6-2018

Verifiably encrypted cascade-instantiable blank signatures to secure progressive decision management

Yujue WANG
Singapore Management University, yjwang@smu.edu.sg

Hwee Hwa PANG
Singapore Management University, hhpang@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1007/s10207-017-0372-2

Published in International Journal of Information Security, 2017 April, Advance Online, pp 1–17.
http://doi.org/10.1007/s10207-017-0372-2

Verifiably encrypted cascade-instantiable blank signatures to secure progressive decision management

Yujue Wang[1] · **HweeHwa Pang**[1] · **Robert H. Deng**[1]

Abstract In this paper, we introduce the notion of verifiably encrypted *cascade-instantiable blank signatures* (CBS) in a multi-user setting. In CBS, there is a *delegation chain* that starts with an *originator* and is followed by a sequence of *proxies*. The originator creates and signs a template, which may comprise fixed fields and exchangeable fields. Thereafter, each proxy along the delegation chain is able to make an instantiation of the template from the choices passed down from her direct predecessor, before generating a signature for her instantiation. First, we present a non-interactive basic CBS construction that does not rely on any shared secret parameters among the users. In verifying an instantiation signature, all the preceding instantiation signatures leading back to the template signature are also verified concurrently. It is formally proved to be secure against collusion attacks by the originator and proxies. Second, we investigate verifiably encrypted CBS to provide fairness between the originator and proxies, where the security model is stricter than basic CBS in that the adversary may also collude with the arbitrator. Efficiency analysis shows that the proposed CBS schemes enjoy linear computation costs. Finally, we extend our scheme to CBS supporting designated instantiations, free instantiations, privately verifiable template signature, identity-based CBS, as well as CBS secure against proxy-key exposure.

✉ Yujue Wang
yjwang@smu.edu.sg

HweeHwa Pang
hhpang@smu.edu.sg

Robert H. Deng
robertdeng@smu.edu.sg

1 School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902, Singapore

Keywords Digital signature · Blank signature · Proxy signature · Sanitizable signature · Redactable signature · Verifiably encrypted signature · Optimistic fair exchange · Delegation chain

1 Introduction

Many real-world applications require signatures to be sequentially generated by users in a way that the subset relationship among messages should be preserved and verifiable. For example, in some XML applications, the XML data need to pass through many entities with security guarantees of integrity and authenticity [9,31,42]. Each entity in the process is able to change the data without interacting with any predecessor, while enabling all the changes to be verified by the successors. In particular, there may be privacy-sensitive components in the XML data that cannot be accessed by lower level entities. In enforcing access control, these components should be excluded from the data at some stage but still be verifiable by the successors without recovering the contents.

In another example, a public electronic ordering or procurement system lets users process purchase orders in turn, thus improving procurement efficiency while saving financial and time costs. Public verifiability of the orders would make the procurement procedure transparent and deter corruption. In the system, the supplier first prepares a structured template according to the buyers' purchasing needs, which contains all the available items along with types, performance parameters, prices, etc. The supplier signs this template and gives the (template, signature) pair to the purchasing manager. The manager makes his choices on some key items, signs his decision and forwards the table to an administrator. If the administrator is convinced that the manager has signed on a subset of the original template, he issues an electronic order by setting the remaining fields in the template and signs the final decision. This final electronic order can be publicly verified on whether it is a subset of the one passed down through the workflow. In case of a dispute among the parties, an arbitrator who is normally offline may intervene to provide a resolution based on the verification information.

In the literature, *optimistic fair exchange protocols* (OFE) [2,4] and *verifiably encrypted signatures* (VES) [8,28] allow users to exchange digital items in a fair manner with the help of a trusted adjudicator. Although many OFE/VES schemes have been proposed in multi-user settings [20,40,45,46], none of them fits our requirement to support a series of modifications as an item passes from a supplier to a list of buyers. Additionally, we need the subset relationship between instances of the item to be verifiable, which is beyond the capability of existing OFE/VES.

Blank digital signature scheme, introduced by Hanser and Slamanig [19], meets our requirements partially. The scheme allows an *originator* to sign a *template*, comprising *fixed fields* as well as *exchangeable fields* with multiple choices. Subsequently, a *proxy* can set the choices for the exchangeable fields to derive an *instantiation* of the template, before affixing to it a signature. The scheme requires the originator and proxy to share a secret parameter, that is, a template dependent private key, which is chosen randomly by the originator. The validity of the template signature can only be verified by the proxy, while the instantiation signature may be publicly verified. Applied to the aforementioned electronic procurement example, the scheme would require the purchasing manager to make all the decisions in producing the electronic order, since only one proxy can derive an instantiation from a template. Thus, the original blank signature scheme is not sufficient to support real-world applications, like the those described above, that involve a hierarchy of buyers.

In this paper, we focus on a setting where a template given by an originator may be instantiated by a succession of proxies. Formally, a template is a set $\mathbb{T}$ of fields T_i for $1 \leq i \leq \ell$, with each T_i allowing choices $m_{i,1}, \ldots, m_{i,s_i}$; hence $\mathbb{T} = \{T_i = \{m_{i,1}, \ldots, m_{i,s_i}\} : 1 \leq i \leq \ell\}$. The number of fields $\ell = |\mathbb{T}|$ is the template *length*, while the total number of choices across all fields is the template *size* s, i.e., $s = \sum_{i=1}^{\ell} s_i$ where $s_i = |T_i|$. If $s_i = 1$, then T_i is a fixed field in the template; otherwise, T_i is an exchangeable field.

For example, suppose that the originator has a template $\mathbb{T} = \{\{a\}, \{b_1, b_2, b_3, b_4\}, \{c\}, \{d_1, d_2, d_3\}\}$; here, $\ell = 4$, $s_1 = s_3 = 1$, $s_2 = 4$, $s_4 = 3$ and $s = 9$. Thus, $T_1 = \{a\}$ and $T_3 = \{c\}$ are fixed fields in $\mathbb{T}$, while T_2 and T_4 are exchangeable fields. The first proxy P_1 may partially instantiate $\mathbb{T}$ as $M_1 = \{\{a\}, \{b_2\}, \{c\}, \{d_1, d_2, d_3\}\}$, making a choice for T_2 among $\{b_1, b_2, b_3, b_4\}$ while leaving $T_4 = \{d_1, d_2, d_3\}$ available to proxy P_2. Alternatively, P_1 may exclude $\{b_2, b_3, d_1\}$ from the template after which P_2 further selects $\{b_4, d_2\}$ to P_2.

Similar to [19], we encode a template $\mathbb{T}$ as a polynomial on variable x:

$$\mathcal{E}(\mathbb{T}) = \prod_{i=1}^{\ell} \prod_{m \in T_i} (x + H(id_T \| m \| i)), \tag{1}$$

where id_T is an unique identifier of $\mathbb{T}$, and $H : \{0, 1\}^* \rightarrow Z_p^*$ is a collision-resistant hash function. Instantiations of the template are encoded in the same way.

1.1 Our contributions

In this paper, we present a suite of *cascade-instantiable blank signature schemes* (CBS). Basic CBS supports multi-level delegations from an originator P_0 to a chain of n proxies $P_1, \ldots, P_n$. The scheme provides strong security guarantee, in that the originator cannot collude with proxies to forge an instantiation of another proxy, and the proxies cannot forge a template signature even when all of them collude. We also introduce an enhanced, verifiably encrypted CBS guaranteeing fairness between an originator P_0, multiple proxies $\{P_1, \ldots, P_n\}$, and an arbitrator. The verifiably encrypted CBS additionally provides security against more powerful attacks involving collusion with the arbitrator. CBS is *public verifiable*, eliminating the need for any shared secret parameter among the users. Our CBS formulation is strictly more general than the original blank signature scheme of [19]. We obtain the following results.

Framework and security model We formalize the frameworks of basic CBS and verifiably encrypted CBS. In basic CBS, an originator prepares a structured template with fixed fields as well as exchangeable fields consisting of multiple choices, signs it and delegates instantiation rights to a proxy. The proxy makes choices on any exchangeable fields in the template, signs the instantiation, and delegates to the next-level proxy the capability to further instantiate the remaining exchangeable fields. In validating an instantiation signature, all instantiation signatures in higher levels tracing back to the template signature are verified in a batch and the verifier is not required to know the content in the original template. In the formal security model, *security against originator* and *security against proxies* capture collusion attacks by malicious originator and proxies.

Verifiably encrypted CBS introduces an arbitrator who only intervenes in case there are disputes between the originator and proxies. In addition to the provisions in basic CBS, the security model of verifiably encrypted CBS considers more severe collusion attacks involving the arbitrator.

Non-interactive constructions We present non-interactive and general constructions for the basic and verifiable encrypted CBS. From a technical standpoint, a polynomial commitment scheme is employed to guarantee the relationship between a template, its instantiations and excluded choices. A challenge in CBS construction is that the subset relationship between instantiations requires the underlying polynomial commitment to be multiplicatively homomorphic, whereas the existing scheme in [27] is only additively homomorphic. We circumvent this problem by introducing an accumulating power $\bar{\omega}_i$ of evaluations of excluded polynomials for every proxy. The CBS constructions are non-interactive in the sense that every proxy operates without interacting with the originator, its predecessors or successors. We employ sequential aggregate signature as a building block, which not only reduces the signature size by combining the template signature and instantiation signatures, but also ensures the correct ordering of these signatures. Our constructions are general and may combine with any available sequential aggregate signature scheme, although one with linear complexity would be desirable.

Extensions We adapt our basic CBS scheme to support five other practical application scenarios. In the first extension, the originator is empowered to designate the exchangeable fields to be instantiated by every proxy in the chain. The second variation is cascade-and-freely-instantiable blank signatures, where the originator and proxies are not required to delegate to specific successors. Instead, at every step, anyone can be a proxy to further instantiate the template obtained from his predecessor. The third adaptation, which makes the template signature privately verifiable by the highest level proxy, offers all the security properties and functionalities of the original blank signature scheme of [19] while being more general and more efficient. The fourth variation employs the multi-level proxy signature scheme of [41] to generate delegations, resulting in a CBS scheme that is secure against key exposure but sacrifices non-interactivity. The fifth extension is CBS employing identity-based sequential aggregate signature that eliminates the burden of managing public key certificates.

1.2 Related work

Optimistic fair exchange (OFE) and verifiably encrypted signatures (VES) OFE allows two users to exchange their digital items in a fair way such that either both of them succeed in obtaining the other's item or both fail [1,2,4]. Usually, fairness is achieved through a trusted third party, e.g., an arbitrator/adjudicator. Similarly, in VES [8,38], a party encrypts her signature for some message using the public key of some trusted adjudicator, and sends the encrypted signature to the receiver. Subsequently, if the sender refuses to reveal her signature, the adjudicator may intervene to recover the signature. One notable way OFE/VES differs from our problem is that, in the former, the signatures of the exchanging parties are on different items, and there is no verifiability of subset relationship between messages.

Huang et al. [23] introduced ambiguous OFE which prevents the verifier from abusing the sender's partial signature. Zhang et al. [49,50] studied OFE and VES in an identity-based setting. Huang et al. [25] investigated the relationship of OFE security between single-user and multi-user settings. From time capsule signatures, Huang et al. [24] presented a generic OFE construction in the standard model. Huang et al. [21] introduced an ambiguous OFE protocol without random oracles, where the sender interacts with the receiver in generating a partial signature. Draper-Gil et al. [16] investigated OFE in a setting with active intermediaries. Huang et al. [22] enhanced OFE security so that the third party cannot learn the resolved signatures. Recently, Hanser et al. [18] introduced a block-box construction of VES from structure-preserving signatures on equivalence classes.

Blank digital signatures Hanser and Slamanig [19] introduced blank digital signatures in a single proxy setting. Given a template and the template signature generated by an originator, only the designated proxy can create a signature on an instantiation of the template. The proxy's behavior is restricted to choices for exchangeable fields, which are explicitly specified in the message template. In their construction, fixed fields and exchangeable fields are encoded in the same manner. Derler et al. [15] noted that all fixed fields can be aggregated together without compromising security; that is, the fixed fields can be concatenated into one string. This optimizes the original scheme of [19] by reducing the degree of the encoded polynomial.

Sanitizable signatures In sanitizable signature, introduced by Ateniese et al. [3], a signer produces a signature on a message with some mutable portions. A designated proxy is able to replace the mutable portions by any elements in the message space, without invalidating the signature. Although sanitizable signature bears some similarities with blank signature in that both involve designated proxy and mutable portions/exchangeable fields, there are obvious differences. First, sanitizable signature emphasizes the *replaceability* of mutable portions and the proxy's choices can be *arbitrary* over the entire message space. Second, the proxy in sanitizable signature has only rights on data replacement and is not required to sign the modified message. Note that Klonowski and Lauks [29] improved sanitizable signatures by limiting the proxy's behavior, where the available choices of mutable portions are predefined strings.

Yuen et al. [47] outlined the properties of existing sanitizable signatures, such as different types of state controllability, sanitized message, designated sanitizer and transparency, and showed the relationships between some of these properties. Brzuska et al. [10] investigated accountability toward signer and proxy in sanitizable signature schemes, which was further refined by Canard and Jambert [12] with the aim of limiting the proxy's capability. The notion of trapdoor sanitizable signature introduced by Canard et al. [13] allows a signer to specify multiple proxies at any time, and a generic construction was given by Yum et al. [48]. Bao et al. [5] introduced hierarchical trapdoor sanitizable signature and presented a generic construction from hierarchical identity-based chameleon hash function. Lai et al. [32] unified accountability and trapdoor properties in sanitizable signature. Brzuska et al. [11] introduced unlinkability in sanitizable signature which prevents outsiders from associating sanitized message-signature pairs to the original message. A typical application of sanitizable signatures in web-service-enabled business processes was investigated in [42].

Redactable signatures Johnson et al. [26] first investigated redactable signature which focuses on the *removability* of a signed message. Informally, anyone who holds a valid message-signature pair is able to generate a signature on a substring of the original signed message by replacing certain parts of the message with a special symbol. Therefore, a redactable signature would not leak the removed parts except for their length. Chang et al. [14] improved redactable signature also hides the length of the removed parts. Brzuska et al. [9] studied redactable signatures specifically for tree-structured data. Kundu et al. [30] investigated a general case which captures redactability over regular strings, trees, graphs and forests. Their scheme possesses leakage-free property so that the redacted parts cannot be inferred by others. Lim et al. [34] presented a more efficient redactable signature construction compared to existing ones based on pairings, where the signature size is not dependent on the number of blocks of a given message. Recently, Pohls and Samelin [39] further enhanced redactable signatures to make them updatable, i.e., the signer can add new blocks to signed messages.

Proxy signatures Mambo et al. [37] introduced proxy signatures and classified delegations in proxy signatures into three types, i.e., full delegation, partial delegation and delegation by warrant. Since then, delegation by warrant has been commonly adopted in proxy-related schemes, where a signer specifies a proxy's legal behavior, which usually contains security policy descriptions, in a warrant. Many studies have been conducted on this topic to support different properties and applications, such as delegation delivery without using a secure channel [33], one-time proxy signatures [44], fully hierarchical proxy signatures [36], security against proxy-key exposure [41], delegator anonymity [17] and security analyses of existing schemes [7,43]. Proxy signature differs from blank signature in three aspects. First, a warrant in proxy signature is usually an abstract description, while a template in blank signature is very specific and has a strict structure. Second, the delegator in a proxy signature scheme is only required to produce a valid delegation, while the originator in a blank signature scheme signs on a template in addition to producing a delegation. Third, a warrant should be known by a verifier for validating proxy signature, whereas the original template should be hidden when verifying an instantiation signature.

1.3 Paper organization

The remainder of this paper is organized as follows. Section 2 briefly recalls some preliminaries that will be used as building blocks in our CBS constructions. We introduce the basic CBS and formalize the corresponding security model in Sect. 3, as well as propose a construction along with security proofs. Section 4 introduces our verifiably encrypted CBS scheme, formalizes the security model, presents a construction and proves its security. Section 5 then discusses some possible extensions of our basic CBS. Finally, Sect. 6 concludes the paper.

2 Preliminaries

2.1 Sequential aggregate signature

A sequential aggregate signature scheme [35] consists of the following four algorithms, where all the given messages and public keys are ordered.

- $\texttt{Setup}(\kappa) \rightarrow \mathsf{pp}$: On input a security parameter $\kappa \in \mathbb{N}$, the setup algorithm outputs public parameters pp.
- $\texttt{KeyGen}(\kappa, \mathsf{pp}) \rightarrow (pk, sk)$: On input security parameter $\kappa \in \mathbb{N}$ and public parameters pp, the key generation algorithm, which is carried out by each user, outputs a pair of public/private keys (pk, sk).
- $\texttt{SASign}((m_1, \ldots, m_{i-1}), (pk_1, \ldots, pk_{i-1}), \sigma_{i-1}, m_i, sk_i, \mathsf{pp}) \rightarrow \sigma_i$: On input a sequential aggregate signature σ_{i-1} over messages $(m_1, \ldots, m_{i-1})$ under distinct public keys $(pk_1, \ldots, pk_{i-1})$, a message m_i, a private key sk_i and public parameters pp, the sequential aggregate signing algorithm, which is carried out by user P_i, outputs signature σ_i for messages $(m_1, \ldots, m_i)$ under $(pk_1, \ldots, pk_i)$. Note that σ_0 is set as empty.
- $\texttt{SAVrfy}((m_1, \ldots, m_i), (pk_1, \ldots, pk_i), \sigma_i, \mathsf{pp}) \rightarrow 0/1$: On input a set of messages $(m_1, \ldots, m_i)$, public keys $(pk_1, \ldots, pk_i)$, a sequential aggregate signature σ_i and public parameters pp, the sequential aggregate signature verification algorithm, which is carried out by a verifier, outputs "1" if σ_i is valid for the given messages under the given public keys, or "0" otherwise.

A sequential aggregate signature scheme is *secure against existential forgery* [35] if no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ can win the following security game with non-negligible probability.

Setup Challenger $\mathcal{C}$ invokes $\texttt{Setup}(\kappa)$ with security parameter κ to obtain public parameters pp. Next, the challenger runs $\texttt{KeyGen}(\kappa, \mathsf{pp})$ to create a pair of public/private keys (pk, sk), and gives public information pp and pk to $\mathcal{A}$.

Queries Adversary $\mathcal{A}$ adaptively issues sequential aggregate signature queries for messages of his choice under public keys including pk. In each query, the adversary submits a sequential aggregate signature σ_{i-1} over messages $(m_1, \ldots, m_{i-1})$ under distinct public keys $(pk_1, \ldots, pk_{i-1})$, and another message m. Here, i is at most n, the maximum number of users. Challenger $\mathcal{C}$ responds with a sequential aggregate signature σ_i over $(m_1, \ldots, m_{i-1}, m)$ under $(pk_1, \ldots, pk_{i-1}, pk)$.

Output Adversary $\mathcal{A}$ outputs a sequential aggregate signature σ'_j over $(m'_1, \ldots, m'_j)$ under distinct public keys $(pk'_1, \ldots, pk'_j)$, where some public key, say pk'_{j*}, must be equal to pk. Also, j is at most n. Adversary $\mathcal{A}$ wins the game if both the following conditions hold:

- $\texttt{SAVrfy}((m'_1, \ldots, m'_j), (pk'_1, \ldots, pk'_j), \sigma'_j, \mathsf{pp}) = 1$;
- $(m'_1, \ldots, m'_{j*})$ has not been queried for a sequential aggregate signature under $(pk'_1, \ldots, pk'_{j*})$.

2.2 Polynomial commitment

Kate et al. [27] proposed an efficient polynomial scheme over bilinear groups such that, for a given polynomial $f(x) \in Z_p[x]$, a committer can produce a polynomial commitment C, along with a witness w_i with respect to the polynomial evaluation $f(i)$ at some random point i. With w_i and C, a verifier can check whether $f(i)$ is indeed the evaluation of $f(x)$ at point i. Their scheme works as follows.

Suppose $G_1 = \langle g \rangle$ is a cyclic group with prime order p and efficient group operations. The group G_1 is bilinear if there exists a cyclic group G_2 with order p and an efficient bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the following properties: (a) bilinearity: $\forall \mu, \nu \in G_1$ and $\forall a, b \in Z_p^*$, $\hat{e}(\mu^a, \nu^b) = \hat{e}(\mu, \nu)^{ab}$; (b) non-degeneracy: $\hat{e}(g, g) \neq 1$.

- $\texttt{KeyGen}(1^\kappa, d)$: Randomly pick a value $\alpha \in_R Z_p^*$ and set the private key $\mathsf{sk} = \alpha$. Compute $u_j = g^{\alpha^j}$ for each $1 \leq j \leq d$ where d is the maximum polynomial degree. Set the public key as $\mathsf{pk} = (\hat{e}, G_1, G_2, p, g, u_1, \ldots, u_d)$.
- $\texttt{Commit}(\mathsf{pk}, f(x))$: Given a polynomial

$$f(x) = \sum_{j=0}^{deg[f]} f_j x^j \bmod p$$

with degree $deg[f]$ at most d, generate the commitment as:

$$C = \prod_{j=0}^{deg[f]} u_j^{f_j}$$

- $\texttt{WitGen}(\mathsf{pk}, f(x), i)$: Compute the polynomial

$$h(x) = \sum_{j=0}^{deg[h]} h_j x^j = \frac{f(x) - f(i)}{x - i} \bmod p$$

which has degree $deg[h]$ at most $d-1$. Produce the witness as:

$$w_i = \prod_{j=0}^{deg[h]} u_j^{h_j}$$

- $\texttt{VrfyEval}(\mathsf{pk}, C, i, f(i), w_i)$: Check whether the following equality holds:

$$\hat{e}(C, g) \stackrel{?}{=} \hat{e}(w_i, g^\alpha / g^i) \hat{e}(g, g)^{f(i)}$$

If so, output "1" which means that $f(i)$ is indeed the evaluation of $f(x)$ at point i; otherwise, output "0".

3 Cascade-instantiable blank signature

In this section, we formulate the basic cascade-instantiable blank signature and its security model. We then present a basic CBS construction based on sequential aggregate signatures and provide the security proofs.

3.1 Definitions and security model

Let the user set be $\mathbf{P} = \{P_0, P_1, \ldots, P_n\}$ and let $\mathbf{PK}_i$ denote the public keys of originator P_0 and proxies $P_1, \ldots, P_i$, i.e., $\mathbf{PK}_i = (pk_0, pk_1, \ldots, pk_i)$; the subscript i denotes the hierarchical position of proxy P_i. A chain of instantiations of a template is valid only if each instantiation preserves the fixed fields in its predecessor, while maintaining or narrowing the choices in each exchangeable field. In this paper, we do not explicitly carry out semantic/sanity checks on the choices in all fixed and exchangeable fields, since their validity and the above mentioned relationship among template, instantiations and excluded choices can be verified in the verification procedures.

Formally, a basic cascade-instantiable blank signature scheme consists of the following algorithms:

- $\texttt{Setup}(\kappa, d) \rightarrow \mathsf{pp}$: On input a security parameter $\kappa \in \mathbb{N}$ and the maximum template size $d \in \mathbb{N}$, the setup algorithm, which is carried out by the system manager, generates public parameters pp.
- $\texttt{KeyGen}(\kappa, \mathsf{pp}) \rightarrow (pk, sk)$: On input security parameter $\kappa \in \mathbb{N}$ and public parameters pp, the key generation algorithm, which is carried out by each user in $\mathbf{P}$, outputs a pair of public/private keys (pk, sk).
- $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1) \rightarrow (\sigma_T, \delta_1)$: On input a template $\mathbb{T}$, public parameters pp, the originator's private key sk_0 and proxy P_1's public key pk_1, the template signing algorithm, which is carried out by the originator, outputs a signature σ_T for the template and a delegation δ_1 for P_1. A unique identifier id_T of template $\mathbb{T}$ is generated and embedded in σ_T.
- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_1) \rightarrow 0/1$: On input a template $\mathbb{T}$, template signature σ_T, public parameters pp, the originator's public key pk_0 and proxy P_1's public key pk_1, the template signature verification algorithm, which is carried out by any verifier (particularly P_1), outputs "1" if σ_T is valid for $\mathbb{T}$ under pk_0 or "0" otherwise.
- $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta_i, \mathsf{pp}, sk_i, \mathbf{PK}_{i+1}) \rightarrow (\sigma_i, \delta_{i+1})$: On input proxy P_{i-1}'s instantiation M_{i-1}, proxy P_i's instantiation M_i, instantiation signature σ_{i-1} produced by P_{i-1}, delegation δ_i for P_i, public parameters pp, proxy P_i's private key sk_i and a set of public keys $\{pk_0, \ldots, pk_{i+1}\}$, the instantiation algorithm, which is carried out by P_i, outputs an instantiation signature σ_i for M_i and a delegation δ_{i+1} for P_{i+1} if both σ_{i-1} and δ_i are valid. Here, M_i is a subset of M_{i-1} for $1 \leq i \leq n$, $M_0 = \mathbb{T}$, and $\sigma_0 = \sigma_T$. Where P_i is the last proxy P_n, pk_{i+1} and δ_{i+1} are set to a special symbol $\perp$. Both σ_i and δ_{i+1} should contain the current delegation δ_i.
- $\texttt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_{i+1}) \rightarrow 0/1$: On input instantiation M_i, instantiation signature σ_i, public parameters pp, and the public keys $\mathbf{PK}_{i+1}$ of originator P_0 and proxies $P_1, \ldots, P_{i+1}$, the instantiation signature verification algorithm, which is carried out by any verifier (particularly P_{i+1}), outputs "1" if σ_i is valid for M_i under $\mathbf{PK}_i$, which also means that the template and instantiation signatures $\sigma_0, \ldots, \sigma_{i-1}$ are all verified, or outputs "0" otherwise.

The identifier id_T should be passed on from σ_T to every instantiation signature σ_i $(1 \leq i \leq n)$, so as to bind the instantiations to the template. We proceed to define formal security model for basic CBS.

A basic CBS scheme is *correct* in the sense that the template signature, all the instantiation signatures and all delegations can be validated to be true if no user's behavior deviates from the scheme.

Definition 1 (*Correctness*) A basic CBS scheme is *correct* if, for a given $\kappa \in \mathbb{N}$, any maximum template size $d \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \texttt{Setup}(\kappa, d)$, any $(pk_i, sk_i) \leftarrow \texttt{KeyGen}(\kappa, \mathsf{pp})$ of originator P_0 and proxies $P_1, \ldots, P_n$, and any template $\mathbb{T}$, the following conditions hold:

- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_1) = 1$, where σ_T is generated as $(\sigma_T, \delta_1) \leftarrow \texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1)$.
- $\texttt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_{i+1}) = 1$ for every $i \in [1, n]$, where σ_i is generated as $(\sigma_i, \delta_{i+1}) \leftarrow \texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta_i, \mathsf{pp}, sk_i, \mathbf{PK}_{i+1})$. This property not only ensures that M_i is a valid i-level instantiation of template $\mathbb{T}$, but also all the preceding instantiations leading back to template $\mathbb{T}$ are valid.
- Every delegation δ_i $(i \in [1, n])$ generated by $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1)$ and $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta_i, \mathsf{pp}, sk_i, \mathbf{PK}_{i+1})$ is validated to be true in the following instantiation.

A secure basic CBS scheme should ensure that even when originator P_0 colludes with all but one proxy P_π, they can neither create an instantiation with a valid instantiation signature for P_π nor forge a delegation to $P_{\pi+1}$. To capture this collusion attack, in the following formal definition, we assume that a PPT adversary $\mathcal{A}$ controls a corrupted set $\mathbf{P}' = \mathbf{P} \backslash \{P_\pi\}$. The template identifier id_T is included in all the signatures and is not explicitly stated in the following security games.

Definition 2 (*Security Against Originator*) A basic CBS scheme is *secure against the originator* if no PPT adversary $\mathcal{A}$, controlling the originator P_0 and all but one proxy P_π, can win the following game by interacting with a challenger $\mathcal{C}$.

Setup Challenger $\mathcal{C}$ invokes $\texttt{Setup}(\kappa, d)$ with security parameter κ and maximum template size d to obtain public parameters pp. Next, the challenger initializes an empty list $\mathcal{L}$, runs $\texttt{KeyGen}(\kappa, \mathsf{pp})$ to create user P_π's public/private keys (pk_π, sk_π), and gives public information pp and pk_π to $\mathcal{A}$.

Queries Adversary $\mathcal{A}$ adaptively submits *instantiation signing queries* to $\mathcal{C}$. In response to each query $(M_{\pi-1}, M_\pi, \sigma_{\pi-1}, \delta_\pi)$, the challenger validates $\sigma_{\pi-1}$ and δ_π, then produces a pair $(\sigma_\pi, \delta_{\pi+1})$, returns $(\sigma_\pi, \delta_{\pi+1})$ to adversary $\mathcal{A}$ and appends the tuple $(M_{\pi-1}, M_\pi, \sigma_{\pi-1}, \delta_\pi, \sigma_\pi, \delta_{\pi+1})$ to $\mathcal{L}$.

Since delegations are produced along with instantiation signatures, delegation queries need not to be posed separately.

Output Adversary $\mathcal{A}$ outputs a tuple $(M_\pi^*, \sigma_\pi^*, \delta_{\pi+1}^*)$ and wins the game if any of the following cases occurs.

- **Case 1**: The pair (M_π^*, σ_π^*) satisfies the conditions:
  - M_π^* has not been requested in instantiation signing queries with $(M_{\pi-1}, M_\pi^*, \sigma_{\pi-1}, \delta_\pi)$ such that $\texttt{IVrfy}(M_{\pi-1}, \sigma_{\pi-1}, \mathsf{pp}, \mathbf{PK}_\pi) = 1$ and δ_π is valid for P_π;
  - $\texttt{IVrfy}(M_\pi^*, \sigma_\pi^*, \mathsf{pp}, \mathbf{PK}_{\pi+1}) = 1$.
- **Case 2**: The delegation $(M_\pi^*, \delta_{\pi+1}^*)$ (when $\pi \neq n$) satisfies the conditions:
  - The same as Case 1, i.e., M_π^* has not been requested in instantiation signing queries;
  - $\delta_{\pi+1}^*$ can be validated to be true in $P_{\pi+1}$'s instantiation Instn.

A secure basic CBS scheme should ensure that even when all the proxies collude, they cannot forge a valid template signature or a delegation to P_1. To capture this collusion attack, in the following formal definition, we allow a PPT adversary $\mathcal{A}$ to control all the proxies, that is, the controlled user set is $\mathbf{P}' = \mathbf{P} \backslash \{P_0\}$.

Definition 3 (*Security Against Proxies*) A basic CBS scheme is *secure against proxies* if no PPT adversary $\mathcal{A}$, controlling all the proxies, can win the following game by interacting with a challenger $\mathcal{C}$.

Setup Challenger $\mathcal{C}$ invokes $\texttt{Setup}(\kappa, d)$ with security parameter κ and maximum template size d to obtain public parameters pp. Next, the challenger initializes an empty list $\mathcal{L}$, runs $\texttt{KeyGen}(\kappa, \mathsf{pp})$ to create originator P_0's public/private keys (pk_0, sk_0), and sends the public information pp and pk_0 to $\mathcal{A}$.

Queries Adversary $\mathcal{A}$ adaptively submits *template signing queries* to $\mathcal{C}$. Upon receiving a template $\mathbb{T}$ along with some parameters from $\mathcal{A}$, challenger $\mathcal{C}$ produce a pair (σ_T, δ_1) which embeds the received parameters in σ_T. Then $\mathcal{C}$ returns (σ_T, δ_1) to $\mathcal{A}$ and appends the tuple $(\mathbb{T}, \sigma_T, \delta_1)$ to $\mathcal{L}$. As in Definition 2, delegation queries need not be posed separately here.

Output Adversary $\mathcal{A}$ outputs a tuple $(\mathbb{T}^*, \sigma_{T^*}, \delta_1^*)$ and wins the game if any of the following cases occurs.

- **Case 1**: The pair $(\mathbb{T}^*, \sigma_{T^*})$ satisfies
  - $\mathbb{T}^*$ has not been requested in template signing queries;
  - $\texttt{TVrfy}(\mathbb{T}^*, \sigma_{T^*}, \mathsf{pp}, \mathbf{PK}_1) = 1$.
- **Case 2**: The pair $(\mathbb{T}^*, \delta_1^*)$ satisfies
  - $\mathbb{T}^*$ has not been requested in template signing queries;
  - The delegation δ_1^* can be validated to be true in P_1's instantiation Instn.

3.2 Basic CBS construction

In this section, we present a construction of basic CBS. Suppose $\mathcal{SAS} = (\texttt{Setup}, \texttt{KeyGen}, \texttt{SASign}, \texttt{SAVrfy})$ denotes a secure sequential aggregate signature scheme. In the construction, $\mathcal{E}(\cdot)$ denotes the expanded expression of the encoding polynomial (Formula 1) on the template and instantiations.

- $\texttt{Setup}(\kappa, d)$: On input κ and d, choose a bilinear pairing $\hat{e} : G_1 \times G_1 \to G_2$ where $G_1 = \langle g \rangle$ and G_2 are cyclic groups with prime order p. Randomly pick a value $\alpha \in_R Z_p^*$ and compute $u_i = g^{\alpha^i}$ for each $i \in [1, d]$. Choose a collision-resistant hash function $H : \{0, 1\}^* \to Z_p^*$. Invoke $pp' \leftarrow \mathcal{SAS}.\texttt{Setup}(\kappa)$. The public parameters are $\mathsf{pp} = (\hat{e}, G_1, G_2, p, u_0 = g, u_1, \ldots, u_d, H, pp')$.
- $\texttt{KeyGen}(\kappa, \mathsf{pp})$: Invoke $(pk, sk) \leftarrow \mathcal{SAS}.\texttt{KeyGen}(\kappa, pp')$.
- $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1)$: Randomly picking a unique identifier $id_T \in_R \{0, 1\}^\kappa$, carry out the following steps.
  - Compute $\psi_T(x) = \mathcal{E}(\mathbb{T}) \in Z_p[x]$ and a commitment
    $$C = \prod_{j=0}^{s} u_j^{\psi_T^{(j)}} \tag{2}$$
    where $\psi_T^{(j)}$ denotes the j-th coefficient of $\psi_T(x)$.
  - Pick a random value $a \in_R Z_p^*$, compute $\varphi(x) = \frac{\psi_T(x) - \psi_T(a)}{x - a}$ and a witness
    $$\omega = \prod_{j=0}^{s-1} u_j^{\varphi^{(j)}} \tag{3}$$
    where $\varphi^{(j)}$ denotes the j-th coefficient of $\varphi(x)$.
  - Invoke
    $$\tau_T \leftarrow \mathcal{SAS}.\texttt{SASign}(id_T \| \ell \| C \| a \| \omega \| pk_1, \varnothing, sk_0)$$

  Let $PT = (id_T, \ell, C, a, \omega)$ be the public parameters associated with template $\mathbb{T}$. Thus, $\sigma_T = (PT, \tau_T)$ and $\delta_1 = (\tau_T)$. Here, $s = \sum_{i=1}^{\ell} s_i \leq d$.
- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_1)$: Compute $\psi_T(x) = \mathcal{E}(\mathbb{T}) \in Z_p[x]$ and check the following equations:
  $$\hat{e}(C, g) \stackrel{?}{=} \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(g, g)^{\psi_T(a)} \tag{4}$$
  and
  $$\mathcal{SAS}.\texttt{SAVrfy}(id_T \| \ell \| C \| a \| \omega \| pk_1, \tau_T, pk_0) \stackrel{?}{=} 1 \tag{5}$$
  If both equations hold, output "1"; otherwise, output "0".

– $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta_i, \textsf{pp}, sk_i, \mathbf{PK}_{i+1})$: Compute $\bar{\psi}_i(x) = \mathcal{E}(M_{i-1} \backslash M_i) \in Z_p[x]$, where $M_{i-1} \backslash M_i$ denotes the set of choices excluded by P_i. Then calculate $\hbar_i = \bar{\psi}_i(a)$ and $\bar{\omega}_i = \bar{\omega}_{i-1}^{\hbar_i}$, where $\bar{\omega}_0 = g$. Invoke $\tau_i \leftarrow \mathcal{SAS}.\texttt{SASign}(id_T \| \hbar_i \| \bar{\omega}_i \| pk_{i+1}, \tau_{i-1}, sk_i)$. Append $(\hbar_i, \bar{\omega}_i)$ to PT. Thus, $\sigma_i = (PT, \tau_i)$ and $\delta_{i+1} = (\tau_i)$. Notice that if an exchangeable field is delegated to proxy P_{i+1}, then all the corresponding choices should be contained in M_i, the instantiation of P_i. Note that σ_i contains a delegation chain from originator P_0 to proxy P_i.
– $\texttt{IVrfy}(M_i, \sigma_i, \textsf{pp}, \mathbf{PK}_{i+1})$: Compute $\psi_i(x) = \mathcal{E}(M_i) \in Z_p[x]$ and $h_i = \psi_i(a)$. Construct

$$\mathtt{m}_0 = id_T \| \ell \| C \| a \| \omega \| pk_1$$

and for every $j \in [1, i]$ construct

$$\mathtt{m}_j = id_T \| \hbar_j \| \bar{\omega}_j \| pk_{j+1}$$

Let $\mathbf{M}_i = (\mathtt{m}_0, \mathtt{m}_1, \ldots, \mathtt{m}_i)$. Check the following equalities:

$$\hat{e}(C, g)^i \stackrel{?}{=} \hat{e}(\omega, u_1/g^a)^i \cdot \hat{e}\left(g, \prod_{j=1}^{i} \bar{\omega}_j^{\left(\prod_{k=j+1}^{i} \hbar_k\right) h_i}\right) \tag{6}$$

and

$$\mathcal{SAS}.\texttt{SAVrfy}(\mathbf{M}_i, \tau_i, \mathbf{PK}_i) \stackrel{?}{=} 1 \tag{7}$$

If both equalities hold, output "1"; otherwise, output "0".

Theorem 1 *The basic CBS scheme proposed above is correct.*

Proof We first consider the correctness of the template signature σ_T. For any given template $\mathbb{T}$, its template signature is associated with the commitment C and witness ω of an evaluation at some point $a \in_R Z_p^*$ of the corresponding encoded polynomial $\mathcal{E}(\mathbb{T})$, as well as a sequential signature τ_T. Equality (4) holds as shown in [27], which ensures both C and ω are generated on template $\mathbb{T}$. The correctness of equality (5) is directly determined by the underlying sequential aggregate signature scheme $\mathcal{SAS}$.

To prove the correctness of the instantiations and their signatures, we need only to show that Equality (6) holds since Equality (7) holds in the same way as Equality (5). In fact, if all the proxies are honest, the following equality holds for the i-th instantiation M_i:

$$\begin{aligned}
&\hat{e}(\omega, u_1/g^a) \cdot \hat{e}(g, \bar{\omega}_i^{h_i}) \\
&\quad = \hat{e}(\omega, u_1/g^a) \cdot \hat{e}\left(g, g^{\left(\prod_{k=1}^{i} \hbar_k\right) h_i}\right) \\
&\quad = \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(g, g)^{\psi_T(a)} \\
&\quad = \hat{e}(C, g)
\end{aligned}$$

Similarly, for the j-th ($1 \le j < i$) instantiation, the following equality holds:

$$\begin{aligned}
&\hat{e}(\omega, u_1/g^a) \cdot \hat{e}\left(g, \bar{\omega}_j^{\left(\prod_{k=j+1}^{i} \hbar_k\right) h_i}\right) \\
&\quad = \hat{e}(\omega, u_1/g^a) \cdot \hat{e}\left(g, g^{\left(\prod_{k=1}^{j} \hbar_k\right)\left(\prod_{k=j+1}^{i} \hbar_k\right) h_i}\right) \\
&\quad = \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(g, g)^{\psi_T(a)} = \hat{e}(C, g)
\end{aligned}$$

Multiplying respective sides of all these i equalities yields Equality (7).

Since all delegations are in fact the sequential aggregate signatures τ_i, their correctness are ensured by Equalities (5) and (7). □

3.3 Security

Theorem 2 *Suppose H is a collision-resistant hash function. The above proposed basic CBS scheme is secure against originator, assuming the underlying sequential aggregate signature scheme $\mathcal{SAS}$ and polynomial commitment scheme are secure.*

Proof Suppose there is an adversary $\mathcal{A}$ that, having control of originator P_0 and all proxies except P_π, breaks the basic CBS scheme. We show that $\mathcal{A}$ can also break the underlying sequential aggregate signature scheme.

Setup Challenger $\mathcal{C}$ initializes an empty query list $\mathcal{L}$, and proceeds as described in Definition 2.

Queries Adversary $\mathcal{A}$ adaptively issues instantiation signing queries. Upon receiving a tuple $(M_{\pi-1}, M_\pi, \sigma_{\pi-1}, \delta_\pi)$ from $\mathcal{A}$, challenger $\mathcal{C}$ validates $\sigma_{\pi-1}$ by running procedure $\texttt{IVrfy}(M_{\pi-1}, \sigma_{\pi-1}, \textsf{pp}, \mathbf{PK}_\pi)$. Since delegation δ_π is an element of $\sigma_{\pi-1}$, it does not need to be validated separately. If $\sigma_{\pi-1}$ is valid, the challenger invokes:

$$\begin{aligned}(\sigma_\pi, \delta_{\pi+1}) \leftarrow \texttt{Instn}(&M_{\pi-1}, M_\pi, \sigma_{\pi-1}, \\ &\delta_\pi, \textsf{pp}, sk_\pi, \mathbf{PK}_{\pi+1})\end{aligned}$$

returns $(\sigma_\pi, \delta_{\pi+1})$ to adversary $\mathcal{A}$, and appends the tuple $(M_{\pi-1}, M_\pi, \sigma_{\pi-1}, \delta_\pi, \sigma_\pi, \delta_{\pi+1})$ to $\mathcal{L}$. If $\sigma_{\pi-1}$ is invalid, $\mathcal{C}$ returns nothing.

In each query, $\mathcal{A}$ is allowed to choose an identifier id_T and a value a for the queried template $\mathbb{T}$, but the challenger will check the uniqueness of id_T. Thus, all the parameters in $PT_{\pi-1} \in \sigma_{\pi-1}$, e.g., $C, \omega, \hbar_i$ and $\bar{\omega}_i$ $(1 \le i < \pi)$, can be computed by $\mathcal{A}$ using parameters pp. This captures a strong case of attacks, where two queries on the same template will be taken as different if distinct identifiers are chosen by the adversary.

Output Adversary $\mathcal{A}$ wins the game by outputting a tuple $(M_\pi^*, \sigma_\pi^*, \delta_{\pi+1}^*)$. In the proposed scheme, delegation $\delta_{\pi+1}^* = \tau_\pi^*$ is an element of $\sigma_\pi^* = (PT_{\pi-1}^*, \hbar_\pi^*, \bar{\omega}_\pi^*, \tau_\pi^*)$. Thus, we only need to consider Case 1 of Definition 2, that is, in the tuple (M_π^*, σ_π^*), M_π^* has not been requested in the form of $(M_{\pi-1}, M_\pi^*, \sigma_{\pi-1}, \delta_\pi)$ such that both $\texttt{IVrfy}(M_{\pi-1}, \sigma_{\pi-1}, \mathsf{pp}, \mathbf{PK}_\pi) = 1$ and $\texttt{IVrfy}(M_\pi^*, \sigma_\pi^*, \mathsf{pp}, \mathbf{PK}_{\pi+1}) = 1$ hold. There should be a random value $a^* \in Z_p^*$ which is chosen by $\mathcal{A}$. There are two cases to consider.

- **Case 1**: $\texttt{IVrfy}(M_\pi^*, \tilde{\sigma}_\pi, \mathsf{pp}, \mathbf{PK}_{\pi+1}) = 1$ holds, where $(\tilde{M}_\pi, \tilde{\sigma}_\pi)$ is an already queried pair and $M_\pi^* \ne \tilde{M}_\pi$. This covers the case where $\mathcal{A}$ can create a new instantiation of some queried template. Since all of C^*, ω^*, $\hbar_i^*$ and $\bar{\omega}_i^*$ $(1 \le i \le \pi)$ can be publicly computed by $\mathcal{A}$, we further distinguish between two situations according to whether $\psi_\pi^*(x) = \tilde{\psi}_\pi(x)$ holds, where $\psi_\pi^*(x) = \mathcal{E}(M_\pi^*)$ and $\tilde{\psi}_\pi(x) = \mathcal{E}(\tilde{M}_\pi)$.
Situation one: $\psi_\pi^*(x) = \tilde{\psi}_\pi(x)$.
In this situation, M_π^* and $\tilde{M}_\pi$ should have the same size, and their templates have the same length. Recall that both $\psi_\pi^*(x)$ and $\tilde{\psi}_\pi(x)$ are constructed using a hash function H. In fact, $\psi_\pi^*(x) = \tilde{\psi}_\pi(x)$ can be rewritten as the following equality:

$$\prod_{T_{i^*}^* \in M_\pi^*} \prod_{m_{i_j}^* \in T_{i^*}^*} (x + H(\tilde{id}_T \| m_{i_j}^* \| i^*)) = \prod_{\tilde{T}_{\tilde{i}} \in \tilde{M}_\pi} \prod_{\tilde{m}_{i_j} \in \tilde{T}_{\tilde{i}}} (x + H(\tilde{id}_T \| \tilde{m}_{i_j} \| \tilde{i}))$$

To satisfy the equality, $\mathcal{A}$ needs to break the second-preimage resistance property of H with non-negligible probability. Specifically, $\mathcal{A}$ must find distinct $m_{i_j}^*$ and $\tilde{m}_{i_j}$ such that $H(\tilde{id}_T \| m_{i_j}^* \| i^*) = H(\tilde{id}_T \| \tilde{m}_{i_j} \| \tilde{i})$. As H is collision-resistant, this situation cannot happen.
Situation two: $\psi_\pi^*(x) \ne \tilde{\psi}_\pi(x)$.
In this situation, M_π^* and $\tilde{M}_\pi$ may have different sizes, while their templates should have the same length. Since both $(M_\pi^*, \tilde{\sigma}_\pi)$ and $(\tilde{M}_\pi, \tilde{\sigma}_\pi)$ satisfy Equality (6), the following equality must hold:

$$h_\pi^* = \psi_\pi^*(\tilde{a}) = \tilde{\psi}_\pi(\tilde{a}) = \tilde{h}_\pi \tag{8}$$

Equality (8) means that $\mathcal{A}$ is able to manipulate the equation $\psi_\pi^*(x) - \tilde{\psi}_\pi(x) = 0$ such that $\tilde{a}$ is a root. To achieve that, $\mathcal{A}$ must manipulate at least one input of H in respect to M_π^*; that is, $\mathcal{A}$ has to find at least one preimage of H containing $m_{i_j} \in M_\pi^*$ such that:

$$\prod_{T_{i^*}^* \in M_\pi^*} \prod_{m_{i_j}^* \in T_{i^*}^*} (\tilde{a} + H(\tilde{id}_T \| m_{i_j}^* \| i^*)) - \prod_{\tilde{T}_{\tilde{i}} \in \tilde{M}_\pi} \prod_{\tilde{m}_{i_j} \in \tilde{T}_{\tilde{i}}} (\tilde{a} + H(\tilde{id}_T \| \tilde{m}_{i_j} \| \tilde{i})) = 0$$

Thus, $\mathcal{A}$ breaks the preimage resistance property of hash function H.
- **Case 2**: Both M_π^* and σ_π^* are fresh. This case implies that $\mathcal{A}$ successfully forges a valid sequential aggregate signature with depth $\pi + 1$, since all of $C^*, \omega^*, \hbar_i^*$ and $\bar{\omega}_i^*$ $(1 \le i \le \pi)$ can be computed by $\mathcal{A}$ using only public parameters.

Combining the cases, adversary $\mathcal{A}$ can only output a valid forgery with probability $\varepsilon = \varepsilon' + \frac{1}{p}$, where ε' denotes the success probability of attacking the underlying sequential aggregate signature scheme. □

Theorem 3 *Suppose H is a collision-resistant hash function. Our proposed basic CBS scheme is secure against proxies, assuming the underlying sequential aggregate signature scheme $\mathcal{SAS}$ and polynomial commitment scheme are secure.*

Proof Suppose there is an adversary $\mathcal{A}$ that, having control of all the proxies, breaks the basic CBS scheme. We show that $\mathcal{A}$ can also break the underlying sequential aggregate signature scheme.

Setup Challenger $\mathcal{C}$ initializes an empty query list $\mathcal{L}$, and proceeds as described in Definition 3.
Queries Adversary $\mathcal{A}$ adaptively submits template signing queries. Upon receiving a template $\mathbb{T}$ as well as parameters (id_T, a) from adversary $\mathcal{A}$, challenger $\mathcal{C}$ invokes $(\sigma_T, \delta_1) \leftarrow \texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1)$ using the received parameters (id_T, a). Then, $\mathcal{C}$ returns (σ_T, δ_1) to $\mathcal{A}$ and appends the tuple $(\mathbb{T}, \sigma_T, \delta_1)$ to $\mathcal{L}$.
In each query, $\mathcal{A}$ is allowed to choose an identifier id_T and a value a for the queried template $\mathbb{T}$, but the challenger will check the uniqueness of id_T. Thus, the parameters C and ω can be computed by $\mathcal{A}$ using parameters pp. This captures a strong case of attacks, where two queries on the same template will be taken as different if distinct identifiers are chosen by the adversary.
Output Adversary $\mathcal{A}$ wins the game by outputting a tuple $(\mathbb{T}^*, \sigma_{T^*}, \delta_1^*)$. Since the delegation $\delta_1^* = \tau_{T^*}$ is an element of $\sigma_{T^*} = (PT^*, \tau_{T^*})$ in the proposed scheme, we only need to consider Case 1 of Definition 3, i.e., in the

tuple $(\mathbb{T}^*, \sigma_{T^*})$, $\mathbb{T}^*$ has not been requested in the form of $(\mathbb{T}^*, id_{T^*}, a^*)$ such that $\texttt{TVrfy}(\mathbb{T}^*, \sigma_{T^*}, \mathsf{pp}, \mathbf{PK}_1) = 1$ holds. There are two cases to consider.

- **Case 1**: $\texttt{TVrfy}(\mathbb{T}^*, \sigma_{\tilde{T}}, \mathsf{pp}, \mathbf{PK}_1) = 1$ holds, where $(\tilde{\mathbb{T}}, \sigma_{\tilde{T}})$ is an already queried pair and $id_{T^*} \neq id_{\tilde{T}}$. This covers the case where $\mathcal{A}$ can produce a forgery for some queried template but with a different identifier. Since C^* and ω^* can be publicly computed by $\mathcal{A}$, we further distinguish between two situations according to whether $\psi^*(x) = \tilde{\psi}(x)$ holds, where $\psi^*(x) = \mathcal{E}(\mathbb{T}^*)$ and $\tilde{\psi}(x) = \mathcal{E}(\tilde{\mathbb{T}})$.
Situation one: $\psi^*(x) = \tilde{\psi}(x)$. In this situation, $\mathbb{T}^*$ and $\tilde{\mathbb{T}}$ should have the same length and size. Recall that both $\psi^*(x)$ and $\tilde{\psi}(x)$ are constructed using a hash function H. In fact, $\psi^*(x) = \tilde{\psi}(x)$ can be rewritten as the following equality:

$$\prod_{T^*_{i^*} \in \mathbb{T}^*} \prod_{m^*_{i_j} \in T^*_{i^*}} (x + H(id_{\tilde{T}} \| m^*_{i_j} \| i^*)) = \prod_{\tilde{T}_{\tilde{i}} \in \tilde{\mathbb{T}}} \prod_{\tilde{m}_{i_j} \in \tilde{T}_{\tilde{i}}} (x + H(id_{\tilde{T}} \| \tilde{m}_{i_j} \| \tilde{i}))$$

To satisfy the equality, $\mathcal{A}$ needs to break the second-preimage resistance property of H with non-negligible probability. Specifically, $\mathcal{A}$ must find distinct $m^*_{i_j}$ and $\tilde{m}_{i_j}$ such that $H(id_{\tilde{T}} \| m^*_{i_j} \| i^*) = H(id_{\tilde{T}} \| \tilde{m}_{i_j} \| \tilde{i})$. As H is collision-resistant, this situation cannot happen.
Situation two: $\psi^*(x) \neq \tilde{\psi}(x)$.
In this situation, $\mathbb{T}^*$ and $\tilde{\mathbb{T}}$ may have different sizes but the same length. Since both $(\mathbb{T}^*, \sigma_{\tilde{T}})$ and $(\tilde{\mathbb{T}}, \sigma_{\tilde{T}})$ satisfy Equality (4), the following equality must hold:

$$\psi^*(\tilde{a}) = \tilde{\psi}(\tilde{a}) \tag{9}$$

Equality (9) means that $\mathcal{A}$ is able to manipulate the equation $\psi^*(x) - \tilde{\psi}(x) = 0$ such that $\tilde{a}$ is a root. To achieve that, $\mathcal{A}$ must manipulate at least one input of H in respect to $\mathbb{T}^*$; that is, $\mathcal{A}$ has to find at least one preimage of H containing $m_{i_j} \in \mathbb{T}^*$ such that:

$$\prod_{T^*_{i^*} \in \mathbb{T}^*} \prod_{m^*_{i_j} \in T^*_{i^*}} (\tilde{a} + H(id_{\tilde{T}} \| m^*_{i_j} \| i^*)) - \prod_{\tilde{T}_{\tilde{i}} \in \tilde{\mathbb{T}}} \prod_{\tilde{m}_{i_j} \in \tilde{T}_{\tilde{i}}} (\tilde{a} + H(id_{\tilde{T}} \| \tilde{m}_{i_j} \| \tilde{i})) = 0$$

Thus, $\mathcal{A}$ breaks the preimage resistance property of hash function H.
- **Case 2**: Both $\mathbb{T}^*$ and σ_{T^*} are fresh. This case implies that $\mathcal{A}$ successfully forges a valid sequential aggregate signature with depth 1, since C^* and ω^* can be computed by $\mathcal{A}$ using only public parameters.

Overall, adversary $\mathcal{A}$ can only output a valid forgery with probability $\varepsilon = \varepsilon' + \frac{1}{p}$, where ε' denotes the success probability of attacking the underlying sequential aggregate signature scheme. □

4 Verifiably encrypted cascade-instantiable blank signature

This section extends the basic CBS scheme of Section 3 to *verifiably encrypted cascade-instantiable blank signatures*. In verifiably encrypted CBS, there is an arbitrator in addition to a set $\mathbf{P}$ of originator and proxies. The originator encrypts her signed commitment on the encoded template using the public key of the arbitrator. This encryption not only preserves verifiability of subset relationship between template and instantiations, as in basic CBS, but also allows the arbitrator to intervene to recover the signed commitment of the originator in case of dispute. If the originator does not cheat, then the resolution of signed commitment by the arbitrator should pass the verification using the originator's parameters.

We define the framework and the security model of verifiably encrypted CBS and then provide a construction along with security proofs.

4.1 Definitions and security model

Formally, a verifiably encrypted CBS scheme comprises the following algorithms:

- $\texttt{Setup}(\kappa, d) \rightarrow (sk_A, \mathsf{pp})$: On input a security parameter $\kappa \in \mathbb{N}$ and the maximum template size $d \in \mathbb{N}$, the setup algorithm, which is carried out by the arbitrator, outputs a private key sk_A and public parameters pp.
- $\texttt{UKeyGen}(\kappa, \mathsf{pp}) \rightarrow (pk_i, sk_i)$: On input security parameter $\kappa \in \mathbb{N}$ and public parameters pp, the user key generation algorithm, which is carried out individually by the originator P_0 and proxies $P_1, \ldots, P_n$, outputs a pair of public/private keys (pk_i, sk_i).
- $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1) \rightarrow (\sigma_T, \delta_1)$: The same as in Sect. 3.1, where σ_T contains an encrypted commitment Π on the encoded template $\mathbb{T}$.
- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_1) \rightarrow 0/1$: The same as in Sect. 3.1.
- $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta_i, \mathsf{pp}, sk_i, \mathbf{PK}_{i+1}) \rightarrow (\sigma_i, \delta_{i+1})$: The same as in Sect. 3.1.
- $\texttt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_{i+1}) \rightarrow 0/1$: The same as in Sect. 3.1.
- $\texttt{Resolve}(sk_A, \mathsf{pp}, \mathbf{PK}_1, \mathbb{T}, \sigma_T) \rightarrow \mathbb{C}/\perp$: On input the arbitrator's private key sk_A, public parameters pp, the

originator's public key pk_0 and proxy P_1's public key pk_1, a template $\mathbb{T}$ and template signature σ_T, the resolution algorithm, which is carried out by the arbitrator, outputs the randomized commitment $\mathbb{C}$ for $\mathbb{T}$ if σ_T is valid for $\mathbb{T}$ under pk_0 or $\perp$ otherwise.

A verifiably encrypted CBS scheme is *correct* in the sense that the template signature, all the instantiation signatures, all delegations and the resolved commitment can be validated to be true if no user's behavior deviates from the scheme.

Definition 4 (*Correctness*) A verifiably encrypted CBS scheme is *correct* if, for a given $\kappa \in \mathbb{N}$, any maximum template size $d \in \mathbb{N}$, any $(sk_A, \mathsf{pp}) \leftarrow \texttt{Setup}(\kappa, d)$, any $(pk_i, sk_i) \leftarrow \texttt{UKeyGen}(\kappa, \mathsf{pp})$ of originator P_0 and proxies $P_1, \ldots, P_n$, and any template $\mathbb{T}$, the following conditions hold:

- The first three conditions are the same as in Definition 1;
- The resolved commitment $\mathbb{C} \leftarrow \texttt{Resolve}(sk_A, \mathsf{pp}, \mathbf{PK}_1, \mathbb{T}, \sigma_T)$ is valid for the encoded polynomial of template $\mathbb{T}$.

A secure verifiably encrypted CBS scheme should ensure that even when originator P_0 colludes with the arbitrator and all but one proxy P_π, they can neither create an instantiation with a valid instantiation signature for P_π nor forge a delegation to $P_{\pi+1}$.

Definition 5 (*Security Against Colluding Originator*) A verifiably encrypted CBS scheme is *secure against the colluding originator* if no PPT adversary $\mathcal{A}$, controlling the originator P_0, the arbitrator and all but one proxy P_π, can win the following game by interacting with a challenger $\mathcal{C}$.

Setup With the public parameters pp outputted by $\mathcal{A}$, challenger $\mathcal{C}$ creates user P_π's public/private keys (pk_π, sk_π) and gives pk_π to $\mathcal{A}$.

Queries As in Definition 2, adversary $\mathcal{A}$ adaptively submits *instantiation signing queries* to challenger $\mathcal{C}$.

Output Adversary $\mathcal{A}$ outputs a tuple $(M^*_\pi, \sigma^*_\pi, \delta^*_{\pi+1})$ and wins the game under the same conditions as in Definition 2.

A secure verifiably encrypted CBS scheme should ensure that even when all the proxies collude with the arbitrator, they cannot forge a valid template signature or a delegation to P_1.

Definition 6 (*Security Against Colluding Proxies*) A verifiably encrypted CBS scheme is *secure against colluding proxies* if no PPT adversary $\mathcal{A}$, controlling all the proxies and the arbitrator, can win the following game by interacting with a challenger $\mathcal{C}$.

Setup With the public parameters pp outputted by $\mathcal{A}$, challenger $\mathcal{C}$ creates originator P_0's public/private keys (pk_0, sk_0) and gives pk_0 to $\mathcal{A}$.

Queries As in Definition 3, adversary $\mathcal{A}$ adaptively submits *template signing queries* to $\mathcal{C}$.

Output Adversary $\mathcal{A}$ outputs a tuple $(\mathbb{T}^*, \sigma_{T^*}, \delta^*_1)$ and wins the game under the same conditions as in Definition 3.

4.2 A verifiably encrypted CBS construction

We present a verifiably encrypted CBS construction based on the basic CBS scheme, where $\mathcal{SAS} = (\texttt{Setup}, \texttt{KeyGen}, \texttt{SASign}, \texttt{SAVrfy})$ also denotes a secure sequential aggregate signature scheme.

- $\texttt{Setup}(\kappa, d)$: On input κ and d, choose a bilinear pairing $\hat{e} : G_1 \times G_1 \to G_2$ where $G_1 = \langle g \rangle$ and G_2 are cyclic groups with prime order p. Randomly pick a value $\alpha \in_R Z^*_p$ and compute $u_i = g^{\alpha^i}$ for each $i \in [1, d]$. Randomly pick a value $x_A \in_R Z^*_p$ and compute $y_A = g^{x_A}$. Choose a collision-resistant hash function $H : \{0,1\}^* \to Z^*_p$. Invoke $pp' \leftarrow \mathcal{SAS}.\texttt{Setup}(\kappa)$. The private key is $sk_A = x_A$ while the public parameters are $\mathsf{pp} = (\hat{e}, G_1, G_2, p, u_0 = g, u_1, \ldots, u_d, y_A, H, pp')$.
- $\texttt{UKeyGen}(\kappa, \mathsf{pp})$: Invoke $\mathcal{SAS}.\texttt{KeyGen}(\kappa, pp')$ to obtain (pk_i, sk_i).
- $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, pk_1)$: Randomly picking a unique identifier $id_T \in_R \{0,1\}^\kappa$ and two values $\beta, \gamma \in_R Z^*_p$, carry out the following steps.
  - Compute $\psi_T(x) = \mathcal{E}(\mathbb{T}) \in Z_p[x]$ and an encrypted commitment $C = (C_1, C_2, C_3)$,

$$C_1 = \left(\prod_{j=0}^{s} u_j^{\psi_T^{(j)}}\right)^{\beta} \cdot y_A^{\gamma}, \;\; C_2 = g^{\beta}, \;\; C_3 = g^{\gamma} \tag{10}$$

    where $\psi_T^{(j)}$ denotes the j-th coefficient of $\psi_T(x)$.
  - Pick a random value $a \in_R Z^*_p$, compute $\varphi(x) = \frac{\psi_T(x) - \psi_T(a)}{x-a}$ and a signed witness

$$\omega = \left(\prod_{j=0}^{s-1} u_j^{\varphi^{(j)}}\right)^{\beta} \tag{11}$$

    where $\varphi^{(j)}$ denotes the j-th coefficient of $\varphi(x)$.
  - Invoke

$$\tau_T \leftarrow \mathcal{SAS}.\texttt{SASign}(id_T \| \ell \| C \| a \| \omega \| pk_1, \varnothing, sk_0).$$

  Let $PT = (id_T, \ell, C, a, \omega)$ be the public parameters associated with template $\mathbb{T}$. Thus, $\sigma_T = (PT, \tau_T)$ and $\delta_1 = (\tau_T)$. Here, $s = \sum_{i=1}^{\ell} s_i \le d$.

– $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_1)$: Compute $\psi_T(x) = \mathcal{E}(\mathbb{T}) \in Z_p[x]$ and check the following equation as well as equation (5):

$$\hat{e}(C_1, g) \stackrel{?}{=} \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_2, g)^{\psi_T(a)} \cdot \hat{e}(C_3, y_A) \tag{12}$$

If both equations hold, output "1"; otherwise, output "0".

– $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta_i, \mathsf{pp}, sk_i, \mathbf{PK}_{i+1})$: The same as in the basic CBS construction in Sect. 3.2.

– $\texttt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_{i+1})$: Compute $\psi_i(x) = \mathcal{E}(M_i) \in Z_p[x]$ and $h_i = \psi_i(a)$. Construct

$$\mathtt{m}_0 = id_T \| \ell \| C \| a \| \omega \| pk_1$$

and for every $j \in [1, i]$ construct

$$\mathtt{m}_j = id_T \| \hbar_j \| \bar{\omega}_j \| pk_{j+1}$$

Let $\mathbf{M}_i = (\mathtt{m}_0, \mathtt{m}_1, \ldots, \mathtt{m}_i)$. Check the following equality as well as Eq. (7):

$$\hat{e}(C_1, g)^i \stackrel{?}{=} (\hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_3, y_A))^i \cdot \hat{e}\left(C_2, \prod_{j=1}^{i} \bar{\omega}_j^{\left(\prod_{k=j+1}^{i} \hbar_k\right) h_i}\right) \tag{13}$$

If both equalities hold, output "1"; otherwise, output "0".

– $\texttt{Resolve}(sk_A, \mathsf{pp}, \mathbf{PK}_1, \mathbb{T}, \sigma_T)$: If both equalities (12) and (5) hold, output the signed commitment $\mathbb{C} = C_1/C_2^{x_A}$; otherwise, output $\perp$.

Theorem 4 *The verifiably encrypted CBS scheme proposed above is correct.*

Proof Building on Theorem 1, we need only to prove the correctness of Equalities (12) and (13) and the resolution of algorithm $\texttt{Resolve}$.

If the originator is honest, the following equality holds for template $\mathbb{T}$:

$$\begin{aligned}\hat{e}(C_1, g) &= \hat{e}\left(\left(\prod_{j=0}^{s} u_j^{\psi_T^{(j)}}\right)^{\beta}, g\right) \cdot \hat{e}(y_A^{\gamma}, g)\\ &= \hat{e}(\omega, g^{\alpha - a}) \cdot \hat{e}(g^{\beta}, g^{\psi_T(a)}) \cdot \hat{e}(y_A, g^{\gamma})\\ &= \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_2, g)^{\psi_T(a)} \cdot \hat{e}(C_3, y_A)\end{aligned}$$

Also, if all the proxies are honest, the following equality holds for the i-th instantiation M_i:

$$\begin{aligned}&\hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_3, y_A) \cdot \hat{e}(C_2, \bar{\omega}_i^{h_i})\\ &\quad = \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_2, g^{(\prod_{k=1}^{i} \hbar_k) h_i}) \cdot \hat{e}(C_3, y_A)\\ &\quad = \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_2, g)^{\psi_T(a)} \cdot \hat{e}(C_3, y_A)\\ &\quad = \hat{e}(C_1, g)\end{aligned}$$

Similarly, for the j-th ($1 \le j < i$) instantiation, the following equality holds:

$$\begin{aligned}&\hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_3, y_A) \cdot \hat{e}\left(C_2, \bar{\omega}_j^{\left(\prod_{k=j+1}^{i} \hbar_k\right) h_i}\right)\\ &= \hat{e}(\omega, u_1/g^a) \cdot \hat{e}\left(C_2, g^{\left(\prod_{k=1}^{j} \hbar_k\right)\left(\prod_{k=j+1}^{i} \hbar_k\right) h_i}\right)\\ &\quad \cdot \hat{e}(C_3, y_A)\\ &= \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_2, g)^{\psi_T(a)} \cdot \hat{e}(C_3, y_A)\\ &= \hat{e}(C_1, g)\end{aligned}$$

Multiplying respective sides of all these i equalities yields Equality (13).

By algorithm $\texttt{Resolve}$, we have

$$\mathbb{C} = C_1/C_2^{x_A} = \left(\prod_{j=0}^{s} u_j^{\psi_T^{(j)}}\right)^{\beta}$$

which satisfies

$$\begin{aligned}\hat{e}(\mathbb{C}, g) &= \hat{e}\left(\left(\prod_{j=0}^{s} u_j^{\psi_T^{(j)}}\right)^{\beta}, g\right)\\ &= \hat{e}(\omega, u_1/g^a) \cdot \hat{e}(C_2, g)^{\psi_T(a)}\end{aligned}$$

It indicates that $\mathbb{C}$ is a signed commitment for encoded template $\mathbb{T}$ by the originator. □

4.3 Security

Theorem 5 *Suppose H is a collision-resistant hash function. The proposed verifiably encrypted CBS scheme is secure against colluding originator, assuming the underlying sequential aggregate signature scheme $\mathcal{SAS}$ and polynomial commitment scheme are secure.*

Theorem 6 *Suppose H is a collision-resistant hash function. Our proposed verifiably encrypted CBS scheme is secure against colluding proxies, assuming the underlying sequential aggregate signature scheme $\mathcal{SAS}$ and polynomial commitment scheme are secure.*

We omit the proofs of the two theorems here as they are similar to Theorems 2 and 3, respectively.

Table 1 Computation costs of the CBS schemes

Algorithm	Computation costs	
	Basic CBS scheme	Verifiably encrypted CBS scheme
`Setup`	dE_G	$(d+1)E_G$
`KeyGen`	E_K	E_K
`TSign`	$(2s-1)E_{G_1} + E_S$	$(2s+4)E_{G_1} + E_S$
`TVrfy`	$1E_{G_1} + 1E_{G_2} + 3E_{pr} + E_V$	$1E_{G_1} + 1E_{G_2} + 4E_{pr} + E_V$
`Instn`	$1E_{G_1} + E_S$	$1E_{G_1} + E_S$
`IVrfy`	$(i+1)E_{G_1} + 2E_{G_2} + 3E_{pr} + E_V$	$(i+1)E_{G_1} + 2E_{G_2} + 4E_{pr} + E_V$

Table 2 Element size of the CBS schemes

Element	Size	
	Basic CBS scheme	Verifiably encrypted CBS scheme
Template signature σ_T	$\kappa + 2\|G_1\| + 2\|Z_p\| + S_{SAS}$	$\kappa + 4\|G_1\| + 2\|Z_p\| + S_{SAS}$
Instantiation signature σ_i	$\kappa + (i+2)\|G_1\| + (i+2)\|Z_p\| + S_{SAS}$	$\kappa + (i+4)\|G_1\| + (i+2)\|Z_p\| + S_{SAS}$
Delegation δ_i	S_{SAS}	S_{SAS}

4.4 Efficiency analysis

The computation costs of the basic CBS and verifiably encrypted CBS schemes are summarized and compared in Table 1 in terms of exponentiation and pairing, the two types of time-consuming computation. In the table, E_{G_1}, E_{G_2} and E_{pr} denote the evaluation cost of exponentiation over group G_1 and G_2, and pairing e, respectively. We use E_K, E_S and E_V to represent the cost of $\mathcal{SAS}$.`KeyGen`, $\mathcal{SAS}$.`SASign` and $\mathcal{SAS}$.`SAVrfy`, respectively. The efficiency of the setup algorithm depends on the maximum template size d, that is, it takes d exponentiations over group G_1 in the basic CBS scheme since $u_i = u_{i-1}^{\alpha}$, while the verifiably encrypted CBS scheme incurs one more exponentiation in computing y_A. Both the template signing algorithm and instantiation signature verification algorithm require a linear number of exponentiations, with the multiple being the template size s and the proxy number i of $\{P_1, \ldots, P_i\}$, respectively.

As shown in Table 2, in both the basic and verifiably encrypted CBS schemes, the template signature σ_T has constant size, which consists of one κ-bit identifier id_T, two/four group elements of G_1, two values in Z_p^* and one signature from the underlying sequential aggregate signature scheme. Here, the template length ℓ is treated as an element of Z_p^*. Two additional elements of G_1 are introduced by C in the verifiably encrypted CBS scheme. Compared to σ_T, the instantiation signature σ_i contains additional elements $\{\hbar_j, \bar{\omega}_j : 1 \leq j \leq i\}$ that are accumulated from P_1 to P_i. Finally, every delegation is effected with only one sequential aggregate signature in both schemes.

5 Extensions

In this section, we extend the basic CBS scheme to support other practical application scenarios. To avoid repeating the formal models and corresponding constructions, we only present brief discussions focusing on the differences from basic CBS. Note that these extensions can be further extended into verifiably encrypted counterpart schemes.

5.1 Cascade-and-designated-instantiable blank signature

In basic CBS, each proxy in the delegation chain has total freedom to not only create an instantiation, but also narrow his successor's choices. For example, he may choose nothing and pass the received instantiation intact to his successor, or exclude some choices and send down the remain ones. In certain applications, each proxy should possess only limited instantiation capability; in particular, the originator should be able to designate which proxy along the chain is to instantiate specific fields in the template. The designated fields associated with different proxies are disjoint. To support such applications, we extend the basic CBS to cascade-and-designated-instantiable blank signature as formalized below.

- `Setup`$(\kappa, d) \rightarrow \mathsf{pp}$: The same as in Sect. 3.1.
- `KeyGen`$(\kappa, \mathsf{pp}) \rightarrow (pk, sk)$: The same as in Sect. 3.1.
- `TSign`$(\mathbb{T}, \mathsf{pp}, sk_0, \mathbf{PK}_n) \rightarrow (\sigma_T, \delta)$: On input a template $\mathbb{T}$, public parameters pp, the originator's private key sk_0 and public keys $\mathbf{PK}_n$, the template signing algorithm, which is carried out by the originator, outputs a

signature σ_T for the template and a delegation δ for all proxies. A unique identifier id_T for template $\mathbb{T}$ is generated and embedded in σ_T.
- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_n) \rightarrow 0/1$: On input a template $\mathbb{T}$, template signature σ_T, public parameters pp and public keys $\mathbf{PK}_n$, the template signature verification algorithm, which is carried out by any verifier (particularly P_1), outputs "1" if σ_T is valid for $\mathbb{T}$ under pk_0 or "0" otherwise.
- $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta, \mathsf{pp}, sk_i, \mathbf{PK}_n) \rightarrow \sigma_i$: On input proxy P_{i-1}'s instantiation M_{i-1}, proxy P_i's instantiation M_i, instantiation signature σ_{i-1} produced by P_{i-1}, delegation δ, public parameters pp, P_i's private key sk_i and public keys $\mathbf{PK}_n$, the instantiation algorithm, which is carried out by P_i, outputs an instantiation signature σ_i for M_i if both σ_{i-1} and δ are valid. Here, for $1 \le i \le n$, M_i is a subset of M_{i-1}, $M_0 = \mathbb{T}$, and $\sigma_0 = \sigma_T$. $M_{i-1} \backslash M_i$ contains all the choices that are excluded by proxy P_i.
- $\texttt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_n) \rightarrow 0/1$: On input instantiation M_i, instantiation signature σ_i, public parameters pp, and the public keys $\mathbf{PK}_n$, the instantiation signature verification algorithm, which is carried out by any verifier (particularly P_{i+1}), outputs "1" if σ_i is valid for M_i under $\mathbf{PK}_i$, which also means that the template and instantiation signatures $\sigma_0, \ldots, \sigma_{i-1}$ are all verified, or "0" otherwise.

The security model of cascade-and-designated-instantiable blank signatures is similar to that of basic CBS, with the following revision to the *correctness* requirement. The *security against originator* requirement is as in Definition 2, except there is no Case 2 in the adversary's output. The *security against proxies* property follows Definition 3.

Definition 7 (*Correctness*) A cascade-and-designated-instantiable blank signature scheme is *correct* if, for a given $\kappa \in \mathbb{N}$, any maximum template size $d \in \mathbb{N}$, any $\mathsf{pp} \leftarrow \texttt{Setup}(\kappa, d)$, any $(pk_i, sk_i) \leftarrow \texttt{KeyGen}(\kappa, \mathsf{pp})$ of originator P_0 and proxies $P_1, \ldots, P_n$, and any template $\mathbb{T}$, the following conditions hold:

- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_n) = 1$, where σ_T is generated as $(\sigma_T, \delta) \leftarrow \texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, \mathbf{PK}_n)$.
- $\texttt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_n) = 1$ for every $i \in [1, n]$, where $\sigma_i \leftarrow \texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta, \mathsf{pp}, sk_i, \mathbf{PK}_n)$.
- The delegation δ generated by $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, \mathbf{PK}_n)$ is validated to be true by all proxies.
- Every proxy has only instantiation rights on the designated exchangeable fields.

We proceed to present a construction. For a given template $\mathbb{T} = \{T_i = \{m_{i,1}, \ldots, m_{i,s_i}\} : 1 \le i \le \ell\}$, each exchangeable field T_i has an associated proxy P_{i_u}. To simplify the notation, we associate the fixed fields with originator P_0. The template is encoded in the form:

$$\mathcal{E}'(\mathbb{T}) = \prod_{i=1}^{\ell} \prod_{m \in T_i} (x + H(id_T \| m \| i \| P_{i_u})), \tag{14}$$

where id_T is an unique identifier of $\mathbb{T}$, and $H : \{0, 1\}^* \rightarrow Z_p^*$ is a collision-resistant hash function. In the system, the template should be transmitted in the form $\mathbb{T} = \{(T_i, P_{i_u}) : 1 \le i \le \ell\}$, where P_{i_u} can be either an identity or a public key. Instantiations are encoded and transmitted in a similar way as the template.

- $\texttt{Setup}(\kappa, d)$: The same as in Sect. 3.2.
- $\texttt{KeyGen}(\kappa, \mathsf{pp})$: The same as in Sect. 3.2.
- $\texttt{TSign}(\mathbb{T}, \mathsf{pp}, sk_0, \mathbf{PK}_n)$: Randomly choosing a unique identifier $id_T \in_R \{0, 1\}^\kappa$, carry out the following steps.
  - Compute $\psi_T(x) = \mathcal{E}'(\mathbb{T}) \in Z_p[x]$ and a commitment C as in Eq. (2).
  - Pick a random value $a \in_R Z_p^*$, and compute a witness ω as in Eq. (3).
  - Invoke

$$\tau_T \leftarrow \mathcal{SAS}.\texttt{SASign}(id_T \| \ell \| s_1 \| \ldots \| s_\ell \| C \| a \| \omega \| pk_1, \varnothing, sk_0)$$

  Let $PT = (id_T, \ell, s_1, \ldots, s_\ell, C, a, \omega)$ be the public parameters associated with template $\mathbb{T}$. Thus, $\sigma_T = (PT, \tau_T)$ and $\delta = (\tau_T)$. Here, $s = \sum_{i=1}^{\ell} s_i \le d$.
- $\texttt{TVrfy}(\mathbb{T}, \sigma_T, \mathsf{pp}, \mathbf{PK}_n)$: Compute $\psi_T(x) = \mathcal{E}'(\mathbb{T}) \in Z_p[x]$. Check Eq. (4) and the following condition:

$$\mathcal{SAS}.\texttt{SAVrfy}(id_T \| \ell \| s_1 \| \ldots \| s_\ell \| C \| a \| \omega \| pk_1, \tau_T, pk_0) \stackrel{?}{=} 1 \tag{15}$$

  If both equations hold, output "1"; otherwise, output "0".
- $\texttt{Instn}(M_{i-1}, M_i, \sigma_{i-1}, \delta, \mathsf{pp}, sk_i, \mathbf{PK}_n)$: Construct a pattern vector $\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,\ell})$, where $s_{i,j} = |T_j|$ for each field T_j in M_i. Specifically, $s_{i,j} = 1$ for all fixed fields, $s_{i,j}$ equals to the number of choices in instantiated exchangeable fields T_j presented to proxies $\{P_1, \ldots, P_i\}$, and $s_{i,j} = s_j$ otherwise. Compute $\bar{\psi}_i(x) = \mathcal{E}'(M_{i-1} \backslash M_i) \in Z_p[x]$. Then calculate $\hbar_i = \bar{\psi}_i(a)$ and $\bar{\omega}_i = \bar{\omega}_{i-1}^{\hbar_i}$, where $\bar{\omega}_0 = g$. Invoke

$$\tau_i \leftarrow \mathcal{SAS}.\texttt{SASign}(id_T \| \hbar_i \| \bar{\omega}_i \| \mathbf{s}_i \| pk_{i+1}, \tau_{i-1}, sk_i)$$

  Append $(\hbar_i, \bar{\omega}_i)$ to PT. Thus, $\sigma_i = (PT, \tau_i)$. It is not necessary to include vector $\mathbf{s}_i$ in PT, since it can be recovered from M_i and $\{s_1, \ldots, s_\ell\}$ by a verifier (including proxy P_{i+1}).

– $\mathtt{IVrfy}(M_i, \sigma_i, \mathsf{pp}, \mathbf{PK}_n)$: Compute $\psi_i(x) = \mathcal{E}'(M_i) \in Z_p[x]$ and $h_i = \psi_i(a)$. Construct

$$\mathtt{m}_0 = id_T \| \ell \| s_1 \| \ldots \| s_\ell \| C \| a \| \omega \| pk_1$$

and for every $j \in [1, i]$ construct

$$\mathtt{m}_j = id_T \| \hbar_j \| \bar{\omega}_j \| \mathbf{s}_j \| pk_{j+1}$$

Let $\mathbf{M}_i = (\mathtt{m}_0, \mathtt{m}_1, \ldots, \mathtt{m}_i)$. Check Equalities (6) and (7). If both equalities hold, output "1"; otherwise, output "0".

The security results below follow the derivations in Sect. 3.3.

Corollary 1 *The cascade-and-designated-instantiable blank signature scheme proposed above is correct.*

The fourth correctness requirement, i.e., each proxy can only instantiate the designated exchangeable fields, is achieved by introducing pattern vector $\mathbf{s}_i$. If some proxy P_i makes choices beyond his designated fields, the resultant pattern vector would differ from the one constructed by the verifier according to the originator's specification. Thus, the proxy's dishonest behavior can be detected by validating the sequential aggregate signature σ_i.

Corollary 2 *Suppose H is a collision-resistant hash function. The cascade-and-designated-instantiable blank signature scheme proposed above is secure against originator, assuming the underlying sequential aggregate signature scheme $\mathcal{SAS}$ and polynomial commitment scheme are secure.*

Corollary 3 *Suppose H is a collision-resistant hash function. The cascade-and-designated-instantiable blank signature scheme proposed above is secure against proxies, assuming the underlying sequential aggregate signature scheme $\mathcal{SAS}$ and polynomial commitment scheme are secure.*

5.2 Cascade-and-freely-instantiable blank signature

Consider a scenario where the originator and proxies do not specify their successors. This allows anyone to generate an instantiation from an existing instantiation. The solution, which may be seen as a relaxed version of basic CBS, involves removing pk_1 and pk_{i+1} from Algorithms `TSign` and `Instn`, respectively. We note that in the construction, $\mathcal{SAS}$ cannot be substituted by an aggregate signature scheme $\mathcal{AS}$. The reason is not only that the instantiation order needs to be preserved, which is realized by $\mathcal{SAS}$, but also because $\mathcal{AS}$ has a different aggregate mechanism that requires all the instantiation signatures to be produced and combined together.

5.3 Cascade-instantiable blank signature with template privacy

In the original blank signature scheme of Hanser and Slamanig [19], the template satisfies indistinguishability property against an external adversary. That is, in the challenge phase of the security game for template privacy, the challenger randomly chooses two distinct templates sharing some common fields, signs and gives the templates and signatures to the adversary. The adversary is then allowed to issue instantiation queries on the common fields. The scheme ensures that the adversary cannot distinguish between the two challenge template signatures at the end of the game.

It is easy to adapt our basic CBS scheme to provide such template indistinguishability, as follows. In `TSign`, the originator P_0 picks a random value $\rho \in_R Z_p^*$ for each template, raises both C and ω to the power of ρ, and inserts g^ρ into PT. In this sense, ρ is a template-dependant private key, which should be known to the highest level proxy P_1. Similar to [19], the template signature in the resultant scheme is privately verifiable by proxy P_1. When the scheme is applied with just the originator and one proxy, it achieves exactly the same functionality of the original scheme in [19]; hence, our scheme is strictly the more general between the two. At the same time, our scheme allows the signatures of template and instantiation(s) to be sequentially aggregated and verified concurrently, which is more efficient than verifying them separately as in [19].

5.4 Cascade-instantiable blank signature secure against key exposure

Schuldt et al. [41] investigated multi-level proxy signatures with security against proxy-key exposure. Applying their scheme to generate the delegation chain leads to a CBS scheme that enjoys the same security property. However, this strong security property also brings with it some disadvantages. For example, the delegation procedure necessitates interactions among the users. Moreover, since the delegations are separately generated, the template/instantiation signature sizes increase correspondingly.

5.5 Identity-based cascade-instantiable blank signature

In identity-based (ID) crypto-systems, a user's identity is his public key. These schemes could alleviate the burden of maintaining public key certificates. Many cryptographic primitives in identity-based setting have been proposed to date. In an ID-based CBS model, algorithm `KeyGen` would be replaced by a key extraction algorithm `KeyExt` which takes a user's identity and produces a private key. All the public keys in algorithms `TSign`, `TVrfy`, `Instn` and `IVrfy` would then be replaced by the corresponding user identities. Together with an identity-based sequential aggregate signature scheme such as the one in [6], we can derive an identity-based CBS construction.

6 Conclusion

Blank signature schemes possess the notable feature that a proxy has total freedom to create an instantiation of a template of exchangeable fields under the originators's explicit regulation, with the originator and proxy signing the template and instantiation respectively. This paper proposed a basic cascade-instantiable blank signature (CBS) to cater to more complex application scenarios involving a sequence of proxies. Here, each proxy in a delegation chain creates from her direct predecessor's template/instantiation a new instantiation that narrows the successors' choices for the exchangeable fields. We also formalize a new notion of verifiably encrypted CBS that provides for an arbitrator in case of dispute with the originator. Both CBS constructions are built on polynomial commitment and sequential aggregate signature. The constructions are formally proved to be secure against collusion attacks, and enjoy linear computation costs. We also describe several extensions of the basic CBS to cater to additional real-world applications.

In creating an instantiation, the proxy makes choices in the exchangeable fields in her direct predecessor's template or instantiation. Thus, each instantiation is in fact a "subset" of the template or previous instantiations. This paper, following [19], encodes the template and instantiations as polynomials in such a way that the subset relationship is transformed into a multiplicative sub-polynomial. Accordingly, the polynomial commitment scheme [27] is employed to ensure that the relationship is preserved. It would be interesting to find other secure and more efficient ways to capture the subset relationship, which may require different template encoding approaches. Another avenue for future work is to remove the underlying polynomial commitment scheme which is designed specifically on symmetric bilinear groups, and realize CBS over common cyclic groups or asymmetric bilinear groups.

Acknowledgements This work is supported by Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012.

References

1. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 7–17. CCS'97, ACM, New York, NY, USA (1997)
2. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Nyberg, K. (ed.) Advances in Cryptology-EUROCRYPT'98. LNCS, vol. 1403. Springer, Heidelberg (1998)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Anitizable signatures. In: di Vimercati, S., Syverson, P., Gollmann, D. (eds.) Computer Security-ESORICS 2005, LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
4. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line TTP. In: 1998 IEEE Symposium on Security and Privacy, 1998, Proceedings, pp. 77–85 (1998)
5. Bao, F., Deng, R.H., Ding, X., Lai, J., Zhao, Y.: Hierarchical identity-based chameleon hash and its applications. In: Lopez, J., Tsudik, G. (eds.) Applied Cryptography and Network Security, LNCS, vol. 6715, pp. 201–219. Springer, Heidelberg (2011)
6. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 276–285. CCS'07, ACM, New York, NY, USA (2007)
7. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. J. Cryptol. **25**(1), 57–115 (2012)
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) Advances in Cryptology-EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
9. Brzuska, C., Busch, H., Dagdelen, O., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., Schröder, D.: Redactable signatures for tree-structured data: definitions and constructions. In: Zhou, J., Yung, M. (eds.) Applied Cryptography and Network Security, LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
10. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography-PKC 2009, LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
11. Brzuska, C., Fischlin, M., Lehmann, A., Schröder, D.: Unlinkability of sanitizable signatures. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography-PKC 2010, LNCS, vol. 6056, pp. 444–461. Springer, Heidelberg (2010)
12. Canard, S., Jambert, A.: On extended sanitizable signature schemes. In: Pieprzyk, J. (ed.) Topics in Cryptology-CT-RSA 2010, LNCS, vol. 5985, pp. 179–194. Springer, Heidelberg (2010)
13. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) Applied Cryptography and Network Security, LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
14. Chang, E.C., Lim, C.L., Xu, J.: Short redactable signatures using random trees. In: Fischlin, M. (ed.) Topics in Cryptology-CT-RSA 2009, LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)
15. Derler, D., Hanser, C., Slamanig, D.: Blank digital signatures: optimization and practical experiences. In: Camenisch, J., Fischer-Hübner, S., Hansen, M. (eds.) Privacy and Identity Management for the Future Internet in the Age of Globalisation, IFIP Advances in Information and Communication Technology, vol. 457, pp. 201–215. Springer, Berlin (2015)
16. Draper-Gil, G., Zhou, J., Ferrer-Gomila, J.L., Hinarejos, M.F.: An optimistic fair exchange protocol with active intermediaries. Int. J. Inf. Secur. **12**(4), 299–318 (2013)
17. Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) Security and Cryptography for Networks, LNCS, vol. 5229. Springer, Heidelberg (2008)

18. Hanser, C., Rabkin, M., Schröder, D.: Verifiably encrypted signatures: security revisited and a new construction. In: Pernul, G., Yaryan, P., Weippl, E. (eds.) Computer Security-ESORICS 2015, Part I. LNCS, vol. 9326, pp. 146–164. Springer International Publishing, Cham (2015)
19. Hanser, C., Slamanig, D.: Blank digital signatures. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 95–106. ASIA CCS'13, ACM, New York, NY, USA (2013)
20. Huang, Q., Wong, D.S., Susilo, W.: Group-oriented fair exchange of signatures. Inf. Sci. **181**(16), 3267–3283 (2011)
21. Huang, Q., Wong, D.S., Susilo, W.: The construction of ambiguous optimistic fair exchange from designated confirmer signature without random oracles. Inf. Sci. **228**, 222–238 (2013)
22. Huang, Q., Wong, D.S., Susilo, W.: P^2OFE: privacy-preserving optimistic fair exchange of digital signatures. In: Benaloh, J. (ed.) CT-RSA 2014, LNCS, vol. 8366, pp. 367–384. Springer, Heidelberg (2014)
23. Huang, Q., Yang, G., Wong, D.S., Susilo, W.: Ambiguous optimistic fair exchange. In: Pieprzyk, J. (ed.) Advances in Cryptology-ASIACRYPT 2008. LNCS, vol. 5350, pp. 74–89. Springer, Heidelberg (2008)
24. Huang, Q., Yang, G., Wong, D.S., Susilo, W.: A new efficient optimistic fair exchange protocol without random oracles. Int. J. Inf. Secur. **11**(1), 53–63 (2011)
25. Huang, X., Mu, Y., Susilo, W., Wu, W., Xiang, Y.: Further observations on optimistic fair exchange protocols in the multi-user setting. In: Nguyen, P., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 124–141. Springer, Heidelberg (2010)
26. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) Topics in Cryptology-CT-RSA 2002, LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
27. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology-ASIACRYPT 2010, LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010)
28. Kim, K.S., Jeong, I.R.: Efficient verifiably encrypted signatures from lattices. Int. J. Inf. Secur. **13**(4), 305–314 (2014)
29. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) Information Security and Cryptology-ICISC 2006, LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
30. Kundu, A., Atallah, M.J., Bertino, E.: Leakage-free redactable signatures. In: Proceedings of the Second ACM Conference on Data and Application Security and Privacy, pp. 307–316. CODASPY'12, ACM, New York, NY, USA (2012)
31. Kundu, A., Bertino, E.: Structural signatures for tree data structures. Proc. VLDB Endow. **1**(1), 138–150 (2008)
32. Lai, J., Ding, X., Wu, Y.: Accountable trapdoor sanitizable signatures. In: Deng, R.H., Feng, T. (eds.) Information Security Practice and Experience, LNCS, vol. 7863, pp. 117–131. Springer, Heidelberg (2013)
33. Lee, J.Y., Cheon, J.H., Kim, S.: An analysis of proxy signatures: is a secure channel necessary? In: Joye, M. (ed.) Topics in Cryptology-CT-RSA 2003, LNCS, vol. 2612, pp. 68–79. Springer, Heidelberg (2003)
34. Lim, S., Lee, E., Park, C.M.: A short redactable signature scheme using pairing. Secur. Commun. Netw. **5**(5), 523–534 (2012)
35. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology-EUROCRYPT 2004, LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
36. Malkin, T., Obana, S., Yung, M.: The hierarchy of key evolving signatures and a characterization of proxy signatures. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology-EUROCRYPT 2004, LNCS, vol. 3027, pp. 306–322. Springer, Heidelberg (2004)
37. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security, pp. 48–57. CCS'96, ACM, New York, NY, USA (1996)
38. Nishimaki, R., Xagawa, K.: Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted ves. In: Kurosawa, K., Hanaoka, G. (eds.) Public-Key Cryptography-PKC 2013. LNCS, vol. 7778, pp. 405–422. Springer, Heidelberg (2013)
39. Pöhls, H.C., Samelin, K.: On updatable redactable signatures. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security. LNCS, vol. 8479, pp. 457–475. Springer International Publishing, Berlin (2014)
40. Qu, L., Wang, G., Mu, Y.: Optimistic fair exchange of ring signatures. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) Security and Privacy in Communication Networks, pp. 227–242. Springer, Heidelberg (2012)
41. Schuldt, J.C.N., Matsuura, K., Paterson, K.G.: Proxy signatures secure against proxy key exposure. In: Cramer, R. (ed.) Public Key Cryptography-PKC 2008, LNCS, vol. 4939, pp. 141–161. Springer, Heidelberg (2008)
42. Tan, K.W., Deng, R.H.: Applying sanitizable signature to web-service-enabled business processes: going beyond integrity protection. In: IEEE International Conference on Web Services, 2009. ICWS 2009, pp. 67–74 (2009)
43. Wang, G., Bao, F., Zhou, J., Deng, R.H.: Security analysis of some proxy signatures. In: Lim, J.I., Lee, D.H. (eds.) Information Security and Cryptology-ICISC 2003, LNCS, vol. 2971, pp. 305–319. Springer, Heidelberg (2004)
44. Wang, H., Pieprzyk, J.: Efficient one-time proxy signatures. In: Laih, C.S. (ed.) Advances in Cryptology-ASIACRYPT 2003, LNCS, vol. 2894, pp. 507–522. Springer, Heidelberg (2003)
45. Wang, Y., Au, M., Liu, J., Yuen, T., Susilo, W.: Threshold-oriented optimistic fair exchange. In: Lopez, J., Huang, X., Sandhu, R. (eds.) Network and System Security, pp. 424–438. Springer, Heidelberg (2013)
46. Wang, Y., Wu, Q., Wong, D.S., Qin, B., Liu, J., Mao, J.: Optimistic fair exchange of distributed signatures. In: CSC 2014, pp. 85–90. IET (2014)
47. Yuen, T.H., Susilo, W., Liu, J.K., Mu, Y.: Sanitizable signatures revisited. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) Cryptology and Network Security, LNCS, vol. 5339, pp. 80–97. Springer, Heidelberg (2008)
48. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor sanitizable signatures made easy. In: Zhou, J., Yung, M. (eds.) Applied Cryptography and Network Security, LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)
49. Zhang, L., Wu, Q., Qin, B.: Identity-based verifiably encrypted signatures without random oracles. In: Pieprzyk, J., Zhang, F. (eds.) Provable Security, LNCS, vol. 5848, pp. 76–89. Springer, Heidelberg (2009)
50. Zhang, L., Wu, Q., Qin, B.: Identity-based optimistic fair exchange in the standard model. Secur. Commun. Netw. **6**(8), 1010–1020 (2013)