SLIM: A Secure and Lightweight Multi-Authority Attribute-Based Signcryption Scheme for IoT

Bei Gong, *Member, IEEE,* Chong Guo, Chong Guo, Chen Guo, Yao Sun, *Senior Member, IEEE,* Muhammad Waqas, *Senior Member, IEEE,* Sheng Chen, *Fellow, IEEE,*

***Abstract*—Although attribute-based signcryption (ABSC) offers a promising technology to ensure the security of IoT data sharing, it faces a two-fold challenge in practical implementation, namely, the linearly increasing computation and communication costs and the heavy load of single authority based key management. To this end, we propose a Secure and Lightweight Multi-authority ABSC scheme called SLIM in this paper. The signcryption and de-signcryption costs of devices are reduced to a small constant by offloading most of the computation to the edge server. To minimize communication and storage costs, a short and constant-size ciphertext is designed. Moreover, we adopt a hierarchical multi-authority architecture, setting up multiple attribute authorities that manage keys independently to prevent the bottleneck. Rigorous security analysis proves that the SLIM scheme can resist adaptive chosen ciphertext attacks and adaptive chosen message attacks under the standard model. Simulation experiments demonstrate the correctness of our theoretical derivations and the cost reduction of the SLIM scheme in computation, communication and storage.**

***Index Terms*—Attribute-based signcryption, multi-authority, outsourced computation, constant-size ciphertext, access control.**

I. Introduction

With the widespread deployment of the Internet of Things (IoT) in daily life, massive amounts of data are continuously generated [1], and cloud computing offers a promising solution to the data storage problem [2]. Outsourcing data to the cloud reduces the storage requirements of IoT devices. On the other hand, emerging edge computing has opened up new paths for resource-constrained IoT devices to handle complex tasks. Compared to cloud servers, edge servers are geographically closer to devices and can respond more quickly to device requests [3]. Cloud-edge-assisted IoT is the integration of cloud computing, edge computing and IoT. Edge servers have powerful computing capability and high-bandwidth communication capacity, and are responsible for providing low-latency computing services to IoT devices. The cloud has nearly unlimited storage space, and is responsible for providing storage and access services for data to IoT devices. However, the data stored in the cloud is out of the control of the data owner, which makes protecting the security of the data a critical and challenging issue.

Confidentiality and authenticity are two the ESsential requirements of data security [4]. Confidentiality requires that unauthorized entities cannot obtain information from the data [4]. Authenticity requires that the data was generated by a legitimate (or authorized) entity and has not been tampered with [5]. Encryption and signature techniques are the two most important ways to ensure confidentiality and authenticity [6], [7]. As a novel encryption technique, attribute-based encryption (ABE) can not only realize one-to-many encrypted transmission, but also provide access control [8]. Attribute-based signature (ABS) on the other hand allows users to determine the authenticity of data by verifying that the signature attributes satisfy the policy, which solves the problem of the signer's identity exposure in the traditional digital signature scheme [9]. By combining ABE and ABS in one logical step, attribute-based signcryption (ABSC) [10] inherits all the features of ABE and ABS. In addition, the computation and communication costs of ABSC are lower compared to the direct use of ABE and ABS. Hence, ABSC is a promising cryptographic primitive to effectively ensure the security of data.

However, applying ABSC to resource-constrained IoT devices faces a series of challenges. The first is the high computation cost that is common in attribute-based cryptography schemes. In the previous ABSC schemes [11]–[20], the computation cost required by the user increases linearly with the number of attributes involved. Some schemes [12], [21]–[23] even need to perform a large number of bilinear mapping operations that are considered the most complex mathematical operation in cryptography. Such an enormous computation cost is unaffordable for IoT devices. The second is high communication and storage costs. The ciphertext size of most ABSC schemes also scales linearly with the number of attributes. Hence, the more attributes, the more complex the access policy, and the larger the ciphertext size. This not only imposes a high communication cost on devices but also requires paying more to the cloud service provider because of the larger cloud storage space needed. Thus, many existing ABSC schemes suffer from a lack of usability due to their large ciphertext. The third is the presence of system

This work was supported in part by the National Key Research and Development Project of China under Grant 2019YFB2102303, in part by the National Natural Science Foundation of China under Grant 61971014. *(Corresponding author: Bei Gong.)*

Bei Gong is with Beijing Key Laboratory of Trusted Computing, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: gongbei@bjut.edu.cn).

Chong Guo is with Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: chongguo@emails.bjut.edu.cn).

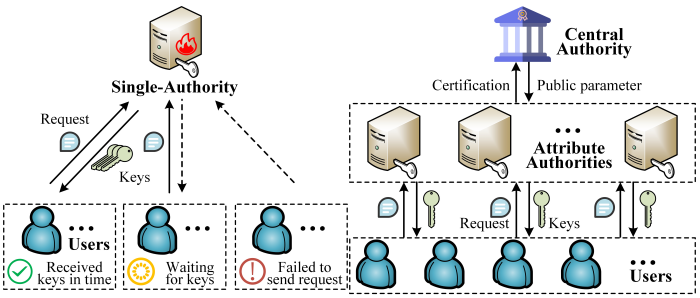
Chong Guo is with Beijing Trusty Cloud Technology Co., LTD, Beijing 100022, China (e-mail: ccieguo@126.com).

Chen Guo is with the China Cybersecurity Review Technology and Certification Center, Beijing 100011, China (e-mail: guochen@isccc.gov.cn).

Yao Sun is with James Watt School of Engineering, University of Glasgow, Glasgow, UK (e-mail: Yao.Sun@glasgow.ac.uk).

Muhammad Waqas is with the School of Computing and Mathematical Sciences, Faculty of Engineering and Science, University of Greenwich, SE10 9LS, London, UK and also with School of Engineering, Edtih Cowan University, 6027 WA, Australia. (e-mail: engr.waqas2079@gmail.com).

Sheng Chen is with School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK (e-mail: sqc@ecs.soton.ac.uk).



(a) Single-authority (b) Multi-authority

Fig. 1. Comparison of single-authority and multi-authority.

bottlenecks. Almost all traditional ABSC schemes adopt the single-authority architecture as shown in Fig. 1 (a). In these schemes, a single authority is responsible for managing all attributes, which means that it must respond to all key requests and undertake the generation, recording and distribution of keys for all attributes. However, the computation resources and communication resources of the authority are not unlimited. Therefore, the authority may not respond in time or even go offline in the face of overloaded requests. This affects the stability and availability of the system.

In this paper, we propose a **S**ecure and **Li**ghtweight **M**ulti-Authority ABSC (SLIM) scheme with outsourced computation and constant-size ciphertext for cloud-edge-assisted IoT. In the SLIM scheme, the legitimate users, i.e., devices in the system, can offload computing tasks to the edge server to reduce its own computation cost. The size of the ciphertext is independent of the number of attributes, to effectively reduce the communication cost of the device and the storage cost of the cloud. Moreover, a multi-authority architecture as illustrated in Fig. 1 (b) is adopted, and a fully trusted central authority (e.g., government agency, etc.) is responsible for building the system and multiple attribute authorities (e.g., functional departments, etc.) independently manage different attributes to prevent system bottlenecks. In summary, the main contributions of SLIM are as follows.

- **Lightweight**. (1) With the assistance of the edge server, the computation cost for the user is constant and low. The user needs only 8 exponentiation operations during signcryption, and only 2 exponentiation operations and 1 bilinear mapping operation during de-signcryption. In contrast, in the existing schemes, the computation cost needed by the user increases linearly with the number of attributes. Thus, the SLIM scheme significantly reduces the computation cost for users. (2) The ciphertext size of the SLIM scheme is constant and small, independent of the number of attributes and shorter than the existing schemes, involving only 5 group elements, which provides lower communication and storage costs.
- **Verifiable outsourced verification**. The SLIM scheme allows users to outsource the verification of the ciphertext to reduce costs, and provides verifiability to prevent the verification results returned by the edge server from mismatching with the ciphertext submitted by the user. Moreover, the SLIM scheme supports public verification, which means that any entity can verify the authenticity of the ciphertext.
- **Multi-authority**. Multiple attribute authorities in the SLIM replace a single authority in the traditional scheme, and are responsible for the complex generation and recording of keys. This disperses the key management burden and avoids the system bottleneck formed by the single authority.
- **Security**. We show that the SLIM guarantees message confidentiality and ciphertext unforgeability by proving indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2) and existential unforgeability under adaptive chosen message attacks (EUF-CMA) in the standard model. Additionally, the privacy of the signcryptor is achieved.

The remainder of this paper is structured as follows. Section II reviews the related prior work on attribute-based signcryption. Section III presents the preliminaries, system model and security models of the proposed SLIM scheme. Subsequently, we give the detailed construction of the SLIM scheme in Section IV. The security and performance of the SLIM are analysed and discussed in Sections V and VI, respectively. We conclude this paper in Section VII.

II. Related Work

To address the storage of massive amounts of IoT data, users may upload data to the cloud. Although ABSC can secure the data, high computation costs, linearly-increasing-size ciphertext and single authority become key challenges for ABSC deployment on IoT devices. We review outsourced computing, constant-size ciphertext and multi-authority architecture adopted in the literature to address these challenges.

To provide attribute-based access control, the signcryption and de-signcryption phases in ABSC are closely linked to the attributes, and hence linearly-increasing high computation cost is unavoidable. To mitigate this problem, the online/offline framework [11] and the outsourcing computing approach [12] were introduced. the outsourcing computing approach, which can offload computation costs, is more widely accepted and adopted than the online/offline framework. Outsourced computing in ABSC can be further divided into outsourcing decryption [12], outsourcing de-signcryption [13]–[16] and outsourcing signcryption [14], [16]. Hong *et al.* [12] proposed a key-policy (KP)-ABSC scheme with outsourced decryption, but the scheme is not very practical because of the lack of public verification and the high cost of verification imposed. Deng *et al.* [13] presented a ciphertext-policy (CP)-ABSC scheme with verifiable outsourced de-signcryption that allows users to de-signcrypt ciphertexts with the assistance of untrusted servers. Xu *et al.* [14] proposed a CP-ABSC that supports not only outsourced de-signcryption but also outsourced signcryption. However, the high verification cost and large ciphertext make this scheme unsuitable for resource-constrained environments. Yu *et al.* [16] designed a more lightweight hybrid key-ciphertext-policy (KCP)-ABSC scheme with low verification cost and small ciphertext. However, the signcryption cost of users in [16] is expensive.

In addition to high computation cost, high communication and storage costs are other constraints limiting ABSC

TABLE I
COMPARISON OF FEATURES FOR THE EXISTING ABSC SCHEMES AND THE PROPOSED SLIM

Scheme	Infrastructure		Security				Functionality				
	Policy	Access Structure	MC	CU	SM	SP	PV	MA	CS	OS	OD
[21]	CP	Access Tree	IND-CCA2	EUF-CMA	✗	✓	✗	✗	✗	✗	✗
[22]	CP	MSP & LSSS	IND-CCA2	EUF-CMA	✓	✓	✗	✗	✗	✗	✗
[17]	KP	LSSS	IND-CCA2	EUF-CMA	✓	✓	✓	✗	✓	✗	✗
[11]	CP	MBF	IND-CCA2	EUF-CMA	✗	✓	✓	✗	✗	✗	✗
[23]	CP	MBF	IND-CCA2	EUF-CMA	✓	✓	✓	✗	✗	✗	✗
[12]	KP	MBF	IND-CPA1	EUF-CMA	✓	✓	✗	✗	✗	✗	✓
[19]	KCP	MBF & Threshold	IND-CCA2	EUF-CMA	✓	✗	✓	✗	✓	✗	✗
[18]	CP	Threshold	IND-CCA2	EUF-CMA	✗	✓	✓	✗	✓	✗	✗
[13]	CP	MSP	IND-CPA2	EUF-CMA	✗	✓	✓	✗	✗	✗	✓
[14]	CP	MBF	IND-CCA2	EUF-CMA	✓	✓	✓	✓	✗	✓	✓
[15]	CP	Threshold	IND-CCA2	EUF-CMA	✓	✓	✓	✗	✓*	✗	✓
[16]	KCP	LSSS	IND-CCA2	EUF-CMA	✓	✓	✓	✗	✗	✓	✓
[20]	KP	LSSS	IND-CCA2	EUF-CMA	-	✓	✓	✓	✓	✗	✗
SLIM	KP	LSSS	IND-CCA2	EUF-CMA	✓	✓	✓	✓	✓	✓	✓

Notes: MC: message confidentiality; CU: ciphertext unforgeability; SM: standard model; SP: signcryptor privacy; PV: public verification; MA: multi-authority; CS: constant-size ciphertext; OS: outsourced signcryption; OD: outsourced decryption/de-signcryption. MBF: monotone Boolean function; MSP: monotone span program. IND-CPA1: indistinguishability under chosen-plaintext attack. ✓*: the scheme does not meet CS strictly.

deployment in the IoT. The linearly increasing ciphertext size with the number of attributes is the main reason for high communication and storage costs. Thus, Rao and Dutta [17] proposed the first KP-ABSC scheme with constant-size ciphertext and adopted the linear secret sharing scheme (LSSS) to provide fine-grained access control. Subsequently, the schemes proposed in [18], [19] also achieve constant-size ciphertext but they lack the expressiveness of the access policy due to the adoption of a threshold. Belguith *et al.* [15] extended the work of [18] by adding an update function, which enables the data owner to update the access policy of the ciphertext stored in the cloud. However, this feature imposes more communication costs on the data owner and more storage costs on the cloud, since the ciphertext involves the entire attribute space. Nevertheless, the data user does not have to download the full ciphertext and can recover the message using a constant size part.

A single authority may cause the attribute authority to become a performance bottleneck, since it is responsible for the generation and distribution of keys for all attributes. The multi-authority (MA) architecture was first introduced in the ABE scheme by Chase [24] and became a crucial feature of attribute-based cryptography. However, there exist very few MA-ABSC schemes in the literature [14], [20], [25], and these schemes require either high computation cost or high communication and storage cost, and are not suitable for IoT.

Table I summarizes the features of the latest ABSC schemes and the proposed SLIM in terms of infrastructure, security and functionality. Our SLIM scheme supports the highest number of crucial features compared with the existing schemes. In particular, the SLIM supports public verification, signcryptor privacy, constant-size ciphertext, outsourced computation and multi-authority. Additionally, users of the SLIM can offload most of the computation to the edge server and the signcryption cost is constant for users, which is extremely attractive. In contrast, the signcryption cost required by the user linearly increases with the number of attributes in all the existing schemes. Later we will show that in the standard

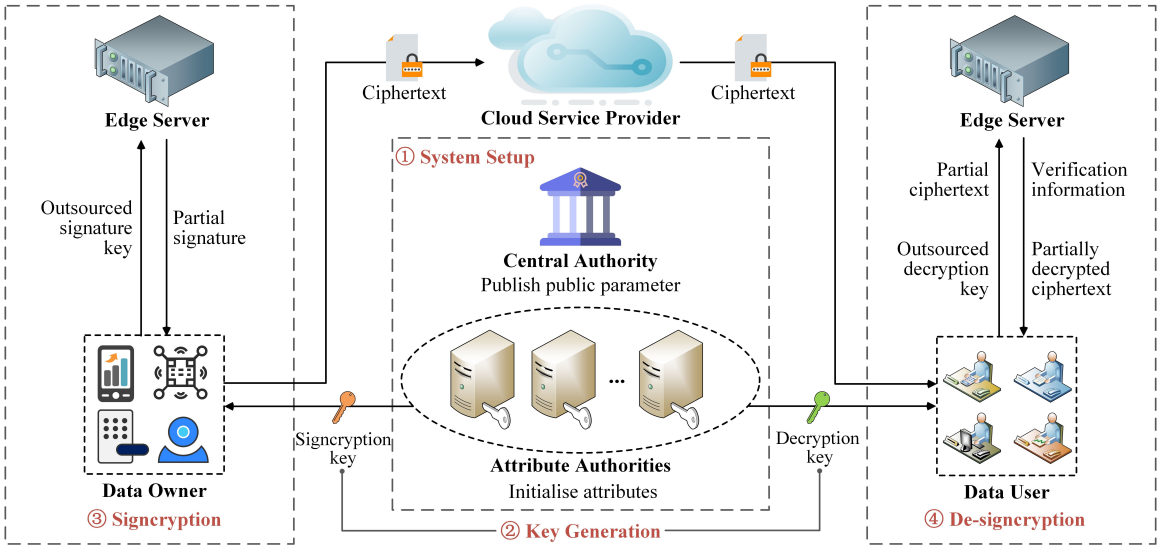


Fig. 2. The architecture of SLIM system.

model, the SLIM satisfies IND-CCA2 under the decisional n-bilinear Diffie–Hellman exponent (n-DBDHE) problem and EUF-CMA under the computation n-Diffie–Hellman exponent (n-CDHE) problem.

III. Framework of SLIM Scheme

A. Preliminaries

In this subsection, we review some background knowledge related to the SLIM scheme. $[X]$ represents a positive integer set $\{1, 2, \cdots, X\}$ and $\mathbb{Z}_p$ denotes the integer field module p, while $\xleftarrow{R}$ denotes a random selection of elements from a group. Due to limited space, bilinear maps, LSSS and security assumptions are described in Supplemental Material A.

B. System Model

As shown in Fig. 2, the SLIM system involves six entities: central authority (CA), attribute authority (AA), data owner (DO), data user (DU), cloud service provider (CSP) and edge server (ES).

1) *Central Authority.* The CA is a fully trusted entity that builds the system and publishes public parameter.
2) *Attribute Authority.* The AA is also a fully trusted entity responsible for the initialisation of attributes and the generation of users' keys. The SLIM scheme has multiple AAs, and the number of AAs is N. Each AA independently manages a set of signature attributes and a set of encryption attributes.
3) *Data Owner.* The DO is the user who provides the data. Before uploading the data to the CSP, the DO signcrypts the data, to ensure that the data is provided by the authorized DO, and specifies the access rights of DUs.
4) *Data User.* The DU is the user who uses the data. After downloading the ciphertext from the CSP, the DU verifies the authenticity and decrypts the valid ciphertext.
5) *Cloud Service Provider.* The CSP is a semi-trusted third-party organization with massive storage space that is responsible for storing ciphertexts. The CSP will verify the authenticity of the ciphertext upon reception and refuse to store invalid ones.
6) *Edge Server.* The ES is a semi-trusted entity deployed at the edge of the network that is responsible for assisting users in handling complex computations involved in the signcryption and de-signcryption.

As shown in Fig. 3, the process flow of the SLIM system consists of four phases: system setup (P1), key generation (P2), signcryption (P3) and de-signcryption (P4). Besides illustrating the process sequence of the four phases, Fig. 3 also describes the algorithms in each phase and the interactions between the entities involved.

Phase 1 (System Setup). This phase includes the $GlobalSetup$ algorithm and the $AuthoritySetup$ algorithm.

1) $GlobalSetup\,(\kappa) \rightarrow PP$: Executed by the CA. Given the security parameter κ, the algorithm generates the public parameter PP.
2) $AuthoritySetup\,(PP) \rightarrow (MSK, PK)$: Executed by each AA. Given the public parameter PP, the algorithm outputs the master secret key MSK and attribute public key PK.

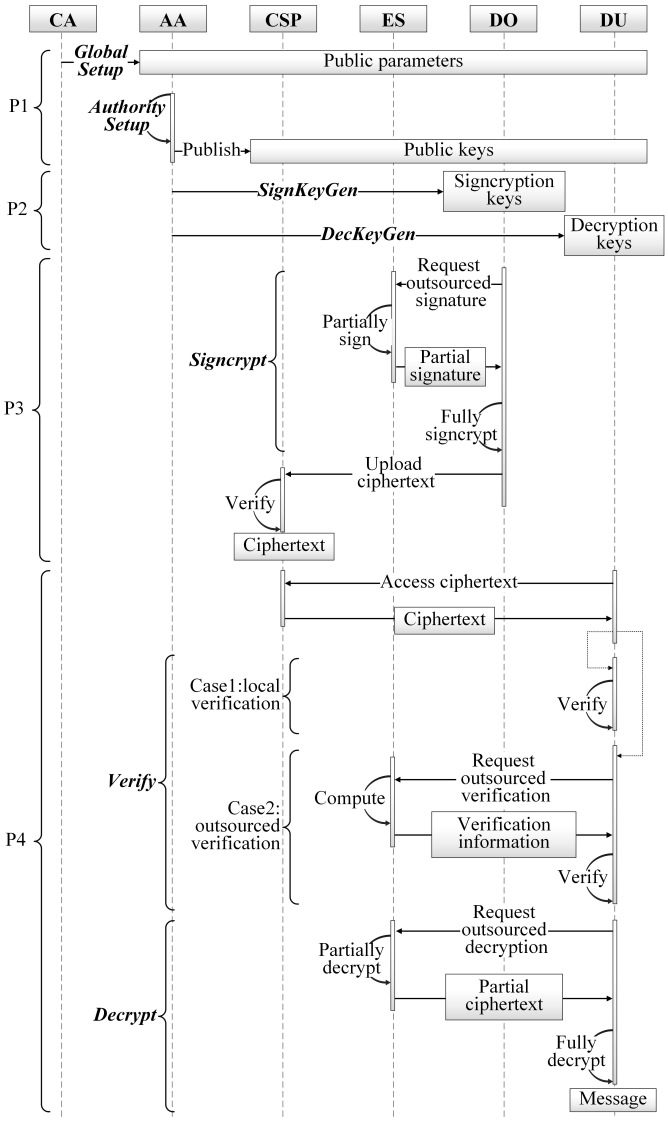


Fig. 3. The process flow diagram of SLIM system.

Phase 2 (Key Generation). This phase is executed independently by each AA and includes the $SignKeyGen$ algorithm and the $DecKeyGen$ algorithm.

1) $SignKeyGen(PP, MSK, PK, (\mathbb{S}, \rho)) \rightarrow (SSK, OSK)$: Given the public parameter PP, own master secret key MSK, attribute public key PK, and the signature LSSS structure $(\mathbb{S}, \rho)$ of the DO, the algorithm outputs the signature key, including the signature secret key SSK and the outsourced signature key OSK.
2) $DecKeyGen(PP, MSK, PK, (\mathbb{D}, \phi)) \rightarrow (DSK, ODK)$: Given the public parameter PP, own master secret key MSK, attribute public key PK, and the decryption LSSS structure $(\mathbb{D}, \phi)$ of the DU, the algorithm outputs the decryption key, including the decryption secret key DSK and the outsourced decryption key ODK.

Phase 3 (Signcryption). This phase is executed by the DO with the assistance of the ES, using the $Signcrypt$ algorithm which is described as follows.

1) $Signcrypt\,(PP, PK, SSK, OSK, M, U_s, U_e) \rightarrow CT$: Given the public parameter PP, attribute public key PK, signature secret key SSK, outsourced signature key OSK, message M, signature attribute set U_s and encryption attribute set U_e, the algorithm outputs the ciphertext CT.

Phase 4 (De-signcryption). This phase includes the $Verify$ algorithm and the $Decrypt$ algorithm.

1) $Verify\,(PP, PK, CT) \rightarrow 1$ or $\perp$: Executed by any entity alone (Case1) or the DU with the assistance a ES (Case2). Given the public parameter PP, attribute public key PK and ciphertext CT, if the ciphertext is valid, the algorithm outputs 1; otherwise it outputs the terminator $\perp$.
2) $Decrypt(PP, PK, DSK, ODK, CT) \rightarrow M$ or $\perp$: Executed by the DU with the assistance of a ES. Given the public parameter PP, attribute public key PK, decryption secret key DSK, outsourced decryption key ODK and ciphertext CT, if the encryption attribute set U_e satisfies the DU's decryption LSSS structure $(\mathbb{D}, \phi)$, the algorithm outputs the message M; otherwise it outputs $\perp$.

C. Security Model

This subsection introduces the security model for the SLIM.

1) Message Confidentiality: The message confidentiality of the SLIM guarantees that the ciphertext cannot be deciphered by unauthorized entities. The security model is defined by the indistinguishability of ciphertexts under selective attribute set and adaptive chosen ciphertext attacks (IND-sAtt-CCA2). This is an attacking game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In the game, $\mathcal{C}$ is in charge of simulating and operating the system. $\mathcal{A}$ acts as an unauthorized entity and tries to distinguish the ciphertext of two random messages. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{SLIM-IND-sAtt-CCA2} = |\Pr[b' = b] - 1/2|$, where b denotes the message selected by $\mathcal{C}$ to generate ciphertext, and b' is the guess given by $\mathcal{A}$.

Definition 1: **(SLIM-IND-sAtt-CCA2 security).** The SLIM scheme is $(\mathcal{T}, q_{SG}, q_{DG}, q_{SC}, q_{DS}, \varepsilon)$-IND-sAtt-CCA2 secure if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ in time at most $\mathcal{T}$ that makes at most q_{SG} times *SignKeyGen Query*, q_{DG} times *DecKeyGen Query*, q_{SC} times *Signcrypt Query* and q_{DS} times *De-signcrypt Query*, its advantage is $Adv_{\mathcal{A}}^{SLIM-IND-sAtt-CCA2} \leq \varepsilon$.

2) Ciphertext Unforgeability: The ciphertext unforgeability of the SLIM ensures that an entity without signing authority cannot generate valid ciphertexts. This security model is defined on the existential unforgeability under selective attribute set and adaptive chosen message attacks (EUF-sAtt-CMA). In the game, $\mathcal{C}$ is responsible for simulating and operating the system and $\mathcal{A}$ acts as an entity without specific signing authority and tries to forge a valid ciphertext. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{SLIM-EUF-sAtt-CMA} = \Pr[\mathcal{A}\text{ wins}]$.

Definition 2: **(SLIM-EUF-sAtt-CMA security).** The SLIM scheme is $(\mathcal{T}, q_{SG}, q_{DG}, q_{SC}, q_{DS}, \varepsilon)$-EUF-sAtt-CMA secure if for any PPT adversary $\mathcal{A}$ in time at most $\mathcal{T}$ that makes at most q_{SG} times *SignKeyGen Query*, q_{DG} times *DecKeyGen Query*, q_{SC} times *Signcrypt Query* and q_{DS} times *De-signcrypt Query*, its advantage is $Adv_{\mathcal{A}}^{SLIM-EUF-sAtt-CMA} \leq \varepsilon$.

3) Signcryptor Privacy: The signcryptor privacy of the SLIM ensures that other entities cannot infer the identity of the signcryptor from the ciphertext. The model is defined by the game of indistinguishability of ciphertext under known-plaintext attacks (IND-sAtt-KPA). In the game, $\mathcal{C}$ maintains the system, and $\mathcal{A}$ acts as a curious entity and tries to distinguish two ciphertexts generated by two different signature keys. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{SLIM-IND-sAtt-KPA} = |\Pr[b' = \mathbf{b}\,] - 1/2\,|$, where b denotes the signature key selected by $\mathcal{C}$ to generate the ciphertext and b' is the guess given by $\mathcal{A}$. It should be noted that $\mathcal{A}$ can only give the guess b' based on the distribution of CT_b.

Definition 3: **(Signcryptor privacy).** The SLIM is said to have signcryptor privacy if the distribution of the ciphertext is independent of the signature key.

IV. Construction of SLIM Scheme

Let $\mathcal{M} = \{0,1\}^{l_m}$ denotes the message space, where l_m is the maximum bit length of the message. $\mathcal{U}_s = \{att_{s,j,i}\}$ and $\mathcal{U}_e = \{att_{e,j,i}\}$ denotes the signature attribute space and encryption attribute space, respectively, where j is the attribute authority label and i is the unique label of the attribute, while $\mathcal{U}_s \bigcap \mathcal{U}_e = \varnothing$. There are N attribute authorities $AA_1, \cdots, AA_N$ in the system. AA_j is independently responsible for key management of $\mathcal{U}_{s,j} \subset \mathcal{U}_s$ and $\mathcal{U}_{e,j} \subset \mathcal{U}_e$. Furthermore, $\mathcal{U}_{s,j} \bigcap \mathcal{U}_{s,j'} = \varnothing$, $\mathcal{U}_{e,j} \bigcap \mathcal{U}_{e,j'} = \varnothing$, $\forall j, j' \in [N]$, $j \neq j'$. We now present the concrete construction of the SLIM Scheme.

Phase 1 (System Setup). This phase uses $GlobalSetup$ and $AuthoritySetup$ to generate public parameters and public/secret key pairs for each attribute authority.

1) $\boldsymbol{GlobalSetup}(\kappa)$: Giventhe the security parameter κ, the CA specifies the message space $\mathcal{M} = \{0,1\}^{l_m}$, the signature attribute space $\mathcal{U}_s = \{att_{s,j,i}\}$ and the encryption attribute space $\mathcal{U}_e = \{att_{e,j,i}\}$. Subsequently, the CA selects two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p, where p is a large prime number satisfying $p \geq \kappa(l+1)$. Then, the CA selects four collision-resistant hash functions $H_1 : \mathbb{G} \rightarrow \mathbb{Z}_p$, $H_2 : \mathbb{G}_T \times \mathbb{G} \times \mathbb{Z}_p \rightarrow \{0,1\}^{l_m}$, $H_3 : \{0,1\}^* \rightarrow \{0,1\}^l$ and $H_4 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, where l is sufficiently large to resist collisions and $l \leq l_m$. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. Define a bilinear group $B = (p, \mathbb{G}, \mathbb{G}_T, e)$. Finally, the CA selects $K_0, T_0, \delta_1, \delta_2, \mu_0, \mu_1, \cdots, \mu_l \xleftarrow{R} \mathbb{G}$, and generates the public parameters: $PP = \big(B, g, K_0, T_0, \delta_1, \delta_2, \mu_0, \mu_1, \cdots, \mu_l, H_1, H_2, H_3, H_4, \mathcal{M}, \mathcal{U}_s, \mathcal{U}_e\big)$.
2) $\boldsymbol{AuthoritySetup}(PP)$: According to PP, the AA_j generates its public/secret key pair. The AA_j chooses a unique $\alpha_j \xleftarrow{R} \mathbb{Z}_p^*$ and computes $Y_j = e(g,g)^{\alpha_j}$. Then, the AA_j selects $K_{j,i} \xleftarrow{R} \mathbb{G}$ for each signature attribute $att_{s,j,i} \in \mathcal{U}_{s,j}$ and $T_{j,i} \xleftarrow{R} \mathbb{G}$ for each encryption attribute $att_{e,j,i} \in \mathcal{U}_{e,j}$. Finally, the AA_j generates the master

secret key $MSK_j = (\alpha_j)$ and public key $PK_j = \left(Y_j, \{K_{j,i}\}_{att_{s,j,i} \in \mathcal{U}_{s,j}}, \{T_{j,i}\}_{att_{e,j,i} \in \mathcal{U}_{e,j}}\right)$.

Phase 2 (Key Generation). This phases is run independently by each attribute authority. The $SignKeyGen$ algorithm generates a signature key for the DO, including a signature secret key and an outsourced signature key. The $DecKeyGen$ algorithm generates a decryption key for the DU, including a decryption secret key and an outsourced decryption key.

1) $\boldsymbol{SignKeyGen}(PP, MSK, PK, (\mathbb{S}, \rho))$: $(\mathbb{S}_j, \rho_j)$ is a signature LSSS structure, where $\mathbb{S}_j$ is an $l_{s,j} \times k_{s,j}$ matrix, ρ_j is a mapping function, and each row $\mathbf{S}_{j,i}$ of $\mathbb{S}_j$ can be mapped onto the signature attribute $att_{s,j,\rho_j(i)} \in \mathcal{U}_{s,j}$. The AA_j selects $\alpha_{j,s,1} \xleftarrow{R} \mathbb{Z}_p$ that requires $\alpha_{j,s,1} < \alpha_j$, computes $\alpha_{j,s,2} = \alpha_j - \alpha_{j,s,1}$ and the component of the signature secret key $S_{j,0} = g^{\alpha_{j,s,2}}$. Then, the AA_j selects $v_{j,s,2}, v_{j,s,3}, \cdots, v_{j,s,k_{s,j}} \xleftarrow{R} \mathbb{Z}_p$, constructs vector $\mathbf{v}_{j,s} = \left(\alpha_{j,s,1}, v_{j,s,2}, v_{j,s,3}, \cdots, v_{j,s,k_{s,j}}\right)$, and computes $\left\{\lambda_{\rho_j}(i) = \mathbf{S}_{j,i}\mathbf{v}_{j,s} : i \in [l_{s,j}]\right\}$. For each row of $\mathbb{S}_j$, the AA_j selects $r_{j,i} \xleftarrow{R} \mathbb{Z}_p$ and computes the components of the outsourced signature key with Eq. (1). Finally, the AA_j sends the signature secret key $SSK_j = (S_{j,0})$ and the outsourced signature key $OSK_j = \left((\mathbb{S}_j, \rho_j), \left\{S_{j,i}, S'_{j,i}, S''_{j,i} : i \in [l_{s,j}]\right\}\right)$ to the DO.

$$\begin{cases} S_{j,i} = g^{\lambda_{\rho_j}(i)} \left(K_0 K_{j,\rho_j(i)}\right)^{r_{j,i}}, \\ S'_{j,i} = g^{r_{j,i}}, \\ S''_{j,i} = \left\{S''_{j,i,x} = K_{j,x}^{r_{j,i}}\right\}_{\forall att_{s,j,x} \in \mathcal{U}_{s,j} \setminus \{\rho_j(i)\}}. \end{cases} \tag{1}$$

2) $\boldsymbol{DecKeyGen}(PP, MSK, PK, (\mathbb{D}, \phi))$: $(\mathbb{D}_j, \phi_j)$ is a decryption LSSS structure, where $\mathbb{D}_j$ is an $l_{e,j} \times k_{e,j}$ matrix, ϕ_j is a mapping function, which can map each row $\mathbf{D}_{j,i}$ of $\mathbb{D}_j$ onto the encryption attribute $\phi_j(i) \in \mathcal{U}_{e,j}$. The AA_j selects $\alpha_{j,e,1} \xleftarrow{R} \mathbb{Z}_p$ that requires $\alpha_{j,e,1} < \alpha_j$ and $\alpha_{j,e,1} \neq \alpha_{j,s,1}$. Next, the AA_j computes $\alpha_{j,e,2} = \alpha_j - \alpha_{j,e,1}$ and the component of the decryption secret key $D_{j,0} = g^{\alpha_{j,e,2}}$. Then, the AA_j selects $v_{j,e,2}, v_{j,e,3}, \cdots, v_{j,e,k_{s,j}} \xleftarrow{R} \mathbb{Z}_p$ to construct vector $\mathbf{v}_{j,e} = \left(\alpha_{j,e,1}, v_{j,e,2}, v_{j,e,3}, \cdots, v_{j,e,k_{s,j}}\right)$, and computes $\left\{\lambda_{\phi_j(i)} = \mathbf{D}_{j,i}\mathbf{v}_{j,e} : i \in [l_{e,j}]\right\}$. The AA_j selects $t_{j,i} \xleftarrow{R} \mathbb{Z}_p$ for each row of $\mathbb{D}_j$ and computes the components of the outsourced decryption key with Eq. (2). Finally, the AA_j returns the decryption secret key $DSK_j = (D_{j,0})$ and the outsourced decryption key $ODK_j = \left((\mathbb{D}_j, \phi_j), \left\{D_{j,i}, D'_{j,i}, D''_{j,i} : i \in [l_{e,j}]\right\}\right)$.

$$\begin{cases} D_{j,i} = g^{\lambda_{\phi_j(i)}} \left(T_0 T_{j,\phi_j(i)}\right)^{t_{j,i}}, \\ D'_{j,i} = g^{t_{j,i}}, \\ D''_{j,i} = \left\{D''_{j,i,x} = T_{j,x}^{t_{j,i}}\right\}_{att_{e,j,x} \in \mathcal{U}_{e,j} \setminus \{\phi_j(i)\}}. \end{cases} \tag{2}$$

Phase 3 (Signcryption). In this phase, the DO signcrypts the message $M \in \mathcal{M}$ with the assistance of the ES and uploads the ciphertext to the CSP. Before signcryption, the DO selects a signature attribute set U_s which satisfies all signature LSSS structures $(\mathbb{S}_j, \rho_j)_{j \in [N]}$, and an encryption attribute set U_e to describe the characteristics of M and constrain the user's access rights. Then, the DO runs the $\boldsymbol{Signcrypt}\left(PP, \{PK_j, SSK_j, OSK_j\}_{j \in [N]}, M, U_s, U_e\right)$ algorithm as follows.

1) For each signature LSSS structure $(\mathbb{S}_j, \rho_j)_{j \in [N]}$, the DO defines a row set $I_{s,j} = (i \in [l_{s,j}] : \rho_j(i) \in U_s)$ and selects a set of constants $\{w_{j,i} \in \mathbb{Z}_p\}_{i \in I_{s,j}}$ that satisfies $\sum_{i \in I_{s,j}} w_{j,i}\mathbf{S}_{j,i} = (1, 0, \cdots, 0)_{k_{s,j}}$. Subsequently, $\{w_{j,i}\}_{i \in I_{s,j}, j \in [N]}$ is sent to the ES for partial signing. The ES selects $\xi \xleftarrow{R} \mathbb{Z}_p$, and then computes the partial signature $\sigma' = (\sigma'_1, \sigma'_2)$ with Eq. (3) and returns it.

$$\begin{cases} \sigma'_1 = g^{\xi} \prod_{i \in I_{s,j}, j \in [N]} \left(S'_{j,i}\right)^{w_{j,i}}, \\ \sigma'_{2,0} = K_0 \prod_{\rho_j(i) \in U_s, j \in [N]} K_{j,i}, \\ \sigma'_{2,j,i} = S_{j,i} \prod_{x \in U_s, x \neq \rho_j(i)} S''_{j,i,x}, \\ \sigma'_2 = \left(\sigma'_{2,0}\right)^{\xi} \prod_{i \in I_{s,j}, j \in [N]} \left(\sigma'_{2,j,i}\right)^{w_{j,i}}. \end{cases} \tag{3}$$

2) To limit the timeliness of the message, the DO generates a timestamp τ to record the current moment, and sets the time limit threshold to $\widehat{\tau}$ to indicate the authenticity period of the message. Then, the DO selects $\beta, \gamma \xleftarrow{R} Z_p$, signcrypts M with Eq. (4) and uploads the ciphertext $CT = \left(U_s, U_e, \tau, \widehat{\tau}, C_1, C_2, C_3, \sigma_1, \sigma_2, \sigma_3\right)$ to the CSP.

$$\begin{cases} C_1 = g^{\beta}, \\ \sigma_1 = g^{\beta\gamma}, \\ C_2 = \left(T_0 \prod_{\phi_j(i) \in U_e, j \in [N]} T_{j,i}\right)^{\beta}, \\ \sigma_2 = \sigma'_1, \\ \mu = H_1(C_1), \\ \chi = H_1(\sigma_2), \\ \Theta = \left(\prod_{j \in [N]} Y_j\right)^{\beta}, \\ C_3 = H_2(\Theta, \sigma_1, \chi) \oplus M, \\ (m_1, m_2, \cdots, m_l) = H_3(\sigma_2 \| U_s \| U_e \| \tau \| \widehat{\tau}), \\ \theta = H_4(\sigma_1 \| \sigma_2 \| C_1 \| C_2 \| C_3 \| U_s \| U_e), \\ \sigma_3 = \left((\delta_1^{\mu}\delta_2)^{\gamma\theta} \left(\mu_0 \prod_{k \in [l]} \mu_k^{m_k}\right)\right)^{\beta} \sigma'_2 \prod_{j \in [N]} S_{j,0}. \end{cases} \tag{4}$$

Phase 4 (De-signcryption). The CSP returns the matching ciphertext CT according to the request of the DU. Before verifying and decrypting the ciphertext, the DU checks the timeliness. Let τ' be the time when the DU receives the ciphertext. The DU continues to de-signcrypt CT if $|\tau' - \tau| \leq \widehat{\tau}$.

1) $\boldsymbol{Verify}\left(PP, \{PK_j\}_{j \in [N]}, CT\right)$: The DU computes $\mathbf{m}$ and θ with Eq. (5), and then decides whether to verify CT with the assistance of the ES.

$$\begin{cases} \mathbf{m} = (m_1, \cdots, m_l) = H_3(\sigma_2 \parallel U_s \parallel U_e \parallel \tau \parallel \widehat{\tau}), \\ \theta = H_4(\sigma_1 \parallel \sigma_2 \parallel C_1 \parallel C_2 \parallel C_3 \parallel U_s \parallel U_e). \end{cases} \tag{5}$$

Case1 (local verification): The DU can directly verify the authenticity of the ciphertext by Eq. (6) if it has computation resources to compute V_1 with Eq. (7) and $V_2 = e(\sigma_3, g)$. The $Verify$ algorithm returns the terminator $\perp$ if Eq. (6) does not hold. Otherwise, 1 is returned and the DU invokes the $Decryption$ algorithm to recover the message.

$$V_1 \stackrel{?}{=} V_2. \tag{6}$$

$$\begin{cases} \mu = H_1(C_1), \\ \zeta_1 = e\left(\left(\delta_1^{\mu}\delta_2\right)^{\theta}, \sigma_1\right), \\ \zeta_2 = e\left(\mu_0 \prod_{k\in[l]} \mu_k^{m_k}, C_1\right), \\ \zeta_3 = e\left(K_0 \prod_{\rho_j(i)\in U_s, j\in[N]} K_{j,i}, \sigma_2\right), \\ V_1 = \zeta_1\zeta_2\zeta_3 \prod_{j\in[N]} Y_j. \end{cases} \tag{7}$$

Case2 (outsourced verification): The DU with insufficient computation resources can verify the ciphertext with the assistance of the ES. Firstly, DU selects $x \xleftarrow{R} Z_p$ as the verification factor and computes $\sigma_3' = \sigma_3^x$. Then, $(\mathbf{m}, \theta, C_1, \sigma_1, \sigma_2, \sigma_3')$ is sent to the ES to assist verification. The ES computes V_1 with Eq. (7) and $V_2' = e(\sigma_3', g)$, and then returns the verification information (V_1, V_2') to the DU. Finally, the DU verifies whether Eq. (8) holds. If Eq. (8) holds, the $Verify$ algorithm returns 1 and the DU will further decrypt the ciphertext. Otherwise, the terminator $\perp$ is returned.

$$V_1^x \stackrel{?}{=} V_2'. \tag{8}$$

2) $\boldsymbol{Decrypt}\left(PP, \{PK_j, DSK_j, ODK_j\}_{j\in[N]}, CT\right)$: The DU first checks whether encryption attribute set U_e satisfies the decryption LSSS structure $(\mathbb{D}_j, \phi_j)_{j\in[N]}$. If it is not satisfied, the $Decrypt$ algorithm is terminated with output $\perp$. Otherwise, the DU generates a row set $\{I_{e,j} = (i \in [l_{e,j}] : \phi_j(i) \in U_e)\}_{j\in[N]}$. Then, the DU selects a constant set $\{\omega_{j,i} \in Z_p\}_{i\in I_{e,j}}$ for each $I_{e,j}$ such that $\sum_{i\in I_{e,j}} \omega_{j,i}\mathbf{D}_{j,i} = (1, 0, \cdots, 0)_{k_{e,j}}$. Subsequently, the DU sends $\left(C_2, \left\{ODK_j, \{\omega_{j,i}\}_{i\in I_{e,j}}\right\}_{j\in[N]}\right)$ to the ES for partial decryption. The ES computes (C_1', C_2') with Eq. (9) and returns it. Finally, the DU recoveries the message M with Eq. (10).

$$\begin{cases} C_1' = \prod_{i\in I_{e,j}, j\in[N]} \left(D_{j,i} \prod_{x\in U_e, x\neq\phi_j(i)} D''_{j,i,x}\right)^{\omega_{j,i}}, \\ C_2' = e\left(C_2, \prod_{i\in I_{e,j}, j\in[N]} \left(D'_{j,i}\right)^{\omega_{j,i}}\right). \end{cases} \tag{9}$$

$$\begin{cases} \chi = H_1(\sigma_2), \\ \Theta = e\left(C_1, C_1' \prod_{j\in[N]} D_{j,0}\right) \Big/ C_2', \\ M = C_3 \oplus H_2(\Theta, \sigma_1, \chi). \end{cases} \tag{10}$$

The verification cost of the SLIM scheme is relatively small (a detailed analysis is given in Subsection VI-A), and IoT devices in general will have enough computation resources to perform local verification. It is worth noting that the public parameters, public key and ciphertext input to the $Verify$ algorithm are all public, and any entity in the system can verify the authenticity of the ciphertext. Therefore, the SLIM scheme supports public verification. In addition, the correctness proof of the SLIM scheme is given in Supplemental Material B.

V. Security Analysis

In this section, we analyse the security of the SLIM scheme according to the security models given in Subsection III-C.

A. Message Confidentiality

Theorem 1: Let the encryption attribute space contain n attributes, and we have the collision resistant hash functions H_1, H_2, H_3 and H_4. If the n-DBDHE problem in $(\mathbb{G}, \mathbb{G}_T)$ is $(\mathcal{T}', \varepsilon')$ hard, the SLIM scheme is $(\mathcal{T}, q_{SG}, q_{DG}, q_{SC}, q_{DS}, \varepsilon)$-IND-sAtt-CCA2 secure, where $\mathcal{T} = \mathcal{T}' - \mathcal{O}\left(|\mathcal{U}_s|^2 (q_{SG} + q_{SC}) + n^2 (q_{DG} + q_{DS})\right)\mathcal{T}_{\exp} - \mathcal{O}(q_{DS})\mathcal{T}_{pair}$, $\varepsilon = \varepsilon' + (q_{DS}/p)$, $\mathcal{T}_{\exp}$ and $\mathcal{T}_{pair}$ are the times of performing exponentiation operation and bilinear mapping, respectively, while $\mathcal{O}(\cdot)$ denotes the time complexity.

Proof: Assume that an adversary $\mathcal{A}$ can break the SLIM scheme. Then we can construct a discriminator $\mathcal{D}$ to solve the n-DBDHE problem. Give a bilinear group $B = (p, \mathbb{G}, \mathbb{G}_T, e)$ and an n-DBDHE instance $(\vec{y}_{a,\beta}, Z)$. $\mathcal{D}$ will attempt to output 1 if $Z = e\left(g_{n+1}, g^{\beta}\right)$; otherwise it will output 0. In the SLIM-IND-sAtt-CCA2 security game, $\mathcal{D}$ plays the challenger $\mathcal{C}$ and interacts with $\mathcal{A}$. The detailed interactions are as follows.

Initialization: $\mathcal{D}$ specifies $\mathcal{M} = \{0,1\}^{l_m}$, $\mathcal{U}_s = \{att_{s,j,i}\}$ and $\mathcal{U}_e = \{att_{e,j,i}\} = \{att_{e,j,1}, \cdots, att_{e,j,n}\}$. $\mathcal{A}$ selects $U_e^* \subset \mathcal{U}_e$ for the challenge and sends it to $\mathcal{D}$.

Setup: $\mathcal{D}$ performs the following steps.

1) Selects $\alpha_j', \alpha_{j,s}, \alpha_{j,e} \xleftarrow{R} \mathbb{Z}_p$ $\forall j \in N$, which requires $\alpha_{j,s} \neq \alpha_{j,e}$, and sets $\alpha_j = \alpha_j' + \alpha_{j,s} + \alpha_{j,e} + a^{n+1}$ implicitly by computing $Y_j = e(g,g)^{\alpha_j'} e(g,g)^{\alpha_{j,s}} e(g,g)^{\alpha_{j,e}} e(g_1, g_n) = e(g,g)^{\alpha_j}$.
2) Selects $\psi_i \xleftarrow{R} \mathbb{Z}_p$ for each $att_{s,j,i} \in \mathcal{U}_s$ and computes $K_{j,i} = g^{\psi_i}$, where $i \in [|\mathcal{U}_s|], j \in [N]$.
3) Selects $\psi_0 \xleftarrow{R} \mathbb{Z}_p$ and sets $K_0 = g^{\psi_0} g_1$.
4) Selects $\eta_i \xleftarrow{R} \mathbb{Z}_p$ for each $att_{e,j,i} \in \mathcal{U}_e$ and computes $T_{j,i} = g^{\eta_i} g_{n+1-i}$, where $i \in [n], j \in [N]$.
5) Selects $\eta_0 \xleftarrow{R} \mathbb{Z}_p$ and sets $T_0 = g^{\eta_0} \prod_{att_{e,j,i}\in U_e^*} T_{j,i}^{-1}$.
6) Sets $C_1^* = g^{\beta} \leftarrow \vec{y}_{a,\beta}$ and $\mu^* = H_1(C_1^*)$.
7) Selects $\upsilon \xleftarrow{R} \mathbb{Z}_p$, and computes $\delta_1 = g_n^{1/\mu^*}$, $\delta_2 = g_n^{-1} g^{\upsilon}$.
8) Selects $u_0, u_1, u_2, \cdots, u_l \xleftarrow{R} \mathbb{Z}_p$ and computes $\mu_i = g^{u_i}$.

Finally, $\mathcal{D}$ sends the public parameter $PP = (B, g, K_0, T_0, \delta_1, \delta_2, \mu_0, \mu_1, \cdots, \mu_l, H_1, H_2, H_3, H_4, \mathcal{M}, \mathcal{U}_s, \mathcal{U}_e)$ and the public key $PK = \{PK_j\}_{j\in[N]} = \left\{Y_j, \{K_{j,i}\}_{att_{s,j,i}\in\mathcal{U}_s}, \{T_{j,i}\}_{att_{e,j,i}\in\mathcal{U}_e}\right\}_{j\in[N]}$ to $\mathcal{A}$.

Query 1: In this phase, $\mathcal{A}$ initiates a series of queries adaptively, and $\mathcal{D}$ responds accordingly.

SignKeyGen Query: $\mathcal{A}$ submits signature LSSS structures $(\mathbb{S}_j, \rho_j)_{j\in[N]}$. $\mathcal{D}$ first computes $S_{j,0} = g^{\alpha_{j,s}}$. Then, $\mathcal{D}$ selects $v_{j,s,2}, \cdots, v_{j,s,k_{s,j}} \overset{R}{\leftarrow} \mathbb{Z}_p$, constructs vector $\mathbf{v}'_{j,s} = (\alpha'_j + \alpha_{j,e}, v_{j,s,2}, \cdots, v_{j,s,k_{s,j}})$, and sets $\mathbf{v}_{j,s} = (a^{n+1}, 0, \cdots, 0) + \mathbf{v}'_{j,s}$ implicitly. As it does not know a^{n+1} and g_{n+1}, $\mathcal{D}$ selects $r'_{j,i} \overset{R}{\leftarrow} \mathbb{Z}_p$ for each row of $\mathbb{S}_j$ and sets $r_{j,i} = r'_{j,i} - \mathbb{S}_{j,i,1} a^n$ implicitly. Subsequently, $\mathcal{D}$ generates the OSK with Eq. (11). Finally, $\mathcal{D}$ returns $\{SK_j = (S_{j,0})\}_{j\in[N]}$ and $\{OSK_j = ((\mathbb{S}_j, \phi_j), \{S_{j,i}, S'_{j,i}, S''_{j,i} : i \in [l_{s,j}]\})\}_{j\in[N]}$.

$$\begin{cases} S_{j,i} = g^{\mathbf{S}_{j,i}\mathbf{v}'_{j,s}} \left(K_0 K_{j,\rho_j(i)}\right)^{r'_{j,i}} g_n^{-\mathbb{S}_{j,i,1}\left(\psi_0 + \psi_{\rho_j(i)}\right)}, \\ S'_{j,i} = g^{r'_{j,i}} g_n^{-\mathbb{S}_{j,i,1}}, \\ S''_{j,i,x} = K_{j,x}^{r'_{j,i}} g_n^{-\mathbb{S}_{j,i,1}\psi_x} : \forall att_{s,j,x} \in \mathcal{U}_{s,j} \backslash \{\rho_j(i)\}, \end{cases} \tag{11}$$

Claim 1: The distributions of SSK and OSK simulated by $\mathcal{D}$ are identical to that generated by the $SignKeyGen$ algorithm.

Due to limited space, the proofs of all the claims are given in Supplemental Material C.

DecKeyGen Query: $\mathcal{A}$ submits decryption LSSS structures $(\mathbb{D}_j, \phi_j)_{j\in[N]}$ that U_e^* cannot satisfy. Hence, there exists vector $\varpi_j = (-1, \varpi_{j,2}, \cdots, \varpi_{j,k_{e,j}}) \in \mathbb{Z}_p^{k_{e,j}}$, such that $\mathbf{D}_{j,i}\varpi_j = 0$, where $\phi_j(i) \in U_e^*$. $\mathcal{D}$ sets $D_{j,0} = g^{\alpha_{j,e}}$, selects $v_{j,e,2}, \cdots, v_{j,e,k_{e,j}} \overset{R}{\leftarrow} \mathbb{Z}_p$, and sets $\mathbf{v}_{j,e} = \mathbf{v}'_{j,e} - (\alpha'_j + \alpha_{j,s} + a^{n+1})\varpi_j$ implicitly, where $\mathbf{v}'_{j,e} = (0, v_{j,e,2}, \cdots, v_{j,e,k_{e,j}})$. If $\phi_j(i) \in U_e^*$, $\lambda_{\phi_j(i)} = \mathbf{D}_{j,i}\mathbf{v}_{j,e} = \mathbf{D}_{j,i}\mathbf{v}'_{j,e}$, since $\mathbf{D}_{j,i}\varpi_j = 0$. Thus, $\mathcal{D}$ selects $t_{j,i} \overset{R}{\leftarrow} \mathbb{Z}_p$ and generates the ODK with Eq. (12). Otherwise, $\phi_j(i) \notin U_e^*$ and $\mathbf{D}_{j,i}\varpi_j \neq 0$. Therefore, $\lambda_{\phi_j(i)} = \mathbf{D}_{j,i}\mathbf{v}_{j,e} = \mathbf{D}_{j,i}\left(v'_{j,e} - (\alpha'_j + \alpha_{j,s})\varpi_j\right) - (\mathbf{D}_{j,i}\varpi_j)\, a^{n+1}$. In this case, $\mathcal{D}$ selects $t'_{j,i} \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets $t_{j,i} = t'_{j,i} + (\mathbf{D}_{j,i}\varpi_j)\, a^{\phi_j(i)}$ implicitly. Then, $\mathcal{D}$ generates the ODK with Eq. (13). Finally, $\mathcal{D}$ returns $\{DK_j = (D_{j,0})\}_{j\in[N]}$ and $\{ODK_j = ((\mathbb{D}_j, \phi_j), \{D_{j,i}, D'_{j,i}, D''_{j,i} : i \in [l_{e,j}]\})\}_{j\in[N]}$.

$$\begin{cases} D_{j,i} = g^{\mathbf{D}_{j,i}\mathbf{v}'_{j,e}} \left(T_0 T_{j,\phi_j(i)}\right)^{t_{j,i}}, \\ D'_{j,i} = g^{t_{j,i}}, \\ D''_{j,i,x} = T_{j,x}^{t_{j,i}} : \forall att_{e,j,x} \in \mathcal{U}_{e,j} \backslash \{\phi_j(i)\}. \end{cases} \tag{12}$$

$$\begin{cases} \varsigma_{1,j,i} = g^{\mathbf{D}_{j,i}\left(\mathbf{v}'_{j,e} - (\alpha'_j + \alpha_{j,s})\varpi_j\right)} \left(T_0 T_{j,\phi_j(i)}\right)^{t'_{j,i}}, \\ \varsigma_{2,j,i} = g_{\phi_j(i)}^{-\eta_0 - \eta_{\phi_j(i)}}, \\ \varsigma_{3,j,i,x} = g_{\phi_j(i)}^{\eta_x} g_{n+1+\phi_j(i)-x}, \\ D_{j,i} = \varsigma_{j,i,1} \left(\varsigma_{2,j,i} \prod_{att_{e,j,x} \in U_e^*} \varsigma_{3,j,i,x}\right)^{-\mathbf{D}_{j,i}\varpi_j}, \\ D'_{j,i} = g^{t'_{j,i}} g_{\phi_j(i)}^{\mathbf{D}_{j,i}\varpi_j}, \\ D''_{j,i,x} = T_{j,x}^{t'_{j,i}} \varsigma_{3,j,i,x}^{\mathbf{D}_{j,i}\varpi_j} : \forall att_{e,j,x} \in \mathcal{U}_{e,j} \backslash \{\phi_j(i)\}. \end{cases} \tag{13}$$

Claim 2: The distributions of DSK and ODK simulated by $\mathcal{D}$ are identical to that generated by the $DecKeyGen$ algorithm.

Signcrypt Query: $\mathcal{A}$ submits $U_s \subset \mathcal{U}_s$, $U_e \subset \mathcal{U}_e$ and $M \in \mathcal{M}$. $\mathcal{D}$ selects $(\mathbb{S}_j, \rho_j)_{j\in[N]}$ that satisfies U_s, invokes *SignKeyGen Query* and then runs $Signcrypt$ algorithm in the SLIM scheme. Finally, $\mathcal{D}$ returns ciphertext CT to $\mathcal{A}$.

De-signcrypt Query: $\mathcal{A}$ submits CT and $(\mathbb{D}_j, \phi_j)_{j\in[N]}$. $\mathcal{D}$ first judge $C_1 \overset{?}{=} C_1^*$. If it is equal, the simulation terminates. The probability of this happening does not exceed $1/p$, because $C_1 = g^\beta$ is random in $\mathcal{A}$'s view. Otherwise, $\mathcal{D}$ runs the $Verify$ algorithm to verify CT and returns $\perp$ when CT is invalid. Then, $\mathcal{D}$ performs the following steps.

a) If U_e^* satisfies $(\mathbb{D}_j, \phi_j)_{j\in[N]}$, $\mathcal{D}$ computes $\chi = H_1(\sigma_2)$, $\Theta = e(C_1, g^{\alpha_{sum}})\, e\left(\left(\delta_1^\mu \delta_2\right)^\beta / C_1^v, g_1\right)^{N/(\mu/\mu^* - 1)}$, $M = C_3 \oplus H_1(\Theta, \sigma_1, \chi)$, and returns M, where $\alpha_{sum} = \sum_{j\in[N]} \alpha'_j + \alpha_{j,s} + \alpha_{j,e}$.
b) Otherwise, $\mathcal{D}$ invokes the *DecKeyGen Query* to generate $\{DSK_j\}_{j\in[N]}$ and $\{ODK_j\}_{j\in[N]}$, runs the $Decrypt$ algorithm to decrypt CT and returns the output.

Claim 3: If $C_1 \neq C_1^*$, the de-signcryption phase simulated by $\mathcal{D}$ is the same as that in the SLIM scheme.

Challenge: $\mathcal{A}$ selects two equal length messages $M_0^*, M_1^* \in \mathcal{M}$ and $U_s^* \subset \mathcal{U}_s$ for $\mathcal{D}$. $\mathcal{D}$ tosses a random coin $b \in \{0,1\}$, selects $\xi, \gamma \overset{R}{\leftarrow} \mathbb{Z}_p$, and generates the challenge ciphertext $CT^* = (U_s^*, U_e^*, \tau^*, \widehat{\tau}^*, C_1^*, C_2^*, C_3^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$ to $\mathcal{A}$ with Eq. (14). Note that C_1^* and μ are defined in **Setup**. Moreover, g_n, g^β and Z are taken from the n-DBDHE instance $(\vec{y}_{a,\beta}, Z)$.

$$\begin{cases} \sigma_1^* = \left(g^\beta\right)^\gamma, C_2^* = \left(g^\beta\right)^{\eta_0} \\ \sigma_2^* = \sigma_1'^* = g^\xi g_n^{-N}, \chi^* = H_1(\sigma_2^*), \\ \sigma_{2,1}'^* = K_0 \prod_{att_{s,j,i} \in U_s^*, j\in[N]} K_{j,i}, \\ \sigma_{2,2}'^* = g_n^{-\psi_0} \prod_{att_{s,j,i} \in U_s^*, j\in[N]} g_n^{-\psi_i}, \\ \sigma_2'^* = \left(\sigma_{2,1}'^*\right)^\xi \left(\sigma_{2,2}'^*\right)^N g^{\sum_{j\in[N]} \alpha'_j + \alpha_{j,e}}, \\ C_3^* = H_2\left(Z^N e\left(g^\beta, g^{\alpha_{sum}}\right), \sigma_1^*, \chi^*\right) \oplus M_b^*, \\ (m_1^*, m_2^*, \cdots, m_l^*) = H_3(\sigma_2^* \parallel U_s^* \parallel U_e^* \parallel \tau^* \parallel \widehat{\tau}^*), \\ \theta^* = H_4(\sigma_1^* \parallel \sigma_2^* \parallel C_1^* \parallel C_2^* \parallel C_3^* \parallel U_s^* \parallel U_e^*), \\ \sigma_3^* = \left(g^\beta\right)^{\gamma\theta^* v + u_0 + \sum_{k\in[l]} m_k^* u_k} \sigma_2'^* g^{\sum_{j\in[N]} \alpha_{j,s}}. \end{cases} \tag{14}$$

Claim 4: If $Z = e\left(g_{n+1}, g^\beta\right)$, the distribution of CT^* simulated by $\mathcal{D}$ is identical to the ciphertext generated by the $Signcrypt$ algorithm under the same input.

Query 2: Same as **Query 1**, except that CT^* cannot be used for the *De-signcrypt Query*.

Guess: $\mathcal{A}$ gives the guess $b' \in \{0,1\}$ for b. If $b' = b$, $\mathcal{D}$ outputs 1 for the n-DBDHE problem, otherwise 0.

Probability Analysis: $\mathcal{D}$ terminates the simulation when $C_1 = C_1^*$ in the *De-signcrypt Query*. The probability of this happening does not exceed q_{DS}/p. If the simulation is not terminated and $Z = e\left(g_{n+1}, g^\beta\right)$, according to Claim 4, we know that the simulation of $\mathcal{D}$ is perfect. Assuming that the advantage of $\mathcal{A}$ winning the game is ε, the probability that $\mathcal{D}$ outputs 1 in the n-DBDHE game is

$$\Pr\left[\mathcal{D}\left(\vec{y}_{a,\beta}, Z = e\left(g_{n+1}, g^\beta\right)\right) = 1\right] > \frac{1}{2} + \varepsilon - \frac{q_{DS}}{p}.$$

If Z is a random element in $\mathbb{G}_T$, $\mathcal{A}$ does not have any advantage. Hence, the probability of $\mathcal{D}$ outputting 1 is

$$\Pr\left[\mathcal{D}\left(\vec{y}_{a,\beta}, Z \overset{R}{\leftarrow} \mathbb{G}\right) = 1\right] = \frac{1}{2}.$$

Therefore, the advantage of $\mathcal{D}$ in solving the n-DBDHE problem in $(\mathbb{G}, \mathbb{G}_T)$ is $Adv_{\mathcal{D}}^{n-DBDHE} > \varepsilon - q_{DS}/p$. ■

B. Ciphertext Unforgeability

Theorem 2: Let the signature attribute space contain n attributes, and the hash functions H_1, H_2, H_3 and H_4 be collision resistant. If the n-CDHE problem is $(\mathcal{T}', \varepsilon')$ hard in $\mathbb{G}$. Then, the SLIM scheme is $(\mathcal{T}, q_{SG}, q_{DG}, q_{SC}, q_{DS}, \varepsilon)$-EUF-sAtt-CMA secure, where $\mathcal{T} = \mathcal{T}' - \mathcal{O}\left(n^2\left(q_{SG}+q_{SC}\right) + \left|\mathcal{U}_e\right|^2\left(q_{DG}+q_{DS}\right)\right)\mathcal{T}_{\text{exp}} - \mathcal{O}\left(q_{SC}+q_{DS}\right)\mathcal{T}_{pair}$, $\varepsilon = \varepsilon'\kappa\left(l+1\right)$, κ is the security parameter and l is the size of the output of H_3.

Proof: Assume that there is an adversary $\mathcal{A}$ that can break the SLIM scheme. We can construct a challenger $\mathcal{C}$ to solve the n-CDHE problem in $\mathbb{G}$. Give a bilinear map group $B = (p, \mathbb{G}, \mathbb{G}_T, e)$ and an n-CDHE instance $\vec{y}_a = (g_1, \cdots, g_n, g_{n+2}, \cdots, g_{2n})$. The goal of $\mathcal{C}$ is to compute g_{n+1}.

Initialization: $\mathcal{C}$ specifies $\mathcal{M} = \{0,1\}^{l_m}$, $\mathcal{U}_s = \{att_{s,j,1}, att_{s,j,2}, \cdots, att_{s,j,n}\}$ and $\mathcal{U}_e = \{att_{e,j,i}\}$. $\mathcal{A}$ selects $U_s^* \subset \mathcal{U}_s$ for the challenge and sends it to $\mathcal{C}$.

Setup: $\mathcal{C}$ performs the following steps.

1) Carries out step 1) of the **Setup** in Subsection V-A.
2) Selects $\psi_i \overset{R}{\leftarrow} \mathbb{Z}_p$ for each $att_{s,j,i} \in \mathcal{U}_s$ and computes $K_{j,i} = g^{\psi_i} g_{n+1-i}$, where $i \in [n], j \in [N]$.
3) Sets $K_0 = g^{\psi_0} \prod_{att_{s,j,i} \in U_s^*, j \in [N]} K_{j,i}^{-1}$, where $\psi_0 \overset{R}{\leftarrow} \mathbb{Z}_p$.
4) Selects $\eta_i \overset{R}{\leftarrow} \mathbb{Z}_p$ for each $att_{e,j,i} \in \mathcal{U}_e$ and computes $T_{j,i} = g^{\eta_i}$, where $i \in [|\mathcal{U}_e|], j \in [N]$.
5) Selects $\eta_0 \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets $T_0 = g^{\eta_0} g_1$.
6) Selects $\upsilon, \upsilon' \overset{R}{\leftarrow} \mathbb{Z}_p$, and computes $\delta_1 = g^{\upsilon}$ and $\delta_2 = g^{\upsilon'}$.
7) Selects $d_0, \cdots, d_l \overset{R}{\leftarrow} \mathbb{Z}_\kappa^*$, $u_0, \cdots, u_l \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and $\varphi \overset{R}{\leftarrow} \mathbb{Z}_l^*$. Computes $\mu_0 = g_n^{p-\kappa\varphi+d_0} g^{u_0}$ and $\mu_i = g_n^{d_i} g^{u_i}$ $\forall i \in [l]$. Note that $\kappa\left(l+1\right) < p$ is required.
8) Defines functions $F(\mathbf{m}) = p - \kappa\varphi + d_0 + \sum_{i\in[l]} m_i d_i$, $J(\mathbf{m}) = u_0 + \sum_{i\in[l]} m_i u_i$ and $K(\mathbf{m}) = d_0 + \sum_{i\in[l]} m_i d_i \bmod \kappa$ for $\mathbf{m} = (m_1, \cdots m_l) \in \{0,1\}^l$.

Finally, $\mathcal{C}$ generates the public parameter $PP = (B, g, K_0, T_0, \delta_1, \delta_2, \mu_0, \mu_1, \cdots, \mu_l, H_1, H_2, H_3, H_4, \mathcal{M}, \mathcal{U}_s, \mathcal{U}_e)$ and the public key $PK = \{PK_j\}_{j\in[N]} = \left\{Y_j, \{K_{j,i}\}_{att_{s,j,i}\in\mathcal{U}_s}, \{T_{j,i}\}_{att_{e,j,i}\in\mathcal{U}_e}\right\}_{j\in[N]}$ to $\mathcal{A}$.

Query 1: $\mathcal{A}$ can initiate a series of queries adaptively. The construction of the *SignKeyGen Query* and *DecKeyGen Query* in this phase is similar to the *DecKeyGen Query* and *SignKeyGen Query* of **Query 1** in Subsection V-A. It only needs to exchange the encryption attribute and signature attribute, as well as the parameters related to the attribute. Moreover, the *Signcrypt Query* and *De-signcrypt Query* are as follows.

Signcrypt Query: $\mathcal{A}$ submits $U_s \subset \mathcal{U}_s$, $U_e \subset \mathcal{U}_e$ and $M \in \mathcal{M}$. $\mathcal{C}$ selects $(\mathbb{S}_j, \rho_j)_{j\in[N]}$ that U_s satisfies. If U_s^* does not satisfy $(\mathbb{S}_j, \rho_j)_{j\in[N]}$, $\mathcal{C}$ invokes *SignKeyGen Query* to generate $\{SSK_j\}_{j\in[N]}$ and $\{OSK_j\}_{j\in[N]}$, and then runs the $Signcrypt$ algorithm to generate the ciphertext to $\mathcal{A}$. Otherwise, $\mathcal{C}$ selects $\xi' \overset{R}{\leftarrow} \mathbb{Z}_p$, and then computes $\sigma_2 = g^{\xi'}$ and $\mathbf{m} = H_3(\sigma_2 \parallel U_s \parallel U_e \parallel \tau \parallel \widehat{\tau})$. If $K(\mathbf{m}) \neq 0$, $\mathcal{C}$ selects $\beta', \gamma \overset{R}{\leftarrow} \mathbb{Z}_p$, generates the ciphertext with Eq. (15) and returns it to $\mathcal{A}$. Otherwise, $\mathcal{C}$ repeat the above step with a new ξ'.

$$\begin{cases} \alpha_{sum} = \sum_{j\in[N]} \alpha_j' + \alpha_{j,s} + \alpha_{j,e}, \chi = H_2(\sigma_2), \\ \Theta = \left(\prod_{j\in[N]} Y_j\right)^{\beta'} \left(e\left(g^{\alpha_{sum}}, g_1\right) e\left(g_2, g_n\right)\right)^{-N/F(\mathbf{m})}, \\ C_1 = g^{\beta'} g_1^{-N/F(\mathbf{m})}, \sigma_1 = C_1^{\gamma}, \mu = H_2(C_1), \\ C_{2,1} = T_0 \prod_{att_{e,j,i}\in U_e} T_{j,i}, C_{2,2} = g_1^{\eta_0} g_2 \prod_{att_{e,j,i}\in U_e} g_1^{\eta_i}, \\ C_2 = C_{2,1}^{\beta'} C_{2,2}^{-N/F(\mathbf{m})}, C_3 = H_1(\Theta, \sigma_1, \chi) \oplus M, \\ \theta = H_4(\sigma_1 \parallel \sigma_2 \parallel C_1 \parallel C_2 \parallel C_3 \parallel U_s \parallel U_e), \\ \sigma_{3,1} = \left(\delta_1^{\mu}\delta_2\right)^{\gamma\theta} g_n^{F(\mathbf{m})} g^{J(\mathbf{m})}, \sigma_{3,2} = K_0 \prod_{att_{s,j,i}\in\mathcal{U}_s} K_{j,i}, \\ \sigma_{3,3} = -N\left(J(\mathbf{m}) + \gamma\theta\left(\mu\upsilon + \upsilon'\right)\right), \sigma_{3,4} = F(\mathbf{m}) g^{\alpha_{sum}}, \\ \sigma_3 = \sigma_{3,1}^{\beta} \sigma_{3,2}^{\xi'} g_1^{\sigma_{3,3}} / \sigma_{3,4}. \end{cases} \tag{15}$$

Claim 5: The distribution of CT simulated by $\mathcal{C}$ is the same as the ciphertext generated by the $Signcrypt$ algorithm.

De-signcrypt Query: $\mathcal{A}$ submits CT and $(\mathbb{D}_j, \phi_j)_{j\in[N]}$. $\mathcal{C}$ first runs the $Verify$ algorithm to verify CT and returns $\perp$ if CT is invalid. Otherwise, $\mathcal{C}$ invokes the *DecKeyGen Query* to generate $\{DSK_j\}_{j\in[N]}$ and $\{ODK_j\}_{j\in[N]}$, runs the $Decrypt$ algorithm to decrypt CT and returns the output to $\mathcal{A}$.

Forgery: $\mathcal{A}$ gives a valid forged ciphertext $CT^* = (U_s^*, U_e^*, \tau^*, \widehat{\tau}^*, C_1^*, C_2^*, C_3^*, \sigma_1^*, \sigma_2^*, \sigma_3^*)$ based on M^*, and CT^* satisfies: a) $Verify\left(PP, \{PK_j\}_{j\in[N]}, CT^*\right) \rightarrow 1$, b) $Decrypt\left(PP, \{PK_j, DK_j, ODK_j\}_{j\in[N]}, CT^*\right) \rightarrow M^*$, c) (U_s^*, U_e^*, M^*) are not submitted to the *Signcrypt Query*. $\mathcal{C}$ computes $\mathbf{m}^* = H_3(\sigma_2^* \parallel U_s^* \parallel U_e^* \parallel \tau^* \parallel \widehat{\tau}^*)$. If $F(\mathbf{m}^*) \neq 0$, the game is terminated. We call this event and its opposite event Abort and $\overline{\text{Abort}}$, respectively. Otherwise, $\mathcal{C}$ computes $\mu^* = H_2(C_1^*)$, $\theta^* = H_4(\sigma_1^* \parallel \sigma_2^* \parallel C_1^* \parallel C_2^* \parallel C_3^* \parallel U_s^* \parallel U_e^*)$ and

$$\begin{aligned} \Upsilon =& \frac{\sigma_4^*}{\left(\sigma_1^*\right)^{\theta^*\left(\mu^*\upsilon+\upsilon'\right)} \left(C_1^*\right)^{J(\mathbf{m}^*)} \left(\sigma_2^*\right)^{\psi_0} g^{\sum_{j\in[N]} \alpha_j' + \alpha_{j,s} + \alpha_{j,e}}} \\ =& g_{n+1}^N. \end{aligned} \tag{16}$$

The proof of Eq. (16) is given in Supplemental Material D. Finally, $\mathcal{C}$ obtains g_{n+1} by computing $\Upsilon^{1/N}$.

Probability Analysis: When CT^* given by $\mathcal{A}$ satisfies $K(\mathbf{m}) = \varphi$, $\mathcal{C}$ simulates successfully and solves the n-CDHE problem. This event is called $\mathcal{C}$ success. Then we have

$$\Pr\left[\mathcal{C} \text{ success}\right] = \Pr\left[\overline{\text{Abort}}\right] = \frac{1}{\kappa\left(l+1\right)}.$$

Assume that the advantage of $\mathcal{A}$ successfully forging ciphertext is ε. Then $\mathcal{C}$ can solve the n-CDHE problem in $\mathbb{G}$ with the advantage $Adv_{\mathcal{C}}^{n-CDHE} = \varepsilon/\kappa\left(l+1\right)$. ■

C. Signcryptor Privacy

Theorem 3: The SLIM scheme has signcryptor privacy.

Proof: In the SLIM scheme, the ciphertext CT includes U_s, U_e, τ, $\widehat{\tau}$, C_1, C_2, C_3, σ_1, σ_2 and σ_3. Among them, U_s, U_e, τ , $\widehat{\tau}$, C_1, C_2, C_3 and σ_1 are irrelevant, since they are generated without signcryptor's signature key. Thus, we only need to focus on σ_2 and σ_3. We can derive $\sigma_2 = g^{\xi'}$ and $\sigma_3 = (\delta_1^{\mu}\delta_2)^{\beta\gamma\theta}$ $\left(\mu_0 \prod_{k\in[l]} \mu_k^{m_k}\right)^{\beta}\left(K_0 \prod_{\rho_j(i)\in U_s, j\in[N]} K_{j,i}\right)^{\xi'} g^{\sum_{j\in[N]}\alpha_j}$ where $\xi' = \xi + \sum_{i\in I_{s,j}, j\in[N]} w_{j,i} r_{j,i}$, and β, γ, ξ are selected at random. Therefore, the distribution of ciphertext is independent of the signature key. Furthermore, the SLIM scheme has signcryptor privacy. ∎

VI. Performance Evaluation

We evaluate the performance of the SLIM scheme in comparison with various existing schemes. The symbols used in this section are defined in Table II. Note that N, l, l_τ and $l_{\widehat{\tau}}$ are constants.

TABLE II
Definitions of Symbols Used in Performance Evaluation

Symbols	Descriptions
l_s, l_e	Numbers of signature/encryption attributes involved
u_s, u_e	Sizes of signature/encryption attribute spaces
E, E_T	Exponentiation operations in groups $\mathbb{G}/\mathbb{G}_T$
P	Bilinear mapping operation
$\lvert\mathbb{G}\rvert, \lvert\mathbb{G}_T\rvert, \lvert\mathbb{Z}_p\rvert$	Bit lengths of elements in groups $\mathbb{G}/\mathbb{G}_T/\mathbb{Z}_p$
$l_\tau, l_{\widehat{\tau}}$	Bit lengths of timestamp and time threshold
msg	Bit length of message

A. Theoretical Performance

1) Computation Cost: In attribute-based cryptography, the bilinear mapping operation has the highest computation cost, followed by the exponentiation operation. The cost of other operations, such as hash operation, constant operation, etc., is negligible. Tables III compares the computation cost of the proposed SLIM scheme with various existing schemes, in terms of the numbers of bilinear mapping operations and exponentiation operations required. Some entities only verify the authenticity of the ciphertext without decrypting it. Therefore, we divide the de-signcryption into decryption and verification for a more detailed comparison.

The schemes [12], [21], [22] cannot provide public verification since the verification algorithm requires the decrypted message, and the de-signcryption of these schemes involves a large number of bilinear mapping operations. For these schemes, each invalid ciphertext wastes lots of computations. This makes them unsuitable for low-reliability environments, where there may be many malicious tampering of ciphertexts by attackers and the percentage of valid ciphertexts in the system is low. Among them, the computation cost of the scheme [22] is linearly related to the number of elements of the MSP matrix ($l_s c$), which is unaffordable for IoT devices. In addition, in the scheme [12], the outsourced decryption cost is as high as $l_e E + 3l_e P$. This means that invalid ciphertexts also waste a large amount of computation resources of the server.

The schemes [17]–[20], [23] do not support outsourced computation, and the user will have to bear the full computation cost. The schemes [17], [19], [20] have similar computation cost, involving a small number of bilinear mapping operations, and they are lightweight. The computation cost of the scheme [18] is only marginally higher than those of the schemes [17], [19], [20], and it is also a lightweight scheme in the existing ABSC schemes without outsourced computation. In contrast, the cost of signcryption and verification for the scheme [23] is expensive, $(4l_s + 2l_e + 6)\,E + E_T$ and $(l_s + 2)\,E + (l_s + 3)\,P$, respectively.

The schemes [13], [15] only support outsourced de-signcryption. Compared with the scheme [15], the scheme [13] has higher decryption cost because the DU needs to generate outsourced decryption key and verify the correctness. If the DU chooses to de-signcrypt the ciphertext locally, the overall computation cost can be reduced to $(l_s + 2l_e + 2)\,E + (l_s + 5)\,P$, but such costs are still very expensive to IoT devices. The scheme [15] adds outsourced decryption based on the scheme [18], which reduces the decryption cost. However, the access policy update function makes signcryption involving the whole attribute space, and the cost is as high as $(u + r - l_e + 4)\,E + 2E_T$. Also most of the computation costs are concentrated on the DO, which is inconsistent with the low computing capability of nodes at the perception layer in the IoT.

The schemes [14], [16] and our SLIM scheme support both outsourced de-signcryption and outsourced signcryption. The decryption of the schemes [14], [16] requires only 1 exponentiation operation in $\mathbb{G}_T$. It is worth noting that in the scheme [14], the DU directly outsources the verification of the ciphertext to the server, and its de-signcryption phase only contains the decryption algorithm. However, the outsourced verification result cannot be judged unless the DU executes the verification algorithm once by itself. Additionally, the verification algorithm of the scheme [14] is very high because of the large number of bilinear mapping operations involved. As for the scheme [16], although it supports outsourced signcryption, the computation cost for the DO still reaches $(2l_e + 10)\,E$, which does not offer an obvious advantage in the existing schemes.

In the SLIM scheme, users offload most of the computation to the ES. In the signcryption phase, the DO requires only $7E + E_T$ to generate a complete ciphertext. This signcryption cost is extremely rare and lightweight, since the signcryption cost in the existing schemes is linearly related to the number of attributes. The cost of de-signcryption for the SLIM scheme is also lower than other schemes except for the scheme [16]. With the assistance of the ES, the DU only needs $E + E_T$ to verify authenticity and P to decrypt ciphertext. Moreover, the cost of local verification is $2E + 4P$, which is acceptable for most entities such as CSPs and common IoT devices. The outsourced computation cost of the SLIM scheme is slightly lower than that of the scheme [16] and significantly lower than the schemes [13], [14], [21]. Overall, our SLIM scheme is more lightweight than the existing ABSC schemes.

2) Communication and Storage Costs: In terms of communication and storage costs, we mainly compare the sizes of

TABLE III
COMPARISON OF THEORETICAL COMPUTATION COST

Scheme	User Signcryption	User De-signcryption		Outsourced Signcryption	Outsourced De-signcryption
		Decryption	Verification		
[21]	$(2l_s + 2l_e + 3)\,E + 2E_T + P$	$(l_s \log l_s + l_e \log l_e + 1)\,E_T + (2l_s + 2l_e + 1)\,P$		-	-
[22]	$(2l_s c + 3l_s + 2l_e + 4)\,E + E_T$	$(2l_s c + c)\,E + l_e E_T + (l_s c + c + 2l_e + 4)\,P$		-	-
[12]	$(2l_s + 2l_e + 1)\,E$	$(l_s + 2)\,E + l_s P$		-	$l_e E + 3l_e P$
[17]	$(2l_s + 9)\,E + E_T$	$2l_e E + 2P$	$2E + 4P$	-	-
[19]	$(2l_s + 7)\,E + E_T$	$2l_e E + 2P$	$E_T + 6P$	-	-
[23]	$(4l_s + 2l_e + 6)\,E + E_T$	$2l_e E + 2P$	$(l_s + 2)\,E + (l_s + 3)\,P$	-	-
[18]	$(l_s + 6)\,E + 2E_T$	$(l_e + 2)\,E + 3E_T + 6P$	$3E + E_T + 3P$	-	-
[20]	$(2l_s + 10)\,E + E_T$	$2l_e E + 2P$	$2E + 4P$	-	-
[13]	$(2l_s + l_e + 4)\,E$	$(l_s + 6)\,E + P$	$8E + 2E_T$	-	$(2l_s + l_e + 2)\,E + (2l_e + 2)\,P$
[15]	$(u + r - l_e + 4)\,E + 2E_T$	$E + E_T$	$3E + E_T + 3P$	-	$E + E_T + 3P$
[14]	$(l_s + 5)\,E + E_T$	E_T	$(l_s + 2)\,E + (l_s + 3)\,P$	$(u_s + 5l_s + 3l_e + 2)\,E$	$3l_e E + l_e E_T + (2l_e + N)\,P$
[16]	$(2l_e + 10)\,E$	E_T	$6E + E_T$	$(2l_s + 2)\,E$	$(2l_e + 2)\,E + 7P$
SLIM	$7E + E_T$	P	$E + E_T$	$(2l_s + 2)\,E$	$(2l_e + 2)\,E + 6P$

Notes: c represents the number of columns of the MSP matrix in [22]; u and r represent the size of the attribute universe and the maximum number of attributes that can be revoked in [15], respectively.

key and ciphertext, where the key includes signature secret key (SSK), decryption secret key (DSK), outsourced signature key (OSK) and outsourced decryption key (ODK). From Table IV, it can be seen that the key sizes of almost all the schemes are related to the number of attributes involved and there are a few schemes, including the SLIM, which can achieve constant ciphertext size.

The keys of the schemes [15]–[20] and the SLIM involve the attribute space, and their key sizes are the largest among the existing schemes. The key sizes of the schemes [12]–[14], [21]–[23] are linear with the number of attributes. For the scheme [13], the AA selects a unique random number as the decryption secret key for each user to decrypt the ciphertext that has been outsourced for decrypting, and the size of the decryption secret key is only $|\mathbb{Z}_p|$. In the SLIM scheme, each AA generates an SSK and a DSK for the user, both are of size $|\mathbb{G}|$. Therefore, the size of the secret key is $N\,|\mathbb{G}|$ for each user. The key only needs to be transmitted once by the user to be stored. Also for outsourced ABSC schemes, the outsourced key does not need to be transmitted repeatedly if the user has established a relationship with the ES.

However, to achieve data sharing, the ciphertext is stored for a long time and transmitted multiple times. Therefore, the size of the ciphertext is the main factor that affects the communication and storage costs. Except for the schemes [17]–[20] and the SLIM scheme, the ciphertext sizes of the other schemes increase with the number of attributes. In the scheme [15], although the ciphertext uploaded bythe the DO to the CSP involves the entire attribute space, the DU only needs to download a part of it for de-signcryption. The schemes [17], [20] involve $6\,|\mathbb{G}|$ and schemes [18], [19] involve $5\,|\mathbb{G}|+|\mathbb{G}_T|$, while the SLIM only involves $5\,|\mathbb{G}|$ which is slightly lower than the other schemes. Since the SLIM scheme allows the DO to set the validity period autonomously, the time threshold is also included in the ciphertext. If the CA sets a globally uniform time threshold, the ciphertext size will be reduced by $l_{\hat{\tau}}$ at the expense of some flexibility. In summary, the proposed SLIM scheme has lower communication and storage costs compared with the prior schemes.

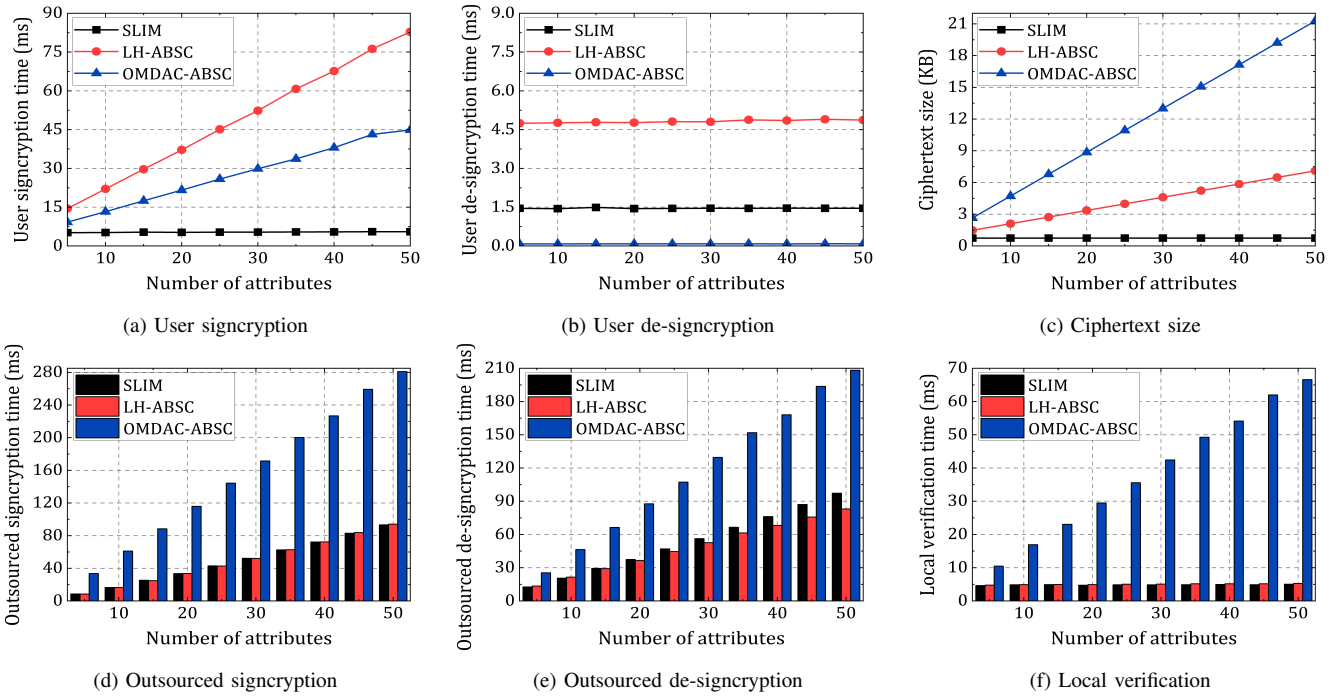
B. Actual Performance

Fig. 4 compares the actual performance of the SLIM scheme with those of the LH-ABSC scheme [16] and OMDAC-ABSC scheme [14]. The simulation experiments are performed on

TABLE IV
COMPARISON OF THEORETICAL COMMUNICATION AND STORAGE COSTS

Scheme	Signcryption Key		Decryption Key		Ciphertext
	SSK	OSK	DSK	ODK	
[21]	$(2l_s + 1)\,\|\mathbb{G}\|$		$(2l_e + 1)\,\|\mathbb{G}\|$		$(2l_s + 2l_e + 2)\,\|\mathbb{G}\| + \|\mathbb{Z}_p\| + msg$
[22]	$(l_s + 2)\,\|\mathbb{G}\|$		$(l_e + 2)\,\|\mathbb{G}\|$		$(l_s + l_e + c + 3)\,\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[23]	$(l_s + 2)\,\|\mathbb{G}\|$		$(l_e + 2)\,\|\mathbb{G}\|$		$(l_s + l_e + 4)\,\|\mathbb{G}\| + l_\tau + msg$
[18]	$(u_s + l_s)\,\|\mathbb{G}\|$		$(u_e + l_e)\,\|\mathbb{G}\|$		$5\,\|\mathbb{G}\| + \|\mathbb{G}_T\| + 2\,\|\mathbb{Z}_p\|$
[17]	$(l_s u_s + l_s)\,\|\mathbb{G}\|$		$(l_e u_e + l_e)\,\|\mathbb{G}\|$		$6\,\|\mathbb{G}\| + l_\pi + msg$
[19]	$(l_s u_s + l_s)\,\|\mathbb{G}\|$		$(3u_e + 3l_e - 3)\,\|\mathbb{G}\|$		$5\,\|\mathbb{G}\| + \|\mathbb{G}_T\|$
[20]	$(l_s u_s + l_s)\,\|\mathbb{G}\|$		$(l_e u_e + l_e)\,\|\mathbb{G}\|$		$6\,\|\mathbb{G}\| + l_\tau + l_\pi + msg$
[12]	$l_s\,\|\mathbb{G}\|$		$l_e\,\|\mathbb{Z}_p\|$	$l_e\,\|\mathbb{G}\|$	$(l_s + 2l_e + 2)\,\|\mathbb{G}\| + msg$
[13]	$(l_s + 2)\,\|\mathbb{G}\|$		$\|\mathbb{Z}_p\|$	$(l_e + 2)\,\|\mathbb{G}\|$	$(2l_s + 2l_e + 3)\,\|\mathbb{G}\| + l_\tau + msg$
[15]	$(u + l_s + l_e)\,\|\mathbb{G}\|$		$(u + l_s + l_e)\,\|\mathbb{G}\|$	$(u + l_s + l_e + 1)\,\|\mathbb{G}\|$	$(u + 6)\,\|\mathbb{G}\| + \|\mathbb{G}_T\| + 2\,\|\mathbb{Z}_p\|$
[14]	$(l_s + N)\,\|\mathbb{G}\|$	$(2l_s + 2l_m + N + 1)\,\|\mathbb{G}\|$	$(l_e + N)\,\|\mathbb{G}\|$	$(l_e + N)\,\|\mathbb{G}\|$	$(l_s + 2l_e + 3)\,\|\mathbb{G}\| + \|\mathbb{G}_T\| + 2l_e\,\|\mathbb{Z}_p\| + l_\tau$
[16]	$2\,\|\mathbb{G}\|$	$(l_s u_s + l_s)\,\|\mathbb{G}\|$	$(l_e + 2)\,\|\mathbb{G}\|$	$(l_e + 2)\,\|\mathbb{G}\|$	$(l_e + 6)\,\|\mathbb{G}\| + l_\tau + msg$
SLIM	$N\,\|\mathbb{G}\|$	$(l_s u_s + l_s)\,\|\mathbb{G}\|$	$N\,\|\mathbb{G}\|$	$(l_e u_e + l_e)\,\|\mathbb{G}\|$	$5\,\|\mathbb{G}\| + l_\tau + l_{\hat{\tau}} + msg$

Notes: l_π represents the bit length of the random value $\pi \in \{0,1\}^{l_\pi}$ selected during the signcryption phase of the schemes [17] and [20]. The scheme [17] states that $l_\pi \approx 40$. l_m represents the maximum value of the signature access structure in the scheme [14].



(a) User signcryption (b) User de-signcryption (c) Ciphertext size

(d) Outsourced signcryption (e) Outsourced de-signcryption (f) Local verification

Fig. 4. Comparison of the actual performance for three different schemes.

an Ubuntu 16.04 LTS with an 11th Intel Core i5 processor 2.6 GHz by utilizing Python 3.8.5 and the Charm-Crypto framework 0.50 [28]. The elliptic curve selected is Type A as $E(F_q) : y^2 = x^3 + x$. The groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p are the subgroups of $E(F_q)$, and the lengths of p and q are 160 bits and 512 bits, respectively. Hence, $|\mathbb{Z}_p| = 160$ bits and $|\mathbb{G}| = |\mathbb{G}_T| = 1024$ bits. In the SLIM scheme, the hash functions H_1 and H_4 are provided by the Charm-Crypto framework, while H_2 and H_3 are constructed based on SHA-512 and SHA-1, respectively. Thus, l and l_m are set to 160 bits and 512 bits, respectively. The symmetric encryption algorithm used in the LH-ABSC scheme is the AES algorithm in ECB mode. In addition, we set the numbers of attributes to $l_s = l_e \in [5,\ 50]$ and the number of authorities to $N = 5$, except for the LH-ABSC as it only supports single-authority. In the experiments, we turn on the pre-computation tables for group exponentiation, which significantly accelerates the speed of E from 0.77 ms to 0.12 ms. The results of Fig. 4 are obtained by averaging over 1000 independent experiments.

It can be seen from Fig. 4a that the user signcryption time of the SLIM scheme is almost constant, approximately 5.7 ms, while those of the other two schemes increase linearly with the number of attributes. In fact, the user signcryption time of the SLIM scheme is also slightly affected by the number of attributes, which increases from 5.6 ms to 5.8 ms, because the multiplication operations required by the DO in signcryption phase is linearly related to the number of attributes. But compared to the exponentiation operation, the multiplication operation is very fast, only approximately 5.3 us, which is negligible by comparison. As shown in Fig. 4b, the user de-signcryption times of all three schemes are roughly constant, approximately 5.0 ms for the LH-ABSC, 1.6 ms for the SLIM, and 84.1 us for the OMDAC-ABSC. It should be noted that for the OMDAC-ABSC scheme, the outsourced entity is fully trusted, and the DU delegates the verification of the ciphertext to it but the DU is unable to verify the returned result. This lack of controllability by the DU is too idealistic in reality. In contrast, our SLIM scheme is practical, as it not only has a lower user de-signcryption cost but also supports verifiable outsourced verification.

Fig. 4c compares the ciphertext sizes of the three schemes. Clearly, the communication and storage costs of the SLIM are constant and much lower than those of the other two schemes. In the experiments, the messages have 512 bits, reaching the maximum length. If the DO needs to share a longer message, there are two solutions. One is to perform a cyclic XOR operation when computing C_3. The other is to introduce a symmetric encryption algorithm to encrypt the message and then signcrypt the symmetric key as in the LH-ABSC scheme. As for sharing multiple messages with the same attributes, both solutions can be adopted as well.

Figs. 4d and 4e show that the outsourced costs of the SLIM and LH-ABSC schemes are similar and both are lower than that of the OMDAC-ABSC scheme. According to the theoretical analysis given in Table III, the outsourced de-signcryption cost of the SLIM scheme should be one bilinear mapping operation (approximately 0.49 ms), lower than that of the LH-ABSC scheme, which can be seen from Fig. 4e when the number of attributes is less than 20. However, as the number of attributes increases, the outsourced decryption cost of the SLIM gradually becomes higher than that of the LH-ABSC scheme. This is because the outsourced decryption of the SLIM also involves l_e^2 multiplication operations.

Fig. 4f compares the computation costs of local verification for the three schemes, which is important for the CSP since it is necessary to verify the the before storing the ciphertext. It is worth noting that the LH-ABSC only provides outsourced verification, and we achieve its local verification by replacing the transformed ciphertext with the original ciphertext brought

into Eq. (1) and checking σ_3. The verification costs of both the SLIM and LH-ABSC schemes are almost constant, approximately 4.87 ms and 5.03 ms, respectively. Although the multiplication operations required for verification in these two schemes are linearly related to the number of attributes, their effect to verification costs is negligible and is even smaller than the effect of fluctuations in the experiments. The verification algorithm of the OMDAC-ABSC scheme involves linear-level exponentiation operations and bilinear mapping operations, and therefore its verification cost is much higher than those of the other two schemes. In addition, because the OMDAC-ABSC cannot judge the correctness of the outsourced verification results, the DU has to pay an unacceptable cost if it needs to confirm the authenticity of the ciphertext,

In summary, the actual performance evaluation of the SLIM scheme matches the theoretical analysis provided in Tables III and IV. The cost of signcryption and de-signcryption are almost constant and lightweight. The size of the ciphertext is not affected by the number of attributes and is smaller than those of the previous schemes. Therefore, the proposed SLIM scheme is feasible and practical for application in resource-constrained IoT environments.

VII. Conclusions

In this paper, we have proposed a secure and lightweight multi-authority attribute-based signcryption scheme called SLIM with outsourced computation and constant-size ciphertext to adapt to the resource-constrained and large-scale characteristics of IoT devices. The proposed SLIM scheme not only has a constant short ciphertext size but also allows users to offload most of the computation cost in signcryption, verification and decryption to edge servers. We have provided the proofs of message confidentiality and ciphertext unforgeability under the standard model and analysed signcryptor privacy. We have evaluated the performance of the SLIM scheme by theoretical comparison and actual simulation with the previous schemes, and the results obtained have confirmed that the SLIM scheme is more lightweight in terms of computation, communication and storage costs. However, the SLIM scheme does not support search functions, such as keyword search, which is very practical for IoT with massive data. In the future, we plan to build an efficient index structure in the SLIM scheme for improving searchability.

References

[1] M. Bansal, I. Chana, and S. Clarke, "A survey on IoT big data: Current status, 13 v's challenges, and future directions," *ACM Computing Surveys*, vol. 53, no. 6, pp. 131-1–131-59, Nov. 2021.

[2] J. Chen, *et al.*, "A survey of compiler testing," *ACM Computing Surveys*, vol. 53, no. 1, pp. 4-1–4-36, Jan. 2021.

[3] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet of Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[4] A. Karati, C.-I. Fan, and R.-H. Hsu, "Provably secure and generalized signcryption with public verifiability for secure data transmission between resource-constrained IoT devices," *IEEE Internet of Things J.*, vol. 6, no. 6, pp. 10431–10440, Dec. 2019.

[5] E. Chandanapriya and G. Murali, "Effective data sharing using advanced ring signature with forward security," in *Proc. ICCES 2016* (Coimbatore, India), Oct. 21-22, 2016, pp. 1–5.

[6] C. Lin, R. Xue, and X. Huang, "Linearly homomorphic signatures with designated combiner," in *Proc. ProvSec 2021* (Guangzhou, China), Nov. 5–8, 2021, pp. 327–345.

[7] M. N. Ghuge and P. N. Chatur, "Collaborative key management in ciphertext policy attribute based encryption for cloud," in *Proc. ICICCT 2018* (Coimbatore, India), Apr. 20-21, 2018, pp. 156–158.

[8] M. Xiao, *et al.*, "Attribute-based hierarchical access control with extendable policy," *IEEE Trans. Information Forensics and Security*, vol. 17, pp. 1868–1883, May 2022.

[9] Y. Li, *et al.*, "SDABS: A flexible and efficient multi-authority hybrid attribute-based signature scheme in edge environment," *IEEE Trans. Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1892–1906, Mar. 2021.

[10] M. Gagné, S. Narayan, and R. Safavi-Naini, "Threshold attribute-based signcryption," in *Proc. SCN 2010* (Amalfi, Italy), Sep. 13-15, 2010, pp. 154–171.

[11] Y. S. Rao, "Attribute-based online/offline signcryption scheme," *Int. J. Communication Systems*, vol. 30, no. 16, pp. 1–20, 2017.

[12] H. Hong, Y. Xia, Z. Sun, and X. Liu, "Provably secure attribute based signcryption with delegated computation and efficient key updating," *KSII Trans. Internet and Information Systems*, vol. 11, no. 5, pp. 2646–2659, May 2017.

[13] F. Deng, *et al.*, "Ciphertext-policy attribute-based signcryption with verifiable outsourced designcryption for sharing personal health records," *IEEE Access*, vol. 6, pp. 39473–39486, Jul. 2018.

[14] Q. Xu, *et al.*, "Secure multi-authority data access control scheme in cloud storage system based on attribute-based signcryption," *IEEE Access*, vol. 6, pp. 34051–34074, Jun. 2018.

[15] S. Belguith, N. Kaaniche, M. Hammoudeh, and T. Dargahi, "PROUD: Verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted IoT applications," *Future Generation Computer Systems*, vol. 111, pp. 899–918, Oct. 2020.

[16] J. Yu, *et al.*, "LH-ABSC: A lightweight hybrid attribute-based signcryption scheme for cloud-fog-assisted IoT," *IEEE Internet of Things J.*, vol. 7, no. 9, pp. 7949–7966, Sep. 2020.

[17] Y. S. Rao and R. Dutta, "Efficient attribute-based signature and signcryption realizing expressive access structures," *Int. J. Information Security*, vol. 15, no. 1, pp. 81–109, 2016.

[18] S. Belguith, *et al.*, "Constant-size threshold attribute based signcryption for cloud applications," in *Proc. SECRYPT 2017* (Madrid, Spain), Jul. 26-28, 2017, pp. 212–225.

[19] G. Yu and Z. Cao, "Attribute-based signcryption with hybrid access policy," *Peer-to-Peer Networking and Applications*, vol. 10, no. 1, pp. 253–261, Jan. 2017.

[20] Y. Zhao, *et al.*, "Efficient multi-authority attribute-based signcryption with constant-size ciphertext," in *Proc. DSC 2021* (Fukushima, Japan), Jan. 30-Feb. 2, 2021, pp. 1–8.

[21] C. Wang and J. Huang, "Attribute-based signcryption with ciphertext-policy and claim-predicate mechanism," in *Proc. 7th Int. Conf. Computational Intelligence and Security* (Sanya, China), Dec. 3-4, 2011, pp. 905–909.

[22] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext-policy attribute-based signcryption," *Future Generation Computer Systems*, vol. 52, pp. 67–76, Nov. 2015.

[23] Y. S. Rao, "A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing," *Future Generation Computer Systems*, vol. 67, pp. 133–151, Feb. 2017.

[24] M. Chase, "Multi-authority attribute based encryption," in *Proc. TCC 2007* (Amsterdam, The Netherlands), Feb. 21-24, 2007, pp. 515–534.

[25] A. Alsharif, *et al.*, "A multi-authority attribute-based signcryption scheme with efficient revocation for smart grid downlink communication," in *Proc. iThings, GreenCom, CPSCom and SmartData 2019* (Atlanta, GA, USA), Jul. 14-17, 2019, pp. 1025–1032.

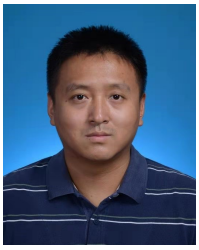
[26] A. Beimel, *Secure Schemes for Secret Sharing and Key Distribution*. Doctor of Science Thesis, Israel Institute of Technology, June 1996.

[27] N. Attrapadung and H. Imai, "Dual-policy attribute based encryption," in *Proc. ACNS 2009* (Paris-Rocquencourt, France), Jun. 2-5, 2009, pp. 168–185.

[28] J. A. Akinyele, *et al.*, "Charm: a framework for rapidly prototyping cryptosystems," *J. Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

Bei Gong received his B.S. degree from Shandong University in 2005, and the Ph.D. degree from the Beijing University of Technology in 2012. In the past five years, he has published more than 30 papers in the first-class SCI / EI and other international famous journals and top international conferences in relevant research fields. His research interests include trusted computing, Internet of things security, mobile Internet of things, mobile edge computing.

Chong Guo received his M.S. degree from Beijing University of Technology, Beijing, China, in 2021. He is working toward the Ph.D degree in computer science and technology with Beijing University of Technology. His research interests including network and information security, Internet of things security and cryptography.



Chong Guo is currently a professorate senior engineer with Beijing Trusty Cloud Technology Co., LTD in Beijing, China. His research interests are computer security theory and cryptographic algorithms. Guo used to work as a technical support engineer for Hewlett-Packard China, assistant manager for China Aerospace Science and Industry Corporation Limited, and director for the examination department of China Cybersecurity Review Technology and Certification Center. In the past five years, with the support of more than 10 national and provincial projects, including National Key Research and Development Program of China and National Natural Science Foundation of China, he has carried out basic theoretical and applied technological research for the security control and trustworthy operation of Internet of Things and cloud computing.

Chen Guo received his B.S. degree from Beijing Jiaotong University in Beijing, China, in 2006 and the M.S. degree from Beijing Jiaotong University in Beijing, China, in 2009. His research interests including information security certification, cybersecurity review and trusted computing.

Yao Sun is currently a Lecturer with James Watt School of Engineering, the University of Glasgow, Glasgow, UK. Dr. Sun has extensive research experience and has published widely in wireless networking research. He has won the IEEE Communication Society of TAOS Best Paper Award in 2019 ICC, IEEE IoT Journal Best Paper Award 2022 and Best Paper Award in 22nd ICCT. He has been the guest editor for special issues of several international journals. He has served as TPC Chair for UCET 2021, and TPC member for a number of international flagship conferences, including ICC 2022, VTC spring 2022, GLOBECOM 2020, WCNC 2019. His research interests include intelligent wireless networking, semantic communications, blockchain system, and resource management in next generation mobile networks. Dr. Sun is a senior member of IEEE.

MUHAMMAD WAQAS (M'18, SM'22) received his B.Sc. and M.Sc. in Electrical Engineering from the Department of Electrical Engineering, University of Engineering and Technology, Peshawar, Pakistan, in 2009 and 2014, respectively. He received his PhD degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2019. From Oct. 2019 to Mar. 2022, he was a Research Associate at the Faculty of Information Technology, Beijing University of Technology, Beijing, China. His current research interests are in the areas of Wireless Communication, vehicular networks, Fog/Mobile Edge Computing, Internet of Things and Machine Learning. He has more than 100 research publications in reputed Journals and Conferences. He is an Associate Editor of the International Journal of Computing and Digital Systems and guest editor of Applied Sciences - MDPI. He is recognised as a Global Talent in the area of Wireless Communications by UK Research and Innovation and a Professional Member of Engineer Australia. He is a senior member of IEEE, a Professional Member of ACM, an IEEE Young Professional, a Member of the Pakistan Engineering Council and an approved supervisor by the Higher Education Commission of Pakistan.

Sheng Chen (IEEE Life Fellow) received his BEng degree from the East China Petroleum Institute, Dongying, China, in 1982, and his PhD degree from the City University, London, in 1986, both in control engineering. In 2005, he was awarded the higher doctoral degree, Doctor of Sciences (DSc), from the University of Southampton, Southampton, UK. From 1986 to 1999, He held research and academic appointments at the Universities of Sheffield, Edinburgh and Portsmouth, all in UK. Since 1999, he has been with the School of Electronics and Computer Science, the University of Southampton, UK, where he holds the post of Professor in Intelligent Systems and Signal Processing. Dr Chen's research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, evolutionary computation methods and optimization. He has published over 700 research papers. Professor Chen has 19,400+ Web of Science citations with h-index 61 and 37,900+ Google Scholar citations with h-index 82. Dr. Chen is a Fellow of the United Kingdom Royal Academy of Engineering, a Fellow of Asia-Pacific Artificial Intelligence Association and a Fellow of IET. He is one of the original ISI highly cited researcher in engineering (March 2004).