Forward-Secure Hierarchical Delegable Signature for Smart Homes

Jianfei Sun, Guowen Xu, Yang Yang, *Senior Member, IEEE*, Xuehuan Yang, Xiaoguo Li, Cong Wu, Zhen Liu, *Member, IEEE*, Guomin Yang, *Senior Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*—Aiming to provide people with great convenience and comfort, smart home systems have been deployed in thousands of homes. In this paper, we focus on handling the security and privacy issues in such a promising system by customizing a new cryptographic primitive to provide the following security guarantees: 1) fine-grained, privacy-preserving authorization for smart home users and integrity protection of communication contents; 2) flexible self-sovereign permission delegation; 3) forward security of previous messages. To our knowledge, no previous system has been designed to consider these three security and privacy requirements simultaneously. To tackle these challenges, we put forward the first-ever efficient cryptographic primitive called the Forward-secure Hierarchical Delegable Signature (FS-HDS) scheme for smart homes. Specifically, we first propose a new primitive, efficient Hierarchical Delegable Signature (HDS) scheme, which is capable of supporting partial delegation capability while realizing privacy-preserving authorization and integrity guarantee. Then, we present an FS-HDS for smart homes with the efficient HDS as the underlying building block, which not only inherits all the desirable features of HDS but also ensures that the past content integrity is not affected even if the current secret key is compromised. We provide comprehensively strict security proofs to prove the security of our proposed solutions. Its performance is also validated via experimental simulations to showcase its practicability and effectiveness.**

***Index Terms*—Smart home, self-sovereign delegation, forward-secure, integrity.**

Received 4 July 2024; revised 9 November 2024 and 20 January 2025; accepted 18 March 2025. Date of publication 26 March 2025; date of current version 18 April 2025. This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative, and the AXA Research Fund. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority. The associate editor coordinating the review of this article and approving it for publication was Dr. Sharif Abuadbba. *(Corresponding author: Guowen Xu.)*

Jianfei Sun, Yang Yang, Guomin Yang, and Robert H. Deng are with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: jfsun@smu.edu.sg; yyang@smu.edu.sg; gmyang@smu.edu.sg; robertdeng@smu.edu.sg).

Guowen Xu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: guowen.xu@uestc.edu.cn).

Xuehuan Yang and Cong Wu are with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: s190113@e.ntu.edu.sg; cong.wu@ntu.edu.sg).

Xiaoguo Li is with the School of Computing and Information Systems, Singapore Management University, Singapore 178902, and also with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: csxgli@cqu.edu.cn).

Zhen Liu is with the School of Electronics Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: liuzhen@sjtu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2025.3555185, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2025.3555185

I. Introduction

In THE rapidly evolving landscape of smart home technology, while the benefits of real convenience and efficiency are undeniable, significant concerns regarding security and privacy persist [1], [2], [3]. The integration of intelligent systems, connecting everything from doorbells to thermostats via the Internet of Things (IoT), also introduces potential vulnerabilities. These vulnerabilities open avenues for cyberattacks, unauthorized data collection, and breaches in authentication, potentially exposing users to risks such as identity theft, surveillance, and unauthorized access [4], [5]. Furthermore, as these devices become more interconnected, a single compromised device could threaten the entire network's security. With the prevalence of smart homes, addressing these privacy and security challenges is imperative to ensure the protection of personal and household privacy and security.

A. Security, Privacy & Efficiency Challenges

Despite various advanced cryptographic methodologies that could be exploited to mitigate some security and privacy issues in smart homes, current state-of-the-art initiatives still exhibit certain deficiencies in security, privacy, and efficiency, as elaborated below.

1) Inadequacy of Fine-Grained, Privacy-Preserving Countermeasures to Tackle Permission Management and Integrity Protection: Smart homes, which often involve managing a variety of IoT devices, necessitate fine-grained access control mechanisms due to the complexity and diversity of the devices and user roles. That is to say, smart home applications need to design access controls over who can perform certain actions within a smart home environment based on attributes rather than identity. For example, attributes could include roles such as "parent", "child", and "guest", or rights like "can adjust the thermostat", "can view camera", or "can unlock door". Since smart IoT devices often operate based on remote instructions from adjusting the thermostat settings to unlocking doors, it is necessary to ensure that these commands are indeed issued by authorized entities and have not been tampered with during transmission. Furthermore, user privacy should also be essentially ensured so that the actions performed do not expose the identity of the user. If user identities linked directly to actions are compromised, it could lead to targeted scams or unwanted solicitation. As one of the effective potential solutions, attribute-based signature (ABS) technologies [6], [7], [8], [9], [10], [11], [15], [16], [17], [18] could provide a secure approach for managing access in smart homes, catering

to the needs of data integrity protection without compromising user privacy. However, existing ABS methodologies are either computationally intensive, owing to the extensive pairing calculations required, or impractical due to the necessity of involving a third party (*e.g.*, cloud) to generate server-aided signature. Consequently, how to *efficiently* and *practically* realize both fine-grained permission management and integrity protection is continuously challenging.

2) Absence of Self-Sovereign Delegation Mechanisms to Flexibly, Efficiently Handle Partial Delegation Capabilities: In smart homes, the ability to flexibly manage access and control permissions is essential for efficient household operations and security. For example, if a parent is late coming home from work, they might need a trusted family friend to temporarily pick up their child from school and bring them home. With the smart home system, a parent can dynamically delegate temporary access to a friend. This approach avoids permanently assigning the same access privileges to the friend. It allows the friend to unlock the door during a specified time, ensuring the child can be sent home safely. An intuitive approach would be to use proxy-based techniques, which are publicly regarded as solutions to realize dynamic delegation authorization. However, this solution is less practical because it requires the delegator to grant access to the entire set of resources he/she owns, rather than allowing the partial delegation of a subset to others. As an alternative methodology, hierarchical delegable approaches [21], [22], [23], [24] have been devised, which enable a delegator to partially delegate his/her capability to other delegatees, such that the delegatees can temporarily have partial access to the resources the delegator authorizes. However, most existing hierarchical delegable approaches (Refer to Section II. A for more details) are inefficient due to prohibitive communication and computation overheads, since non-interactive zero-knowledge (NIZK) proofs are exploited. Hence, how to *efficiently* realize *partial* delegation authorization capabilities without utilizing NIZK proofs remains an unsolved challenge.

3) Intractability of Hindering the Secret Key Exposure Resulting in Security and Privacy Vulnerabilities: In smart homes, safeguarding against the exposure of secret keys instead of privilege revocation, which leads to various security and privacy vulnerabilities, constitutes a formidable challenge. For example, if an attacker were to obtain a residence's secret key, irrespective of whether due to deliberate or technical key compromise, they could potentially manipulate sensitive functionalities such as access controls and system logs. This could include altering the login and entry records of smart security systems, enabling unauthorized access to the home without triggering alarms, or erasing any trace of the intrusion in the system's history. Such security breaches compromise not only the physical safety of the residence but also the privacy of its occupants by revealing household routines and activities. Hence, it is essential to devise a key exposure resistance mechanism to mitigate these vulnerabilities in smart homes. One of the frequently utilized methodologies is the implementation of the tree-based forward-secure (TFS) technique [12], [13], [14], [40] to realize updating secret keys associated with time periods, which are only encoded in leaf nodes of the tree. However, the costs of key updates in the majority of TFS-based solutions are exorbitant since every key update requires finding the intended time periods by traversing from the root node to the leaf node. Another potential approach is to exploit puncturable signature (PS) based solutions [42], [44], [45], [46] to guarantee the timeliness of the secret key, thus realizing forward security. However, most PS-based solutions are inefficient due to the computationally expensive puncturing process of frequent key updates (See Section II-B for more details). Consequently, how to develop *efficient* forward-secure solutions to handle key exposure in smart home applications remains a significant challenge.

B. Solutions and Technical Challenges

1) Imperfections of Current Approaches: As far as we are aware, no existing studies adequately address all the outlined challenges simultaneously. As succinctly summarized above, ABS technologies enable a user to create a signature that proves possession of certain attributes without compromising his/her specific attributes used; hierarchical delegable attribute-based signature/encryption approaches (HABS/HABE) support a delegator with access to a set of resources in delegating a subset of those resources to others; both tree-based forward-secure ABS (TFS-ABS) and PS methodologies enable the timeliness of the secret key, such that past communications are immune from the exposure of the secret key in the future. These cryptographic technologies may be employed to mitigate the above challenges, whereas they are only applicable to tackle certain ones in smart homes.

Concisely, with the feature of providing fine-grained permission management, ABS methodologies enable integrity protection for communications among IoT devices; however, they fall short in addressing the needs of (2) & (3). Apart from inheriting the properties that ABS enables, HABS approaches are additionally capable of supporting self-sovereign delegation, whereas existing HABS technologies are infeasible for the requirement of (3). Besides, the computation and communication costs of existing HABS are prohibitive, thus inappropriate for resource-constrained IoT devices due to the NIZK involved; both TFS-ABS and PS techniques enable key exposure resistance while allowing for fine-grained permission management and integrity protection, nevertheless, neither of them caters to the requirements of (2). Furthermore, existing TFS-ABS approaches are computationally intensive due to the use of NIZK proofs for attribute privacy protection, while current PS solutions do not consider the privacy issue.

2) Potential Solutions & Technical Challenges: Intuitively, the security, privacy, and efficiency requirements stated above might be met by integrating the aforementioned technologies. The most natural methods involve the application of the TFS technique to HABS schemes or the incorporation of PS technology into HABS schemes. However, elegantly realizing the technical convergence of these technologies to devise an *efficient* forward-secure hierarchical delegable signature (FS-HDS) methodology is not trivial but intractable for the following reasons: (a) for the convergence of TFS and HABS, although TFS techniques typically employ a

hierarchical structure of keys as those in HABS, the distinct hardness assumptions they depend on lead to some security vulnerabilities in the integrated scheme; (b)for the integration of PS and HABS, the secret keys in HABS are structured (*i.e.*, well-formed) while those in PS are structureless (*i.e.*, ill-formed). Once unstructured keys are embedded into structured secret keys, the original structure of HABS' secret keys is likely to be damaged, which further leads to the failure of self-sovereign delegation. Besides, despite technically perfect combinations, merely achieving a technically seamless integration of these technologies is insufficient to devise such an *efficient* FS-HDS scheme. The fundamental reason leading to this inefficiency is the use of NIZK proofs for signature generation and verification in current HABS schemes, and how to implement an efficient HABS without using NIZK technologies remains a technical challenge.

C. Our Contributions

In this paper, we design an efficient FS-HDS, the first-ever efficient Forward-secure Hierarchical Delegable Signature without using NIZK proof techniques for smart homes, which elegantly addresses the most concerned challenges in smart homes, such as efficient fine-grained permission management, integrity and privacy protection, flexible self-sovereign delegation, and key exposure resistance. The key technical novelties are as follows: (I) We observe that a wicked identity-based encryption (IBE) can be efficiently converted into an efficient wicked identity-based signature (IBS) with a skillful transformation approach instead of the known general one, which innovatively overcomes the inefficiency issue of the general transformation solution, *i.e.*, a randomness is encoded with the encryption algorithm of IBE and the verifier determines the consistency with the random number by decoding it with the secret key (given as signature parts); Then, the wicked IBS is used to design our privacy-preserving hierarchical delegable signature (HDS); (II) with the HDS, we also construct an FS-HDS, which inherits all desirable features of the HDS while additionally achieving forward security. The primary contributions of this paper include the following:

- *Efficient fine-grained authorization*: To allow for the granularity and effectiveness of access control in smart homes, both our HDS and FS-HDS employ a fine-grained authorization mechanism that precisely delineates user permissions based on their attributes, which contributes to the smart homes' scalability.
- *Integrity and privacy protection*: To ensure that the commands sent to smart IoT devices are not tampered with during transmission, both our HDS and FS-HDS methodologies empower a sender to generate a signature with his/her secret key associated with a collection of attributes, such that the integrity of transmitted commands is preserved while simultaneously safeguarding the privacy of the sender's attribute information.
- *Flexible self-sovereign delegation*: To enable a user with access to a set of resources to *individually* grant another user access to a specific subset of those resources, both our HDS and FS-HDS exploit self-sovereign delegation technique to support a delegator in delegating his/her partial capabilities to others. Compared to other inflexible delegations from the authority, flexibility and practicability are greatly enhanced by self-sovereign delegation.
- *Key exposure resistance*: To mitigate the risks associated with key exposure in smart homes, our FS-HDS employs the TFS technique in our HDS to realize the forward security of previous messages even if key exposure occurs instead of the occurrence of privilege revocation.

Additionally, comprehensively rigorous security proofs have been provided to demonstrate the FS-HDS with forward security and unforgeability. Experimental evaluations have also been conducted, showcasing the efficiency of our FS-HDS.

II. Related Work

A. Hierarchical Attribute-Based Signature Schemes

Attribute-based signature (ABS) [6], [7] was a cryptographic primitive designed to provide privacy-preserving authentication of messages, which enables a signer to create a signature that proves possession of certain attributes without compromising the specific attributes used. Standard ABS schemes primarily focus on two security properties: user privacy and unforgeability. Specifically, user privacy can be protected by preventing the disclosure of the signer's identity and the specific attributes used in the signing, while maintaining unforgeability by ensuring that only a signer with the requisite attributes can produce a valid signature for a given policy. Subsequently, numerous ABS schemes with various features have been devised, such as decentralized ABS [8], [9], [10], [11], forward-secure ABS [12], [13], [14], registered ABS [15], outsourced ABS [19], [20] and lattice-based ABS [16], [17], [18], etc. However, most existing ABS schemes are neither flexible nor scalable due to the need for a trusted third-party (*e.g.*, cloud server) to complete efficient signature generation and the lack of support for hierarchical delegation of attributes.

The hierarchical ABS (HABS) scheme, an advanced cryptographic protocol, was recently proposed by Dragan et al. [21], which overcomes some limitations of traditional ABS schemes by allowing for controlled delegation of attributes from a root authority through various intermediate entities to the users [25]. In HABS [21], intermediate entities have the capability to delegate attributes to any entity within the system, and users are able to obtain attributes from any authorized entity within the hierarchy. However, this methodology is inefficient since HABS realizes the delegation capability by utilizing a tag-based signature (TBS) and requiring it to incorporate all previous public keys from the delegation chain. To eliminate the dependency on successive releases of TBS from higher to lower authorities on the authorization path, Gardham et al. [22] proposed a new HABS with short key and optimal signature primarily based on homomorphic trapdoor commitments (HTC) and NIZK proofs. While the work [22] has significantly optimized the HABS key lengths and delegation efficiency [21], its efficiency, regardless of computation and communication overheads, is still prohibitive due to the need for multiple TBS and the associated NIZK proofs across

TABLE I
PROPERTY-WISE COMPARISONS AMONG SIMILAR RELATED WORKS

Type of scheme	Key-exposure Resistance	Privacy Preserving	Fine-granularity	Key Delegation	High Efficiency	Construction Structure
ABS [6], [7], [15]	✗	✓	✓	✗	✓	Pairing-based
HABS [21], [22]	✗	✓	✓	✓	✗	Pairing-based
HABS [I, II] [24]	✗	✗	✓	✓	✗/ ✓	Pairing-based
HABS [23]	✗	✓	✓	✓	✗	Lattice-based
FS-IBS [36], [37]	✓	✗	✗	✗	✓	Pairing-based
FS-ABS [12]–[14], [40]	✓	✗	✓	✗	✓	Pairing-based
PS [43]	✓	✗	✓	✗	✗	iO-based
PS [42], [44]–[46]	✓	✗	✓	✗	✗	Pairing-based
Our HDS	✗	✓	✓	✓	✓	Pairing-based
Our FS-HDS	✓	✓	✓	✓	✓	Pairing-based

Note: "✓": this property the work provides; "✗": this feature the work does not support; Privacy-preserving: do not leak identity or attributes; "✗/ ✓" means that the basic work (*i.e.*, [I]) does not support this feature but the improved one (*i.e.*, [II]) supports it;

the hierarchical structure. Subsequently, Gardham et al. [23] also formulated a revocable lattice-based HABS with the methodologies as those in [22]. Very recently, Kumar et al. [24] have realized a general concrete HABS for supporting the key delegation in end-to-end IoT applications based on the hierarchical identity-based encryption scheme (HIBE) [28]. However, the HABS [24] construction is also inefficient since the general signature transformation solution in the same vein as has been done in [26] and [27] is exploited to extend a HIBE to a HABS scheme. Subsequently, they also presented an improved HABS in the full version of [24], however, the scheme cannot realize the forward security. Please note that the schemes (viewed as weak HABS schemes) in [24] actually do not consider attribute privacy of a sgner since the attribute pattern/policy in the signature can directly reveal the attribute privacy a signer owns.

To summarize, while there have been attempts to develop HABS schemes, designing a highly efficient HABS solution without NIZK proofs used remains one of the focuses for continuous exploration.

B. Forward-Secure Signature Schemes

The concept of forward security (FS) was initially introduced by Anderson [29], which was originally explored in the context of key exchange protocols, to the domain of digital signatures in order to lessen the consequences of signing key compromises. Subsequently, Bellare et al. [30] formally formulated the concept of a forward-secure digital signature and introduced its first practical implementation based on the hardness of factoring large numbers. Subsequently, numerous FS signature schemes [31], [32], [33] mainly based on tree-based HIBE solutions were suggested in terms of security and efficiency. The idea of these FS solutions has been broadened to include FS threshold signature (FS-TS) [34], [35], FS identity/attribute-based signature (FS-IBS) [12], [13], [14], [36], [37], [40], FS multi-signature (FS-MS) [38], [39] and FS lattice-based signature (FS-LBS) [41], etc. Although the above FS signature schemes enable forward security, most of them fail to support fine granularity.

Recent advancements in some puncturable signature schemes [42], [43], [44], [45], [46] ensuring fine-grained forward-security have been made. Specifically, Green et al. [42] formalized the concept of puncturable encryption (PE) that provides fine-grained forward-security property. In [42], this technique involves dynamically altering encryption keys after each message, effectively rendering past keys inaccessible and preventing retrospective decryption of messages even if a key is compromised. After that, Bellare et al. [43] presented the idea of puncturable signature (PS) based on the PE solution, providing a design reliant on indistinguishability obfuscation (iO) and one-way functions. Halevi et al. [44] defined a PS scheme that necessitates frequent updates to the signers' public keys. Li et al. [45] proposed a puncturable signature scheme that incorporates the principles of bloom filter encryption. Wei et al. [46] put forth two puncturable multi-signature schemes to achieve compact and efficient forward-security. The PS-based technique enables robust protection against key compromises by allowing dynamic key updates after each use. Nevertheless, this approach also incurs significant key management overhead and computational costs due to a computationally expensive puncturing process of frequent key updates.

In summary, the above FS signature schemes maintain the integrity of previously signed messages by periodically updating and invalidating the current signing key, ensuring that security persists even if a key compromise occurs. However, most existing FS signature schemes are neither efficient nor capable of supporting privacy-preserving authentication of messages and flexible fine-grained key delegation. TABLE. I provides a comparative conclusion of characteristics across various existing works.

III. BASIC KNOWLEDGE AND DEFINITIONS

A. Hardness Assumptions

Definition 1: Given a tuple $(g, g^{\alpha}, g^{\beta}) \in \mathbb{G}_0^3$, the goal of any adversary $\mathcal{A}$ is to compute $g^{\alpha\beta} \in \mathbb{G}_0$. We say the Computational Diffie-Hellman (CDH) problem holds [28] if the advantage in solving CDH assumption is negligible.

Definition 2: Given a tuple $(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{\mathcal{L}}}) \in \mathbb{G}_0^{\mathcal{L}-1}$, the goal of any adversary $\mathcal{A}$ is to compute $g^{\alpha^{\mathcal{L}+1}} \in \mathbb{G}_0$. We say the Computational Diffie-Hellman Exponent (CDHE) problem holds [39] if the advantage in solving the CDHE assumption is negligible.

B. Attribute Pattern

An attribute pattern $\mathcal{P}$ is defined as an attribute vector $(\mathcal{P}_1, \ldots, \mathcal{P}_\ell) \in \{0,1\}^* \cup \{*\}$ of length ℓ, where $*$ denotes a special wildcard symbol [28]. Each element $\mathcal{P}_i$ in the attribute pattern $\mathcal{P}$ can be either a specific identity string or the wildcard. A user possessing the secret key for a particular pattern $\mathcal{P}$ can generate secret keys for any pattern $\mathcal{P}'$ that matches $\mathcal{P}$. We define a pattern $\mathcal{P}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_\ell)$ as matching $\mathcal{P}$, denoted $\mathcal{P}' \in_* \mathcal{P}$, if and only if $\mathcal{P}'_i = \mathcal{P}_i$ or $\mathcal{P}_i = *$. An access pattern $\mathcal{P}^*$ is also defined as an access vector $(\mathcal{P}^*_1, \ldots, \mathcal{P}^*_\ell) \in \{0,1\}^* \cup \{*\}$. If $\mathcal{P} \in_* \mathcal{P}^*$ holds, we say that the attribute pattern $\mathcal{P}$ matches the access pattern $\mathcal{P}^*$.

C. Encoding Time Periods

In previous tree-based forward-secure signatures, the time periods $\mathcal{T}$ were only associated with leaf nodes using the pre-order traversal algorithm [48], [49]. This resulted in an amortized complexity of key updates of $\mathcal{O}(\log \mathcal{T})$ exponentiations. In our method, all nodes related to time periods $\mathcal{T}$ are encoded with the pre-order traversal algorithm to achieve a complexity of key updates of $\mathcal{O}(1)$ exponentiations [39].

In a binary tree with depth $\mathcal{L}-1$, there are $2^{\mathcal{L}}-1$ nodes corresponding to time periods $\mathcal{T}$ in $[2^{\mathcal{L}}-1]$. The nodes of the tree with depth $\mathcal{L}-1$ and strings in $\{1,2\}^{\leq \mathcal{L}-1}$ are encoded, where 1 and 2 denote taking the left and right branches, respectively. Note that we use the strings $\{1,2\}$ instead of $\{0,1\}$ to encode the time periods, as strings of length exactly $\mathcal{L}-1$ are required, thus requiring to pad strings in $\{1,2\}^{\leq \mathcal{L}-1}$ with zeroes [24].

The association between $\mathbf{t} = t_1 \| t_2 \| \ldots \in \{1,2\}^{\leq \mathcal{L}-1}$ and $t \in [2^{\mathcal{L}}-1]$, for any integer $\mathcal{L}$, is explicitly described as a bijection $t(\mathbf{t}) = 1 + \sum_{j=1}^{|\mathbf{t}|}(2 - t_i + 2^{\mathcal{L}-i}(t_i - 1))$.

For example, for $\mathcal{L} = 3$, this maps ε, 1, 11, 12, 2, 21, 22 to $1, 2, 3, 4, 5, 6, 7$. The inverse of this bijection can be described as:

$$\begin{aligned} \mathbf{t}(1) &= \varepsilon, \\ \mathbf{t}(t) &= \mathbf{t}(t-1)\|1 \quad \text{if } |\mathbf{t}(t-1)| \neq \mathcal{L}-1, \\ \mathbf{t}(t) &= \bar{\mathbf{t}}\|2 \quad \text{if } |\mathbf{t}(t-1)| = \mathcal{L}-1, \end{aligned}$$

where $\bar{\mathbf{t}}$ is the longest string such that $\bar{\mathbf{t}}\|1$ is a prefix of $\mathbf{t}(t-1)$.

The bijection implies a natural precedence relation over $\{1,2\}^{\leq \mathcal{L}-1}$, where $\mathbf{t} \preceq \mathbf{t}'$ if and only if either $\mathbf{t}$ is a prefix of $\mathbf{t}'$ or there exists $\bar{\mathbf{t}}$ such that $\bar{\mathbf{t}}\|1$ is a prefix of $\bar{\mathbf{t}}$ and $\bar{\mathbf{t}}\|2$ is a prefix of $\mathbf{t}'$. *In this paper, we also write $\mathbf{t}$ and $\mathbf{t}+1$ as $t, t+1$.*

We define a set $\Phi_{\mathbf{t}}$ associated with any $\mathbf{t} \in \{1,2\}^{\leq \mathcal{L}-1}$ as $\Phi_{\mathbf{t}} = \mathbf{t} \cup \{\bar{\mathbf{t}}\|2 : \bar{\mathbf{t}}\|1$ is a prefix of $\mathbf{t}\}$, which contains $\mathbf{t}$ along with all the "right-hand siblings" of nodes on the path from $\mathbf{t}$ to the root. This set is the smallest set of nodes that includes a prefix of all $\mathbf{t}' \succeq \mathbf{t}$. For example, for $\mathcal{L} = 3$, the following information can be concluded: $\Phi_1 = \{1,2\}$, $\Phi_{11} = \{11, 12, 2\}$, $\Phi_{12} = \{12, 2\}$.

The following properties for the sets $\Phi_{\mathbf{t}}$ must hold:

- $\mathbf{t}' \succeq \mathbf{t}$ if and only if there exists $\mathbf{x} \in \Phi_{\mathbf{t}}$ such that $\mathbf{x}$ is a prefix of $\mathbf{t}'$.
- For all $\mathbf{t}$, $\Phi_{\mathbf{t}+1} = \Phi_{\mathbf{t}} \setminus \{\mathbf{t}\}$ if $|\mathbf{t}| = \mathcal{L}-1$, or $\Phi_{\mathbf{t}+1} = (\Phi_{\mathbf{t}} \setminus \{\mathbf{t}\}) \cup \{\mathbf{t}\|1, \mathbf{t}\|2\}$ if $|\mathbf{t}| < \mathcal{L}-1$.
- For all $\mathbf{t}' \succeq \mathbf{t}$, for all $\mathbf{x}' \in \Phi_{\mathbf{t}'}$, there must exist $\mathbf{x} \in \Phi_{\mathbf{t}}$ such that $\mathbf{x}$ is a prefix of $\mathbf{x}'$.

In the above properties, the first one is used for signature verification, and the last two are used for fast key updates.

D. The Framework of FS-HDS Scheme

Our FS-HDS contains five algorithms: **Setup**, **KeyDerive**, **KeyUpdate**, **Sign** and **Verify**. The **Setup** algorithm is run by a trusted authority (TA) to generate the public parameter and the master secret key; the **KeyDerive** algorithm is used to produce the secret key for signers; the **KeyUpdate** algorithm is used to update the secret key for the time period to an updated secret key for the subsequent time period; the **Sign** algorithm is utilized by the signer to create a signature on a message in a specified time period; the **Verify** algorithm is executed by any verifier to check the validity of the signature of a given message. The detailed algorithms of **Setup**, **KeyDerive**, **KeyUpdate**, **Sign** and **Verify** are described as follows:

- $(\mathsf{pp}, \mathsf{msk}) \leftarrow$ **Setup** $(\lambda, \ell, \mathcal{T})$: On the input of a security parameter λ, the maximum number ℓ of levels, the maximum number time slots $\mathcal{T}$, TA generates the public parameter pp and master secret key msk.
- $\mathsf{sk}_{\mathcal{P},t} \leftarrow$ **KeyDerive** $(\mathsf{msk}, \mathsf{pp}, \mathcal{P}, t)$: On the input of the master secret key msk, the public parameter pp, an authorization pattern $\mathcal{P}$ and a time period t, the TA generates a secret key $\mathsf{sk}_{\mathcal{P},t}$ with regard to $\mathcal{P}$ for a user. Particularly, the initial secret key for the first time period is set as $\mathsf{sk}_{\mathcal{P},1}$. It is also worth noting that anyone owning the secret key can create a new delegated key $\mathsf{sk}'_{\mathcal{P}',t}$ for pattern $\mathcal{P}'$, *i.e.*, $\mathsf{sk}'_{\mathcal{P}',t} \leftarrow$ **KeyDerive** $(\mathsf{sk}_{\mathcal{P},t}, \mathsf{pp}, \mathcal{P}', t)$.
- $\mathsf{sk}_{\mathcal{P},t+1} \leftarrow$ **KeyUpdate** $(\mathsf{sk}_{\mathcal{P},t})$: On the input of the secret key $\mathsf{sk}_{\mathcal{P},t}$ for the time period t, it can produce the new updated secret key $\mathsf{sk}_{\mathcal{P},t+1}$ for the next period $t+1$. It is worth noting that it can also flexibly offer key updates by performing $\mathsf{sk}_{\mathcal{P},t'} \leftarrow$ **KeyUpdate** $(\mathsf{sk}_{\mathcal{P},t})$ as long as $t' \geq t$.
- $\sigma \leftarrow$ **Sign** $(\mathsf{sk}_{\mathcal{P},t}, m, \mathcal{P}^*, t)$: On the input of secret key sk with the access pattern $\mathcal{P}$, a message $m \in \{0,1\}^*$, a time period t and a pattern policy $\mathcal{P}^*$, the signer returns a signature σ with its policy $\mathcal{P}^*$.
- ("1" *or* "$\perp$") $\leftarrow$ **Verify** $((\sigma, \mathcal{P}^*), m, \mathsf{pp}, t)$: On the input of a message m, a signature σ with its access pattern (*i.e.*, policy) $\mathcal{P}^*$, the time period t and the public parameter pp, the verifier determines the validity of the given signature. If it is valid, the verifier returns "1"; otherwise, the verifier returns an abortion symbol "$\perp$".

Correctness: The standard consistency constraint of digital signature must be satisfied, *i.e.*, if $\sigma \leftarrow$ **Sign** $(\mathsf{sk}_{\mathcal{P}}, m, \mathcal{P}^*, t)$ holds, then "1" $\leftarrow$ **Verify** $((\sigma, \mathcal{P}^*), m, \mathsf{pp}, t)$ always holds.

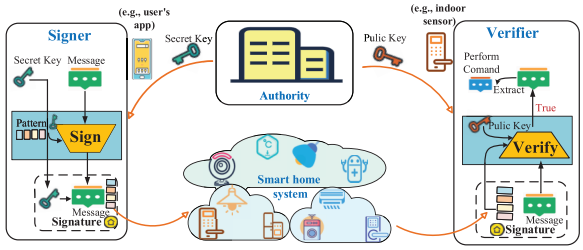


Fig. 1. System Model for Smart Homes.

E. Security Definition of FS-HDS

Unforgeability under chosen-pattern-and-message attacks for hierarchical forward-secure signature is formalized between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ via the following game:

- **Init**: $\mathcal{A}$ first gives the challenge pattern $\mathcal{P}^* = (\mathcal{P}_1^*, \ldots, \mathcal{P}_\ell^*)$ and the challenge time period t^* to $\mathcal{C}$.
- **Setup**: $\mathcal{C}$ executes **Setup** to generate the public parameter pp and the master secret key msk. Subsequently, $\mathcal{C}$ securely stores msk while pp is sent to $\mathcal{A}$.
- **Key Update**: If the current time period t (initially $t = 1$) is less than $\mathbf{T}$, then this oracle updates the key $\mathsf{sk}_{\mathcal{P},t}$ for the time period t to $\mathsf{sk}_{\mathcal{P},t+1}$.
- **Signing Phase**: Upon receiving a message m, the oracle utilizes the Signing oracle with the current $\mathsf{sk}_{\mathcal{P},t}$ and m, resulting in the signature σ.
- **Break-in**: This experiment records the break-in time $\bar{t} \leftarrow t$ and provides the current signing key $\mathsf{sk}_{\mathcal{P},\bar{t}}$ to $\mathcal{A}$. This oracle can only be queried once, and subsequent queries to the key update or signing oracles by $\mathcal{A}$ are disallowed.
- **Forgery**: $\mathcal{A}$ outputs $(\sigma^*, m^*, \mathcal{P}^*, t^*)$, where the pattern itself $\mathcal{P}^*$ does not emerge in any KeyDerive queries during time period t^*. Furthermore, any **Sign** on $(m^*, \mathcal{P}', t^*)$ during time period t^*, where $\mathcal{P}'$ is the $\mathcal{P}^*$, has also not been queried in this phase.

The game is won by $\mathcal{A}$ if the response of the **Verify** on $(\sigma^*, m^*, \mathcal{P}^*, t^*)$ always outputs "1". Besides, the break in oracle was queried, then it did so in a time period $\bar{t} \geq t^*$. The $\mathcal{A}$'s advantage is defined as $Adv_{FS}^{Sig}(\mathcal{A})$.

IV. Models, Goals and Technical Overview

A. System Architecture for Smart Homes

Our FS-HDS based smart home architecture mainly consists of three types of entities: Authority, Signers and Verifiers, as shown in Fig. 1. The *authority* in smart homes is commonly the home manager (*i.e.*, householder), which is a fully honest entity responsible for initializing the public parameters of smart homes and distributing the private key to other family members by implementing the Setup & KeyDerive algorithms; The *signers* can be home members or delegatees of home members, who can make requests via their apps to smart IoT devices (*e.g.*, smart door lock). Through these devices, they can perform some commands (*e.g.,* opening door, cooking on time, etc.) within the specified time. Specifically, the signer selects an attribute pattern (*i.e.*, policy) instead of his/her identity and utilizes his/her secret key to generate a signature regarding the requests by performing the Sign algorithm; It is worth learning that a family member acting as a delegator can also carry out the KeyDerive algorithm for a delegatee to realize self-independent delegation of his/her partial capabilities. The delegatee can also make some service requests to some IoT devices as the delegator makes; The *verifiers* are smart IoT devices, which can confirm whether the signature for the command is valid via signature verification. Once the validity of the command is authenticated by calling the Verify algorithm, they are required to follow the instructions and execute the operation commands. It is important to note that signers and verifiers are also general users who could maliciously launch various attacks, including forgery and collusion attacks, etc., to compromise the integrity of ciphertexts generated by others.

B. Threat Model and Security Goals

In our smart homes with our FS-HDS, four various attacks are considered in our threat model. Specifically, (I) any attackers try to launch collusion attacks to derive a legitimate secret key, thus further launching forgery attacks to forge a valid signature; (II) malicious adversaries who intentionally launch forgery attacks aim to compromise data authenticity by intercepting-then-altering or substituting raw data even if they do not have the legitimate secret keys; (III) any adversary who captures the current compromised secret key intends to undermine the validity and originality of previously issued signatures; (IV) any adversary, regardless of whether they are authorized or unauthorized smart IoT devices, attempts to learn some privacy information (e.g., identity, attribute) from a valid signature. Taking into account these attacks in smart homes, we have formally stated our security objectives as follows:

- *Collusion attack resistance*: Any adversary in smart homes lacking authorized secret keys is incapable of reconstructing a legitimate secret key by combining partial keys, irrespective of whether these partial keys are obtained from authorized or unauthorized ones.
- *Authenticity of requested contents*: Once a signature for the requested content has been generated, it cannot be tampered with or forged by any malicious user unless they possess the *same* legitimate secret keys.
- *Forward security of previous contents*: Even if current secret keys are compromised to an adversary, the contents previously communicated remain forgery resistant and unaffected.
- *Identity/Attribute anonymity*. The access pattern attached in a signature indicating partial attributes the signer possesses is embedded in the signature, any user/smart device can neither deduce the signer's complete set of attributes nor ascertain the signer's identity from the signature alone. In other words, it offers privacy by allowing the signer to remain anonymous while ensuring that the verifier can confirm that the signer possesses the required attributes.

Besides, the high efficiency is also one of our design goals, ensuring rapid processing and minimal computational load.

This efficiency facilitates quick signature and verification in smart homes.

V. Efficient HDS Construction

A. Basic Ideas and Technical Overview

Unlike the previously inefficient methodologies (*e.g.*, [21], [22], [23], [24]) that require the use of Non-Interactive Zero-Knowledge (NIZK) proofs to prove knowledge of each tag-based signature at every delegation of each attribute required to meet the policy, we have devised an efficient privacy-preserving hierarchical delegable signature (HDS) mechanism without using NIZK proofs. The reason previous approaches failed to devise an efficient privacy-preserving HDS without NIZK proofs is the difficulty in producing well-formed (*i.e.*, structured) secret keys. Our HDS is achieved by seamlessly integrating hierarchical delegable ID-based cryptography [26] and the standard signature technique. Specifically, to circumvent the challenge of producing a well-formed secret key, where a secret key in our HDS involves two parts: one is an integrated secret key, and the others are multiple well-formatted secret keys, we exploit the component padding technique [26] to incorporate new factors into the integrated key component and insert the formatted secret keys produced using random values associated with these factors into distributed well-formatted key components. It is worth noting that the term "structure" refers to the property that a secret key is *well-formed*, meaning that if a secret key is used to generate a new key (e.g., in the context of key delegation or transformation), the structure of the newly generated key remains consistent with the original key. For instance, the number of sub-keys or the key format remains unchanged, ensuring structural consistency between the original and derived keys. In contrast, "structureless (unstructured)" refers to the property that a secret key is *ill-formed*, implying that if a secret key is used to generate a new key, the structure of the newly generated key changes. For example, the number of sub-keys in the derived key may differ from that of the original key, or the format of the key might not adhere to the original structure.

B. Concrete Construction of HDS

- **Setup** (λ, ℓ): On the input of a security parameter λ, the maximum number ℓ of attributes, the root identity performs the following executions:
  1) Select random generators $g, h_0, \ldots, h_\ell \in \mathbb{G}_0$ and $\tau \in \mathbb{Z}_p$ and compute $g_1 = g^\tau$.
  2) Select a hash function $\mathcal{H}$ that is defined as $\mathcal{H} : \{0,1\}^* \times \mathbb{G}_0 \to \mathbb{Z}_p$.
  3) Publish the public parameter $\mathsf{pp} = (g, g_1, h_0, \ldots, h_\ell, \mathcal{H})$ and keep the master secret key $\mathsf{msk} = h_0^\tau$ secret.
- **KeyDerive** $(\mathsf{msk}, \mathsf{pp}, \mathcal{P})$: On the input of the master secret key msk, the public parameters pp, a pattern $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_\ell)$, the root proceeds with the following executions:
  1) Let $\mathcal{I} = \overline{\mathcal{W}}(\mathcal{P})$. For all $i \in \mathcal{I}$, pick $r_i \in \mathbb{Z}_p$ and let $b_i \leftarrow g^{r_i}$, $a \leftarrow h_0^\tau \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i}$, where $\mathcal{H}_i(x)$ is set as $\mathcal{H}_i(x_i) = g_1^x \cdot h_i = (g^\tau)^{x_i} \cdot h_i$ and $\overline{\mathcal{W}}(\mathcal{P})$ refers to the complementary set containing all non-wildcard indices in $\mathcal{P}$.
  2) The secret key for pattern $\mathcal{P}$ is $\mathsf{sk}_\mathcal{P} = (a, \{b_i\}_{i \in \mathcal{I}})$.

  Anyone owning this secret key can create a new key $\mathsf{sk}_{\mathcal{P}'}$ for the pattern $\mathcal{P}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_\ell) \in_* \mathcal{P}$ as follows:
  1) $\mathcal{P}' \in_* \mathcal{P}$ indicates that $\mathcal{I} \in \mathcal{I}'$. For all $i \in \mathcal{I}$, select $r_i \in \mathbb{Z}_p$ and calculate $b'_i \leftarrow b_i \cdot g^{r_i}$; for $i \in \mathcal{I}' \backslash \mathcal{I}$, select $r_i \in \mathbb{Z}_p$ and compute $b'_i \leftarrow g^{r_i}$. Besides, calculate $a' \leftarrow a \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}')} \mathcal{H}_i(\mathcal{P}'_i)^{r_i}$.
  2) The secret key for the pattern $\mathcal{P}'$ is $\mathsf{sk}_{\mathcal{P}'} = (a', \{b'_i\}_{i \in \mathcal{I}'})$.
- **Sign** $(\mathsf{sk}_\mathcal{P}, m, \mathcal{P}^*)$: On the input of a message $m \in \{0,1\}^*$, the secret key $\mathsf{sk}_\mathcal{P}$, a pattern policy $\mathcal{P}^* = (\mathcal{P}^*_1, \ldots, \mathcal{P}^*_\ell)$ such that $\mathcal{P}^* \in_* \mathcal{P}$, the signer implements the following steps:
  1) Let $\mathcal{I}^* = \overline{\mathcal{W}}(\mathcal{P}^*)$, where $\overline{\mathcal{W}}(\mathcal{P}^*)$ refers to the complementary set containing all non-wildcard indices in $\mathcal{P}^*$. For all $i \in \mathcal{I}^*$, select $r_i \in \mathbb{Z}_p$ and calculate $b^*_i \leftarrow b_i \cdot g^{r_i}$; for $i \in \mathcal{I}^* \setminus \mathcal{I}$, select $r_i \in \mathbb{Z}_p$ and compute $b^*_i \leftarrow g^{r_i}$. Besides, calculate $a^* \leftarrow a \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} \mathcal{H}_i(\mathcal{P}^*_i)^{r_i}$. Here notice that $(a^*, \{b^*_i\}_{i \in \mathcal{I}^*})$ is well-formed.
  2) Pick $s \in \mathbb{Z}_p$ and compute $x = h_0^s$.
  3) Calculate $t = \mathcal{H}(m, x)$.
  4) Repeat Steps 2) & 3) such that the event $s + t = 0$ is forbidden to occur.
  5) For $i \in \mathcal{I}^*$, compute $y_i = (b^*_i)^{s+t}$.
  6) Compute $z = (a^*)^{s+t}$.
  7) Output the signature $\sigma = (x, \{y_i\}_{i \in \mathcal{I}^*}, z)$ with the picked pattern policy $\mathcal{P}^*$ and signed message m.
- **Verify** $(m, (\sigma, \mathcal{P}^*), \mathsf{pp})$: On the input of a message m, a valid signature σ with its policy $\mathcal{P}^*$ and the public parameter pp, the verifier implements the following steps to determine the validity of the given signature:
  1) Perform the calculation $t = \mathcal{H}(m, x)$.
  2) Output "1" if $e(z, g) = e\left(g_1, h_0^t \cdot x \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} y_i^{\mathcal{P}^*_i}\right) \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} e(y_i, h_i)$ holds; otherwise return an abortion symbol "$\perp$".

Remark 1: The signer uses his/her screct key $\mathsf{sk}_\mathcal{P}$ and combines the selected pattern policy $\mathcal{P}^*$ to produce the signature σ, where $\mathcal{P}^* \in_* \mathcal{P}$. As defined in $\mathcal{I} = \overline{\mathcal{W}}(\mathcal{P})$ and $\mathcal{I}^* = \overline{\mathcal{W}}(\mathcal{P}^*)$, it is easily learned $\mathcal{I} \in \mathcal{I}^*$. In other words, the adversary can only obtain the attributes of pattern policy but fails to truly learn the attributes the signer owns, thus realizing the attribute privacy of the signer.

VI. Efficient FS-HDS Construction

A. Basic Ideas and Technical Overview

At first glance, forward security can be easily realized in existing HDS schemes via TFS or PS technologies. However, potential issues that could emerge from integrating TFS or PS into existing HDS solutions will be outlined: (I) incapability of resolving inefficiencies, the reason originates from the inefficiencies of existing HDS methodologies; (II) the security uncertainties of devised FS-HDS or the occur of various

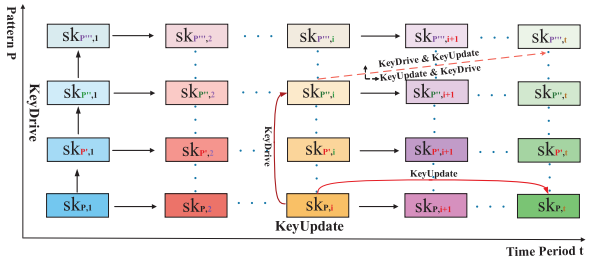


Fig. 2. Roadmap for KeyUpdate.

security vulnerabilities (*e.g.*, collusion attacks), the reason leading to this is the significant difference of the hardness assumptions the distinct cryptographic solutions depend on; (III) the failure of delegation realization due to the structure destruction of secret key. Hence, the challenge of devising an efficient FS-HDS lies in achieving the same level of security as the TFS approach while ensuring that the secret key's structure is not undermined. In this paper, we also design an efficient privacy-preserving forward-secure HDS (FS-HDS) based on our HDS, which overcomes the technical challenges (I)-(III). In more detail, we avail of the (structured) TFS to realize the structure-preserving secret key of our FS-HDS, thus ensuring the well-formed property of a secret key. Besides, since the hardness assumption TFS scheme depends on is essentially the extension of the one our efficient HDS relies on, the security and efficiency of our FS-HDS can be guaranteed. Our FS-HDS not only supports resilient key delegation of secret key from $\mathsf{sk}_{\mathsf{P},\mathbf{i}}$ to $\mathsf{sk}_{\mathsf{P}''',\mathbf{i}}$ if $\mathsf{P} \subset \mathsf{P}'''$, but also enables more flexible and fast-forward key update of the secret key $\mathsf{sk}_{\mathsf{P},\mathbf{i}}$ for the time period to $\mathsf{sk}_{\mathsf{P},\mathbf{t}}$ for any time $\mathbf{t} \geq \mathbf{i}$. Via the above two flexible processing ways (i.e., KeyDerive-then-KeyUpdate or KeyUpdate-then-KeyDerive), the secret key $\mathsf{sk}_{\mathsf{P}'',\mathbf{i}}$ can be easily updated to the secret key $\mathsf{sk}_{\mathsf{P}''',\mathbf{i}}$. The technical roadmap realizing the above process has been shown in Fig. 2. To summarize, the KeyDerive-then-KeyUpdate solution is advantageous in structured, privilege-based hierarchies where initial delegation is critical before updating by time, while the KeyUpdate-then-KeyDerive one is more flexible and efficient in environments with frequent, time-sensitive key updates followed by self-sovereign delegation.

B. Concrete FS-HDS Construction

- **Setup** $(\lambda, \ell, \mathcal{T})$: On the input of a security parameter λ, the maximum number ℓ of levels, the maximum number time slots $\mathcal{T} = 2^{\mathcal{L}} - 1$, the root identity performs the following executions:
  1) Select random generators $g, h_0, \ldots, h_\ell, u_0, \ldots, u_{\mathcal{L}} \in \mathbb{G}_0$ and $\tau \in \mathbb{Z}_p$ and compute $g_1 = g^\tau$.
  2) Select a hash function $\mathcal{H}$ that is defined as $\mathcal{H} : \{0,1\}^* \times \mathbb{G}_0 \to \mathbb{Z}_p$.
  3) Publish the public parameter $\mathsf{pp} = (g, g_1, h_0, \ldots, h_\ell, u_0, \ldots, u_{\mathcal{L}}, \mathcal{H})$ and keep the master secret key $\mathsf{msk} = h_0^\tau$ secret.
- **KeyDerive** $(\mathsf{msk}, \mathsf{pp}, \mathcal{P}, \mathbf{t})$: On the input of the master secret key msk, the public parameters pp, a pattern $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_\ell)$, the time period $\mathbf{t}$, the root proceeds with the following executions to compute initial secret key $\mathsf{sk}_{\mathcal{P},1} \leftarrow \widehat{\mathsf{sk}}_{\mathcal{P},\varepsilon}$ for the first time period $\mathbf{t} = \varepsilon$:
  1) Let $\mathcal{I} = \overline{\mathcal{W}}(\mathcal{P})$. For all $i \in \mathcal{I}$, pick $r_i, r \in \mathbb{Z}_p$ and let $b_i \leftarrow g^{r_i}$, $a \leftarrow h_0^\tau \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} \cdot u_0^r$, where $\mathcal{H}_i(x)$ is set as $\mathcal{H}_i(x_i) = g_1^x \cdot h_i = (g^\tau)^{x_i} \cdot h_i$. Besides, it also computes $d = g^r$, $e_k = u_k^r$ for $k \in [1, \mathcal{L}]$.
  2) The initial secret key for pattern $\mathcal{P}$ is $\widehat{\mathsf{sk}}_{\mathcal{P},\varepsilon} = (a, \{b_i\}_{i \in \mathcal{I}}, d, \{e_i\}_{i \in [1,\mathcal{L}]})$.

  Anyone owning the secret key $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$ can create a new key $\mathsf{sk}_{\mathcal{P}',\mathbf{t}}$ for the pattern $\mathcal{P}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_\ell) \in_* \mathcal{P}$ as follows:
  1) $\mathcal{P}' \in_* \mathcal{P}$ indicates that $\mathcal{I} \in \mathcal{I}'$. For all $i \in \mathcal{I}$, select $r_i \in \mathbb{Z}_p$ and calculate $b'_i \leftarrow b_i \cdot g^{r_i}$; for $i \in \mathcal{I}' \backslash \mathcal{I}$, select $r_i \in \mathbb{Z}_p$ and compute $b'_i \leftarrow g^{r_i}$. Besides, calculate $a' \leftarrow a \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}')} \mathcal{H}_i(\mathcal{P}'_i)^{r_i}$, $d' = d$ and $e'_i = e_i$. The final format of a delegated key is $a' \leftarrow h_0^\tau \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}')} \mathcal{H}_i(\mathcal{P}'_i)^{r_i} \cdot \left(u_0 \prod_{j=1}^{k} u_j^{t_j}\right)^r$, $b'_i = g^{r_i}$, $d' = g^r$ and $e'_i = u_i^r$.
  2) The delegable key for the pattern $\mathcal{P}'$ is $\mathsf{sk}_{\mathcal{P}',\mathbf{t}} \leftarrow \widehat{\mathsf{sk}}_{\mathcal{P}',\mathbf{t}} = (a', \{b'_i\}_{i \in \mathcal{I}'}, d', \{e'_i\}_{i \in [k+1,\mathcal{L}]})$, where $\mathbf{t} \in \{1,2\}^k$.

  It is worth noting that the initial delegable key for the pattern $\mathcal{P}'$ is $\mathsf{sk}_{\mathcal{P}',1} \leftarrow \widehat{\mathsf{sk}}_{\mathcal{P}',\varepsilon} = (a', \{b'_i\}_{i \in \mathcal{I}'}, d', \{e'_i\}_{i \in [1,\mathcal{L}]})$.
- **KeyUpdate** $(\mathsf{sk}_{\mathcal{P},\mathbf{t}})$: On the input of the secret key $\mathsf{sk}_{\mathcal{P},t}$ with its current time period $\mathbf{t}$, it computes the follows steps to update its secret key $\mathsf{sk}_{\mathcal{P},t}$ for the time period t to $\mathsf{sk}_{\mathcal{P},t+1}$ for the next time period:
  1) Each $\mathbf{w} \in \{1,2\}^k$ is associated with a secret key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}}$ of the form as

$$\begin{aligned}\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}} &= (a, \{b_i\}_{i \in \mathcal{I}}, d, \{e_i\}_{i \in [k+1,\mathcal{L}]}) \\ &= \Big(h_0^\tau \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} \cdot \left(u_0 \prod_{j=1}^{k} u_j^{w_j}\right)^r, \\ &\quad g^{r_1}, \ldots, g^{r_i}, g^r, u_{k+1}^r, \ldots, u_{\mathcal{L}}^r\Big), \end{aligned} \tag{1}$$

  for $r \in \mathbb{Z}_p$. Given $\mathsf{sk}_{\mathcal{P},\mathbf{w}}$, one can compute a new key for $\mathsf{sk}_{\mathcal{P},\mathbf{w}'}$ any $\mathbf{w}' \in \{1,2\}^{k'}$ that contains $\mathbf{w}$ as a prefix as follows:

$$\begin{aligned}\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'} &= \left(\hat{a}', \{\hat{b}'_i\}_{i \in \mathcal{I}}, \hat{d}', \{\hat{e}'_i\}_{i \in \left[k'+1,\mathcal{L}\right]}\right) \\ &= \Big(a \cdot \prod_{j=k+1}^{k'} e_j^{w_j} \cdot \left(u_0 \prod_{j=i}^{k'} u_i^{w_i}\right)^{r'}, g^{r_1}, \ldots, \\ &\quad g^{r_i}, d \cdot d^{r'}, e_{k'+1} \cdot u_{k'+1}^{r'}, \ldots, e_{\mathcal{L}} \cdot u_{\mathcal{L}}^{r'}\Big), \end{aligned} \tag{2}$$

  for $r' \in \mathbb{Z}_p$. Hence, the secret key $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$ at time period $\mathbf{t}$ is given by $\mathsf{sk}_{\mathcal{P},\mathbf{t}} = \{\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}} : \mathbf{w} \in \Psi_t\}$. Based on the first property, this secret key contains a secret key $\mathsf{sk}_{\mathcal{P},\mathbf{w}}$ for a prefix $\mathbf{w}$ of all nodes $\mathbf{t}' \geq \mathbf{t}$.
  2) To proceed with the update of $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$ to $\mathsf{sk}_{\mathcal{P},\mathbf{t}+1}$, the signer uses the second property to update its secret key, that is, if $|\mathbf{t}| \leq \mathcal{L} - 1$, then the signer looks up $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}} = (\hat{a}', \{\hat{b}'_i\}_{i \in \mathcal{I}}, \hat{d}', \{\hat{e}'_i\}_{i \in [|\mathbf{t}|+1,\mathcal{L}]}) \in \mathsf{sk}_{\mathcal{P},\mathbf{t}}$ and calculates $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}\|1} = (\hat{a}' \cdot \hat{e}'_{|\mathbf{t}|+1}, \{\hat{b}'_i\}_{i \in \mathcal{I}}, \hat{d}', \{\hat{e}'_i\}_{i \in [|\mathbf{t}|+2,\mathcal{L}]}) \in \mathsf{sk}_{\mathcal{P},\mathbf{t}}$. Besides, the signer computes $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}\|2}$ from $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$ with the Equation (2). Finally, the signer sets

$\mathsf{sk}_{\mathcal{P},\mathbf{t}+1} = (\mathsf{sk}_{\mathcal{P},\mathbf{t}} \setminus \widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}}) \cup (\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}\|1}, \widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}\|2})$ and deletes $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$. If $|\mathbf{t}| = \mathcal{L}-1$, the signer directly sets $\mathsf{sk}_{\mathcal{P},\mathbf{t}+1} = (\mathsf{sk}_{\mathcal{P},\mathbf{t}} \setminus \widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}})$ and deletes $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$.

Remark 2: To enable more flexible and fast-forward key update of the secret key $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$ for the time period to $\mathsf{sk}_{\mathcal{P},\mathbf{t}'}$ for any time $t' \geq t$, the new secret key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'}$ for all nodes $\mathbf{w}' \in \Phi_{\mathbf{t}'} \setminus \Phi_{\mathbf{t}}$ of the signer can be also derived by using the Equation (2) to derive the key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}} \in \widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{t}}$, such that $\mathbf{w}'$ contains $\mathbf{w}$, which must hold due to the third property of $\Phi_{\mathbf{t}}$. Then the signer can set $\mathsf{sk}_{\mathcal{P},\mathbf{t}'} \leftarrow \{\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'} : \mathbf{w}' \in \Phi_{\mathbf{t}'}\}$ and delete $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$.

- **Sign** $(\mathsf{sk}_{\mathcal{P},\mathbf{t}}, m, \mathcal{P}^*, \mathbf{t})$: On the input of a message $m \in \{0,1\}^*$, the secret key $\mathsf{sk}_{\mathcal{P},\mathbf{t}}$, a pattern policy $\mathcal{P}^* = (\mathcal{P}_1^*, \ldots, \mathcal{P}_\ell^*)$ such that $\mathcal{P} \in_* \mathcal{P}^*$ and a time period $\mathbf{t} \in \{1,2\}^{\leq \mathcal{L}-1}$, the signer implements the following steps:
  1) Let $\mathcal{I}^* = \overline{\mathcal{W}}(\mathcal{P}^*)$. For all $i \in \mathcal{I}^*$, select $r_i \in \mathbb{Z}_p$ and calculate $b_i^* \leftarrow b_i \cdot g^{r_i}$; for $i \in \mathcal{I}^* \setminus \mathcal{I}$, select $r_i \in \mathbb{Z}_p$ and compute $b_i^* \leftarrow g^{r_i}$. Besides, calculate $a^* \leftarrow a \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} \mathcal{H}_i(\mathcal{P}_i^*)^{r_i}$. Besides, set $d^* = d$, $e_j^* = e_j$ for $j \in [|\mathbf{t}|+1, \mathcal{L}]$. Here notice that $(a^*, \{b_i^*\}_{i \in \mathcal{I}^*}, d^*, \{e_j^*\}_{j \in [|\mathbf{t}|+1,\mathcal{L}]})$ is well-formed.
  2) Pick $s \in \mathbb{Z}_p$ and compute $x = h_0^s$.
  3) Calculate $\varphi = \mathcal{H}(m, x)$.
  4) Repeat Steps 2) & 3) such that the event $s + \varphi = 0$ is forbidden to occur.
  5) For $i \in \mathcal{I}^*$, compute $y_i = (b_i^*)^{s+\varphi}$.
  6) Select $r' \in \mathbb{Z}_p$ and compute $f = (d^* \cdot g^{r'})^{s+\varphi}$ and $z = \left(a^* \cdot \left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}\right)^{r'}\right)^{s+\varphi}$.
  7) Output the signature $\sigma = (x, \{y_i\}_{i \in \mathcal{I}^*}, f, z)$ with the picked pattern policy $\mathcal{P}^*$.
- **Verify** $(m, (\sigma, \mathcal{P}^*), \mathsf{pp}, \mathbf{t})$: On the input of a message m, a valid signature σ with its policy $\mathcal{P}^*$, the public parameter pp and the time period $\mathbf{t}$, the verifier implements the following steps to determine the validity of the given signature:
  1) Perform the calculation $\varphi = \mathcal{H}(m, x)$.
  2) Output "1" if $e(z, g) = e\left(g_1, h_0^\varphi \cdot x \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} y_i^{\mathcal{P}_i^*}\right) \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} e(y_i, h_i) \cdot e\left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}, f\right)$ holds; otherwise return an abortion symbol "$\perp$".

Remark 3: Although our FS-HDS achieves the key exposure resistance via the realization of forward security, our FS-HDS does not consider the privilege revocation. Realize the privilege revocation in our FS-HDS is actually not difficult by introducing the KUNode algorithm [50] to our FS-HDS.

VII. Correctness and Security Proof of FS-HDS Scheme

Theorem 1: The signature can be publicly verified by anyone if the signature is validly produced with a valid secret key.

Proof: Anyone can perform the following calculations to check the signature validity of our FS-HDS scheme:

$$
\begin{aligned}
& e\left(g_1, h_0^\varphi \cdot x \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} y_i^{\mathcal{P}_i^*}\right) \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} e(y_i, h_i) e\left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}, f\right) \\
&= e\left(g^\tau, (h_0)^{s+\varphi} \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} (b_i^*)^{(s+\varphi)\mathcal{P}_i^*}\right) \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} e(g^{r_i^*}, h_i^{s+\varphi}) \\
&\quad \cdot e\left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}, g^{(r+r')(s+\varphi)}\right) \\
&= \left[e\left(g^\tau, h_0 \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} (b_i^*)^{\mathcal{P}_i^*}\right) \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} e(g^{r_i^*}, h_i) \right. \\
&\quad \left. \cdot e\left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}, g^{r+r'}\right)\right]^{s+\varphi} \\
&= \left[e\left(g, h_0^\tau \cdot g^{\tau \sum_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} r_i^* \mathcal{P}_i^*} \left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}\right)^{r+r'}\right) \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} \right. \\
&\quad \left. \times e(g^{r_i^*}, h_i)\right]^{s+\varphi} \\
&= \left[e\left(g, h_0^\tau \cdot g_1^{\sum_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} r_i^* \mathcal{P}_i^*} \cdot \left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}\right)^{r+r'}\right) \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)}^{\mathcal{L}} \right. \\
&\quad \left. \times e(g, h_i^{r_i^*})\right]^{s+\varphi} \\
&= \left[e\left(g, h_0^\tau \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} (g_1^{\mathcal{P}_i^*} h_i)^{r_i^*}\right) \cdot \left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}\right)^{r+r'}\right]^{s+\varphi} \\
&= \left[e\left(g, h_0^\tau \cdot \prod_{i \in \overline{\mathcal{W}}(\mathcal{P}^*)} \mathcal{H}_i(\mathcal{P}_i^*)^{r_i^*}\right) \cdot \left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}\right)^{r+r'}\right]^{s+\varphi} \\
&= e\left(g, \left(a^* \cdot \left(u_0 \prod_{j=1}^{|\mathbf{t}|} u_j^{t_j}\right)^{r'}\right)^{s+\varphi}\right) = e(g, z).
\end{aligned}
$$

If the above equation holds, the signature σ is valid. ∎

Theorem 2: If the CDHE assumption holds, our FS-HDS is forward secure against existential forgery under selective pattern attacks and adaptive chosen-pattern-and-message attacks.

Proof: It is assuming that there is an adversary $\mathcal{A}$ that has an advantage ϵ in successfully attacking our scheme, then we can present that an algorithm $\mathcal{C}$ can be constructed to solve the CDHE problem. In other words, given $(A_0 = g, A_1 = g^\alpha, g^{\alpha^2}, \ldots, A_{\mathcal{L}} = g^{\alpha^{\mathcal{L}}})$, $\mathcal{C}$ is capable of computing $g^{\alpha^{\mathcal{L}+1}}$. In this proof, we let $g_1 = g^\alpha$ and $h_0 = A_{\mathcal{L}} = g^{\alpha^{\mathcal{L}}}$. The selective pattern game is simulated via the interaction between $\mathcal{C}$ and $\mathcal{A}$ in the following:

- **Init**: When the game starts, $\mathcal{A}$ first gives the challenge pattern $\mathcal{P}^* = (\mathcal{P}_1^*, \ldots, \mathcal{P}_\ell^*)$ and the challenge time period $\mathbf{t}^*$ to $\mathcal{C}$. Let $\mathbf{w}^* \in \{0,1,2\}^{\mathcal{L}-1}$ such that $\mathbf{w}^* = w_1^* \| \ldots \| w_{\mathcal{L}-1}^* = \mathbf{t}^* \| 0^{\mathcal{L}-1-|\mathbf{t}^*|}$.

- **Setup**: To produce the system public parameters, $\mathcal{C}$ first randomly selects $\alpha_1, \ldots, \alpha_\ell \in \mathbb{Z}_p$, defines $h_i = g_1^{-\mathcal{P}_i^*} g^{\alpha_i}$ for all $i \in [1, \ell]$. Also, it chooses $\gamma_0, \ldots, \gamma_{\mathcal{L}}$ and sets $u_0 = g^{\gamma_0} \prod_{j=1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}$ and $u_j = g^{\gamma_j} A_{\mathcal{L}-j+1}$ for $j \in [1, \mathcal{L}]$. Besides, the functions $\mathcal{H}_i(.)$ is defined as the functions $\mathcal{H}_i(x) = g_1^x h_i = (g^\tau)^{x-\mathcal{P}_i^*} g^{\alpha_i}$. Finally, the system public parameter $\mathsf{pp} = (g, g_1, h_0, h_1, \ldots, h_\ell, u_1, \ldots, u_{\mathcal{L}})$ is sent to $\mathcal{A}$. Note that the master secret key $h_0^\tau = g^{\alpha^{\mathcal{L}+1}}$ is unknown to $\mathcal{C}$.
- **Key Update**: $\mathcal{C}$ does not need to simulate anything other than tracking the current time period $\mathbf{t}$.
- **Signing Phase**: In this phase, we first describe how to respond to the signing query for the policy $\mathcal{P}$ in time period $\mathbf{t} \neq \mathbf{t}^*$. Then, describe the case $\mathbf{t} = \mathbf{t}^*$ and $\mathcal{P} \not\models \mathcal{P}^*$. In the following, the above queries are simulated:
  - Case I: $\mathbf{t} \neq \mathbf{t}^*$ and $\mathcal{P} \not\models \mathcal{P}^*$: It is not hard to learn that $\mathbf{w} \neq \mathbf{w}^*$ and $\mathcal{P}_i \neq \mathcal{P}_i^*$. Then let $\mathbf{w}' = w_1 \| \ldots \| w_k$ represent the shortest prefix of $\mathbf{w}$ that is not $\mathbf{w}^*$. We describe how $\mathcal{C}$ can derive a valid key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'}$. Based on it, we can easily get both $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}}$ and a signature for $\mathbf{t}$. Recall that $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'}$ has the structure

$$\begin{aligned}\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}} &= (a, \{b_i\}_{i\in\mathcal{I}}, d, \{e_i\}_{i\in[k+1,\mathcal{L}]}) \\ &= \Big(h_0^\tau \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} \cdot \Big(u_0 \prod_{j=1}^{k} u_j^{w_j}\Big)^{r}, \\ &\quad g^{r_1}, \ldots, g^{r_i}, g^{r}, u_{k+1}^{r}, \ldots, u_{\mathcal{L}}^{r}\Big)\end{aligned}$$

    for uniformly distributed values of $r, r_1, \ldots, \{r_i\}_{i\in\mathcal{I}}$. Focusing on the first part a first, we can get that

$$\begin{aligned}a &= h_0^\tau \cdot \Big(u_0 \prod_{j=1}^{k} u_j^{w_j}\Big)^{r} \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} \\ &= A_{\mathcal{L}}^{\alpha} \cdot \Big(\Big(g^{\gamma_0} \prod_{j=1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\Big) \cdot \Big(\prod_{j=1}^{k} (g^{\gamma_j} A_{\mathcal{L}-j+1})^{w_i}\Big)\Big)^{r} \\ &\quad \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} \\ &= g^{\alpha^{\mathcal{L}+1}} \cdot \Big(g^{\gamma_0+\sum_{j=1}^{k}\gamma_j w_j} \cdot A_{\mathcal{L}-k+1}^{w_k-w_k^*} \cdot \prod_{j=k+1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\Big)^{r} \\ &\quad \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} ((g^{\alpha})^{\mathcal{P}_i-\mathcal{P}_i^*} g^{\alpha_i})^{r_i},\end{aligned}$$

    where the above equation holds due to the fact $w_j = w_j^*$ for $1 \leq j < k$ and $w_k \neq w_k^*$. Let us denote the three factors in $\left(g^{\gamma_0+\sum_{j=1}^{k}\gamma_j w_j} \cdot A_{\mathcal{L}-k+1}^{w_k-w_k^*} \cdot \prod_{j=k+1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\right)$ as E_1, E_2, E_3 and let $E = \prod_{i=1}^{3} E_i$. If we set $r = r + \frac{\alpha^k}{w_k^*-w_k} \mod q$ for a random $r' \in \mathbb{Z}_p$, then we can get

$$a = g^{\alpha^{\mathcal{L}+1}} \cdot E^{r'} \cdot E^{\frac{\alpha^k}{w_k^*-w_k}} \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i}.$$

    In more detail, $E^{\frac{\alpha^k}{w_k^*-w_k}}$ can be computed as the products of:

$$\begin{aligned}E_1^{\frac{\alpha^k}{w_k^*-w_k}} &= A_k^{\frac{\gamma_0+\sum_{j=1}^{k}\gamma_j w_j}{w_k^*-w_k}}, \\ E_2^{\frac{\alpha^k}{w_k^*-w_k}} &= A_{\mathcal{L}-k+1}^{-\alpha^k} = g^{-\alpha^{\mathcal{L}+1}}, \\ E_3^{\frac{\alpha^k}{w_k^*-w_k}} &= \prod_{j=k+1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{\frac{-w_j^*}{w_k^*-w_k}} = \prod_{j=0}^{\mathcal{L}-k-2} A_{\mathcal{L}-j}^{\frac{-w_{k+1+j}^*}{w_k^*-w_k}}.\end{aligned}$$

    As described above, it is straightforward to see that a can be computed for $1 \leq k \leq \mathcal{L}-1$. The other key components can be efficiently computed as:

$$\begin{aligned}b_i &= g^{r_i}, d = g^{r'} A_k^{\frac{1}{w_k^*-w_k}}, \\ e_j &= u_i^{r'} \cdot A_{\mathcal{L}+k-j+1} = u_{k+j+1}^{r'} \cdot A_{\mathcal{L}-j},\end{aligned}$$

    where $i \in \mathcal{I}$ and $j = 0, \ldots, \mathcal{L}-k-1$.
    From the key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'} = (a, \{b_i\}_{i\in\mathcal{I}}, d, \{e_i\}_{i\in[k+1,\mathcal{L}]})$ for $\mathbf{w}'$, $\mathcal{C}$ can produce a new key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}}$ for $\mathbf{w}$ and create a signature as that in the real signing algorithm.
  - Case II: $\mathbf{t} \neq \mathbf{t}^*$ and $\mathcal{P} \models \mathcal{P}^*$: This case is almost identical to that in Case I except the part of $\prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} = \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} g^{\alpha_i r_i}$. That is,

$$\begin{aligned}a &= h_0^\tau \cdot \Big(u_0 \prod_{j=1}^{k} u_j^{w_j}\Big)^{r} \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} \mathcal{H}_i(\mathcal{P}_i)^{r_i} \\ &= A_{\mathcal{L}}^{\alpha} \cdot \Big(\Big(g^{\gamma_0} \prod_{j=1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\Big) \cdot \Big(\prod_{j=1}^{k} (g^{\gamma_j} A_{\mathcal{L}-j+1})^{w_i}\Big)\Big)^{r} \\ &\quad \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} g^{\alpha_i r_i} \\ &= g^{\alpha^{\mathcal{L}+1}} \cdot \Big(g^{\gamma_0+\sum_{j=1}^{k}\gamma_j w_j} \cdot A_{\mathcal{L}-k+1}^{w_k-w_k^*} \cdot \prod_{j=k+1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\Big)^{r} \\ &\quad \cdot \prod_{i\in\overline{\mathcal{W}}(\mathcal{P})} g^{\alpha_i r_i}.\end{aligned}$$

    As illustrated as those in Case I, here we omit the above process. Hence, it is easy to conclude that the key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}'} = (a, \{b_i\}_{i\in\mathcal{I}}, d, \{e_i\}_{i\in[k+1,\mathcal{L}]})$ for $\mathbf{w}'$ can be produced. Based on this key, $\mathcal{C}$ can generate a new key $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}}$ for $\mathbf{w}$ and create a signature as that in the real signing algorithm.
  - Case III: $\mathbf{t} = \mathbf{t}^*$ and $\mathcal{P} \not\models \mathcal{P}^*$: For the signing key query with $\mathbf{t} = \mathbf{t}^*$ and $\mathcal{P} \not\models \mathcal{P}^*$, $\mathcal{C}$ can produce the secret key using a similar way as above, apart from the fact that $\mathcal{P}_i \neq \mathcal{P}_i^*$ instead of $w_k \neq w_k^*$. Namely, letting $\mathbf{w} = \mathbf{t} \| 0^{\mathcal{L}-1-|\mathbf{t}|}$, if we set $\widetilde{r_i} = r_i - \frac{\alpha^{\mathcal{L}}}{\mathcal{P}_i-\mathcal{P}_i^*}$, $\mathcal{C}$ can produce the signing key as

$$\begin{aligned}a &= h_0^\tau \cdot \Big(u_0 \prod_{j=1}^{k} u_j^{w_j}\Big)^{r} \cdot \mathcal{H}_i(\mathcal{P}_i)^{r_i} \cdot h_0^{\frac{-\alpha_i}{\mathcal{P}_i-\mathcal{P}_i^*}} \\ &= A_{\mathcal{L}}^{\alpha} \cdot \Big(\Big(g^{\gamma_0} \prod_{j=1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\Big) \cdot \Big(\prod_{j=1}^{\mathcal{L}-1} (g^{\gamma_j} A_{\mathcal{L}-j+1})^{w_i}\Big)\Big)^{r}\end{aligned}$$

$$\cdot h_0^{\frac{-\alpha_i}{\mathcal{P}_i-\mathcal{P}_i^*}} \cdot ((g^\alpha)^{\mathcal{P}_i-\mathcal{P}_i^*} g^{\alpha_i})^{r_i}$$
$$= g^{\alpha \mathcal{L}+1} \cdot \left(g^{\gamma_0+\sum_{j=1}^k \gamma_j w_j}\right)^r \cdot h_0^{\frac{-\alpha_i}{\mathcal{P}_i-\mathcal{P}_i^*}} (g_1^{\mathcal{P}_i-\mathcal{P}_i^*} g^{\alpha_i})^{r_i-\frac{\alpha \mathcal{L}}{\mathcal{P}_i-\mathcal{P}_i^*}} \cdot g^{\frac{\alpha \mathcal{L} \alpha_i}{\mathcal{P}_i-\mathcal{P}_i^*}}$$
$$= g^{\alpha \mathcal{L}+1} \cdot \left(g^{\gamma_0+\sum_{j=1}^k \gamma_j w_j}\right)^r \cdot \mathcal{H}_i(\mathcal{P}_i)^{\widetilde{r}_i}$$
$$= h_0^\alpha \cdot \left(g^{\gamma_0+\sum_{j=1}^k \gamma_j w_j}\right)^r \cdot \mathcal{H}_i(\mathcal{P}_i)^{\widetilde{r}_i}.$$

Hence, the secret key defined above for the pattern with the time period $\mathbf{t}$ is $\mathsf{sk}_{\mathcal{P}} = \left(a = h_0^\alpha \cdot \left(u_0 \prod_{j=1}^k u_j^{w_j}\right)^r \cdot \prod_{v \in \overline{\mathcal{W}}(\mathcal{P}) \setminus i} (\mathcal{H}_v(\mathcal{P}_v)^{r_v}) \cdot \mathcal{H}_i(\mathcal{P}_i)^{\widetilde{r}_i}, b_1 = g^{r_1}, \ldots, b_{i-1} = g^{r_{i-1}}, b_i = g^{\widetilde{r}_i}, \ldots, b_\ell = g^{r_v}\right)$, $d = g^r$, $e_{k+1} = u_{k+1}^r, \ldots, e_{\mathcal{L}} = u_{\mathcal{L}}^r$ where $r, r_1, \ldots, r_{i-1}, \widetilde{r}_i, \ldots, r_v$ are uniform in $\mathbb{Z}_p$. Therefore, $\mathcal{C}$ can generate a valid secret key of the queried pattern in time periods $\mathbf{t}$ for $\mathcal{A}$.

- *Hashing*: A list L storing the answers of the hash oracle is maintained by $\mathcal{C}$, which takes charges of checking the list L for responding to the $\mathcal{A}$'s queries on the hash value of (m, x). If the value for this query can be checked, the value as the answer is given to $\mathcal{A}$; otherwise, $\mathcal{C}$ randomly picks $\varphi \in \mathbb{Z}_p$, inserts the entry (x, m, φ) into the list and returns it to $\mathcal{A}$.
- *Signing for selected pattern and time period*: For the pattern $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_\ell)$ and $\mathbf{t}$. That is, $\mathcal{P}$ either equals to $\mathcal{P}^*$ or is the subset of $\mathcal{P}^*$ and $\mathbf{t} = \mathbf{t}^*$, $\mathcal{C}$ randomly selects $\varphi, r_0 \in \mathbb{Z}_p$ and computes $x = h_0^{-\varphi} g^{r_0}$ until no entry (m, x, φ) where $\varphi^* \neq -\varphi$ is checked. Then, $\mathcal{C}$ randomly chooses $r, r_1, \ldots, r_u \in \mathbb{Z}_p$ and calculates $f = g^r, y_j = {h_0}^{r_j}$, where $j \in [1, u]$, $u = |\mathcal{I}| = |\overline{\mathcal{W}}(\mathcal{P})| \leq \ell$. In addition, $\mathcal{C}$ computes $z = g_1^{r_0} \cdot \prod_{j=1}^u h_0^{r_j \alpha_j} \cdot g^{r\left(\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i\right)}$. Therefore, it can be inferred that $(x, y_1, \ldots, y_u, z)$ is a valid signature in the following:

$$e\left(g_1, h_0^\varphi \cdot x \cdot \prod_{j=1}^u y_j^{\mathcal{P}_j}\right) \cdot \prod_{j=1}^u e(y_j, h_j) \cdot e\left(u_0 \prod_{j=1}^{\mathcal{L}-1} u_j^{t_j}, g^r\right)$$
$$= e\left(g^\alpha, h_0^\varphi \cdot h_0^{-\varphi} g^{r_0} \prod_{j=1}^u {h_0}^{r_j \mathcal{P}_j}\right) \prod_{j=1}^u e({h_0}^{r_j}, g_1^{-\mathcal{P}_j} g^{\alpha_j}) \times e\left(\left(g^{\gamma_0} \prod_{j=1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\right) \cdot \left(\prod_{j=1}^{\mathcal{L}-1} (g^{\gamma_j} A_{\mathcal{L}-j+1})^{w_i}\right), g^r\right)$$
$$= e\left(g, g^{\alpha r_0} \cdot \prod_{j=1}^u {h_0}^{\alpha r_j \mathcal{P}_j}\right) \cdot \prod_{j=1}^u e({h_0}^{r_j}, g_1^{-\mathcal{P}_j} g^{\alpha_j}) \times e\left(g, g^{r\left(\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i\right)}\right)$$
$$= e(g, g^{\alpha r_0}) \cdot e\left(g, \prod_{j=1}^u {h_0}^{\alpha r_j \mathcal{P}_j}\right) \cdot \prod_{j=1}^u e({h_0}^{r_j}, g_1^{-\mathcal{P}_j} g^{\alpha_j}) \times e\left(g, g^{r\left(\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i\right)}\right)$$
$$= e\left(g, g^{\alpha r_0} \cdot \prod_{j=1}^u {h_0}^{\alpha_j r_j} \cdot g^{r\left(\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i\right)}\right)$$
$$= e(g, z).$$

- *Signing for any other patterns and time periods*: For the pattern $\mathcal{P}$ which is not the subset of $\mathcal{P}^*$ and $\mathbf{t} \neq \mathbf{t}^*$, $\mathcal{C}$ creates the secret key of the pattern via the above simulations and refers to the signing algorithm to produce the signature on m.

- **Break in**: Here $\mathcal{C}$ requires to simulate $sk_{\mathcal{P},\bar{\mathbf{t}}}$, where $\bar{t} \geq \mathbf{t}$. This in turn needs to simulate $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}}$ for all $\mathbf{w} \in \Phi_{\bar{t}}$, due to the first property, all of $\mathbf{w}$ are neither the prefixes of $\mathbf{t}^*$ and the prefixes of $\mathbf{w}^*$, thus we can simulate $\widehat{\mathsf{sk}}_{\mathcal{P},\mathbf{w}}$ as before.
- **Forgery**: $\mathcal{A}$ outputs a forgery $(m^*, x^*, y_1^*, \ldots, y_v^*, f^*, z^*)$ such that $(x^*, y_1^*, \ldots, y_v^*, f^*, z^*)$ is a legitimate signature on m^* in time period $\mathbf{t}^*$, where $v = |\mathcal{I}^*| = |\overline{\mathcal{W}}(\mathcal{P}^*)| \leq \ell$.
  - Addressing CDHE problem via the forking lemma: Similar to that in the forking lemma, the CDHE problem can be solved by $\mathcal{C}$, which responds $\mathcal{A}$ with the same random value but distinct choices of $\mathcal{H}$ to derive two legitimate signatures $(x^*, y_1^*, \ldots, y_v^*, f^*, z^*)$ and $(x^*, {y_1'}^*, \ldots, {y_v'}^*, f'^*, z'^*)$. These two signatures in time period $\mathbf{t}^*$ are supposed to be valid signatures m^* concerning the hash functions $\mathcal{H}$ and $\mathcal{H}'$ having various values $\varphi \neq \varphi'$ on (m, x^*).
    For $i \in [1, v]$, since $y_i = b_i^{s+\varphi}$, $\mathcal{C}$ can compute $(y_i/y_i')^{(\varphi-\varphi')^{-1}}$ to capture b_i. Similarly, $\mathcal{C}$ can compute $(z/z')^{(\varphi-\varphi')^{-1}}$ and $(f/f')^{(\varphi-\varphi')^{-1}}$ to derive a and d. In the following, we prove that $(h_0)^\alpha = g^{\alpha \mathcal{L}+1} = a/\left(\prod_{i=1}^v b_i^{\alpha_i} \cdot d^{\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i^*}\right)$ holds:
$$a = h_0^\alpha \cdot \prod_{i=1}^v \mathcal{H}_i(\mathcal{P}_i^*)^{r_i} \cdot \left(u_0 \prod_{j=1}^{\mathcal{L}-1} u_j\right)^r$$
$$= h_0^\alpha \cdot \prod_{i=1}^v [(g^\tau)^{(\mathcal{P}_i^*)^{r_i}} (g_1^{-\mathcal{P}_i^*} g^{\alpha_i})^{r_i}] \cdot \left(g^{\gamma_0} \prod_{j=1}^{\mathcal{L}-1} A_{\mathcal{L}-j+1}^{-w_j^*}\right)^r \cdot \left(\prod_{j=1}^{\mathcal{L}-1} (g^{\gamma_j} A_{\mathcal{L}-j+1})^{w_i^*}\right)^r$$
$$= h_0^\alpha \cdot \prod_{i=1}^v g^{\alpha_i r_i} \cdot g^{r\left(\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i^*\right)}$$
$$= h_0^\alpha \cdot \prod_{i=1}^v b_i^{\alpha_i} \cdot d^{\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i^*}.$$
    Based on the above process, it is easily learned that $h_0^\alpha = g^{\alpha \mathcal{L}+1} = a/\left(\prod_{i=1}^v b_i^{\alpha_i} \cdot d^{\gamma_0+\sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i^*}\right)$ holds.
  - Solving CDHE problem without forking lemma: If we consider the following outsider attacks: *i.e.,* only adaptive chosen message attacks can be allowed to launch but the adaptive chosen pattern attacks are forbidden. To prove the security, only *the signing for any other pattern* simulation is required to modify such that the list L will store the selected s in the signing phase.

After the simulation, $\mathcal{A}$ outputs a forgery $(m^*, x^*, y_1^*, \ldots, y_v^*, z^*)$ such that $(x^*, y_1^*, \ldots, y_v^*, z^*)$ is a legitimate signature on m^* in time period $\mathbf{t}^*$. Here, it is worthwhile to mention that the probability of obtaining the value of (m^*, x^*) in time period $\mathbf{t}^*$ without asking is negligible, and $\mathcal{C}$ is likely to know the discrete logarithm of φ concerning h_0, where h_0 implicitly equals to $g^{\alpha^{\mathcal{L}+1}}$. For obtaining the value of φ, three approaches can be found from the simulations: (1) the signing oracle for chosen patterns; (2) the signing oracle for any other patterns; and (3) the hash oracle itself.

Iff when φ from the second approach as the query result is answered to $\mathcal{C}$, the discrete logarithm of h_0 regarding φ can be obtained. Under this attack model, $\mathcal{A}$ cannot produce the forged signature via the first approach. As a result, the probability of getting the discrete logarithm of h_0 regarding φ equals to $Q_{\mathcal{S}^*}/[\mathcal{T}(Q_{\mathcal{S}^*} + Q_{\mathcal{H}})]$, where $Q_{\mathcal{S}^*}$ and $Q_{\mathcal{H}}$ denote the number of signature requests for any other patterns, time periods and hashing, respectively. Due to the fact that the *Key Derivation and Update* queries are not permitted, the probability $Q_{\mathcal{S}^*}/(Q_{\mathcal{S}^*} + Q_{\mathcal{H}})$ is a non-negligible value.

In this case, $\mathcal{C}$ can derive (s, x^*) holding $h_0^s = x^*$ via checking (m^*, x^*) from the list L. Therefore, for $i \in [1, v]$, since $y_i = b_i^{s+\varphi}$, $\mathcal{C}$ can compute $y_i^{(s+\varphi)^{-1}}$ to capture b_i. Similarly, $\mathcal{C}$ can calculate $z^{(s+\varphi)^{-1}}$ and $f^{(s+\varphi)^{-1}}$ to derive a and d. As proved before, we can also calculate $(h_0)^\alpha = g^{\alpha^{\mathcal{L}+1}} = a/\left(\prod_{i=1}^{v} b_i^{\alpha_i} \cdot d^{\gamma_0 + \sum_{j=1}^{\mathcal{L}-1} \gamma_i w_i^*}\right)$. ■

Remark 4: We have also rendered the correctness, security definitions and proofs of our HDS (under CDH, DDH assumptions). Due to the limited space, please refer to Supplementary Material.

VIII. Performance Evaluation

This section begins with a theoretical analysis by comparing computational and communication costs. Subsequently, we assess the performance through experimental simulations to demonstrate the practicality of our solutions.

A. Theoretical Analysis

The computation costs (Comp.Cos) and communication costs (Comm.Cos) of related works are summarized in TABLE II, focusing primarily on the most time-intensive operations such as exponentiation and bilinear pairings. For the sake of comparison, we define p and e_0 as the time required for one bilinear pairing and one exponentiation in $\mathbb{G}_0/\mathbb{G}_1$, respectively. Given that different algorithms involve varying parameters, we also introduce the following definitions: $\mathcal{L}$: the maximum number of attributes allowed in the system or policy; $\ell(\ell')$: the maximum number of attributes in the (delegated) pattern or policy, (here $\ell \leq \mathcal{L}$); $\mathcal{L}'$: the depth of the time-encoded tree; a/a': the number of non-wildcards in the original/delegated pattern; c: the number of non-wildcards in the access policy; x: the number of non-wildcards in a secret key, which equals a or a' since the original or delegated secret key may be used in the signature algorithm; $|\Phi_{\mathbf{t}}|$: the length of a set containing the time $\mathbf{t}$ and all the "right-hand siblings" of nodes on the path from $\mathbf{t}$ to the root; k: the length of the string $\{1, 2\}^k$ indicating the time $\mathbf{t}$. In our comparisons, we mainly choose to compare our schemes exclusively with [22] and [24] since the most prominent unstructured scheme [22] realizes similar functionalities and the methodology [24] is the only structured one.

From TABLE II, we can easily conclude that our HDS is almost lower than [22], [24] and FS-HDS in terms of computation and communication costs involving all algorithms. In [24], there are two schemes: one is a basic signature scheme called KHA+ and the other one is an improved scheme named KHA+*. Specifically, as illustrated in TABLE II, it is evident that the computation costs of KeyDerive and KeyDerive* in KHA+ and KHA+* [24], as well as in HDS, exhibit a linear relationship with the number of non-wildcards, while these costs in FS-HDS are influenced by both the number of non-wildcards and the time $\mathbf{t}$. Additionally, the computational expenses of the Sign operation in KHA+, KHA+* and HDS are affected by the number of non-wildcards in both the access pattern and the secret key. However, in FS-HDS, these costs also depend on the time $\mathbf{t}$. Furthermore, the computation costs of Verify in KHA+, KHA+* and HDS are solely determined by the number of non-wildcards in the access pattern. In FS-HDS, the costs are related to the number of non-wildcards in the access pattern and the time $\mathbf{t}$. We can also learn that the computation cost of the KeyUpdate only realized in our FS-HDS is linearly related to both the length of a set $|\Phi_{\mathbf{t}}|$ and the number attributes in an attribute pattern. Note that KeyDerive is conducted with the master secret key by authority and KeyDerive* is performed with his/her secret key by a delegator.

From TABLE II, we can also draw several conclusions regarding communication cost comparisons. Specifically, it is clear that the communication costs of KeyDerive and KeyDerive* in KHA+, KHA+* and HDS are directly related to the number of non-wildcards. In contrast, for FS-HDS, these costs are influenced by both the number of non-wildcards and the time parameter $\mathbf{t}$. Moreover, the communication costs of the Sign in KHA+ are determined by the number of non-wildcards in both the access pattern and the secret key, and those in KHA+* are constant. However, in FS-HDS, these costs are influenced solely by the number of non-wildcards in the access pattern. Additionally, the communication costs for KeyUpdate in FS-HDS are associated with the number of non-wildcards in the access pattern and the encoded string indicating $\mathbf{t}$.

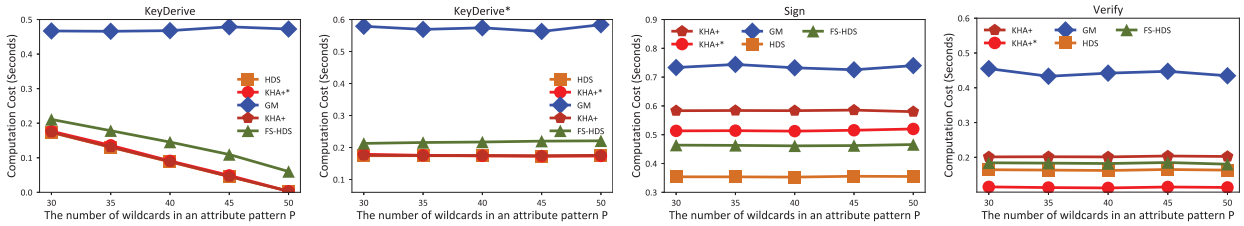
To summarize, KHA+* realizes lower constant-size signature communication while our FS-HDS can provide lower computation costs of signature generation and more practical properties concluded in TABLE I.

B. Experimental Analysis

The experimental performance evaluation was carried out using Python 3.6.13, Charm 0.43, the PBC-0.5.14 library,

TABLE II
COMPUTATION AND COMMUNICATION COST COMPARISONS

Overhead \ Algorithm		Setup	KeyDerive	KeyDerive*	KeyUpdate	Sign	Verify
Comp.Cos	KHA+ [24]	$2e_0$	$(2a+1+\mathcal{L}-\ell)e_0$	$[2(a'-a)+(\mathcal{L}-\ell)]e_0$	\	$(3\mathcal{L}+2x-2\ell+4)e_0+p$	$(c+\mathcal{L}-\ell+1)p$
	KHA+* [24]	e_0	$(2a+3)e_0$	$[2(a'-a)+3]e_0$	\	$(3c-2x+7)e_0$	$2p+ce_0$
	GM [22]	$11e_0+5p$	$(14\ell+8)e_0+4p$	$(14\ell'+8)e_0$	\	$(4\ell+8)p+31e_0$	$(4\ell+8)p+9e_0$
	HDS	$2e_0$	$(2a+1)e_0$	$2(a'-a)e_0$	\	$(2c-x+2)e_0$	$(c+2)p+(c+1)e_0$
	FS-HDS	$2e_0$	$(2a+1+2\|\Phi_{\mathbf{t}}\|)e_0$	$[2(a'-a)+\|\Phi_{\mathbf{t}}\|+2]e_0$	$[(\mathcal{L}'+k'-k)(\|\Phi_{\mathbf{t}}\|-\|\Phi_{\mathbf{t}}\cap\Phi_{\mathbf{i}}\|)]e_0$	$[(2c-x+2)+(\|\mathbf{t}\|+2)]e_0$	$(c+2+1)p+(c+1+\|\mathbf{t}\|)e_0$
Comm.Cos	KHA+ [24]	$(\mathcal{L}+1)\|\mathbf{G}_0\|$	$(a+\mathcal{L}-\ell)\|\mathbf{G}_0\|$	$(a'+\mathcal{L}-\ell)\|\mathbf{G}_0\|$	\	$(2\mathcal{L}+x-\ell+1)\|\mathbf{G}_0\|+\|\mathbf{G}_1\|$	\
	KHA+* [24]	$(\ell+1)\|\mathbf{G}_0\|$	$(a+2)\|\mathbf{G}_0\|+a\|\mathbb{Z}_p\|$	$(a'+2)\|\mathbf{G}_0\|+a'\|\mathbb{Z}_p\|$	\	$2\|\mathbf{G}_0\|$	\
	GM [22]	$(\ell+14)\|\mathbf{G}_0\|$	$(2\ell+4)\|\mathbf{G}_0\|$	$(2\ell+4)\|\mathbf{G}_0\|$	\	$(60\ell+20)\|\mathbf{G}_0\|$	\
	HDS	$(\ell+1)\|\mathbf{G}_0\|$	$(a+1)\|\mathbf{G}_0\|$	$(a'+1)\|\mathbf{G}_0\|$	\	$(c+2)\|\mathbf{G}_0\|$	\
	FS-HDS	$(\ell+1+\mathcal{L}')\|\mathbf{G}_0\|$	$(a+2+\mathcal{L}')\|\mathbf{G}_0\|$	$(a'+2+\mathcal{L}')\|\mathbf{G}_0\|$	$[\|\Phi_{\mathbf{t}}\|(x+2+\mathcal{L}')]\|\mathbf{G}_0\|$	$(c+3)\|\mathbf{G}_0\|$	\



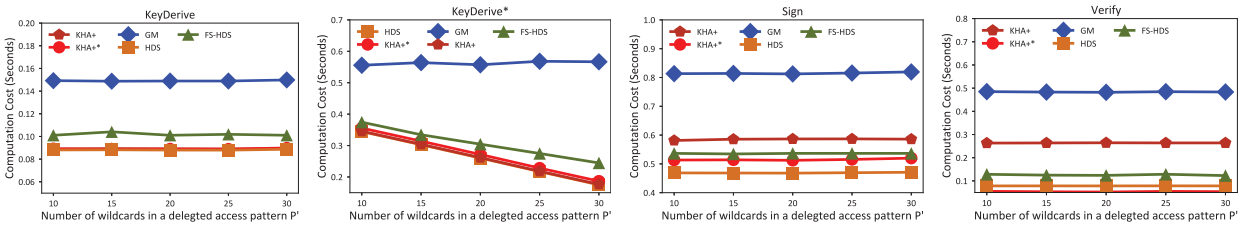
(a) KeyDerive algorithm (b) KeyDerive* algorithm (c) Sign algorithm (d) Verify algorithm

Fig. 3. Running time for various algorithms when $|\mathcal{P}|_* \in [30, 50]$, $|\mathcal{P}'|_* = 30$, $|\mathcal{P}^*|_* = 20$.

and OpenSSL 1.1.1. Simulations were performed on a laptop equipped with an Intel Core i5-11400H CPU @ 2.70GHz (12 cores) and 16GB RAM, running 64-bit Ubuntu 22.04.4 LTS. Additionally, a Raspberry Pi 4 Model B, featuring a Broadcom BCM 2711 Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz and 2GB RAM, running Raspbian, was used to simulate IoT devices in smart homes [47]. Since many variables are involved in various algorithms, to better simulate our experimental performance, we have controlled unique variables in our simulations and used a binary tree with its depth 6 for time period encoding. To simplify, we define the number of wildcards in an attribute pattern hidden in a secret key as $|\mathcal{P}|*$, in a delegated secret key as $|\mathcal{P}'|$, in an access pattern as $|\mathcal{P}|_*$.

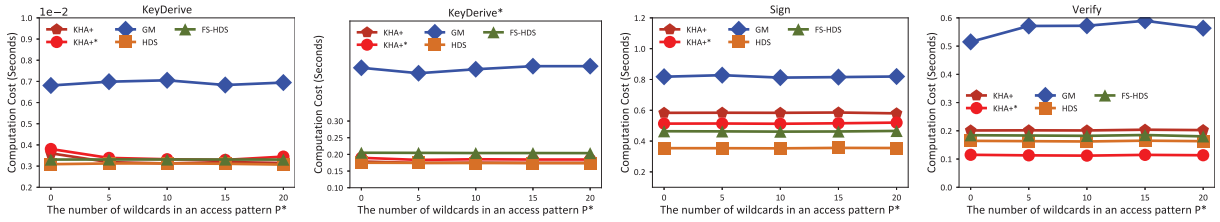
Fig. 3 exhibits a comparative analysis of the running time for the KeyDerive, KeyDerive*, Sign, and Verify algorithms on Raspberry Pi 4, within the parameters where $|\mathcal{P}|_* \in [30, 50]$, $|\mathcal{P}'|_* = 30$, $|\mathcal{P}^*|_* = 20$. As seen from Fig. 3(a), the computation time for KeyDerive in KHA+, KHA+*, HDS, and FS-HDS decreases linearly relative to the number of wildcards increased in the attribute pattern hidden in the secret key and those in GM are almost stable. Notably, the costs associated with these algorithms are proportional to the number of non-wildcards in the attribute pattern. This is because the pattern consists of both wildcard and non-wildcard elements, where an increase in the number of wildcards naturally results in a decrease in the number of non-wildcards. We can also from Figs. 3(b), 3(c), 3(d) reveal that the computation overheads for KeyDerive, Sign, and Verify remain relatively stable across KHA+, KHA+*, GM, HDS, and FS-HDS. Additionally, the signature and verification processes in HDS and FS-HDS demonstrate markedly lower costs when compared to those in KHA+ and GM, and the verification in KHA+* is slightly more efficient than that in our HDS and FS-HDS.

Fig. 4 reveals the running time comparisons for KeyDerive, KeyDerive*, Sign, and Verify algorithms on Raspberry Pi 4 when $|\mathcal{P}|_* = 50$, $|\mathcal{P}'|_* = 30$, $|\mathcal{P}^*|_* \in [0, 20]$. As depicted in Fig. 4(b), we can observe that the KeyDerive* computation time in KHA+, KHA+*, HDS, and FS-HDS show a linear decrease as the number of wildcards in the attribute pattern of



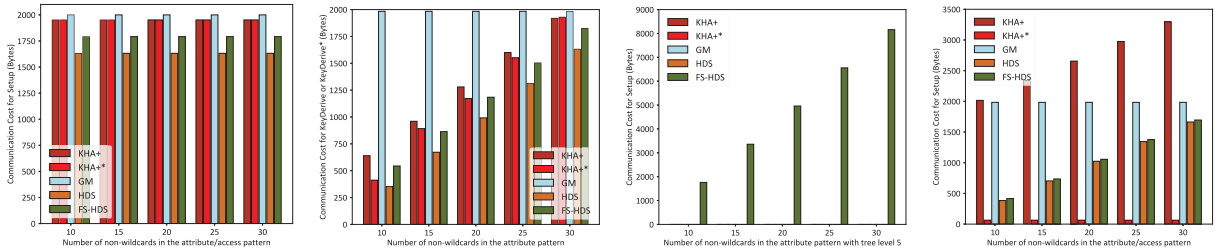
(a) KeyDerive algorithm (b) KeyDerive* algorithm (c) Sign algorithm (d) Verify algorithm

Fig. 4. Running time for distinct algorithms when $|\mathcal{P}|_* = 50$, $|\mathcal{P}'|_* = 30$, $|\mathcal{P}^*|_* \in [0, 20]$.



(a) KeyDerive algorithm (b) KeyDerive* algorithm (c) Sign algorithm (d) Verify algorithm

Fig. 5. Running time for different algorithms when $|\mathcal{P}|_* = 40$, $|\mathcal{P}'|_* \in [10, 30]$, $|\mathcal{P}^*|_* = 10$.



(a) Setup (b) KeyDerive (c) KeyUpdate (d) Sign

Fig. 6. Communication consumption for various algorithms.

a delegated secret key increases. This is because the pattern contains both wildcard and non-wildcard elements, and as the number of wildcards increases, the number of non-wildcards correspondingly decreases. From Figs. 4(a), 4(c), 4(d), we can see that the running time for KeyDerive, Sign, and Verify are consistently stable. In addition, the computation costs of the signature and verification algorithms in our HDS and FS-HDS remain lower compared to those in KHA+, GM, and the work in KHA+* is more efficient than our HDS and FS-HDS in terms of the Verify algorithm.

Fig. 5 provides insights into the running time efficiency of KeyDerive, KeyDerive*, Sign, and Verify algorithms on Raspberry Pi 4, with parameters set at $|\mathcal{P}|_* = 40$, $|\mathcal{P}'|_* \in [10, 30]$, $|\mathcal{P}^*|_* = 10$. As shown in Fig. 5(a), it is evident that the computation costs for these algorithms in KHA+, KHA+*, GM, HDS, and FS-HDS are essentially stable. Furthermore, from Figs. 5(c) and 5(d) the costs associated with our signature and verification algorithms in HDS and FS-HDS continue to be more efficient than those in KHA+ and GM.

Figure 6 illustrates the comparisons of communication overhead for conducting Setup, KeyDerive, KeyDerive*, Sign, and generating corresponding parameters. From Figs. 6(a) and 6(b), it demonstrates that our HDS and FS-HDS incur significantly lower communication costs for producing public parameters and (delegated) secret keys compared to KHA+ and GM. Furthermore, as shown in Fig. 6(c), it is evident that the KeyUpdate communication costs in FS-HDS increase linearly with the number of non-wildcards when the tree depth remains constant. As indicated from Fig. 6(d), our HDS and FS-HDS also feature lower communication costs for signature generation than those in KHA+ and GM but realize a slightly higher verification communication cost.

Since our FS-HDS is the only solution that incorporates the KeyUpdate feature, while other systems do not, this capability significantly enhances the security and flexibility of FS-HDS, making it particularly well-suited for smart homes. Hence, we have proceeded with the experimental simulations. Specifically, for time periods corresponding to the left leaf nodes, the KeyUpdate execution time reaches up to 1s. In contrast, for the right leaf nodes, the time required is considerably less, only 0.07s. This significant difference originates from the secret keys for the time periods corresponding to the right leaf nodes have been pre-generated by their parent node, whereas those for the left leaf nodes must be calculated by themselves.

In summary, our FS-HDS incurs significantly lower costs than KHA+ and GM across both the signature and verification and provides a slightly higher computation cost than KHA+*, while also enabling rapid key updates. Since the KHA+,

KHA+* and GM have demonstrated practicality in smart home applications through their experimental validation, the superior efficiency of our FS-HDS system further enhances its suitability for smart homes, making it more effective for this application.

IX. Conclusion

In this paper, we first suggested an efficient privacy-preserving hierarchical delegable signature scheme (HDS) without NIZK proofs, primarily addressing efficient fine-grained authorization, data integrity, and flexible delegation in smart homes. Then, we designed the first-ever efficient forward-secure HDS (FS-HDS) methodology with the efficient HDS as the underlying primitive, which, in addition to handling the issues that HDS enables in smart homes, also realizes forward security for key exposure resistance. We also demonstrated the forward-secure unforgeability of FS-HDS via comprehensively rigorous security proofs. The effectiveness and practicability of our methodologies were validated through experimental simulations for smart homes. In future work, our efforts will continue to design a more secure versatile FS-HDS scheme capable of withstanding chosen ciphertext attacks (CCA) for smart homes. One possible solution is to incorporate Non-Interactive Zero-Knowledge (NIZK) Proofs to ensure that any decryption attempt can be verified without compromising security. Another promising direction is to develop attribute-specified self-sovereign delegation, which enables a delegator to selectively control which attributes within their attribute pattern can be delegated. One potential solution is to exploit secret sharing and reconstruction techniques to ensure only authorized ones can reconstruct complete permissions.

References

[1] B. Hammi, S. Zeadally, R. Khatoun, and J. Nebhen, "Survey on smart homes: Vulnerabilities, risks, and countermeasures," *Comput. Secur.*, vol. 117, Jun. 2022, Art. no. 102677.

[2] H. Chi, Q. Zeng, and X. Du, "Detecting and handling IoT interaction threats in multi-platform multi-control-channel smart homes," in *Proc. 32nd USENIX Secur. Symp.*, Anaheim, CA, USA, 2023, pp. 1559–1576.

[3] Y. Wan, K. Xu, G. Xue, and F. Wang, "IoTArgos: A multi-layer security monitoring system for Internet-of-Things in smart homes," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 874–883.

[4] E. Zeng and F. Roesner, "Understanding and improving security and privacy in multi-user smart homes: A design exploration and in-home user study," in *Proc. USENIX Secur. 19*, Jan. 2019, pp. 159–176.

[5] A. Siddiqui et al., "Survey on unified threat management (UTM) systems for home networks," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 4, pp. 2459–2509, 4th Quart., 2024, doi: 10.1109/COMST.2024.3382470.

[6] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur.*, Apr. 2010, pp. 60–69.

[7] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology—CT-RSA 2011*. Heidelberg, Germany: Springer, 2011, pp. 376–392.

[8] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *Proc. PKC*. Berlin, Germnay: Springer, 2013, pp. 125–142.

[9] Q. Tao, X. Cui, and A. Iftekhar, "A novel lightweight decentralized attribute-based signature scheme for social co-governance," *Inf. Sci.*, vol. 654, Jan. 2024, Art. no. 119839.

[10] A. E. Kaafarani, E. Ghadafi, and D. Khader, "Decentralized traceable attribute-based signatures," in *Proc. CT-RSA*, Jan. 2014, pp. 327–348.

[11] J. Li, Y. Chen, J. Han, C. Liu, Y. Zhang, and H. Wang, "Decentralized attribute-based server-aid signature in the Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4573–4583, Mar. 2022.

[12] Z. Kang, J. Li, J. Shen, J. Han, Y. Zuo, and Y. Zhang, "TFS-ABS: Traceable and forward-secure attribute-based signature scheme with constant-size," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9514–9530, Sep. 2023.

[13] Y. Bao, W. Qiu, and X. Cheng, "Efficient and fine-grained signature for IIoT with resistance to key exposure," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9189–9205, Jun. 2021.

[14] J. Wei, W. Liu, and X. Hu, "Forward-secure threshold attribute-based signature scheme," *Comput. J.*, vol. 58, no. 10, pp. 2492–2506, Oct. 2015.

[15] Y. Zhang, J. Zhao, Z. Zhu, J. Gong, and J. Chen, "Registered attribute-based signature," in *Proc. PKC*. Cham, Switzerland: Springer, Jan. 2024, pp. 133–162.

[16] F. Luo and S. Al-Kuwari, "Attribute-based signatures from lattices: Unbounded attributes and semi-adaptive security," *Designs, Codes Cryptography*, vol. 90, no. 5, pp. 1157–1177, May 2022.

[17] A. E. Kaafarani and S. Katsumata, "Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices," in *Proc. PKC*, Jan. 2018, pp. 89–119.

[18] R. Tsabary, "An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both," in *Proc. 15th Int. Conf. Theory Cryptogr.* Switzerland: Springer, 2017, pp. 489–518.

[19] Y. Chen, J. Li, C. Liu, J. Han, Y. Zhang, and P. Yi, "Efficient attribute based server-aided verification signature," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3224–3232, Nov. 2022.

[20] Z. Kang, J. Li, Y. Zuo, Y. Zhang, and J. Han, "OABS: Efficient outsourced attribute-based signature scheme with constant-size," *IEEE Internet of Things Journal*, vol. 11, no. 23, pp. 38167–38177, Dec. 2024, doi: 10.1109/JIOT.2024.3444827.

[21] C. C. Drăgan, D. Gardham, and M. Manulis, "Hierarchical attribute-based signatures," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, Jan. 2018, pp. 213–234.

[22] D. Gardham and M. Manulis, "Hierarchical attribute-based signatures: Short keys and optimal signature length," in *Proc. ACNS*, Jan. 2019, pp. 89–109.

[23] D. Gardham and M. Manulis, "Revocable hierarchical attribute-based signatures from lattices," in *Proc. ACNS*, Jan. 2022, pp. 459–479.

[24] S. Kumar, Y. Hu, and M. Andersen, "JEDI: Many-to-many end-to-end encryption and key delegation for IoT," in *Proc. USENIX Secur.*, 2019, pp. 1519–1536.

[25] Y. Zhu, G.-J. Ahn, H. Hu, D. Ma, and S. Wang, "Role-based cryptosystem: A new cryptographic RBAC system based on role-key hierarchy," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 2138–2153, Dec. 2013.

[26] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," in *Proc. ASIACRYPT*. Berlin, Switzerland: Springer, 2002, pp. 548–566.

[27] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. CRYPTO*, Jan. 2001, pp. 213–229.

[28] M. Abdalla, E. Kiltz, and G. Neven, "Generalized key delegation for hierarchical identity-based encryption," in *Proc. ESORICS*, vol. 4734, Jan. 2007, pp. 139–154.

[29] R. Anderson and R. Needham, "Robustness principles for public key protocols," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1995, pp. 236–247.

[30] M. Bellare and S. K. Miner, "A forward-secure digital signature scheme," in *Proc. 19th Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, Aug. 1999, pp. 431–448.

[31] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," in *Proc. ASIACRYPT*. Cham, Switzerland: Springer, 2000, pp. 116–129.

[32] X. Boyen, H. Shacham, E. Shen, and B. Waters, "Forward-secure signatures with untrusted update," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 191–200.

[33] G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifying," in *Proc. 21st Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2001, pp. 332–354.

[34] M. Abdalla, S. Miner, and C. Namprempre, "Forward-secure threshold signature schemes," in *Proc. CT-RSA*, vol. 2001. Cham, Switzerland: Springer, 2001, pp. 441–456.

[35] R. Kurek, "Efficient forward-secure threshold public key encryption," in *Proc. ACISP*. Cham, Switzerland: Springer, Jan. 2020, pp. 330–349.

[36] J. Yu, R. Hao, F. Kong, X. Cheng, J. Fan, and Y. Chen, "Forward-secure identity-based signature: Security notions and construction," *Inf. Sci.*, vol. 181, no. 3, pp. 648–660, Feb. 2011.

[37] H. Ko, G. Jeong, J.-H. Kim, J. Kim, and H. Oh, "Forward secure identity-based signature scheme with RSA," in *Proc. IFIP TC*. Cham, Switzerland: Springer, Jan. 2019, pp. 314–327.

[38] S. S. M. Chow, L. C. K. Hui, S. M. Yiu, and K. P. Chow, "Forward-secure multisignature and blind signature schemes," *Appl. Math. Comput.*, vol. 168, no. 2, pp. 895–908, Sep. 2005.

[39] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee, "Pixel: Multi-signatures for consensus," in *Proc. 29th USENIX Secur. Symp.*, Aug. 2020, pp. 2093–2110.

[40] T. H. Yuen, J. K. Liu, X. Huang, M. H. Au, W. Susilo, and J. Zhou, "Forward secure attribute-based signatures," in *Proc. ICICS,*. Berlin, Germany: Springer, Jan. 2012, pp. 167–177.

[41] Y. Tao, R. Zhang, and Y. Ji, "Forward security of fiat-shamir lattice signatures," in *Proc. ACNS*. Cham, Switzerland: Springer, Jan. 2023, pp. 607–633.

[42] M. D. Green and I. Miers, "Forward secure asynchronous messaging from puncturable encryption," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 305–320.

[43] M. Bellare, I. Stepanovs, and B. Waters, "New negative results on differing-inputs obfuscation," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 2016, pp. 792–821.

[44] S. Halevi, Y. Ishai, A. Jain, I. Komargodski, A. Sahai, and E. Yogev, "Non-interactive multiparty computation without correlated randomness," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, 2017, pp. 181–211.

[45] X. Li, J. Xu, X. Fan, Y. Wang, and Z. Zhang, "Puncturable signatures and applications in proof-of-stake blockchain protocols," in *Proc. IEEE TIFS*, Jan. 2020, pp. 3872–3885.

[46] J. Wei, G. Tian, D. Wang, F. Guo, W. Susilo, and X. Chen, "Pixel++ *and Pixel*+++: Compact and efficient forward-secure multi-signatures for PoS blockchain consensus," in *Proc. USENIX Secur. 2024*, 2024, pp. 6237–6254.

[47] T. M. Jois et al., "SocIoTy: Practical cryptography in smart home contexts," in *Proc. PoPET*, Oct. 2024, pp. 447–464.

[48] F. Zhu, X. Yi, A. Abuadbba, I. Khalil, S. Nepal, and X. Huang, "Authenticated data sharing with privacy protection and batch verification for healthcare IoT," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 1, pp. 32–42, Jan. 2023.

[49] F. Zhu, X. Yi, A. Abuadbba, I. Khalil, S. Nepal, and X. Huang, "Forward-secure edge authentication for graphs," *Comput. J.*, vol. 65, no. 7, pp. 1653–1665, Jul. 2022.

[50] J. H. Seo and K. Emura, "Revocable identity-based encryption revisited: Security model and construction," in *Proc. PKC*. Berlin, Germany: Springer, Jan. 2013, pp. 216–234.