module of BSSN-SDNs

BSSN-SDNs extended the functionality of the SDN controller by adding a security service negotiation module *SSN* and a blockchain client *BlockApp*, and deployed a customized security service negotiation smart contract *SSN_SC* on the blockchain. The functional module is shown in Figure 6. *SSN* contains a local security policy database and a security service negotiation engine. *BlockApp*, as the blockchain client in the SDN domain contains the smart contract invocation function and the simple payment verification function (SPV) for receipts.

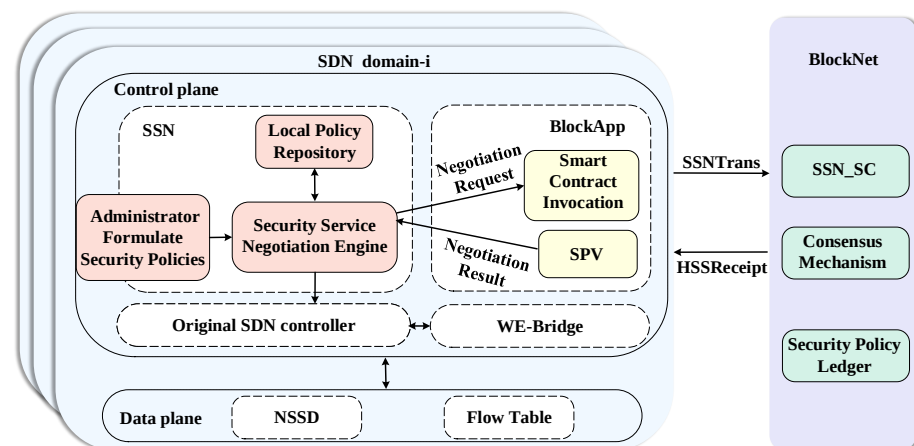


Figure 6. The function model of BSSN-SDNs.

The main functions of each module are as follows:

- (1) *SSN*: Write the security policies formulated by the administrator into the local policy repository through the security service negotiation engine; use the *HMAC()* to generate that newly formulated security policies into secret state security policies and submit them to the smart contract invocation model; as well as match the ciphertext security policy with the receipt *HSSReceipt* to obtain the corresponding plaintext security service and issue them to the controller.
- (2) *BlockApp*: The smart contract invocation module encapsulates the secret state security policy into a security service negotiation transaction *SSNTrans* and uploads it to the blockchain, and performs SPV validation on the secret state security service receipts *HSSReceipt* which issued by the blockchain.
- (3) *BlockNet*: The blockchain *BlockNet* receives the security service negotiation transaction *SSNTrans* and verifies the signature of the transaction; the legitimate transaction will automatically trigger the execution of the security service negotiation smart con-

- tract *SSN_SC*, realize the policy publishing and updating, and the interdomain secret state policy matching, and write the published and updated policies into the security policy ledger through the consensus mechanism; and it sends the matched secret state security service to the relevant SDN domains through the *HSSReceipt*.
- (4) The controller can send the negotiated security services to the data plane through the southbound interface. In addition, the controller calculates the forwarding path based on the inter-domain topology obtained by the WE-Bridge and the intra-domain topology and sends the forwarding paths down to the data plane.
 - (5) The data plane performs encryption and decryption operations for incoming data flow according to the security service parameters in *NSSD*, thus providing on-demand security services for SDN interdomain communication, and forwards the data flow based on the forwarding table.

3.6. Workflow of BSSN-SDNs

The workflow of BSSN-SDNs is shown in Figure 7, where SDN_S and SDN_D denote a source domain and a destination domain participating in the interdomain security service negotiation in the federation chain, respectively, and *BlockNet* represent the blockchain network. The overall workflow involves the following three main phases:

- (1) Security Policy Preprocessing Phase. Each SDN domain formulates the security policy from this domain to other domains and encrypts the security policy into a secret state security policy employing *HMAC()*.
- (2) Security Service Negotiation Phase. Firstly, each SDN domain uploads the secret state security policy to the blockchain through the security service negotiation transaction *SSNTrans*, and then triggers the security service negotiation smart contract *SSN_SC* to automatically execute the secret state security policy publishing, updating, and matching, and finally sends the results of the security service negotiation to the relevant SDN domains through the secret state security service receipts *HSSReceipt*.
- (3) Security Service Acquisition Phase. The blockchain clients within the SDN domain use SPV to verify the secret state security service receipts. If the verification is approved, the security service negotiation engine will map the ciphertext security policy in the receipt *HSSReceipt* to the corresponding plaintext security service, thus acquiring the negotiated security service.

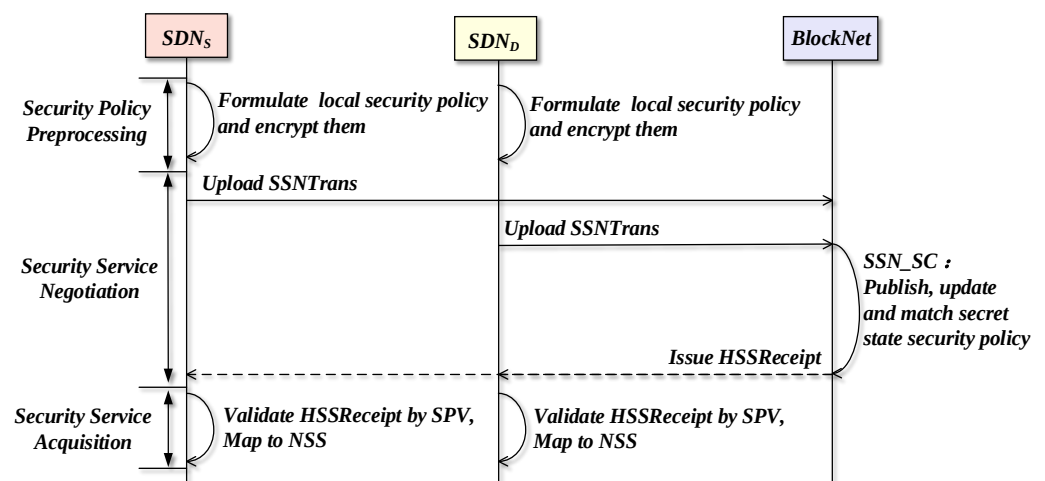


Figure 7. The overall workflow of BSSN-SDNs.

Figure 8 presents a schematic diagram of the transformation of the security service negotiation data during processing.

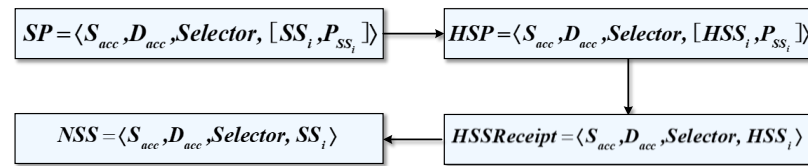


Figure 8. The transformation of the security service negotiation data.

Firstly, the security service negotiation engine encrypts the plaintext of the security policy $SP = \langle S_{acc}, D_{acc}, Selector, [SS_i, P_{SS_i}] \rangle$ defined by the manager into a secret state security policy $HSP = \langle S_{acc}, D_{acc}, Selector, [HSS_i, P_{SS_i}] \rangle$.

Then, HSP is encapsulated as $SSNTrans$ and uploaded to the blockchain. After blockchain negotiation, the security service receipt $HSSReceipt = \langle S_{acc}, D_{acc}, Selector, HSS \rangle$ is issued.

After the receipt is verified and passed, the security service engine matches the secret state security policy HSP according to the receipt $HSSReceipt$, thus mapping to the corresponding plaintext security policy SP , and obtaining the negotiated security service $NSS = \langle S_{acc}, D_{acc}, Selector, SS \rangle$.

4. Security Service Negotiation on Blockchain

4.1. Confidentiality Protection for Security Service Negotiation on Blockchain

Since the information on the chain is publicly available but the security service negotiation needs to guarantee the confidentiality of the negotiated content, BSSN-SDNs assigns public-private key pairs $\langle Q_{ID}, S_{ID} \rangle$ to each domain based on He-IBS (an identity-based signature mechanism). He-IBS generates the master private keys s by the key generation center and discloses the system parameters $\langle G_1, G_2, \hat{e}, P, P_{pub}, H, h \rangle$ [30]. Equation (1) and Equation (2) respectively describe the public-private key pairs generated for the source domain S and the destination domain D , and the bilinear pair computation shown in Equation (3) is used to realize the non-interactive SDN interdomain key negotiation to obtain the shared key $\hat{K}_{S,D}$ of the source and destination domains.

$$\langle Q_{ID_S}, S_{ID_S} \rangle = \langle h(ID_S), sh(ID_S) \rangle \quad (1)$$

$$\langle Q_{ID_D}, S_{ID_D} \rangle = \langle h(ID_D), sh(ID_D) \rangle \quad (2)$$

$$\hat{K}_{S,D} = \hat{e}(S_{ID_S}, Q_{ID_D}) = \hat{e}(S_{ID_D}, Q_{ID_S}) \quad (3)$$

Here Q_{ID_S} and S_{ID_S} are the public and private keys of the source domain, and Q_{ID_D} and S_{ID_D} are the public and private keys of the destination domain.

The domain administrator formulates security policies from this domain to other domains and writes them to the local policy repository. When the submission of the security policy is confirmed, the security service negotiation engine automatically encrypts the security services in the security policy into the corresponding secret state security services HSS_i and reassembles them into a secret state security policy $HSP_{S,D}$, and finally submits the set of encrypted security policies set $HSPset_S$ to the blockchain client, as shown in Equations (4)–(6).

$$HSS_i = HMAC_{\hat{K}_{S,D}}(SS_i, \hat{K}_{S,D}) \quad (4)$$

$$HSP_{S,D} = \langle S_{acc}, D_{acc}, \bigcup_j (Selector_j, [HSS_i, P_{SS_i}]) \rangle \quad (5)$$

$$HSPset_S = \bigcup_k HSP_{S,D_k} \quad (6)$$

Here i denotes the serial number of a security service in a security policy and j denotes the serial number of a security policy in the policy set of this domain.

4.2. Smart Contracts for Security Service Negotiation on Blockchain

After the blockchain receives the transaction $SSNTrans$ uploaded by the blockchain client of an SDN domain, it first verifies the transaction signature. And after it passes the verification, it submits the security service policies set $HSPset_S$ in the $SSNTrans$ to the security service negotiation smart contract SSN_SC , to execute the interdomain security service negotiation of SND based on the blockchain.

The customized security service negotiation smart contract SSN_SC on the blockchain is shown in Algorithm 1. It is primarily responsible for enforcing the secret state security policies, publishing, updating, and matching. SSN_SC receives the set of secret state security policies submitted by each domain as input and outputs the result of security service negotiation. It works in the following main steps.

- (1) For each secret state security policy $HSP_i = \langle S_{acc}, D_{acc}, Selector, [HSS_i, P_{SS_i}] \rangle$ in the security policy set $HSPset_S$, obtain its domain identifications $\langle S_{acc}, D_{acc} \rangle$ in HSP_i , and obtain its serial number $\langle S_{id}, D_{id} \rangle$ according to the domain identifications, then write the policy HSP_i into $HSPstore_S[D_{id}]$ which is the corresponding security policy storage unit of the SDN domain S_{acc} in the blockchain ledger. (steps 2–9).
- (2) Obtain the corresponding secret state security policy formulated by the destination domain D_{acc} from the security policy storage unit $HSPstore_D[S_{id}]$, then write it to a temporary variable HSP_{temp} . (step 10).
- (3) If HSP_{temp} is not empty, then the secret state security policies HSP_{temp} and HSP_i are both sorted in descending order of security service priority (steps 11–13). And a match between HSP_{temp} and HSP_i will be performed in steps 14–28.
- (4) The temporary variable mark is used to record the existence of a matching security service. (step 14).
- (5) For each security service in HSP_{temp} , starting from the highest priority, match each security service in HSP_i in turn, and if there is a match, write it to the matched set of security policies $HSPset_{match}$ and end the two-layer loop, otherwise, until the two-layer traversal is completed. (steps 15–28).
- (6) If the destination domain's secret security policy on $\langle S_{acc}, D_{acc}, Selector \rangle$ is empty, or there is no matching security service in the security policies of the source and destination domains, no operation will be performed on $HSPset_{match}$. (steps 29–30).
- (7) When each policy in the set $HSPset_S$ has completed the above processing, the set of matched security policies $HSPset_{match}$ is returned. (step 31).

$HSPset_{match}$ contains a collection of secret state security services obtained by security policy matching, that is $[(S_{acc}, D_{acc}, Selector, HSS)]$. The blockchain organizes each secret state security service in $HSPset_{match}$ into a secret security service receipt $HSSReceipt$ and sends them to the relevant source domain S_{acc} and destination domain D_{acc} .

Algorithm 1 SSN_SC

Input: $HSPset_S$;

Output: $HSPset_{match}$;

```

1:  $HSP_{temp} = \emptyset$ ;
2: for  $HSP_i$  in  $HSPset_S$ 
3:    $\langle S_{acc}, D_{acc} \rangle \leftarrow HSP_{temp}.getAcc()$ 
4:    $\langle S_{id}, D_{id} \rangle \leftarrow \langle S_{acc}, D_{acc} \rangle.getID()$ 
5:   if  $HSPstore_S[D_{id}, Selector]$  is null
6:     publish  $HSP_i$  in  $HSPstore_S[D_{id}, Selector]$ 
7:   else
8:     update  $HSPstore_S[D_{id}, Selector]$  by  $HSP_i$ 
9:   end if
10:   $HSP_{temp} \leftarrow HSPstore_D.getHSP(S_{id}, Selector)$ 

```

```

11:    if  $HSP_{temp} \neq \emptyset$ 
12:         $HSP_{temp}.rank(Priority)$ 
13:         $HSP_i.rank(Priority)$ 
14:        mark = 0
15:        for  $HSS_j$  in  $HSP_{temp}$ 
16:            for  $HSS_k$  in  $HSP_i$ 
17:                if  $HSS_j == HSS_k$ 
18:                     $HSPset_{match}[\langle S_{acc}, D_{acc}, Selector \rangle] \leftarrow HSS_k$ 
19:                    mark++
20:                    break
21:                else
22:                    continue
23:                end if
24:            end for
25:            if mark != 0
26:                break
27:            end if
28:        end for
29:    end if
30: end for
31: return  $HSPset_{match}$ 

```

5. Discussions

5.1. Timeliness of Security Service Negotiation for BSSN-SDNs

BSSN-SDNs can realize automatic updating of policies and ensure the timeliness of security service negotiation. According to the account state mechanism of blockchain, there are $\delta_{t+1} = \gamma(\delta_t, T)$, where δ_t is the account state of a smart contract at time t , T is a transaction that has been consensus of the blockchain nodes, and γ is the state transition function triggered by the transaction T , thus the account state mechanism makes a transaction update the account state of its related smart contract [31]. Similarly, in BSSN-SDNs, assuming η_t is the security policy set of domain acc stored in the SSN_SC account at the moment t . When domain acc uploads a new security service negotiation transaction $SSNTrans_{acc}$ to update its policies, SSN_SC executes $SSNTrans_{acc}$ and consensus confirms it, the security policy set of domain acc in the SSN_SC account is automatically updated to the current newest policy set η_{t+1} , that is $\eta_{t+1} = \gamma(\eta_t, SPNTrans_{acc})$. By analogy, BSSN-SDNs ensure that the set of security policies of each SDN domain on the chain is always kept current and up-to-date. Also, since the security service receipts are based on the most up-to-date security policy matching results, the BSSN-SDNs guarantee the timeliness of the security service negotiation results.

5.2. Security and Trustworthiness of Security Service Negotiation for BSSN-SDNs

To discuss the security and trustworthiness of BSSN-SDNs, we can review the threat model described in Section 3.1. And assumes that the BSSN-SDNs are deployed in open, untrusted, and decentralized SDN interdomain environments. The control plane in each domain is reliable, the interactions within the control plane are secure and trusted, and the southbound communication between the controller plane and the data plane is secure and trusted.

Figure 9 gives the security measures taken by the BSSN-SDNs during the security service negotiation process. Then we consider the following threats.

(1) Eavesdropping attack.

In the process of (a)–(e), the security service is uploaded, matched, and issued completely in ciphertext so that eavesdropping attacks can be prevented during the negotiation process.

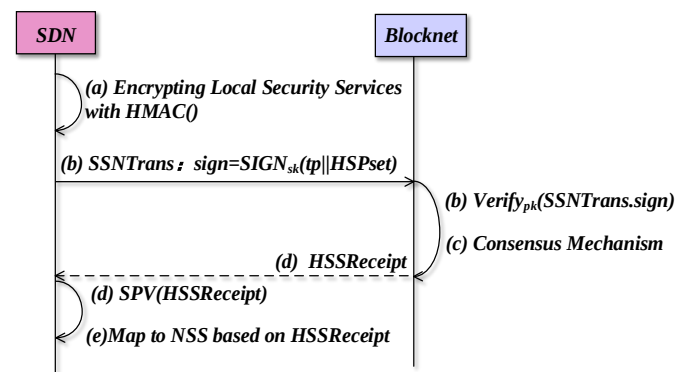


Figure 9. The security measures taken by the BSSN-SDNs.

(2) Tampering attack.

As shown in step (b), the uploading process can detect tampering because the transaction containing the secret state security policy is signed when it is uploaded to the blockchain, and the blockchain verifies the signature after receiving the transaction. As shown in step (c), since the consensus mechanism is used in the on-chain negotiation process, the on-chain negotiation process can detect tampering. As shown in step (d), the blockchain sends down the negotiated secret security service in the form of receipts to the *BlockApp* within the SDN domain. The *BlockApp* uses the SPV to verify the receipt, so the issuing process can detect tampering. In summary, BSSN-SDNs have the ability to detect tampering.

(3) Identity forgery.

As shown in step (b), it is necessary to sign the uploaded transactions with the domain's private key, so BSSN-SDNs prevent malicious or compromised controllers from uploading false transactions with forged identities.

(4) Replay attacks.

As shown in step (b), since the signature mechanism includes timestamps when uploading the blockchain, it is possible to detect malicious or compromised controllers launching replay message attacks to the blockchain.

(5) Management transparency and single point of failure.

Since security services are negotiated using the consortium blockchain co-managed by the SDN domains, the negotiation process mitigates the single point of failure and management transparency issues.

As mentioned above, BSSN-SDNs help to ensure confidentiality, integrity, authenticity, and anti-replay attacks and mitigate single points of failure and management transparency problems. Therefore, BSSN-SDNs is an SDN interdomain security service negotiation scheme with security and trustworthiness.

When SDN inter-domain security service negotiation has higher security requirements, further security measures are required. For example, in addition to protecting the confidentiality of the negotiated content, the privacy of the negotiating party's identity needs to be protected. Especially when a more serious security threat, a global adversary, is encountered, the communication needs to be guaranteed to be anonymous to protect true privacy. For example, [32] is an enhanced anonymous communication scheme that has the advantage over Tor [33] of protecting sender anonymity and relationship anonymity and can be applied to traditional IPsec, data-plane-based, or control-layer-based inter-domain secure service negotiation for the SDN interdomain. For the blockchain-based SDN inter-domain secure service negotiation proposed in this paper, to provide privacy protection, techniques such as zero-knowledge proofs, homomorphic encryption, ring signatures, and attribute cryptosystems can be used to protect the anonymity of the negotiating parties.

5.3. Comparison of BSSN-SDNs with Similar Schemes

Compare BSSN-SDNs with the following security service negotiation schemes, as shown in Table 1.

Table 1. Comparison of BSSN-SDNs with similar schemes.

Scheme	IPsec	Open vSwitch	[15,16]		[19]	[21]	BSSN-SDNs
Application Scenarios	legacy network	SDN	SDN intradomain		between legacy network and SDN	SDN with NVF	SDN interdomain
Deployment Locations	host or router	host or OVS switch	SDN controller and SDN border switch		SDN controller and data plane	SDN controller and VNF nodes	SDN and blockchain
IKE SA negotiation	✓ ¹	✓	✓	✗ ²	✓	✗	✗
IPsec SA negotiation	✓	✓	✓	✗	✓	✗	✓
where to negotiate?	host or router	host or OVS switch	host or switch	SDN controller issue	SDN controller with host in legacy network	SDN controller issue to peer VNFs	smart contract on blockchain

¹ The symbol “✓” indicates that the scheme conducts the corresponding negotiations. ² The symbol “✗” indicates that the scheme does not conduct the corresponding negotiations.

Firstly, it can be found that most of the schemes are applied in traditional networks (such as IPsec) or within SDN domains (such as [15,16]), while BSSN-SDNs are applied in SDN multi-domains.

Secondly, the end-to-end manual configuration of security parameters used by IPsec and OVS is an error-prone and non-scalable task, especially for SDN inter-domain cross-organization scenarios.

Thirdly, to ensure the security of security service negotiation, the following schemes either require two-phase negotiation, such as using IKE SA negotiation to ensure the security of IPsec SA negotiation, or the IPsec SA is directly issued by the controller, which sacrifices the personalized security service negotiation among endpoints. However, in BSSN-SDNs, blockchain naturally guarantees the security and trustworthiness of SDN interdomain security services negotiation, which avoids an additional round of negotiation specifically to ensure the security of security service negotiation. At the same time, BSSN-SDNs provide personalized negotiations based on the security requirements of the negotiators.

Finally, in a multi-domain environment, combining SDN with blockchain to solve cross-domain problems is advantageous. Because SDN is logically centralized compared to traditional networks, the SDN controllers in each domain have a global view and global control capability. Interaction with blockchain nodes through SDN controllers can facilitate the implementation of required functions, such as the SDN interdomain security service negotiation in this scheme.

6. Experiments and Evaluation

The experiments evaluate the performance of BSSN-SDNs in two aspects: the performance of blockchain-based security service negotiation and the performance of secure service negotiation for the SDN interdomain.

6.1. Simulation Setup

In order to verify the performance of BSSN-SDNs, we utilized Mininet to build the simulated SDN multi-domain environment. Then we adopt Floodlight as the controller and develop the secure service negotiation function and the *BlockApp* function as the controller applications. In addition, we deploy the FISCO BCOS [34] as the blockchain node function. FISCO BCOS is an open-source consortium blockchain platform, which fits well with the

BSSN-SDNs. The secure service policies are transmitted and stored in JSON file format. The experiments are conducted on an Ubuntu 18.04 system on a VMware Workstation 15 Pro virtual machine. The host is configured with an Intel Core i7-10750H 2.6 GHz CPU and 32 GB of RAM.

The topology is built as shown in Figure 10. Six SDN domains are connected sequentially, with four switches in the data plane of each domain.

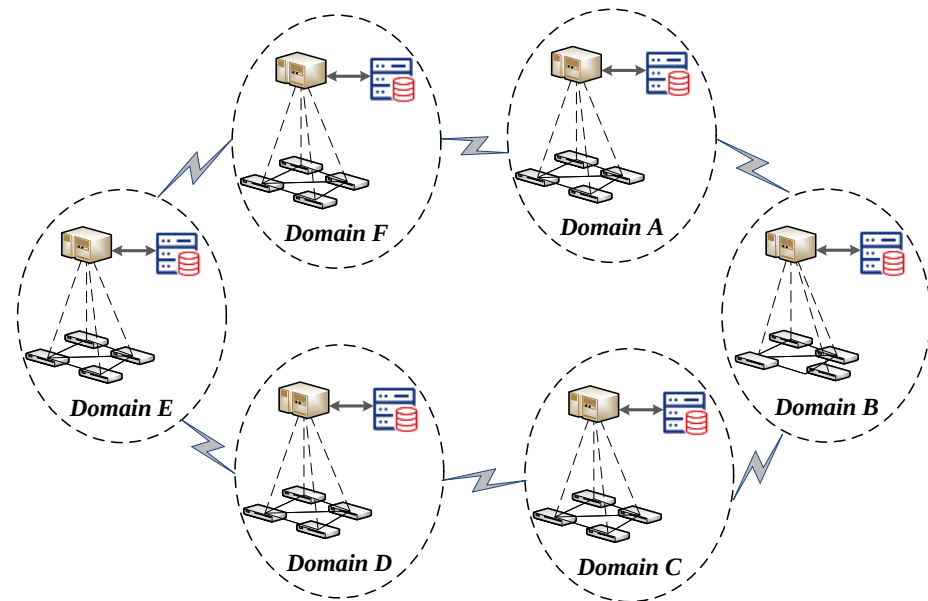


Figure 10. The experimental topology for BSSN-SDNs.

6.2. Experimental Results

6.2.1. Performance Evaluation of Blockchain-Based Security Service Negotiations

Task 1: Time overhead of blockchain-based security service negotiation

This experiment tested the time consumed by the blockchain to execute secure service negotiation when uploading different numbers of secure service policies in batches. Each transaction in the experiment contains 10 policies, and the block timeout is set to 1 s. Moreover, BPFC-SDNs implement cross-domain forwarding control policy sharing based on blockchain. It solves the problems of cross-domain policy agnosticism and policy conflict for SDN inter-domain forwarding control by issuing the global shared policy to each SDN domain and achieving cross-domain policy collaboration by each domain controller. Meanwhile, blockchain-based technology ensures the security and trustworthiness of the policy collaboration process [23]. Because the performance of block-chain-based forwarding control policy sharing was tested in BPFC-SDNs, BPFC-SDNs is used as a comparison scheme for BSSN-SDNs to compare and analyze the time overhead of blockchain-based task execution between the two. The experimental results are shown in Figure 11.

The experimental results in Figure 11 show that as the number of security service policies uploaded in batches goes from 500 to 2000, the time for the blockchain to complete the security service negotiation in BSSN-SDNs rises slowly from about 1.2 s to about 1.83 s, and when the number of policies uploaded in batches reaches 2500, the negotiation time rises steeply to about 3 s. This is because, compared to batch up-loading 2000 policies, batch uploading 2500 policies requires the generation of a new block, at which point the waiting time for generating a new block brings a significant increase in the negotiation time.

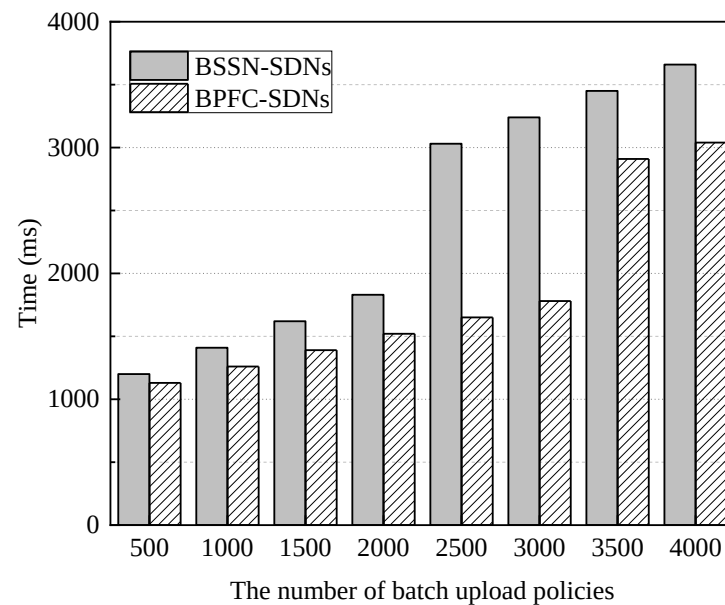


Figure 11. The time overhead of blockchain-based secure service negotiation.

In addition, it can be seen in Figure 11 that for batch uploading the same number of policies, the blockchain execution time of BSSN-SDNs is longer than that of BPFC-SDNs. This is because in BPFC-SDNs, the blockchain policy management smart contract only performs policy publishing, and its smart contract has a time complexity of $O(n)$, whereas the security service negotiation smart contract in BSSN-SDNs needs to perform policy matching in addition to performing policy publishing, and its smart contract has a time complexity of $O(Cn)$ ($1 \leq C \leq 2$). Therefore, when the same number of policies are uploaded, the blockchain-based execution time in BSSN-SDNs is longer than that in BPFC-SDNs. But it can be found that the core of BPFC-SDNs is forwarding control, whose main function is to control data forwarding based on policies by the controller when the data flow arrives, while the core of BSSN-SDNs is security service negotiation, whose main function is to complete the security service negotiation by the blockchain before the data flow arrives. Therefore, the time spent on blockchain-based security service negotiation in BSSN-SDNs hardly affects data forwarding, so although the overhead of BSSN-SDNs is longer than that of BPFC-SDNs, it is acceptable.

Task 2: the throughput and average time overhead of blockchain-based secure service negotiation

Figures 12 and 13 show the average time overhead for blockchain to complete a security service negotiation and the throughput of performing a security service negotiation when different numbers of policies are uploaded in a bath during the above experiments. It can be found that when the batch uploading of security service policies reaches about an integer multiple of 2000 policies, the average time overhead of security service negotiation is the smallest, which is about 0.92 ms, and the throughput is the largest in test, which is about 1093 completed negotiations per second. This is because when the number of policies received by the blockchain in a batch is about an integer multiple of 2000, every block generated is filled, and this is when the blockchain performance, that is, the number of transactions that can be processed per second, is near its peak in this experimental setting. Compared to BPFC-SDNs, BSSN-SDNs have a larger average latency and a smaller throughput, which is because the contract complexity of BSSN-SDNs is slightly higher than that of BPFC-SDNs.

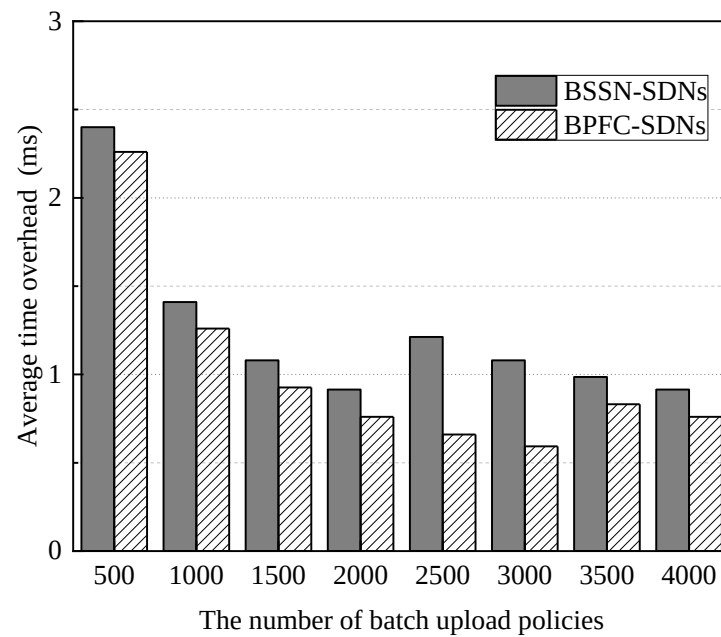


Figure 12. The average time overhead of blockchain-based secure service negotiation.

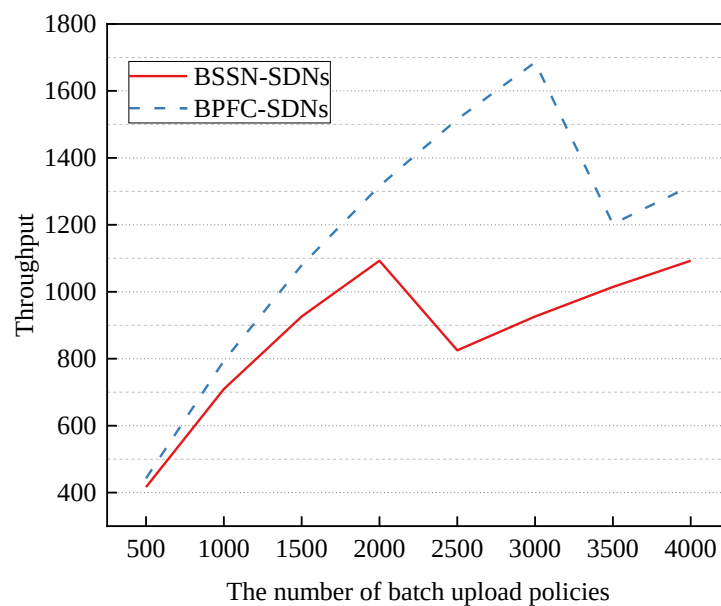


Figure 13. The throughput of blockchain-based secure service negotiation.

6.2.2. Performance Evaluation of Security Service Negotiation for SDN Interdomain

In BSSN-SDNs, SDN interdomain security service negotiation is triggered when an SDN domain formulates and submits a set of new security service policies. In this experiment, the total latency of SDN interdomain security service negotiation includes the time overhead of the following six processes: (1) SDN controller application SSN encrypts the security policy; (2) *BlockApp* generates the security service negotiation transaction; (3) *BlockApp* submits the transaction *SSNTrans* to the blockchain, and the blockchain validates the transaction; (4) the blockchain executes the security service negotiation and generates the block by consensus; (5) issues the negotiation result to the relevant SDN domain controllers; (6) the controllers map the security service receipt to *NSS* and write it to the local security service repository.

Task 3: Total latency for two domains with different distances to complete a security service negotiation

The experiments tested the total delay for BSSN-SDNs to perform SDN interdomain security service negotiation and the time for the blockchain to perform security service negotiation (the 4th phase mentioned above) when a domain performs one security service negotiation with a domain at a different distance. The distance here refers to the number of hops at the domain level. The experiment tested the total delay for BSSN-SDNs to complete one SDN interdomain security service negotiation and the time for the blockchain to complete one security service negotiation in the case that domain A negotiates security services with five other domains B\C\D\E\F, respectively. The negotiation experiments for different distances are repeated 30 times individually, and the averages are taken for the test times. The experimental results are shown in Figures 14 and 15.

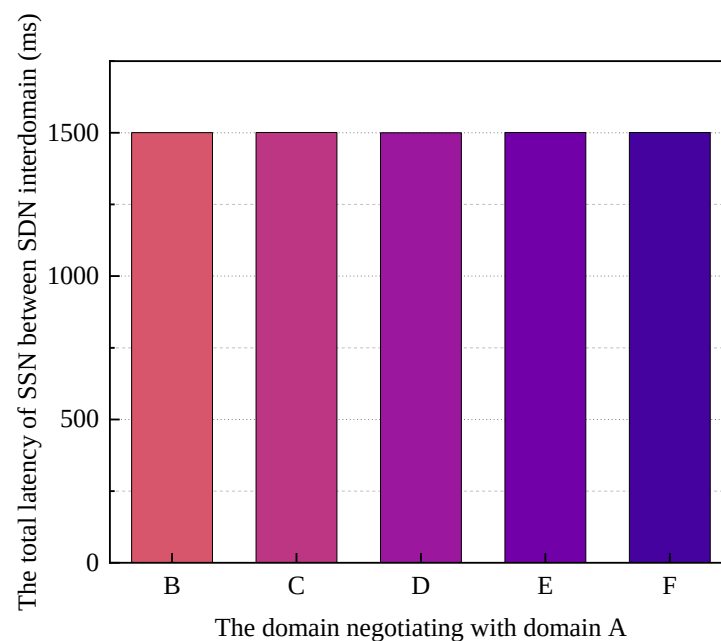


Figure 14. The total latency of secure service negotiation between SDN interdomains.

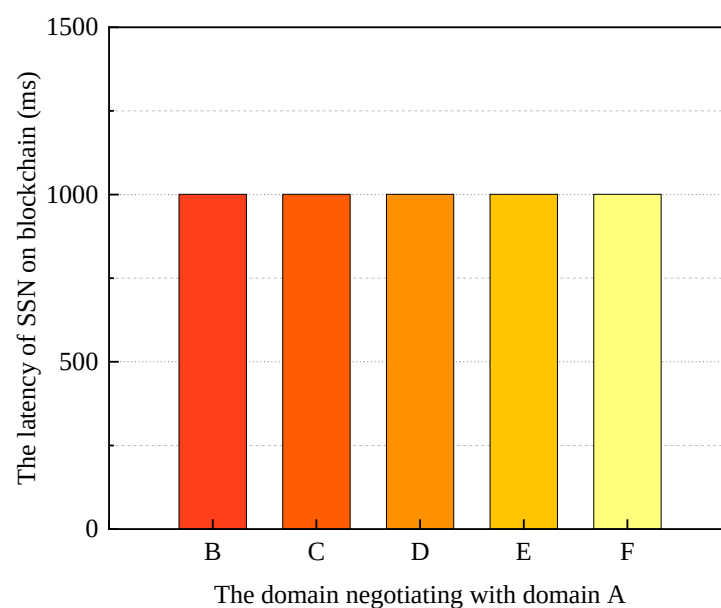


Figure 15. The latency of secure service negotiation on blockchain.

From the experimental results in Figures 14 and 15, it can be seen that although the distance (distance here refers to the number of hops at the domain level, not the physical distance) between the two domains participating in the negotiation is different, the blockchain-based interdomain security service negotiation latency of SDN is similar, and the total interdomain SDN latency and the on-blockchain latency in this experimental environment are about 1.5 s and 1 s, respectively. Therefore, in the SDN multidomain environment, the blockchain-based security service negotiation delay of BSSN-SDNs is weakly correlated with the number of hops between SDN domains.

Task 4: Total latency for multiple domains to complete security service negotiation

The experiments measured the total delay of BSSN-SDNs to complete SDN interdomain security service negotiation and the latency of blockchain to complete security service negotiation when different numbers of SDN domains request security service negotiation at the same time, respectively. The security service negotiation policy in the experiment is configured so that one successful security service negotiation can be completed between every two domains. Each experiment is repeated 30 times, and the test time is averaged. The experimental results are shown in Figure 16.

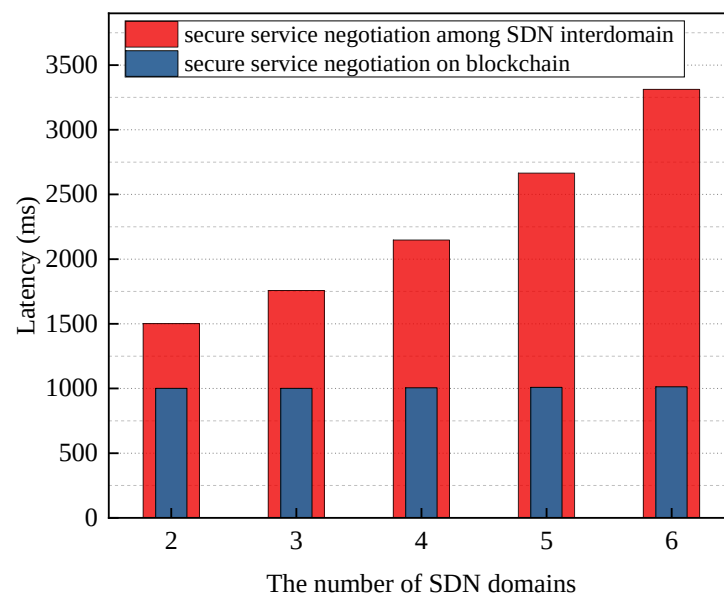


Figure 16. The latency of secure service negotiation among different numbers of domains.

The experimental results show that, firstly, for different numbers (2–6) of domains re-requesting security service negotiation at the same time, the latency of the blockchain to execute security service negotiation is growing slowly and takes similar time, all slightly more than 1 s. This is due to the fact that when there are few negotiation policies, the blockchain takes a large percentage of the time to generate the block. Secondly, the total delay of the SDN interdomain security service negotiation increases gradually with the number of domains involved in the negotiation and shows a quadratic tendency, as shown in Figure 17. This is because when there are n ($2 \leq n \leq 6$) domains performing security service negotiation at the same time and the policy is configured to allow every two domains to be able to complete a successful security service negotiation, the number of times SDN interdomain performs security service negotiation is about $n(n - 1)$.

In summary, the security service negotiation latency of BSSN-SDNs positively correlates with the number of policies to be negotiated as well as the number of domains participating in the negotiation, and it is weakly correlated with the hop distance between the participating domains.

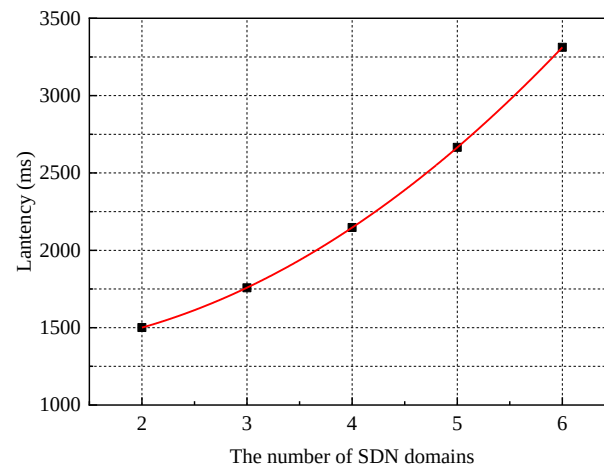


Figure 17. The order 2-regression for the total latency of SSN among SDN interdomains.

7. Conclusions

In this paper, BSSN-SDNs, a blockchain-based SDN interdomain security service negotiation scheme, is proposed. It can automatically perform security service negotiation to meet the demand for dynamic security service negotiation between SDN domains as well as to solve the problem of the lack of security and trustworthiness of SDN interdomain security service negotiation. It is discussed that BSSN-SDNs can guarantee the timeliness, security, and trustworthiness of security service negotiation within the SDN interdomain. The experimental results show that the delay in blockchain-based security service negotiation is within an acceptable range and positively correlates with the number of policies and domains involved in the negotiation. In the future, it is expected that the application can be extended into the field of multi-domain IoT.

Author Contributions: Conceptualization, Y.M. and C.C.; methodology, Y.M. and P.W.; validation, J.X. and L.Y.; formal analysis, Y.M.; writing—original draft preparation, Y.M.; writing—review and editing, Y.M. and C.C.; visualization, Y.M.; project administration, C.C.; funding acquisition, C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 61572517, and the Science and Technology Project of Henan Province, grant number 222102210070.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- McKeown, N. Software-defined networking. *INFOCOM Keynote Talk*. **2009**, *17*, 30–32.
- Flauzac, O.; González, C.; Hachani, A.; Nolot, F. SDN based Architecture for IoT and Improvement of the Security. In Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwangju, Republic of Korea, 24–27 March 2015; pp. 688–693. [\[CrossRef\]](#)
- Nisar, K.; Jimson, E.R.; Hijazi, M.H.A.; Welch, I.; Hassan, R.; Aman, A.H.M.; Sodhro, A.H.; Pirbhulal, S.; Khan, S. A Survey on the Architecture, Application, and Security of Software Defined Networking: Challenges and Open Issues. *Internet Things* **2020**, *12*, 100289. [\[CrossRef\]](#)
- Rahman, A.; Islam, M.J.; Band, S.S.; Muhammad, G.; Hasan, K.; Tiwari, P. Towards a Blockchain-SDN-based Secure Architecture for Cloud Computing in Smart Industrial IoT. *Digit. Commun. Netw.* **2023**, *9*, 411–421. [\[CrossRef\]](#)
- Tong, W.; Dong, X.; Shen, Y.; Jiang, X.; Zhang, Z. A Blockchain-driven Data Exchange Model in Multi-domain IoT with Controllability and Parallelity. *Future Gener. Comput. Syst.* **2022**, *135*, 85–94. [\[CrossRef\]](#)
- Makhdoom, I.; Zhou, I.; Abolhasan, M.; Lipman, J.; Ni, W. PrivySharing: A Blockchain-based Framework for Privacy-preserving and Secure Data Sharing in Smart Cities. *Comput. Secur.* **2020**, *88*, 101653. [\[CrossRef\]](#)
- Siddiqui, S.; Hameed, S.; Shah, S.A.; Khan, A.K.; Aneiba, A. Smart Contract-based Security Architecture for Collaborative Services in Municipal Smart Cities. *J. Syst. Archit.* **2023**, *135*, 102802. [\[CrossRef\]](#)
- Singh, P.; Masud, M.; Hossain, M.S.; Kaur, A. Cross-domain Secure Data Sharing using Blockchain for industrial IoT. *J. Parallel Distrib. Comput.* **2021**, *156*, 176–184. [\[CrossRef\]](#)

9. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 25 March 2024).
10. Yang, Z.; Wang, R.; Wu, D.; Yang, B.; Zhang, P. Blockchain-enabled Trust Management Model for the Internet of Vehicles. *IEEE Internet Things J.* **2021**, *10*, 12044–12054. [\[CrossRef\]](#)
11. Liu, Y.; Wang, J.; Yan, Z.; Wan, Z.; Jäntti, R. A Survey on Blockchain-based Trust Management for Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 5898–5922. [\[CrossRef\]](#)
12. Zhao, Y.; Zhao, J.; Jiang, L.; Tan, R.; Niyato, D.; Li, Z.; Lyu, L.; Liu, Y. Privacy-preserving Blockchain-based Federated Learning for IoT Devices. *IEEE Internet Things J.* **2020**, *8*, 1817–1829. [\[CrossRef\]](#)
13. Qi, Y.; Hossain, M.S.; Nie, J.; Li, X. Privacy-preserving Blockchain-based Federated Learning for Traffic Flow Prediction. *Future Gener. Comput. Syst.* **2021**, *117*, 328–337. [\[CrossRef\]](#)
14. Gai, K.; Wu, Y.; Zhu, L.; Zhang, Z.; Qiu, M. Differential Privacy-based blockchain for Industrial internet-of-things. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4156–4165. [\[CrossRef\]](#)
15. Lopez-Millan, G.; Marin-Lopez, R.; Pereniguez-Garcia, F. Towards a Standard SDN-based IPsec Management Framework. *Comput. Stand. Interfaces* **2019**, *66*, 103357. [\[CrossRef\]](#)
16. López-Millán, G.; Marín-López, R.; Pereñíguez-García, F.; Canovas, O.; Espín, J.A.P. Analysis and Practical Validation of a Standard SDN-based Framework for IPsec Management. *Comput. Stand. Interfaces* **2023**, *83*, 103665. [\[CrossRef\]](#)
17. Hauser, F.; Häberle, M.; Schmidt, M.; Menth, M. P4-ipsec: Site-to-site and Host-to-site VPN with IPsec in p4-based SDN. *IEEE Access* **2020**, *8*, 139567–139586. [\[CrossRef\]](#)
18. Hauser, F.; Schmidt, M.; Häberle, M.; Menth, M. P4-MACsec: Dynamic Topology Monitoring and Data Layer Protection with MACsec in P4-based SDN. *IEEE Access* **2020**, *8*, 58845–58858. [\[CrossRef\]](#)
19. Son, J.; Xiong, Y.; Tan, K.; Wang, P.; Gan, Z.; Moon, S. Protego: Cloud-Scale Multitenant IPsec Gateway. In Proceedings of the 2017 USENIX Annual Technical Conference, Santa Clara, CA, USA, 12–14 July 2017; pp. 473–485. Available online: <https://www.usenix.org/conference/atc17/technical-sessions/presentation/son> (accessed on 7 February 2024).
20. Coly, A.; Mbaye, M. S-SDS: A Framework for Security Deployment as Service in Software Defined Networks. In Proceedings of the Innovations and Interdisciplinary Solutions for Underserved Areas: Third EAI International Conference, Cairo, Egypt, 14–15 February 2019; Proceedings 3. Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 92–103. [\[CrossRef\]](#)
21. Gunleisen, H.; Kemmerich, T.; Gkioulos, V. Dynamic Setup of IPsec VPNs in Service Function Chaining. *Comput. Netw.* **2019**, *160*, 77–91. [\[CrossRef\]](#)
22. Wibowo, F.X.; Gregory, M.A.; Ahmed, K.; Gomez, K.M. Multi-domain Software Defined Networking: Research Status and Challenges. *J. Netw. Comput. Appl.* **2017**, *87*, 32–45. [\[CrossRef\]](#)
23. Ma, Y.; Wang, Z.; Chang, C.; Wu, P. BPFC-SDNs: A Blockchain-Based and Policy-Oriented Forwarding Control for the SDN Interdomain. *Secur. Commun. Netw.* **2023**, *1*, 1104565. [\[CrossRef\]](#)
24. Hu, J.; Reed, M.; Thomos, N.; Al-Naday, M.F.; Yang, K. Securing SDN-controlled IoT Networks through Edge Blockchain. *IEEE Internet Things J.* **2020**, *8*, 2102–2115. [\[CrossRef\]](#)
25. Gao, J.; Agyekum, K.O.B.O.; Sifah, E.B.; Acheampong, K.N.; Xia, Q.; Du, X.; Guizani, M.; Xia, H. A Blockchain-SDN-enabled Internet of Vehicles Environment for Fog Computing and 5G Networks. *IEEE Internet Things J.* **2019**, *7*, 4278–4291. [\[CrossRef\]](#)
26. Ren, W.; Sun, Y.; Luo, H.; Guizani, M. SiLedger: A Blockchain and ABE-based Access Control for Applications in SDN-IoT Networks. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4406–4419. [\[CrossRef\]](#)
27. Ahmad, S.; Mir, A.H. Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers. *J. Netw. Syst. Manag.* **2021**, *29*, 1–59. [\[CrossRef\]](#)
28. Antichi, G.; Castro, I.; Chiesa, M.; Fernandes, E.L.; Lapeyrade, R.; Kopp, D.; Han, J.H.; Bruyere, M.; Dietzel, C.; Gusat, M.; et al. Endeavour: A scalable SDN Architecture for Real-world Ixps. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 2553–2562. [\[CrossRef\]](#)
29. Karakus, M.; Duresi, A. A survey: Control Plane Scalability Issues and Approaches in Software-Defined Networking (SDN). *Comput. Netw.* **2017**, *112*, 279–293. [\[CrossRef\]](#)
30. Hess, F. Efficient Identity based Signature Schemes based on Pairings. In Proceedings of the Selected Areas in Cryptography: 9th Annual International Workshop, St. John's, NL, Canada, 15–16 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 310–324. [\[CrossRef\]](#)
31. Luu, L.; Chu, D.H.; Olickel, H.; Saxena, P.; Hobor, A. Making Smart Contracts Smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 254–269. [\[CrossRef\]](#)
32. Buccafurri, F.; De Angelis, V.; Idone, M.F.; Labrini, C.; Lazzaro, S. Achieving Sender Anonymity in Tor against the Global Passive Adversary. *Appl. Sci.* **2022**, *12*, 137. [\[CrossRef\]](#)
33. Dingledine, R.; Mathewson, N.; Syverson, P. Tor: The Second-generation Onion Router. In Proceedings of the 13th Conference on USENIX Security Symposium, San Diego, CA, USA, 9–13 August 2004; USENIX Association: Berkeley, CA, USA, 2004; pp. 303–320. [\[CrossRef\]](#)
34. FISCO-BCOS. Available online: https://www.fisco.org.cn/fisco_20.html (accessed on 17 March 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.