

Static Group Signature with Post-Compromise Accountability

Kyosuke Yamashita^{1*}

^{1*}School of Science and Engineering, Chuo University, Tokyo, Japan.

Corresponding author(s). E-mail(s): kyamashita887@g.chuo-u.ac.jp;

Abstract

Signatures with Post-Compromise Accountability (SPCA), introduced by Dayanikli et al. at SCN'26, are signature schemes that enable signatures generated before the compromise of a signing key to be distinguished from those produced after the compromise. In this paper, we extend this notion to static group signatures and introduce Group Signatures with Post-Compromise Accountability (GSPCA). GSPCA preserves the anonymity and traceability guarantees of group signatures while enabling pre-compromise signatures to be revalidated after key compromise. The coexistence of opening and revalidation algorithms gives rise to a new consistency requirement, which we formalize as opening-revalidation consistency. We also show that accountable keys must be generated locally by the users, since post-compromise unforgeability becomes vacuous if the group manager holds them. We then present a generic construction of GSPCA in the framework of Bellare, Micciancio, and Warinschi from public key encryption secure against adaptive chosen-ciphertext attacks, existentially unforgeable signatures, and simulation-sound non-interactive zero-knowledge proofs, and prove it secure in the standard model. The construction places an accountable signature inside the ciphertext that the group manager already decrypts for opening. Since revalidation, like opening, is performed by the group manager, this hides the accountable signature from every other party and lets opening and revalidation operate on a single decryption.

Keywords: Cryptography, Group signatures, Post-compromise accountability, Generic construction

1 Introduction

1.1 Background

Digital signatures underpin authenticity and integrity across essentially all digital systems, from software updates to financial transactions. Their security, however, rests entirely on the secrecy of the signing key: once an adversary obtains it, every unforgeability guarantee the scheme provided is lost. Key compromise is not a rare edge case but a recurring reality — signing keys leak through malware, side-channel attacks, misconfigured infrastructure, or outright human error, and large-scale incidents involving stolen code-signing certificates and compromised public key infrastructures are well documented [1].

A variety of mechanisms mitigate the *long-term* consequences of key compromise. For instance, key-evolving signatures [2] let a signer rotate its key so that signatures produced *after* a detected compromise regain their security guarantees. Threshold signatures [3, 4] and multi-signatures [5] instead distribute signing capability across several parties, so that an adversary must compromise more than one key to forge a signature. What neither approach addresses, however, is the signatures produced *before* detection: once a compromise is discovered, every signature issued in the intervening window becomes indistinguishable from an adversarial forgery, since nothing in the scheme lets a verifier tell them apart. Given that compromises are routinely detected only after a substantial delay, this leaves a real and often sizable gap during which legitimate, honestly generated signatures lose their validity retroactively, together with the signer’s ability to stand behind them.

Signatures with Post-Compromise Accountability (SPCA) were recently proposed to close exactly this gap [1], building on an idea sketched by Błażkiewicz et al. [6]. The core idea is to equip the signer with a second, *accountable* key pair whose existence remains completely hidden during normal operation, so that a conventional key-compromise attack targeting the ordinary signing key does not expose it. Only once the signer suspects that its ordinary key has been compromised does it reveal the accountable public key; a verifier holding it can then revalidate previously issued signatures and determine which of them were produced by the legitimate signer (who alone holds the accountable secret key) rather than by an adversary in possession only of the compromised ordinary key. Because the accountable key is never exposed through the standard signing process, a compromise of the ordinary key alone gives an adversary no means of attacking it, so this fallback guarantee can be provided even though it protects only a bounded window between compromise and detection.

The same motivation applies, arguably with even higher stakes, to group signatures. A group signature scheme lets any member of a group sign on the group’s behalf so that a verifier learns only that *some* member signed, while a designated group manager retains the ability to open a signature and identify its author when necessary [7, 8]. This combination — anonymity for honest use, traceability when required — makes group signatures a natural fit for a wide range of settings that need both accountability and privacy, and the individual signing key held by each group member is exposed to exactly the same kind of compromise risk as in the single-signer setting: it may reside on an ordinary end-user device, or in an embedded component with a

comparatively weak physical security boundary. As one concrete illustration, group signatures have been proposed for privacy-preserving vehicle-to-vehicle authentication, where a designated tracing authority can identify the sender only in the event of a reported incident [9]; there, the signing key resides in an on-board electronic control unit, a class of component that has been shown vulnerable to physical key-extraction attacks such as power analysis and fault injection [10]. Crucially, the consequences of key compromise are in some ways more severe here than in the single-signer setting: because a group signature can be *opened*, a forged signature produced with a compromised key does not merely fail to verify as authentic in the abstract — it can be opened and attributed, by the group manager, to a specific, identifiable member, who then has no way to demonstrate that the signature was not their own. Post-compromise accountability is thus at least as relevant to group signatures as it is to ordinary ones; the presence of an opening mechanism gives compromise a concrete victim.

This raises the natural question we study in this work.

Can a group signature scheme be made post-compromise accountable, and if so, how?

1.2 Our Contribution

In this work, we introduce *Group Signatures with Post-Compromise Accountability* (GSPCA), an extension of static group signatures by Bellare, Micciancio, and Warinschi (BMW) [7]. Our contribution is primarily definitional. Realizing GSPCA is not a matter of transplanting the SPCA definitions into the group setting: the opening mechanism of group signatures interacts with the revalidation mechanism of post-compromise accountability, and the resulting guarantees are meaningful only under specific modeling choices. Once these are fixed, the construction turns out to be a simple and modular extension of BMW with standard-model security, which we regard as evidence that the model is the right one rather than as a limitation.

A new security requirement.

Beyond the security requirements inherited from group signatures (traceability and anonymity) and from SPCA (post-compromise unforgeability, indistinguishability, and accountable-key committing), we identify an additional requirement that is specific to the combination of the two settings: *Opening-Revalidation Consistency* (Definition 14). Group signatures provide an **Open** algorithm with no counterpart in SPCA, whereas post-compromise accountability provides a **Reval** algorithm with no counterpart in ordinary group signatures. Once these two mechanisms coexist, the existing security requirements do not by themselves guarantee that the identity determined by **Open** is consistent with the accountable public key under which **Reval** accepts. We therefore formalize opening-revalidation consistency, which requires that a valid signature opening to a user i cannot revalidate with respect to the accountable public key of a different user j , unless both users are corrupted. This requirement is not automatic even in our construction, where **Open** and **Reval** read the same decryption. An adversary that has seen a signature of an honest user i and corrupted a user j may

try to produce a fresh ciphertext that still opens to i but carries an accountable signature under j 's key; ruling this out requires the non-malleability provided by security against adaptive chosen-ciphertext attacks (IND-CCA2).

Modeling choices.

Three further design decisions shape our syntax and security model.

First, we require the accountable key pair $(\text{ask}_i, \text{apk}_i)$ of every user to be generated locally by the user via the user key generation algorithm UKG, rather than by the group manager as part of the group key generation algorithm GKG. This is not merely a stylistic choice inherited from SPCA: if the group manager held every ask_i , it could produce an accountable signature on behalf of any user without that user's involvement, which would make post-compromise unforgeability (Definition 13) vacuous as a security guarantee. Local generation of accountable keys is therefore structurally necessary for post-compromise unforgeability to be meaningful. This introduces UKG as a new algorithm not present in the BMW syntax, but leaves group key generation, signing, and verification otherwise close to the BMW framework.

Second, we restrict the revalidation algorithm *Reval* to be executable only by the holder of a master secret key gmsk , rather than allowing public revalidation. SPCA distinguishes private accountability, where only a designated party can revalidate signatures, from public accountability, where anyone can, and observes that the two lead to substantially different security guarantees. In the group setting, since anonymity already restricts the ability to trace a signature to the group manager alone, we adopt the analogous choice of making revalidation a group-manager-only operation as well, consistent with the fact that *Open* itself is similarly restricted.

Third, our anonymity notion gives the adversary every accountable key pair $(\text{ask}_i, \text{apk}_i)$, in addition to every ordinary signing key as in BMW. Since each accountable key pair is generated by its owner, a coalition of group members trivially knows its own accountable keys, and a worst-case anonymity notion must account for this. This does not contradict the non-disclosure guarantee underlying post-compromise accountability, which concerns an external adversary who compromises an ordinary key sk_i without ever obtaining ask_i . In our construction, this stronger anonymity notion comes at no cost, since the encryption randomness is independent of all accountable keys.

Construction in the standard model.

Our construction follows the generic BMW framework for static group signatures, which utilizes a standard signature scheme, an IND-CCA2 secure public key encryption (PKE) scheme, and a simulation-sound non-interactive zero-knowledge proof system (SS-NIZK): a signer authenticates a message under a per-user key, encrypts her identity together with a certificate and this inner signature under a group-wide PKE key, and proves via an SS-NIZK that the ciphertext was formed correctly, so that only the group manager, holding the decryption key, can trace a signature back to its signer. We extend it in the most direct way. The signer additionally computes an accountable signature σ_{acc} on $(i, \mathbf{m})$ under her local accountable key ask_i and includes it in the plaintext encrypted to the group manager. The revalidation algorithm *Reval* decrypts

the ciphertext exactly as **Open** does and verifies σ_{acc} under the supplied accountable public key, with respect to the identity contained in the plaintext. The reference signing algorithm required for indistinguishability places a fixed dummy string of the same length in place of σ_{acc} . The only property we need from the accountable signature scheme beyond existential unforgeability under adaptive chosen-message attacks (EUF-CMA) is key binding, namely that a signature is valid under at most one public key, which is achieved by a well-known transformation that appends the public key to the signature [11]. In contrast, SPCA requires its accountable signature scheme to be pseudorandom, because the accountable signature is embedded into the randomness of a publicly visible signature, as well as to satisfy exclusive ownership. Since our accountable signature is only ever visible to the group manager, pseudorandomness is unnecessary, and the accountable signature need not even hide the accountable public key; only the binding requirement, in the form of key binding, remains. We prove all six security requirements in the standard model, under exactly the assumptions of the BMW construction: an IND-CCA2 secure PKE scheme, EUF-CMA secure signature schemes, and an SS-NIZK proof system.

Why the plaintext, and not the encryption randomness.

SPCA ties the accountable signature to randomness consumed elsewhere in the scheme, because an ordinary signature offers no other place in which the accountable signature can be hidden. A group signature, in contrast, already carries a ciphertext that only the group manager can decrypt, and in our model the group manager is also the only party that revalidates. Placing σ_{acc} in the plaintext therefore hides it from exactly the parties from which it must be hidden, while the encryption randomness remains uniform and independent of all keys. Deriving the encryption randomness from σ_{acc} instead would make it depend on accountable keys that the anonymity adversary holds, a situation that ordinary IND-CCA2 security, which assumes uniformly sampled randomness, does not cover. Conversely, the plaintext approach relies on revalidation being restricted to the group manager; supporting public revalidation would require a different mechanism, which we leave to future work.

Limitation: no non-repudiation.

Like SPCA [1], GSPCA does not provide non-repudiation against a malicious legitimate signer who deliberately avoids the accountable signing procedure. In the SPCA model, the accountable signing key is intended to reside in a hardened component, such as secure hardware or a trusted execution environment, and the adversary is assumed to be unable to compromise this key even when the ordinary signing key is compromised. This assumption protects the accountable signing key from an external key-compromise adversary, but does not force the legitimate signer to use it when generating a signature. Indeed, SPCA explicitly observes that a malicious signer may instead invoke the reference signing algorithm using only the ordinary signing key and later deny responsibility for the resulting signature. The same limitation carries over to GSPCA: since indistinguishability requires the existence of a reference signing algorithm that does not use the accountable signing key, a legitimate group member can deliberately generate a valid signature using this reference signing algorithm rather

than the accountable signing algorithm. Such a signature remains valid and opens to that member, but need not successfully revalidate under the member’s accountable public key. Consequently, a failure of revalidation cannot by itself distinguish a post-compromise forgery from a signature deliberately generated by a dishonest member without using the accountable signing key. Preventing such deliberate repudiation would require an additional enforcement mechanism that ensures use of the accountable signing procedure, and is outside the scope of our model.

1.3 Related Work

A long line of work mitigates the impact of signing-key compromise without addressing the accountability of signatures already issued. Key-evolving signatures — forward-secure [12], key-insulated [13], and intrusion-resilient [14] schemes — let a signer periodically update its key so that signatures produced *after* an update remain secure even if a previous key was exposed. Threshold and multi-signature schemes [3–5] instead distribute signing capability across several parties, raising the bar for a successful forgery. As discussed in Section 1.1, none of these approaches determines whether a signature produced *before* a compromise is detected originates from the legitimate signer or from an adversary already in possession of the key; this is precisely the gap that SPCA, and in turn GSPCA, is designed to close.

Group signatures have been formalized both in a *static* setting, where the group of members is fixed once and for all at setup [7], and in a *dynamic* setting, where members may join the group over time [8]. GSPCA is formalized in the static setting, following the same choice as the original BMW framework; we leave a dynamic extension to future work.

Group signatures already come equipped with mechanisms for handling misbehaving members, and it is worth clarifying why these do not already provide the guarantee GSPCA targets. Revocation mechanisms [15] allow the group manager to exclude a member from the group *going forward*, so that its future signatures are no longer accepted; they say nothing about signatures already produced under a key that has since been compromised. Traceable signatures [16] let the group manager issue a per-member tracing trapdoor that can later be used to identify all signatures produced by that member, but this tracing capability is a property of the scheme from the outset and is not tied to any notion of key compromise: unlike GSPCA, there is no distinction between a signature produced by the legitimate member and one produced by an adversary who has since obtained that member’s signing key. In short, none of these mechanisms address the question GSPCA is built around: distinguishing, after the fact, an honestly generated signature from one forged with a compromised key.

SPCA ties accountability to randomness consumed elsewhere in the scheme, using the notion of signatures with public randomness recovery formalized by Jaeger and Stracovsky [17]. This is conceptually related to anamorphic signatures [18], which use the randomness of a probabilistic signature as a covert channel to communicate a hidden message under coercion. Our construction needs no such hidden channel: the ciphertext that the BMW framework addresses to the group manager already provides a channel that only the group manager can read, and we use it to carry the accountable signature.

Incoercible signatures [19] address a different threat: an adversary who actively coerces the signer, and who must be handed a signature that looks genuine but can later be revealed as fake by a trusted authority. This resembles our use of signatures with hidden extra structure, but the threat model and the trust assumptions differ substantially: incoercible signatures rely on a trusted authority to expose the deception, whereas GSPCA relies only on the legitimate signer’s own accountable key and requires no external trusted party.

2 Preliminaries

Throughout this paper, we denote by $\lambda \in \mathbb{N}$ a security parameter. For $n \in \mathbb{N}$, we let $[n] = \{1, 2, \dots, n\}$. A polynomial function and a negligible function are denoted by $\text{poly}()$ and $\text{negl}()$, respectively. Probabilistic polynomial-time is abbreviated as PPT. For an NP language $\mathcal{L}$, the corresponding NP relation is denoted by $R_{\mathcal{L}}$. If an algorithm X has a subroutine A , we write $X.A$. An arbitrary value is denoted by a dot “.”.

In the following, we define the cryptographic primitives required for our GSPCA construction. Besides the standard notions, our construction uses in particular two known tools, which we recall right after the corresponding primitives: key binding for signature schemes, a perfect variant of exclusive ownership, which we require of the accountable signature scheme, and the left-or-right (multi-challenge) formulation of IND-CCA2 security, which simplifies our proofs.

Definition 1 (Signature) A signature scheme $\Pi = (\text{KG}, \text{Sig}, \text{Vrf})$ consists of three PPT algorithms that work as follows:

$\text{KG}(1^\lambda) \rightarrow (\text{sk}, \text{pk})$: The key generation algorithm, given a security parameter 1^λ , outputs a secret key sk and a public key pk .

$\text{Sig}(\text{sk}, \text{m}) \rightarrow \sigma$: The signing algorithm, given a secret key sk and a message m , outputs a signature σ .

$\text{Vrf}(\text{pk}, \text{m}, \sigma) = 1/0$: The (deterministic) verification algorithm, given a public key pk , a message m , and a signature σ , outputs 1 (meaning “valid”) or 0 (meaning “invalid”).

The signature scheme Π is correct if, for any sufficiently large security parameter λ and any message m , it holds that

$$\Pr \left[(\text{sk}, \text{pk}) \leftarrow \text{KG}(1^\lambda); \sigma \leftarrow \text{Sig}(\text{sk}, \text{m}) : \text{Vrf}(\text{pk}, \text{m}, \sigma) = 1 \right] \geq 1 - \text{negl}(\lambda).$$

Definition 2 (EUF-CMA) A signature scheme $\Pi = (\text{KG}, \text{Sig}, \text{Vrf})$ is existentially unforgeable under an adaptive chosen-message attack (EUF-CMA) if for any sufficiently large security parameter λ , and any PPT adversary $\mathcal{A}$, it holds that $\Pr[\text{ExpEUFSig}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$, where ExpEUFSig is defined as follows:

$$\frac{\text{ExpEUFSig}_{\Pi, \mathcal{A}}(\lambda)}{L := \emptyset; (\text{sk}, \text{pk}) \leftarrow \text{KG}(1^\lambda); \\ (\text{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\text{O}_{\text{Sig}}}(\text{pk}) : \\ \text{Output } 1 \text{ if } \text{Vrf}(\text{pk}, \text{m}^*, \sigma^*) = 1 \wedge \text{m}^* \notin L, \text{ otherwise } 0}$$

where O_{Sig} works as follows: Given a message m , it returns $\sigma \leftarrow \text{Sig}(\text{sk}, \text{m})$, and updates $L := L \cup \{\text{m}\}$.

Definition 3 (Key Binding) A signature scheme $\text{SIG} = (\text{KG}, \text{Sig}, \text{Vrf})$ is *key-binding* if, for every security parameter λ , every pair of distinct bit strings pk_0, pk_1 (not necessarily in the range of KG), every message m , and every signature σ , it holds that

$$\text{SIG.Vrf}(\text{pk}_0, \text{m}, \sigma) = 1 \implies \text{SIG.Vrf}(\text{pk}_1, \text{m}, \sigma) = 0.$$

Key binding is a perfect variant of the exclusive-ownership notions studied in [11, 20, 21]: it rules out the existence of a second public key altogether, rather than merely requiring that finding one be computationally infeasible. Key binding is achieved by a well-known transformation that appends the public key to the signature. Given any signature scheme $\text{SIG} = (\text{KG}, \text{Sig}, \text{Vrf})$, let SIG^{kb} be the scheme with the same key generation, whose signing algorithm outputs $(\text{pk}, \text{SIG.Sig}(\text{sk}, \text{m}))$ on input (sk, m) for the public key pk corresponding to sk , and whose verification algorithm, on input $(\text{pk}, \text{m}, (\text{pk}', \sigma))$, outputs 1 if and only if $\text{pk}' = \text{pk}$ and $\text{SIG.Vrf}(\text{pk}, \text{m}, \sigma) = 1$. This is a variant of the second transformation of Pornin and Stern [11], which appends $H(\text{pk})$ instead of pk and thereby obtains exclusive ownership under the collision resistance of H ; appending pk itself makes the binding unconditional.

Fact 1 *If SIG is correct and EUF-CMA secure, then SIG^{kb} is correct, EUF-CMA secure, and key-binding.*

Indeed, a forgery $(\text{m}^*, (\text{pk}^*, \sigma^*))$ against SIG^{kb} yields the forgery (m^*, σ^*) against SIG , since pk^* must equal the challenge key; and a signature valid under pk_0 has public key component pk_0 , so it fails the public key check under any $\text{pk}_1 \neq \text{pk}_0$. In particular, an EUF-CMA secure and key-binding signature scheme, as required by our construction, exists whenever an EUF-CMA secure signature scheme exists.

Definition 4 (Public Key Encryption) A public key encryption (PKE) scheme consists of the following three PPT algorithms $(\text{KG}, \text{Enc}, \text{Dec})$:

$\text{KG}(1^\lambda) \rightarrow (\text{sk}, \text{pk})$: Given 1^λ , output a secret key sk and a public key pk .

$\text{Enc}(\text{pk}, \text{m}; r) \rightarrow c$: Given a public key pk , a plaintext m , and a randomness r , output a ciphertext c .

$\text{Dec}(\text{sk}, c) = \text{m}/\perp$: Given a secret key sk and a ciphertext c , output a plaintext m or $\perp$, where $\perp$ indicates c is invalid.

We write $c \leftarrow \text{Enc}(\text{pk}, \text{m})$ when r is sampled uniformly from the randomness space.

Definition 5 (Correctness) A PKE scheme $\Pi = (\text{KG}, \text{Enc}, \text{Dec})$ is correct if

$$\Pr \left[(\text{sk}, \text{pk}) \leftarrow \text{KG}(1^\lambda) : \exists \text{m}, r \text{ s.t. } \text{Dec}(\text{sk}, \text{Enc}(\text{pk}, \text{m}; r)) \neq \text{m} \right] \leq \text{negl}(\lambda).$$

Definition 6 (IND-CCA2 Security) A PKE scheme $\Pi = (\text{KG}, \text{Enc}, \text{Dec})$ is indistinguishable under adaptive chosen ciphertext attack (IND-CCA2), if for any security parameter λ and

any stateful PPT adversary $\mathcal{A}$, it holds that $|\Pr[\text{ExpPKE}_{\mathcal{A},\Pi}(\lambda) = 1] - 1/2| \leq \text{negl}(\lambda)$ where ExpPKE is defined as follows:

$$\begin{array}{l} \text{ExpPKE}_{\mathcal{A},\Pi}(\lambda) \\ \hline L := \emptyset; (\text{sk}, \text{pk}) \leftarrow \Pi.\text{KG}(1^\lambda); \\ (\mathbf{m}_0, \mathbf{m}_1) \leftarrow \mathcal{A}^\mathcal{O}(\text{pk}); \\ b \leftarrow \{0, 1\}; c^* \leftarrow \Pi.\text{Enc}(\text{pk}, \mathbf{m}_b); \\ b' \leftarrow \mathcal{A}^\mathcal{O}(c^*); \\ \text{Output } 1 \text{ if } b' = b, \text{ otherwise } 0 \end{array}$$

The oracle $\mathcal{O}$ is that, given a ciphertext c , returns $\text{Dec}(\text{sk}, c)$ and updates $L := L \cup \{c\}$. Further, if c^* is given after the challenger creates it, then $\mathcal{O}$ returns $\perp$. The two challenge messages are required to satisfy $|\mathbf{m}_0| = |\mathbf{m}_1|$.

Definition 7 (LoR-CCA Security [22]) Let $\Pi = (\text{KG}, \text{Enc}, \text{Dec})$ be a PKE scheme. For $\beta \in \{0, 1\}$ and a PPT adversary $\mathcal{A}$, the experiment $\text{ExpLoR}_{\Pi,\mathcal{A}}^\beta(\lambda)$ generates $(\text{sk}, \text{pk}) \leftarrow \Pi.\text{KG}(1^\lambda)$, initializes $L := \emptyset$, and outputs the bit $\beta' \leftarrow \mathcal{A}^{\text{O}_{\text{LR}}, \text{O}_{\text{Dec}}}(\text{pk})$, where:

- O_{LR} , given $(\mathbf{m}_0, \mathbf{m}_1)$ with $|\mathbf{m}_0| = |\mathbf{m}_1|$, computes $c \leftarrow \Pi.\text{Enc}(\text{pk}, \mathbf{m}_\beta)$, updates $L := L \cup \{c\}$, and returns c ;
- O_{Dec} , given c , returns $\perp$ if $c \in L$ and $\Pi.\text{Dec}(\text{sk}, c)$ otherwise.

The advantage of $\mathcal{A}$ is $\text{Adv}_{\Pi,\mathcal{A}}^{\text{lor}}(\lambda) := |\Pr[\text{ExpLoR}_{\Pi,\mathcal{A}}^0(\lambda) = 1] - \Pr[\text{ExpLoR}_{\Pi,\mathcal{A}}^1(\lambda) = 1]|$.

For an adversary $\mathcal{B}$ in the IND-CCA2 experiment of Definition 6, we write $\text{Adv}_{\Pi,\mathcal{B}}^{\text{cca}}(\lambda) := |2 \Pr[\text{ExpPKE}_{\mathcal{B},\Pi}(\lambda) = 1] - 1|$.

Fact 2 ([22]) For every PPT adversary $\mathcal{A}$ that makes at most $Q = Q(\lambda)$ queries to O_{LR} , there exists a PPT adversary $\mathcal{B}$ such that

$$\text{Adv}_{\Pi,\mathcal{A}}^{\text{lor}}(\lambda) \leq Q \cdot \text{Adv}_{\Pi,\mathcal{B}}^{\text{cca}}(\lambda).$$

In particular, if Π is IND-CCA2 secure, then $\text{Adv}_{\Pi,\mathcal{A}}^{\text{lor}}(\lambda) \leq \text{negl}(\lambda)$ for every PPT $\mathcal{A}$.

Definition 8 (Simulation-Sound NIZK) Let $\mathcal{L}$ be an NP-language. Let $\Pi = (\text{CRS}, \text{Prv}, \text{Vrf})$ be a tuple of PPT algorithms that work as follows.

$\text{CRS}(1^\lambda) \rightarrow \text{crs}$: Given 1^λ , output a *common reference string* (CRS) crs .

$\text{Prv}(\text{crs}, x, w) \rightarrow \pi$: Given a CRS crs and a pair of a statement and a witness $(x, w) \in R_{\mathcal{L}}$, output a proof π .

$\text{Vrf}(\text{crs}, x, \pi) \rightarrow 1/0$: Given a CRS crs , a statement x , and a proof π , output a bit $b \in \{0, 1\}$, where 1 means “accept” and 0 means “reject.”

The tuple Π is a simulation-sound non-interactive zero-knowledge proof system (SS-NIZK) for $\mathcal{L}$ if it is complete, simulation-sound, and zero-knowledge, defined as follows:

Completeness. For any λ , any $\text{crs} \leftarrow \Pi.\text{CRS}(1^\lambda)$ and any $(x, w) \in R_{\mathcal{L}}$, it holds that $\Pr[\Pi.\text{Vrf}(\text{crs}, x, \Pi.\text{Prv}(\text{crs}, x, w)) = 1] \geq 1 - \text{negl}(\lambda)$.

(Adaptive) Zero-Knowledge. There exist PPT algorithms SimCRS and Sim that work as follows.

$\text{SimCRS}(1^\lambda) \rightarrow (\text{crs}, \tau)$: Given a security parameter 1^λ , output a CRS crs and a trapdoor τ .

$\text{Sim}(\text{crs}, x, \tau)$: Given a CRS crs , a statement x (not necessarily satisfying $x \in \mathcal{L}$) and a trapdoor τ , output a simulated proof π .

Further, for any λ and any PPT adversary $\mathcal{A}$, it holds that

$$|\Pr[\text{crs} \leftarrow \Pi.\text{CRS}(1^\lambda) : 1 \leftarrow \mathcal{A}^{\mathcal{O}_0(\cdot, \cdot)}(\text{crs})] - \Pr[(\text{crs}, \tau) \leftarrow \Pi.\text{SimCRS}(1^\lambda) : 1 \leftarrow \mathcal{A}^{\mathcal{O}_1(\cdot, \cdot)}(\text{crs})]| \leq \text{negl}(\lambda)$$

where $\mathcal{O}_0$ and $\mathcal{O}_1$ are oracles that, given (x, w) , output $\perp$ if $(x, w) \notin R_{\mathcal{L}}$, otherwise they output $\Pi.\text{Prv}(\text{crs}, x, w)$ and $\Pi.\text{Sim}(\text{crs}, x, \tau)$, respectively.

Simulation-Soundness. Let $\mathcal{Q}_{\text{Sim}}$ be initially empty. For any PPT adversary $\mathcal{A}$, it holds that

$$\Pr \left[\begin{array}{l} (\text{crs}, \tau) \leftarrow \Pi.\text{SimCRS}(1^\lambda); (x^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sim}}(\cdot)}(\text{crs}) : \\ \Pi.\text{Vrf}(\text{crs}, x^*, \pi^*) = 1 \wedge x^* \notin \mathcal{L} \wedge (x^*, \pi^*) \notin \mathcal{Q}_{\text{Sim}} \end{array} \right] \leq \text{negl}(\lambda),$$

where the simulation oracle $\mathcal{O}_{\text{Sim}}$, on input a statement x , computes $\pi \leftarrow \Pi.\text{Sim}(\text{crs}, x, \tau)$, updates $\mathcal{Q}_{\text{Sim}} := \mathcal{Q}_{\text{Sim}} \cup \{(x, \pi)\}$, and returns π .

We stress that the condition $(x^*, \pi^*) \notin \mathcal{Q}_{\text{Sim}}$ does not require x^* itself to be fresh. In particular, even if the adversary has previously obtained $\pi \leftarrow \Pi.\text{Sim}(\text{crs}, x^*, \tau)$, it must still be infeasible to produce a different accepting proof $\pi^* \neq \pi$ for the same false statement $x^* \notin \mathcal{L}$.

We note that the simulated CRS is computationally indistinguishable from the honestly generated CRS.

Some of our security proofs require only ordinary soundness, which is weaker than simulation-soundness. For later use, we explicitly define this property as follows. For any sufficiently large security parameter λ and any PPT adversary $\mathcal{A}$, it holds that

$$\Pr [\text{crs} \leftarrow \Pi.\text{CRS}(1^\lambda); (x^*, \pi^*) \leftarrow \mathcal{A}(\text{crs}) : x^* \notin \mathcal{L} \wedge \Pi.\text{Vrf}(\text{crs}, x^*, \pi^*) = 1] \leq \text{negl}(\lambda).$$

For the simulation-sound NIZK considered here, ordinary soundness follows from simulation-soundness together with the computational indistinguishability of the honestly generated and simulated CRS distributions.

3 Definition of GSPCA

In this section, we formalize group signatures with post-compromise accountability (GSPCA). We consider the static setting, where the set of ($n = \text{poly}(\lambda)$) group members is fixed at setup. In what follows, n denotes the number of group members. We first define the syntax and correctness of GSPCA, and then introduce the security requirements capturing anonymity, traceability, and post-compromise accountability, together with the consistency requirements that arise from simultaneously supporting opening and revalidation.

3.1 Syntax

Definition 9 (GSPCA) A group signature scheme with post-compromise accountability (GSPCA) consists of the six PPT algorithms ($\text{UKG}, \text{GKG}, \text{AccGSig}, \text{Vrf}, \text{Open}, \text{Reval}$) that

work as follows:

$\text{UKG}(1^\lambda) \rightarrow (\text{ask}, \text{apk})$: Given a security parameter 1^λ , it outputs an accountable signing key ask and an accountable public key apk . Note that this algorithm is run by users locally.

$\text{GKG}(1^\lambda) \rightarrow (\text{gmsk}, \text{gpk}, \{\text{sk}_i\}_{i \in [n]})$: Given a security parameter 1^λ , it outputs a master secret key gmsk , a group public key gpk , and a set of signing keys of users $\{\text{sk}_i\}_{i \in [n]}$.

$\text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, \text{m}) \rightarrow \sigma$: Given a group public key gpk , a user's signing key sk_i , a user's accountable signing key ask_i , and a message m , it outputs a signature σ .

$\text{Vrf}(\text{gpk}, \text{m}, \sigma) = 1/0$: Given a group public key gpk , a message m , and a signature σ , it outputs 1 (meaning valid) or 0 (meaning invalid).

$\text{Open}(\text{gmsk}, \text{gpk}, \text{m}, \sigma) = i/\perp$: Given a master secret key gmsk , a group public key gpk , a message m , and a signature σ , it outputs a user's identity $i \in [n]$ or $\perp$.

$\text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}_i, \text{m}, \sigma) = 1/0$: Given a master secret key gmsk , a group public key gpk , a user's accountable public key apk_i , a message m , and a signature σ , it outputs 1 (meaning that σ is created by using ask_i) or 0 (meaning that σ is not created by using ask_i).

3.2 Requirements

We require a GSPCA scheme to satisfy the following correctness and security properties. At a high level, anonymity and traceability capture the standard privacy and accountability guarantees of group signatures, while post-compromise unforgeability captures the additional guarantee sought after compromise of the ordinary signing keys. In particular, post-compromise security is considered after exposure of the ordinary group signing keys, while the accountable signing keys remain secret. Opening-revalidation consistency, indistinguishability, and accountable-key committing ensure that the additional accountability mechanism interacts consistently with the underlying group-signature functionality. We first provide the oracles used in the experiments in Fig. 1. The experiments are depicted in Figs. 2 and 3.

Correctness requires that an honestly generated signature be accepted by the verification algorithm, opened to the actual signer, and successfully revalidated with respect to the signer's accountable public key.

Definition 10 (Correctness) For any sufficiently large security parameter λ , any $n = \text{poly}(\lambda)$, any $i \in [n]$, and any message m , let $(\text{ask}_j, \text{apk}_j) \leftarrow \text{UKG}(1^\lambda)$ for any $j \in [n]$, $(\text{gmsk}, \text{gpk}, \{\text{sk}_j\}_{j \in [n]}) \leftarrow \text{GKG}(1^\lambda)$, and $\sigma \leftarrow \text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, \text{m})$. A GSPCA scheme Π is correct if

$$\Pr \left[\begin{array}{l} \text{Vrf}(\text{gpk}, \text{m}, \sigma) = 1 \\ \wedge \text{Open}(\text{gmsk}, \text{gpk}, \text{m}, \sigma) = i \\ \wedge \text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}_i, \text{m}, \sigma) = 1 \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

Anonymity requires that a signature does not reveal which of two users generated it, even when the adversary is given all users' signing keys and accountable key pairs and is allowed to make opening and revalidation queries, except for queries that trivially

$O_{Op}(m, \sigma)$ 1 : $L_{Op} := L_{Op} \cup \{(m, \sigma)\};$ 2 : return $Open(gmsk, gpk, m, \sigma)$	$O_b(i, m)$ 1 : if $b = 0$ then 2 : return $RefGSig(gpk, sk_i, m)$ 3 : return $AccGSig(gpk, sk_i, ask_i, m)$
$O_{Rv}(i, m, \sigma)$ 1 : $L_{Rv} := L_{Rv} \cup \{(i, m, \sigma)\};$ 2 : return $Reval(gmsk, gpk, apk_i, m, \sigma)$	$O_{Comp}(i)$ 1 : if $comp_i = 1$ then return $\perp$ 2 : else $comp_i := 1$ 3 : return (sk_i, apk_i)
$O_{AC}(i, m)$ 1 : $\sigma \leftarrow AccGSig(gpk, sk_i, ask_i, m)$ 2 : $L_{AC} := L_{AC} \cup \{(i, m)\};$ 3 : return σ	$O_{AC}^{AKC}(i, m)$ 1 : $\sigma \leftarrow AccGSig(gpk, sk_i, ask_i, m)$ 2 : if $comp_i = 0$ 3 : then $L_{AC}^{AKC} := L_{AC}^{AKC} \cup \{(i, m, \sigma)\};$ 4 : return σ
$O_{Cor}(i)$ 1 : $L_{Co} := L_{Co} \cup \{i\};$ 2 : return (sk_i, ask_i)	

Fig. 1 Oracles for GSPCA. The algorithm $RefGSig$ in O_b is defined in Definition 15.

reveal the signer of the challenge signature. Giving every ask_i to the adversary reflects that each member generates, and therefore knows, its own accountable key pair; see Section 1.2 for a discussion.

Definition 11 (Anonymity) A GSPCA scheme Π satisfies anonymity if for any sufficiently large security parameter λ and any PPT algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, it holds that $\Pr[\text{ExpAnon}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq 1/2 + \text{negl}(\lambda)$ where ExpAnon is the experiment as described in Fig. 2.

Traceability requires that every valid signature be openable to a registered user. Moreover, a valid signature that opens to an uncorrupted user must have been obtained by requesting a signature on the same message on behalf of that user. Thus, even corrupted users, possibly colluding with the group manager, cannot produce an untraceable signature or falsely attribute a fresh signature to an uncorrupted user. Following the static setting of BMW, this single requirement subsumes non-frameability; separating the two requires a dynamic model as in [8]. The adversary is additionally given every accountable public key apk_i . Since traceability does not involve revalidation, this only strengthens the notion, so that traceability holds regardless of whether accountable public keys have been revealed; it plays no further role.

Definition 12 (Traceability) A GSPCA scheme Π satisfies traceability if for any sufficiently large security parameter λ and any PPT algorithm $\mathcal{A}$, it holds that $\Pr[\text{ExpTrc}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$ where ExpTrc is the experiment as described in Fig. 2.

ExpAnon_{Π,ℳ}(λ)
1: $L_{Op} := \emptyset; L_{Rv} := \emptyset;$ 2: $\forall i \in [n], (ask_i, apk_i) \leftarrow UKG(1^\lambda); (gmsk, gpk, \{sk_i\}_{i \in [n]}) \leftarrow GKG(1^\lambda);$ 3: $(i_0^*, i_1^*, m^*, st) \leftarrow \mathcal{A}_0^{Op, Orv}(gpk, \{sk_i\}_{i \in [n]}, \{(ask_i, apk_i)\}_{i \in [n]});$ 4: $b \leftarrow \{0, 1\};$ 5: $\sigma^* \leftarrow AccGSig(gpk, sk_{i_b}^*, ask_{i_b}^*, m^*);$ 6: $b' \leftarrow \mathcal{A}_1^{Op, Orv}(\sigma^*, st);$ 7: if $b' = b \wedge i_0^* \neq i_1^* \wedge (m^*, \sigma^*) \notin L_{Op} \wedge (i_0^*, m^*, \sigma^*), (i_1^*, m^*, \sigma^*) \notin L_{Rv}$ 8: then return 1 9: else return 0
ExpTrc_{Π,ℳ}(λ)
1: $L_{Co} := \emptyset; L_{AC} := \emptyset$ 2: $\forall i \in [n], (ask_i, apk_i) \leftarrow UKG(1^\lambda); (gmsk, gpk, \{sk_i\}_{i \in [n]}) \leftarrow GKG(1^\lambda);$ 3: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{O_{AC}, O_{Cor}}(gmsk, gpk, \{apk_i\}_{i \in [n]});$ 4: if $Vrf(gpk, m^*, \sigma^*) = 0$ then return 0 5: if $Open(gmsk, gpk, m^*, \sigma^*) = \perp$ then return 1 6: if $\exists i^* \in [n]$ s.t. $Open(gmsk, gpk, m^*, \sigma^*) = i^* \wedge i^* \notin L_{Co} \wedge (i^*, m^*) \notin L_{AC}$ 7: then return 1 8: else return 0
ExpPCU_{Π,ℳ}(λ)
1: $L_{AC} := \emptyset;$ 2: $\forall i \in [n], (ask_i, apk_i) \leftarrow UKG(1^\lambda); (gmsk, gpk, \{sk_i\}_{i \in [n]}) \leftarrow GKG(1^\lambda);$ 3: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{O_{AC}}(gpk, \{sk_i\}_{i \in [n]}, \{apk_i\}_{i \in [n]});$ 4: if $Vrf(gpk, m^*, \sigma^*) = 0 \vee Open(gmsk, gpk, m^*, \sigma^*) = \perp$ then return 0 5: if $Open(gmsk, gpk, m^*, \sigma^*) = i^* \wedge Reval(gmsk, gpk, apk_{i^*}, m^*, \sigma^*) = 1$ 6: $\wedge (i^*, m^*) \notin L_{AC}$ then return 1 7: else return 0

Fig. 2 Security experiments for anonymity (ExpAnon), traceability (ExpTrc), and post-compromise unforgeability (ExpPCU) of GSPCA.

Post-compromise unforgeability captures accountability after the ordinary signing keys have been compromised. Even when the adversary obtains all users' signing keys and accountable public keys, but not their accountable signing keys, it cannot produce a fresh valid signature that both opens to a user and successfully revalidates with respect to that user's accountable public key.

Definition 13 (Post-Compromise Unforgeability) A GSPCA scheme Π is post-compromise unforgeable if for any sufficiently large security parameter λ and any PPT algorithm $\mathcal{A}$, it

ExpORCon$_{\Pi, \mathcal{A}}(\lambda)$
1 : $L_{AC} := \emptyset; L_{Co} := \emptyset;$ 2 : $\forall i \in [n], (\text{ask}_i, \text{apk}_i) \leftarrow \text{UKG}(1^\lambda); (\text{gmsk}, \text{gpk}, \{\text{sk}_i\}_{i \in [n]}) \leftarrow \text{GKG}(1^\lambda);$ 3 : $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{O}_{AC}, \text{O}_{Cor}}(\text{gpk}, \{\text{apk}_i\}_{i \in [n]});$ 4 : $i^* \leftarrow \text{Open}(\text{gmsk}, \text{gpk}, m^*, \sigma^*);$ 5 : if $\text{Vrf}(\text{gpk}, m^*, \sigma^*) = 0 \vee i^* = \perp$ then return 0 6 : if $\exists j \in [n] \text{ s.t. } j \neq i^* \wedge \text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}_j, m^*, \sigma^*) = 1 \wedge \{i^*, j\} \not\subseteq L_{Co}$ 7 : then return 1 8 : else return 0
ExpIND$_{\Pi, \text{RefGSig}, \mathcal{A}}(\lambda)$
1 : $\forall i \in [n], (\text{ask}_i, \text{apk}_i) \leftarrow \text{UKG}(1^\lambda); (\text{gmsk}, \text{gpk}, \{\text{sk}_i\}_{i \in [n]}) \leftarrow \text{GKG}(1^\lambda);$ 2 : $b \leftarrow \{0, 1\};$ 3 : $b' \leftarrow \mathcal{A}^{\text{O}_b}(\{\text{sk}_i\}_{i \in [n]}, \text{gpk});$ 4 : if $b' = b$ then return 1 5 : else return 0
ExpAKC$_{\Pi, \mathcal{A}}(\lambda)$
1 : $L_{AC}^{\text{AKC}} := \emptyset; \forall i \in [n], \text{comp}_i := 0;$ 2 : $\forall i \in [n], (\text{ask}_i, \text{apk}_i) \leftarrow \text{UKG}(1^\lambda); (\text{gmsk}, \text{gpk}, \{\text{sk}_i\}_{i \in [n]}) \leftarrow \text{GKG}(1^\lambda);$ 3 : $(i^*, m^*, \sigma^*, \text{apk}^*) \leftarrow \mathcal{A}^{\text{O}_{AC}^{\text{AKC}}, \text{O}_{Comp}}(\text{gpk});$ 4 : if $(i^*, m^*, \sigma^*) \in L_{AC}^{\text{AKC}} \wedge \text{apk}^* \neq \text{apk}_{i^*} \wedge \text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}^*, m^*, \sigma^*) = 1$ 5 : return 1 6 : else return 0

Fig. 3 Security experiments for opening-revalidation consistency (ExpORCon), indistinguishability (ExpIND), and accountable-key committing (ExpAKC) of GSPCA.

holds that $\Pr[\text{ExpPCU}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$ where ExpPCU is the experiment as described in Fig. 2.

Opening-revalidation consistency requires the identities determined by the opening and revalidation procedures to be consistent. More precisely, a valid signature that opens to a user i must not successfully revalidate with respect to the accountable public key of a different user j , unless both users are corrupted. This prevents an adversary from involving an honest user in an inconsistency between opening and revalidation.

Definition 14 (Opening-Revalidation Consistency) A GSPCA scheme Π satisfies opening-revalidation consistency if for any sufficiently large security parameter λ and any PPT algorithm $\mathcal{A}$, it holds that $\Pr[\text{ExpORCon}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$ where ExpORCon is the experiment as described in Fig. 3.

Indistinguishability requires that accountable signatures be computationally indistinguishable from valid reference signatures generated without an accountable signing key. In particular, a signature itself does not reveal whether the accountable signing procedure or the reference signing procedure was used.

Definition 15 (Indistinguishability) A GSPCA scheme Π satisfies indistinguishability if there exists a PPT algorithm $\text{RefGSig}(\text{gpk}, \text{sk}_i, \mathbf{m}) \rightarrow \sigma$ that does not take the accountable signing key ask_i as input, such that:

1. for every honestly generated key tuple, every $i \in [n]$, and every message $\mathbf{m}$, a signature $\sigma \leftarrow \text{RefGSig}(\text{gpk}, \text{sk}_i, \mathbf{m})$ is valid and opens to i except with negligible probability; and
2. for every PPT adversary $\mathcal{A}$, it holds that $|\Pr[\text{ExpIND}_{\Pi, \text{RefGSig}, \mathcal{A}}(\lambda) = 1] - 1/2| \leq \text{negl}(\lambda)$ where ExpIND is the experiment as described in Fig. 3.

Accountable-key committing requires an honestly generated pre-compromise signature to be bound to the accountable public key used when the signature was created. Even after the ordinary signing key is compromised, an adversary cannot cause such a signature to revalidate with respect to a different, adversarially chosen accountable public key.

Definition 16 (Accountable-Key Committing) A GSPCA scheme Π satisfies accountable-key committing if for any sufficiently large security parameter λ and any PPT adversary $\mathcal{A}$, it holds that $\Pr[\text{ExpAKC}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$ where ExpAKC is the experiment described in Fig. 3.

We emphasize that the latter three requirements capture distinct aspects of the accountability mechanism. Opening-revalidation consistency concerns agreement between the identity determined by **Open** and the accountable public key accepted by **Reval**. Indistinguishability ensures that the use of the accountable signing key is hidden during normal operation. Accountable-key committing, in contrast, prevents an honestly generated pre-compromise signature from later being associated with a different accountable public key. None of these properties is intended to replace traceability or post-compromise unforgeability; rather, they complement them by constraining how revalidation interacts with opening and with the accountable keys.

4 Construction of GSPCA

We now present our generic construction of GSPCA, which follows the BMW framework [7].

4.1 The Construction

Let $E = (\text{KG}, \text{Enc}, \text{Dec})$ be a correct and IND-CCA2 secure PKE scheme with randomness space R . Let $S = (\text{KG}, \text{Sig}, \text{Vrf})$ be an EUF-CMA secure signature scheme. Let $S_{\text{acc}} = (\text{KG}, \text{Sig}, \text{Vrf})$ be an EUF-CMA secure and key-binding signature scheme.

Finally, let ZK be a simulation-sound NIZK proof system for the NP language $\mathcal{L}$ defined later.

We now provide an overview of our construction. A user i 's accountable key pair is exactly an S_{acc} key pair, generated locally as $(\text{ask}_i, \text{apk}_i) \leftarrow \text{S}_{\text{acc}}.\text{KG}(1^\lambda)$ by the user in the role of UKG. Given 1^λ , the group manager (GM) generates $\text{crs} \leftarrow \text{ZK}.\text{CRS}(1^\lambda)$, $(\text{sk}_E, \text{pk}_E) \leftarrow \text{E}.\text{KG}(1^\lambda)$, and $(\text{sk}_S, \text{pk}_S) \leftarrow \text{S}.\text{KG}(1^\lambda)$. For each $i \in [n]$, it also computes $(\text{SK}_i, \text{PK}_i) \leftarrow \text{S}.\text{KG}(1^\lambda)$ and $\text{cert}_i \leftarrow \text{S}.\text{Sig}(\text{sk}_S, (i, \text{PK}_i))$. The master secret key is $\text{gmsk} = \text{sk}_E$, the group public key is $\text{gpk} = (\text{crs}, \text{pk}_E, \text{pk}_S)$, and user i 's signing key is $\text{sk}_i = (i, \text{SK}_i, \text{PK}_i, \text{cert}_i)$.

To sign a message m , user i computes an accountable signature $\sigma_{\text{acc}} \leftarrow \text{S}_{\text{acc}}.\text{Sig}(\text{ask}_i, (i, m))$ and an ordinary signature $\sigma' \leftarrow \text{S}.\text{Sig}(\text{SK}_i, m)$, samples fresh randomness $r \leftarrow \mathcal{R}$, and encrypts

$$c = \text{E}.\text{Enc}(\text{pk}_E, (0^\lambda, m, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}}); r).$$

The plaintext is encoded as a bit string in a uniquely decodable way. As is standard, we assume that identities in $[n]$, public keys and signatures of S , and signatures of S_{acc} have fixed lengths; we denote the length of signatures of S_{acc} by ℓ_{acc} . Hence the length of the plaintext depends only on λ , n , and $|m|$, and replacing σ_{acc} by any other string of length ℓ_{acc} does not change it. The prefix 0^λ is redundancy used only in the security analysis (Section 4.2). It then proves that c is well formed with respect to the following language:

$$\mathcal{L} = \left\{ (\text{pk}_E, \text{pk}_S, m, c) \left| \begin{array}{l} \exists i \in [n], \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}}, r \text{ s.t.} \\ \text{S}.\text{Vrf}(\text{pk}_S, (i, \text{PK}_i), \text{cert}_i) = 1 \\ \wedge \text{S}.\text{Vrf}(\text{PK}_i, m, \sigma') = 1 \\ \wedge \text{E}.\text{Enc}(\text{pk}_E, (0^\lambda, m, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}}); r) = c \end{array} \right. \right\}.$$

Here σ_{acc} ranges over all bit strings of length ℓ_{acc} ; the language places no further constraint on it. The signer computes

$$\pi \leftarrow \text{ZK}.\text{Prv}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), (i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}}, r))$$

and outputs $\sigma := (\pi, c)$.

Opening follows the BMW framework: the GM verifies π , decrypts c , checks the certificate and the ordinary signature contained in the plaintext, and outputs the identity contained in the plaintext. Revalidation performs the same steps and then verifies the last component σ_{acc} of *the same* plaintext, under the supplied accountable public key, with respect to the message (i, m) , where i is the identity contained in the plaintext. Since both algorithms read the identity and the accountable signature from a single decryption, the identity with respect to which **Reval** verifies σ_{acc} is exactly the identity output by **Open**.

The full construction Π is given in Fig. 4.

UKG(1^λ) 1 : $(ask, apk) \leftarrow S_{acc}.KG(1^\lambda)$ 2 : return (ask, apk) GKG(1^λ) 1 : $crs \leftarrow ZK.CRS(1^\lambda)$ 2 : $(sk_E, pk_E) \leftarrow E.KG(1^\lambda)$ 3 : $(sk_S, pk_S) \leftarrow S.KG(1^\lambda)$ 4 : $\forall i \in [n], (SK_i, PK_i) \leftarrow S.KG(1^\lambda)$ 5 : $\forall i \in [n], cert_i \leftarrow S.Sig(sk_S, (i, PK_i))$ 6 : $gmsk := sk_E$ 7 : $gpk := (crs, pk_E, pk_S)$ 8 : $\forall i \in [n], sk_i := (i, SK_i, PK_i, cert_i)$ 9 : return $(gmsk, gpk), \{sk_i\}_{i \in [n]}$ AccGSig(gpk, sk_i, ask_i, m) 1 : Parse $gpk = (crs, pk_E, pk_S)$ 2 : Parse $sk_i = (i, SK_i, PK_i, cert_i)$ 3 : $\sigma_{acc} \leftarrow S_{acc}.Sig(ask_i, (i, m))$ 4 : $\sigma' \leftarrow S.Sig(SK_i, m)$ 5 : $M := (0^\lambda, m, i, PK_i, cert_i, \sigma', \sigma_{acc})$ 6 : $r \leftarrow R$; 7 : $c := E.Enc(pk_E, M; r)$ 8 : $X := (pk_E, pk_S, m, c)$ 9 : $W := (i, PK_i, cert_i, \sigma', \sigma_{acc}, r)$ 10 : $\pi \leftarrow ZK.Prv(crs, X, W)$ 11 : return $\sigma := (\pi, c)$ Vrf(gpk, m, σ) 1 : Parse $gpk = (crs, pk_E, pk_S)$ 2 : Parse $\sigma = (\pi, c)$ 3 : if $ZK.Vrf(crs, (pk_E, pk_S, m, c), \pi) = 1$ 4 : then return 1 5 : else return 0	Open($gmsk, gpk, m, \sigma$) 1 : Parse $gmsk = sk_E$ 2 : Parse $gpk = (crs, pk_E, pk_S)$ 3 : Parse $\sigma = (\pi, c)$ 4 : if $ZK.Vrf(crs, (pk_E, pk_S, m, c), \pi) = 0$ 5 : then return $\perp$ 6 : Parse $E.Dec(sk_E, c)$ 7 : as $(0^\lambda, m', i, PK, cert, \sigma', \sigma_{acc})$ 8 : if parsing fails, then return $\perp$ 9 : if $i \in [n]$ 10 : $\wedge S.Vrf(pk_S, (i, PK), cert) = 1$ 11 : $\wedge m' = m \wedge S.Vrf(PK, m, \sigma') = 1$ 12 : then return i 13 : else return $\perp$ Reval($gmsk, gpk, apk_i, m, \sigma$) 1 : Parse $gmsk = sk_E$ 2 : Parse $gpk = (crs, pk_E, pk_S)$ 3 : Parse $\sigma = (\pi, c)$ 4 : if $ZK.Vrf(crs, (pk_E, pk_S, m, c), \pi) = 0$ 5 : then return 0 6 : Parse $E.Dec(sk_E, c)$ 7 : as $(0^\lambda, m', i, PK, cert, \sigma', \sigma_{acc})$ 8 : if parsing fails, then return 0 9 : if $i \notin [n]$ 10 : $\vee S.Vrf(pk_S, (i, PK), cert) = 0$ 11 : $\vee m' \neq m \vee S.Vrf(PK, m, \sigma') = 0$ 12 : then return 0 13 : if $S_{acc}.Vrf(apk_i, (i, m), \sigma_{acc}) = 1$ 14 : then return 1 15 : else return 0
--	--

Fig. 4 The construction Π of GSPCA.

Correctness.

Let $\sigma = (\pi, c) \leftarrow \text{AccGSig}(gpk, sk_i, ask_i, m)$ for honestly generated keys. The pair consisting of (pk_E, pk_S, m, c) and $(i, PK_i, cert_i, \sigma', \sigma_{acc}, r)$ is in $R_{\mathcal{L}}$ by the correctness of S , so $\text{Vrf}(gpk, m, \sigma) = 1$ by the completeness of ZK , except with negligible probability. By the correctness of E , c decrypts to $(0^\lambda, m, i, PK_i, cert_i, \sigma', \sigma_{acc})$, so all checks of Open

pass and `Open` returns i . By the correctness of S_{acc} , $S_{\text{acc}}.\text{Vrf}(\text{apk}_i, (i, m), \sigma_{\text{acc}}) = 1$, so $\text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}_i, m, \sigma) = 1$. Hence Π is correct.

Reference signing.

The reference signing algorithm $\text{RefGSig}(\text{gpk}, \text{sk}_i, m)$ required by Definition 15 is identical to AccGSig except that it uses the fixed dummy string $\sigma_{\perp} := 0^{\ell_{\text{acc}}}$ in place of σ_{acc} , and therefore does not need ask_i :

1. Parse $\text{gpk} = (\text{crs}, \text{pk}_E, \text{pk}_S)$ and $\text{sk}_i = (i, \text{SK}_i, \text{PK}_i, \text{cert}_i)$.
2. Compute $\sigma' \leftarrow S.\text{Sig}(\text{SK}_i, m)$ and sample $r \leftarrow R$.
3. Compute $c \leftarrow E.\text{Enc}(\text{pk}_E, (0^\lambda, m, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\perp}); r)$.
4. Set $X := (\text{pk}_E, \text{pk}_S, m, c)$ and $W := (i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\perp}, r)$.
5. Compute $\pi \leftarrow \text{ZK.Priv}(\text{crs}, X, W)$ and return $\sigma := (\pi, c)$.

Since $\mathcal{L}$ does not constrain the last plaintext component, $(X, W) \in R_{\mathcal{L}}$.

4.2 Security Analysis

In this subsection, we prove that Π satisfies the six security requirements of Section 3; together with the correctness shown above, they follow from Lemmas 1, 2, 3, 4, 5, and 6 below. In the proofs below, we condition, whenever stated, on the event that the accountable public keys $\{\text{apk}_i\}_{i \in [n]}$ are pairwise distinct. The complementary event has negligible probability by the EUF-CMA security of S_{acc} : otherwise, an adversary that runs $S_{\text{acc}}.\text{KG}$ itself and signs a fresh message with the resulting signing key would produce a forgery under the challenge key with non-negligible probability. We also write Bad_{corr} for the event that the correctness condition of Definition 5 fails for the generated key pair $(\text{sk}_E, \text{pk}_E)$; this event has negligible probability.

We call a plaintext of E *well-formed* if it begins with 0^λ and parses as a tuple $(0^\lambda, m, i, \text{PK}, \text{cert}, \sigma', \sigma_{\text{acc}})$ as in Section 4.1. For each plaintext length, we fix the malformed plaintext $M_{\perp}$ of that length consisting only of ones. Since $M_{\perp}$ does not begin with 0^λ , unless Bad_{corr} occurs, no encryption of $M_{\perp}$ is also an encryption of a well-formed plaintext. This is the only purpose of the prefix 0^λ ; the proofs of anonymity and opening-revalidation consistency use $M_{\perp}$ to replace honestly generated plaintexts.

Lemma 1 (Anonymity) *The construction Π satisfies anonymity if E is correct and IND-CCA2 secure, S_{acc} is correct, EUF-CMA secure, and key-binding, and ZK is zero-knowledge and simulation-sound.*

Proof Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be any PPT adversary against the anonymity of Π . Throughout the proof, we condition on the event that the accountable public keys are pairwise distinct. Let Adm denote the event

$$i_0^* \neq i_1^* \wedge (m^*, \sigma^*) \notin L_{\text{Op}} \wedge (i_0^*, m^*, \sigma^*), (i_1^*, m^*, \sigma^*) \notin L_{\text{Rv}}.$$

Since L_{Op} and L_{Rv} only grow during the experiment, once a query falsifies Adm , the event Adm remains false. For each Hybrid h , let $P_h := \Pr[b' = 1 \wedge \text{Adm} \text{ in Hybrid } h]$. For $\beta \in \{0, 1\}$,

we write

$$M_\beta^* := (0^\lambda, \mathbf{m}^*, i_\beta^*, \text{PK}_{i_\beta^*}, \text{cert}_{i_\beta^*}, \sigma'_\beta, \sigma_{\text{acc}, \beta}),$$

where $\sigma'_\beta \leftarrow \text{S.Sig}(\text{SK}_{i_\beta^*}, \mathbf{m}^*)$ and $\sigma_{\text{acc}, \beta} \leftarrow \text{S}_{\text{acc}}.\text{Sig}(\text{ask}_{i_\beta^*}, (i_\beta^*, \mathbf{m}^*))$. Note that $|M_0^*| = |M_1^*| = |M_\perp|$ by the encoding in Section 4.1. We write $x^* := (\text{pk}_E, \text{pk}_S, \mathbf{m}^*, c^*)$ for the statement of the challenge signature.

Hybrid 0

Hybrid 0 is ExpAnon with $b = 0$. That is, $c^* \leftarrow \text{E.Enc}(\text{pk}_E, M_0^*; r^*)$ for $r^* \leftarrow \mathcal{R}$, and $\pi^* \leftarrow \text{ZK.Priv}(\text{crs}, x^*, (i_0^*, \text{PK}_{i_0^*}, \text{cert}_{i_0^*}, \sigma'_0, \sigma_{\text{acc}, 0}, r^*))$.

Hybrid 1

Hybrid 1 is identical to Hybrid 0 except that GKG runs $(\text{crs}, \tau) \leftarrow \text{ZK.SimCRS}(1^\lambda)$ instead of $\text{crs} \leftarrow \text{ZK.CRS}(1^\lambda)$, and the challenge proof is generated as $\pi^* \leftarrow \text{ZK.Sim}(\text{crs}, x^*, \tau)$.

We show $|P_0 - P_1| \leq \text{negl}(\lambda)$ by a reduction $\mathcal{D}$ to the zero-knowledge property of ZK . Given crs and oracle access to $\mathcal{O}_\beta$ from its challenger, $\mathcal{D}$ generates all other keys of ExpAnon itself, including sk_E , and runs $\mathcal{A}$ on the input prescribed by ExpAnon . It answers $\mathcal{O}_{\text{Op}}$ and $\mathcal{O}_{\text{Rv}}$ by running $\Pi.\text{Open}$ and $\Pi.\text{Reval}$ with sk_E . To generate the challenge, it computes M_0^* and c^* as in Hybrid 0 and obtains π^* by querying $\mathcal{O}_\beta$ on $(x^*, (i_0^*, \text{PK}_{i_0^*}, \text{cert}_{i_0^*}, \sigma'_0, \sigma_{\text{acc}, 0}, r^*))$, which is in $R_\mathcal{C}$. Finally, it outputs 1 if and only if $b' = 1 \wedge \text{Adm}$. If $\mathcal{O}_\beta$ returns honest proofs, $\mathcal{D}$ perfectly realizes Hybrid 0, and if it returns simulated proofs, $\mathcal{D}$ perfectly realizes Hybrid 1. Hence $|P_0 - P_1| \leq \text{negl}(\lambda)$.

Hybrid 2

Hybrid 2 is identical to Hybrid 1 except that the challenge ciphertext encrypts the malformed plaintext, namely, $c^* \leftarrow \text{E.Enc}(\text{pk}_E, M_\perp; r^*)$. The challenge proof is still generated as $\pi^* \leftarrow \text{ZK.Sim}(\text{crs}, x^*, \tau)$.

Before comparing Hybrids 1 and 2, we bound the probability of the following event in Hybrid 2. Let F be the event that, after c^* is generated, $\mathcal{A}$ submits to $\mathcal{O}_{\text{Op}}$ or $\mathcal{O}_{\text{Rv}}$ a query whose signature component has the form (π, c^*) for a message $\mathbf{m}$ such that $\text{ZK.Vrf}(\text{crs}, (\text{pk}_E, \text{pk}_S, \mathbf{m}, c^*), \pi) = 1$ and $((\text{pk}_E, \text{pk}_S, \mathbf{m}, c^*), \pi) \neq (x^*, \pi^*)$. Intuitively, F captures an adversary that reuses the challenge ciphertext c^* in a valid signature other than the challenge signature itself, either by attaching a different accepting proof to c^* or by using c^* for a different message. Such reuse need not be accidental; the adversary may attempt it deliberately, for instance by re-randomizing π^* . In the reduction to IND-CCA2 security below, queries containing c^* cannot be forwarded to the decryption oracle and are instead answered using M_0^* . These answers are correct when c^* encrypts M_0^* , but may be incorrect when c^* encrypts $M_\perp$; apart from queries on the challenge signature itself, which are handled separately, this can happen only on queries captured by F . Claim 1 shows that, once c^* encrypts $M_\perp$, every statement containing c^* is false, so that no adversary, however it proceeds, can produce such a signature except with negligible probability.

Claim 1 $\Pr[F \text{ in Hybrid 2}] \leq \text{negl}(\lambda)$.

Proof We construct a reduction $\mathcal{C}$ against the simulation-soundness of ZK . Given crs and oracle access to $\mathcal{O}_{\text{Sim}}$, the algorithm $\mathcal{C}$ generates all other keys of ExpAnon itself, including sk_E , and runs $\mathcal{A}$ exactly as in Hybrid 2, answering $\mathcal{O}_{\text{Op}}$ and $\mathcal{O}_{\text{Rv}}$ by running $\Pi.\text{Open}$ and

$\Pi.\text{Reval}$ with sk_E . It computes $c^* \leftarrow E.\text{Enc}(\text{pk}_E, M_\perp; r^*)$ and obtains π^* by querying $\mathcal{O}_{\text{Sim}}$ on x^* . When $\mathcal{A}$ makes the first query that triggers F , with statement $x = (\text{pk}_E, \text{pk}_S, m, c^*)$ and proof π , the algorithm $\mathcal{C}$ outputs (x, π) .

The view of $\mathcal{A}$ in this simulation is identical to that in Hybrid 2. Assume $\neg \text{Bad}_{\text{corr}}$. We claim that $x \notin \mathcal{L}$. Indeed, if $x \in \mathcal{L}$, then $c^* = E.\text{Enc}(\text{pk}_E, M'; r')$ for some well-formed plaintext M' and some r' , and hence $E.\text{Dec}(\text{sk}_E, c^*) = M'$. On the other hand, $c^* = E.\text{Enc}(\text{pk}_E, M_\perp; r^*)$, and hence $E.\text{Dec}(\text{sk}_E, c^*) = M_\perp$. Since M' begins with 0^λ and $M_\perp$ does not, $M' \neq M_\perp$, which is a contradiction. Moreover, the only pair in $\mathcal{Q}_{\text{Sim}}$ is (x^*, π^*) , and $(x, \pi) \neq (x^*, \pi^*)$ by the definition of F . Thus $\mathcal{C}$ breaks simulation-soundness whenever $F \wedge \neg \text{Bad}_{\text{corr}}$ occurs, and $\Pr[F \text{ in Hybrid 2}] \leq \text{negl}(\lambda) + \Pr[\text{Bad}_{\text{corr}}] \leq \text{negl}(\lambda)$. $\square$

We now show $|P_1 - P_2| \leq \text{negl}(\lambda)$ by a reduction $\mathcal{B}$ to the IND-CCA2 security of E .

Setup Phase

Given a public key pk from its IND-CCA2 challenger, $\mathcal{B}$ sets $\text{pk}_E := \text{pk}$. It generates $(\text{ask}_i, \text{apk}_i) \leftarrow \text{S}_{\text{acc}}.\text{KG}(1^\lambda)$ for each $i \in [n]$, $(\text{crs}, \tau) \leftarrow \text{ZK}.\text{SimCRS}(1^\lambda)$, $(\text{sk}_S, \text{pk}_S) \leftarrow \text{S}.\text{KG}(1^\lambda)$, and, for each $i \in [n]$, $(\text{SK}_i, \text{PK}_i) \leftarrow \text{S}.\text{KG}(1^\lambda)$ and $\text{cert}_i \leftarrow \text{S}.\text{Sig}(\text{sk}_S, (i, \text{PK}_i))$. It sets $\text{gpk} := (\text{crs}, \text{pk}_E, \text{pk}_S)$ and $\text{sk}_i := (i, \text{SK}_i, \text{PK}_i, \text{cert}_i)$, initializes $\text{L}_{\text{Op}} := \emptyset$ and $\text{L}_{\text{Rv}} := \emptyset$, and runs $\mathcal{A}_0$ on $(\text{gpk}, \{\text{sk}_i\}_{i \in [n]}, \{(\text{ask}_i, \text{apk}_i)\}_{i \in [n]})$. Note that $\mathcal{B}$ does not know sk_E .

Challenge Phase

When $\mathcal{A}_0$ outputs $(i_0^*, i_1^*, m^*, \text{st})$, the algorithm $\mathcal{B}$ computes M_0^* , submits $(M_0^*, M_\perp)$ to its challenger, and receives c^* . It computes $\pi^* \leftarrow \text{ZK}.\text{Sim}(\text{crs}, x^*, \tau)$, sets $\sigma^* := (\pi^*, c^*)$, and runs $\mathcal{A}_1(\sigma^*, \text{st})$. Throughout, $\mathcal{B}$ answers the oracle queries of $\mathcal{A}$ as follows.

$\text{O}_{\text{Op}}(m, \sigma)$: $\mathcal{B}$ updates $\text{L}_{\text{Op}} := \text{L}_{\text{Op}} \cup \{(m, \sigma)\}$, parses $\sigma = (\pi, c)$, and returns $\perp$ if $\text{ZK}.\text{Vrf}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), \pi) = 0$. If c^* is not yet defined or $c \neq c^*$, $\mathcal{B}$ queries its decryption oracle on c and runs the remaining steps of $\Pi.\text{Open}$ on the returned plaintext. If $c = c^*$, $\mathcal{B}$ runs the remaining steps of $\Pi.\text{Open}$ on M_0^* ; that is, it returns i_0^* if $m = m^*$ and $\perp$ otherwise.

$\text{O}_{\text{Rv}}(i, m, \sigma)$: $\mathcal{B}$ updates $\text{L}_{\text{Rv}} := \text{L}_{\text{Rv}} \cup \{(i, m, \sigma)\}$, parses $\sigma = (\pi, c)$, and returns 0 if $\text{ZK}.\text{Vrf}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), \pi) = 0$. If c^* is not yet defined or $c \neq c^*$, $\mathcal{B}$ queries its decryption oracle on c and runs the remaining steps of $\Pi.\text{Reval}$, with respect to apk_i , on the returned plaintext. If $c = c^*$, $\mathcal{B}$ runs the remaining steps of $\Pi.\text{Reval}$ on M_0^* ; that is, it returns $\text{S}_{\text{acc}}.\text{Vrf}(\text{apk}_i, (i_0^*, m^*), \sigma_{\text{acc}, 0})$ if $m = m^*$ and 0 otherwise.

When $\mathcal{A}_1$ outputs b' , the algorithm $\mathcal{B}$ outputs 1 if and only if $b' = 1 \wedge \text{Adm}$, which it evaluates from its own lists. Since $\mathcal{B}$ never submits c^* to its decryption oracle, it is a legitimate IND-CCA2 adversary.

Analysis

Let β denote the challenge bit of $\mathcal{B}$'s challenger, so that c^* encrypts M_0^* if $\beta = 0$ and $M_\perp$ if $\beta = 1$. Apart from the generation of c^* and the answers to queries involving c^* , $\mathcal{B}$ realizes Hybrids 1 and 2 exactly.

Case $\beta = 0$. Here c^* is distributed exactly as in Hybrid 1. Under $\neg \text{Bad}_{\text{corr}}$, $E.\text{Dec}(\text{sk}_E, c^*) = M_0^*$, so $\Pi.\text{Open}$ and $\Pi.\text{Reval}$ applied to any query containing c^* compute exactly the answers that $\mathcal{B}$ computes from M_0^* . Hence $|\Pr[\mathcal{B} = 1 \mid \beta = 0] - P_1| \leq \Pr[\text{Bad}_{\text{corr}}]$.

Case $\beta = 1$. Here c^* is distributed exactly as in Hybrid 2. Consider a query containing c^* , with statement $x = (\text{pk}_E, \text{pk}_S, m, c^*)$ and proof π , and assume $\neg \text{Bad}_{\text{corr}}$, so that $E.\text{Dec}(\text{sk}_E, c^*) = M_\perp$. If $\text{ZK}.\text{Vrf}(\text{crs}, x, \pi) = 0$, both $\mathcal{B}$ and the real oracle reject. If $\text{ZK}.\text{Vrf}(\text{crs}, x, \pi) = 1$ and $(x, \pi) \neq (x^*, \pi^*)$, the event F occurs. It remains to consider $(x, \pi) = (x^*, \pi^*)$, that is, a

query on the challenge pair $(\mathbf{m}^*, \sigma^*)$ itself. An opening query on $(\mathbf{m}^*, \sigma^*)$, and a revalidation query $(i, \mathbf{m}^*, \sigma^*)$ with $i \in \{i_0^*, i_1^*\}$, falsify **Adm**, so the answers to such queries do not affect the event $b' = 1 \wedge \text{Adm}$. For a revalidation query $(i, \mathbf{m}^*, \sigma^*)$ with $i \notin \{i_0^*, i_1^*\}$, the real $\Pi.\text{Reval}$ returns 0 because $M_\perp$ is not well-formed. The algorithm $\mathcal{B}$ also returns 0 except with negligible probability: by the correctness of S_{acc} , $\sigma_{\text{acc},0}$ is valid under $\text{apk}_{i_0^*}$ for $(i_0^*, \mathbf{m}^*)$ except with negligible probability, and since $\text{apk}_i \neq \text{apk}_{i_0^*}$, key binding gives $\text{S}_{\text{acc}}.\text{Vrf}(\text{apk}_i, (i_0^*, \mathbf{m}^*), \sigma_{\text{acc},0}) = 0$. Hence $|\Pr[\mathcal{B} = 1 \mid \beta = 1] - P_2| \leq \Pr[\text{F in Hybrid 2}] + \Pr[\text{Bad}_{\text{corr}}] + \text{negl}(\lambda)$.

Combining the two cases with Claim 1,

$$\begin{aligned} |P_1 - P_2| &\leq |\Pr[\mathcal{B} = 1 \mid \beta = 0] - \Pr[\mathcal{B} = 1 \mid \beta = 1]| + \text{negl}(\lambda) \\ &= \text{Adv}_{\mathcal{E}, \mathcal{B}}^{\text{cca}}(\lambda) + \text{negl}(\lambda) \leq \text{negl}(\lambda). \end{aligned}$$

Hybrid 3

Hybrid 3 is identical to Hybrid 2 except that $c^* \leftarrow \text{E.Enc}(\text{pk}_E, M_1^*; r^*)$; the challenge proof is still simulated. The bound $|P_2 - P_3| \leq \text{negl}(\lambda)$ follows by the argument for Hybrids 1 and 2 with the roles of i_0^* and i_1^* exchanged. More precisely, a reduction $\mathcal{B}'$ submits the challenge pair $(M_1^*, M_\perp)$, answers every query involving c^* by running the remaining steps of $\Pi.\text{Open}$ or $\Pi.\text{Reval}$ on M_1^* , and outputs 1 if and only if $b' = 1 \wedge \text{Adm}$. Claim 1 bounds the probability of F in Hybrid 2 as before, so $|P_2 - P_3| \leq \text{Adv}_{\mathcal{E}, \mathcal{B}'}^{\text{cca}}(\lambda) + \text{negl}(\lambda) \leq \text{negl}(\lambda)$.

Hybrid 4

Hybrid 4 is identical to Hybrid 3 except that $\text{crs} \leftarrow \text{ZK.CRS}(1^\lambda)$ is used and the challenge proof is generated as $\pi^* \leftarrow \text{ZK.Priv}(\text{crs}, x^*, (i_1^*, \text{PK}_{i_1^*}, \text{cert}_{i_1^*}, \sigma_1', \sigma_{\text{acc},1}, r^*))$. By the same reduction to the zero-knowledge property as for Hybrids 0 and 1, $|P_3 - P_4| \leq \text{negl}(\lambda)$. Hybrid 4 is exactly ExpAnon with $b = 1$.

We have $P_0 = \Pr[b' = 1 \wedge \text{Adm} \mid b = 0]$ and $P_4 = \Pr[b' = 1 \wedge \text{Adm} \mid b = 1]$, and the triangle inequality gives $|P_0 - P_4| \leq \text{negl}(\lambda)$. Since ExpAnon outputs 1 only if $b' = b$ and **Adm** holds,

$$\begin{aligned} \Pr[\text{ExpAnon}_{\Pi, \mathcal{A}}(\lambda) = 1] &= \frac{1}{2} \Pr[b' = 0 \wedge \text{Adm} \mid b = 0] + \frac{1}{2} \Pr[b' = 1 \wedge \text{Adm} \mid b = 1] \\ &\leq \frac{1}{2}(1 - P_0) + \frac{1}{2}P_4 \leq \frac{1}{2} + \frac{1}{2}|P_4 - P_0| \leq \frac{1}{2} + \text{negl}(\lambda). \end{aligned}$$

Hence Π satisfies anonymity. $\square$

Lemma 2 (Traceability) *The construction Π satisfies traceability if S is EUF-CMA secure, E is correct, and ZK is sound.*

Proof Let $\mathcal{A}$ be a PPT adversary in ExpTrc , and write $\sigma^* = (\pi^*, c^*)$. By inspection of Fig. 2, $\text{ExpTrc}_{\Pi, \mathcal{A}}(\lambda)$ outputs 1 only if one of the following two mutually exclusive events occurs.

Bad₁: $\Pi.\text{Vrf}(\text{gpk}, \mathbf{m}^*, \sigma^*) = 1$ and $\Pi.\text{Open}(\text{gmsk}, \text{gpk}, \mathbf{m}^*, \sigma^*) = \perp$.

Bad₂: $\Pi.\text{Vrf}(\text{gpk}, \mathbf{m}^*, \sigma^*) = 1$ and $\Pi.\text{Open}(\text{gmsk}, \text{gpk}, \mathbf{m}^*, \sigma^*) = i^*$ for some $i^* \in [n]$ with $i^* \notin \text{L}_{\text{Co}}$ and $(i^*, \mathbf{m}^*) \notin \text{L}_{\text{AC}}$.

Hence $\Pr[\text{ExpTrc}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \Pr[\text{Bad}_1] + \Pr[\text{Bad}_2]$.

Bounding $\Pr[\text{Bad}_1]$.

Let $X^* := (\text{pk}_E, \text{pk}_S, \text{m}^*, c^*)$. Since $\text{crs} \leftarrow \text{ZK.CRS}(1^\lambda)$ is honestly generated, the soundness of ZK guarantees that, except with negligible probability, $\text{ZK.Vrf}(\text{crs}, X^*, \pi^*) = 1$ implies $X^* \in \mathcal{L}$. In that case, there exist $i \in [n]$, PK , cert , σ' , σ_{acc} , and r such that $\text{S.Vrf}(\text{pk}_S, (i, \text{PK}), \text{cert}) = 1$, $\text{S.Vrf}(\text{PK}, \text{m}^*, \sigma') = 1$, and

$$\text{E.Enc}(\text{pk}_E, (0^\lambda, \text{m}^*, i, \text{PK}, \text{cert}, \sigma', \sigma_{\text{acc}}); r) = c^*.$$

Under $\neg \text{Bad}_{\text{corr}}$, c^* decrypts to this plaintext, so all checks of $\Pi.\text{Open}$ succeed and $\Pi.\text{Open}$ returns $i \neq \perp$. Hence $\Pr[\text{Bad}_1] \leq \text{negl}(\lambda)$.

Structure of Bad_2 .

If Bad_2 occurs, then, since $\Pi.\text{Open}$ returns $i^* \neq \perp$, the plaintext $\text{E.Dec}(\text{sk}_E, c^*)$ parses as $(0^\lambda, \text{m}^*, i^*, \text{PK}, \text{cert}, \sigma', \sigma_{\text{acc}})$ with $\text{S.Vrf}(\text{pk}_S, (i^*, \text{PK}), \text{cert}) = 1$ and $\text{S.Vrf}(\text{PK}, \text{m}^*, \sigma') = 1$. These values are obtained directly by decryption; no witness extraction from the proof is used. We split Bad_2 according to whether $\text{PK} = \text{PK}_{i^*}$.

Claim 2 $\Pr[\text{Bad}_2 \wedge \text{PK} \neq \text{PK}_{i^*}] \leq \text{negl}(\lambda)$.

Proof We construct a PPT reduction $\mathcal{B}$ against the EUF-CMA security of S .

Setup Phase

On receiving a challenge public key pk and oracle access to O_{Sig} , the algorithm $\mathcal{B}$ sets $\text{pk}_S := \text{pk}$. For every $i \in [n]$, it generates $(\text{SK}_i, \text{PK}_i) \leftarrow \text{S.KG}(1^\lambda)$ and obtains cert_i by querying O_{Sig} on (i, PK_i) . It additionally generates $(\text{sk}_E, \text{pk}_E) \leftarrow \text{E.KG}(1^\lambda)$, $\text{crs} \leftarrow \text{ZK.CRS}(1^\lambda)$, and $(\text{ask}_i, \text{apk}_i) \leftarrow \text{S}_{\text{acc}}.\text{KG}(1^\lambda)$ for every $i \in [n]$. It sets $\text{gmsk} := \text{sk}_E$, $\text{gpk} := (\text{crs}, \text{pk}_E, \text{pk}_S)$, and $\text{sk}_i := (i, \text{SK}_i, \text{PK}_i, \text{cert}_i)$, and gives $\mathcal{A}$ the input prescribed by ExpTrc .

Challenge Phase

Since $\mathcal{B}$ knows every SK_i and every ask_i , it answers $\text{O}_{\text{AC}}(i, \text{m})$ by running $\Pi.\text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, \text{m})$ and $\text{O}_{\text{Cor}}(i)$ by returning $(\text{sk}_i, \text{ask}_i)$, maintaining L_{AC} and L_{Co} as in ExpTrc . When $\mathcal{A}$ outputs (m^*, σ^*) , the algorithm $\mathcal{B}$ checks, using sk_E , whether Bad_2 occurs, and aborts if not. Otherwise, it decrypts c^* to obtain PK and cert as above, and outputs $((i^*, \text{PK}), \text{cert})$ if $\text{PK} \neq \text{PK}_{i^*}$, aborting otherwise.

Analysis

The simulation is perfect. If $\text{Bad}_2 \wedge \text{PK} \neq \text{PK}_{i^*}$ occurs, then $\text{S.Vrf}(\text{pk}_S, (i^*, \text{PK}), \text{cert}) = 1$. The only signing queries of $\mathcal{B}$ are (i, PK_i) for $i \in [n]$; for $i = i^*$ the second component differs from PK , and for $i \neq i^*$ the first component differs from i^* . Hence $((i^*, \text{PK}), \text{cert})$ is a valid fresh forgery, and $\Pr[\text{Bad}_2 \wedge \text{PK} \neq \text{PK}_{i^*}] \leq \Pr[\text{ExpEUFSSig}_{\text{S}, \mathcal{B}}(\lambda) = 1] \leq \text{negl}(\lambda)$. $\square$

Claim 3 $\Pr[\text{Bad}_2 \wedge \text{PK} = \text{PK}_{i^*}] \leq \text{negl}(\lambda)$.

Proof We construct a PPT reduction $\mathcal{B}$ against the EUF-CMA security of S .

Setup Phase

On receiving a challenge public key $\text{pk}^\dagger$ and oracle access to O_{Sig} , the algorithm $\mathcal{B}$ chooses $\hat{i} \leftarrow [n]$ uniformly and sets $\text{PK}_{\hat{i}} := \text{pk}^\dagger$. For every $i \neq \hat{i}$, it generates $(\text{SK}_i, \text{PK}_i) \leftarrow \text{S.KG}(1^\lambda)$. It generates $(\text{sk}_S, \text{pk}_S) \leftarrow \text{S.KG}(1^\lambda)$ and computes $\text{cert}_i \leftarrow \text{S.Sig}(\text{sk}_S, (i, \text{PK}_i))$ for every $i \in [n]$, including $i = \hat{i}$. It also generates $(\text{sk}_E, \text{pk}_E) \leftarrow \text{E.KG}(1^\lambda)$, $\text{crs} \leftarrow \text{ZK.CRS}(1^\lambda)$, and $(\text{ask}_i, \text{apk}_i) \leftarrow \text{S}_{\text{acc}}.\text{KG}(1^\lambda)$ for every $i \in [n]$. It sets $\text{gmsk} := \text{sk}_E$ and $\text{gpk} := (\text{crs}, \text{pk}_E, \text{pk}_S)$, defines $\text{sk}_i := (i, \text{SK}_i, \text{PK}_i, \text{cert}_i)$ for every $i \neq \hat{i}$, and gives $\mathcal{A}$ the input prescribed by ExpTrc .

Challenge Phase

The algorithm $\mathcal{B}$ answers the queries of $\mathcal{A}$ as follows, maintaining L_{AC} and L_{Co} as in ExpTrc .

$\text{O}_{\text{AC}}(i, \text{m})$: If $i \neq \hat{i}$, $\mathcal{B}$ returns $\Pi.\text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, \text{m})$. If $i = \hat{i}$, $\mathcal{B}$ obtains σ' by querying O_{Sig} on m , and completes the remaining steps of $\Pi.\text{AccGSig}$ itself: it computes $\sigma_{\text{acc}} \leftarrow \text{S}_{\text{acc}}.\text{Sig}(\text{ask}_{\hat{i}}, (\hat{i}, \text{m}))$, samples $r \leftarrow \text{R}$, computes

$$c \leftarrow \text{E.Enc}(\text{pk}_E, (0^\lambda, \text{m}, \hat{i}, \text{PK}_{\hat{i}}, \text{cert}_{\hat{i}}, \sigma', \sigma_{\text{acc}}); r)$$

and $\pi \leftarrow \text{ZK.Priv}(\text{crs}, (\text{pk}_E, \text{pk}_S, \text{m}, c), (\hat{i}, \text{PK}_{\hat{i}}, \text{cert}_{\hat{i}}, \sigma', \sigma_{\text{acc}}, r))$, and returns (π, c) .

$\text{O}_{\text{Cor}}(i)$: If $i = \hat{i}$, $\mathcal{B}$ aborts. Otherwise, it returns $(\text{sk}_i, \text{ask}_i)$.

When $\mathcal{A}$ outputs (m^*, σ^*) , the algorithm $\mathcal{B}$ checks, using sk_E , whether $\text{Bad}_2 \wedge \text{PK} = \text{PK}_{i^*} \wedge i^* = \hat{i}$ occurs, and aborts if not. Otherwise, it decrypts c^* to obtain σ' and outputs (m^*, σ') .

Analysis

Until $\mathcal{B}$ aborts, the simulation is perfect, and $\hat{i}$ is independent of the view of $\mathcal{A}$. Suppose that $\text{Bad}_2 \wedge \text{PK} = \text{PK}_{i^*}$ occurs and $\hat{i} = i^*$. Since $i^* \notin \text{L}_{\text{Co}}$, $\mathcal{A}$ never queried $\text{O}_{\text{Cor}}(\hat{i})$, so $\mathcal{B}$ does not abort. Since $(i^*, \text{m}^*) \notin \text{L}_{\text{AC}}$, $\mathcal{B}$ never queried O_{Sig} on m^* . Moreover, $\text{S.Vrf}(\text{pk}^\dagger, \text{m}^*, \sigma') = \text{S.Vrf}(\text{PK}, \text{m}^*, \sigma') = 1$. Hence (m^*, σ') is a valid fresh forgery, and

$$\Pr[\text{ExpEUF}_{\text{Sig}, \mathcal{B}}(\lambda) = 1] \geq \frac{1}{n} \Pr[\text{Bad}_2 \wedge \text{PK} = \text{PK}_{i^*}].$$

Since $n = \text{poly}(\lambda)$, the claim follows. $\square$

Combining the bound on Bad_1 with Claims 2 and 3 gives $\Pr[\text{ExpTrc}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda)$. $\square$

Lemma 3 (Post-Compromise Unforgeability) *The construction Π is post-compromise unforgeable if S_{acc} is EUF-CMA secure.*

Proof Let $\mathcal{A}$ be a PPT adversary in ExpPCU . We construct a PPT reduction $\mathcal{B}$ against the EUF-CMA security of S_{acc} .

Setup Phase

On receiving a challenge public key pk and oracle access to O_{Sig} , the algorithm $\mathcal{B}$ chooses $i^* \leftarrow [n]$ uniformly and sets $\text{apk}_{i^*} := \text{pk}$. For every $i \neq i^*$, it generates $(\text{ask}_i, \text{apk}_i) \leftarrow \text{S}_{\text{acc}}.\text{KG}(1^\lambda)$. It then runs $(\text{gmsk}, \text{gpk}, \{\text{sk}_i\}_{i \in [n]}) \leftarrow \text{GKG}(1^\lambda)$ entirely by itself; this step does not involve any accountable key, so $\mathcal{B}$ obtains $\text{gmsk} = \text{sk}_E$. It gives $\mathcal{A}$ the tuple $(\text{gpk}, \{\text{sk}_i\}_{i \in [n]}, \{\text{apk}_i\}_{i \in [n]})$ and initializes $\text{L}_{\text{AC}} := \emptyset$.

Challenge Phase

The algorithm $\mathcal{B}$ answers $\text{O}_{\text{AC}}(i, m)$ as follows. If $i \neq i^*$, it returns $\Pi.\text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, m)$. If $i = i^*$, it obtains σ_{acc} by querying O_{Sig} on (i^*, m) , computes $\sigma' \leftarrow \text{S.Sig}(\text{SK}_{i^*}, m)$, samples $r \leftarrow \mathcal{R}$, computes

$$c \leftarrow \text{E.Enc}(\text{pk}_E, (0^\lambda, m, i^*, \text{PK}_{i^*}, \text{cert}_{i^*}, \sigma', \sigma_{\text{acc}}); r)$$

and

$$\pi \leftarrow \text{ZK.Priv}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), (i^*, \text{PK}_{i^*}, \text{cert}_{i^*}, \sigma', \sigma_{\text{acc}}, r)),$$

and returns (π, c) . In either case, it updates $\text{L}_{\text{AC}} := \text{L}_{\text{AC}} \cup \{(i, m)\}$.

When $\mathcal{A}$ outputs (m^*, σ^*) with $\sigma^* = (\pi^*, c^*)$, the algorithm $\mathcal{B}$ computes $i := \Pi.\text{Open}(\text{gmsk}, \text{gpk}, m^*, \sigma^*)$ and aborts unless $\Pi.\text{Vrf}(\text{gpk}, m^*, \sigma^*) = 1$ and $i = i^*$. Otherwise, it decrypts c^* , parses the plaintext as $(0^\lambda, m^*, i^*, \text{PK}, \text{cert}, \sigma', \sigma_{\text{acc}}^*)$, and outputs $((i^*, m^*), \sigma_{\text{acc}}^*)$.

Analysis

Since $\mathcal{B}$ holds gmsk , sk_S , and every SK_i , the view of $\mathcal{A}$ is distributed exactly as in ExpPCU , and i^* is independent of this view. No property of E or ZK is used. Suppose that $\mathcal{A}$ wins and that the index i output by $\Pi.\text{Open}$ equals i^* . Then $\Pi.\text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}_{i^*}, m^*, \sigma^*) = 1$. Since $\Pi.\text{Reval}$ decrypts the same ciphertext c^* and verifies its last component with respect to the identity contained in the plaintext, we have $\text{S}_{\text{acc}}.\text{Vrf}(\text{pk}, (i^*, m^*), \sigma_{\text{acc}}^*) = 1$. Furthermore, $(i^*, m^*) \notin \text{L}_{\text{AC}}$, so $\mathcal{B}$ never queried (i^*, m^*) to O_{Sig} . Thus $\mathcal{B}$ outputs a valid forgery, and

$$\Pr[\text{ExpPCU}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq n \cdot \Pr[\text{ExpEUF-Sig}_{\text{S}_{\text{acc}}, \mathcal{B}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

□

Lemma 4 (Opening-Revalidation Consistency) *The construction Π satisfies opening-revalidation consistency if S is EUF-CMA secure, S_{acc} is correct, EUF-CMA secure, and key-binding, E is correct and IND-CCA2 secure, and ZK is zero-knowledge.*

Proof Let $\mathcal{A}$ be a PPT adversary in ExpORCon . Throughout, we condition on the event that the accountable public keys are pairwise distinct. Write $\sigma^* = (\pi^*, c^*)$, and let i^* denote the output of $\Pi.\text{Open}$ on (m^*, σ^*) . By inspection of Fig. 3, $\text{ExpORCon}_{\Pi, \mathcal{A}}(\lambda)$ outputs 1 only if $\Pi.\text{Vrf}(\text{gpk}, m^*, \sigma^*) = 1$, $i^* \neq \perp$, and the set

$$J^* := \{j \in [n] \setminus \{i^*\} : \Pi.\text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}_j, m^*, \sigma^*) = 1 \wedge \{i^*, j\} \not\subseteq \text{L}_{\text{Co}}\}$$

is nonempty. In that case, we let j^* be the minimum index in J^* . For each $i \in [n]$, we let $\mathcal{C}_i$ denote the set of ciphertext components of the signatures returned by $\text{O}_{\text{AC}}(i, \cdot)$; this is analytical bookkeeping only.

Proof overview.

We first record the structure of a winning output, which follows from the definitions of $\Pi.\text{Open}$ and $\Pi.\text{Reval}$ alone. We then show that the ordinary public key in the decrypted plaintext is the registered key PK_{i^*} , and partition the remaining winning event into four cases. If j^* is uncorrupted, we obtain a forgery against S_{acc} . If j^* is corrupted but (i^*, m^*) was never queried to O_{AC} , we obtain a forgery against S . If c^* was returned by $\text{O}_{\text{AC}}(i^*, \cdot)$, the same accountable signature would be valid under two distinct accountable public keys, contradicting key binding. Finally, if c^* is a fresh ciphertext, we use IND-CCA2 security to replace the ciphertexts returned for i^* by encryptions of $M_\perp$, after which a winning output

yields a forgery against $\mathcal{S}$. The last case is the only one in which the adversary may combine an honest user's signature with an accountable signature of its own; it is excluded by the non-malleability provided by IND-CCA2 security.

Structural properties of a winning output.

Suppose that $\mathcal{A}$ wins. Since $\Pi.\text{Open}$ returns $i^* \neq \perp$, the plaintext $\mathbf{E}.\text{Dec}(\text{sk}_E, c^*)$ parses as $(0^\lambda, m^*, i^*, \text{PK}, \text{cert}, \sigma', \sigma_{\text{acc}}^*)$ with $\mathbf{S}.\text{Vrf}(\text{pk}_S, (i^*, \text{PK}), \text{cert}) = 1$ and $\mathbf{S}.\text{Vrf}(\text{PK}, m^*, \sigma') = 1$. Since $\Pi.\text{Reval}$ decrypts the same ciphertext, $j^* \in J^*$ implies $\mathbf{S}_{\text{acc}}.\text{Vrf}(\text{apk}_{j^*}, (i^*, m^*), \sigma_{\text{acc}}^*) = 1$. Neither the soundness of ZK nor the correctness of $\mathbf{E}$ is needed for these two facts.

Bounding $\Pr[\text{PK} \neq \text{PK}_{i^*}]$.

We construct a reduction $\mathcal{B}_0$ against the EUF-CMA security of $\mathcal{S}$. On receiving pk and oracle access to $\mathbf{O}_{\text{Sig}}$, the algorithm $\mathcal{B}_0$ sets $\text{pk}_S := \text{pk}$, generates $(\text{SK}_i, \text{PK}_i) \leftarrow \mathbf{S}.\text{KG}(1^\lambda)$ and obtains cert_i by querying $\mathbf{O}_{\text{Sig}}$ on (i, PK_i) for every $i \in [n]$, and generates $(\text{sk}_E, \text{pk}_E) \leftarrow \mathbf{E}.\text{KG}(1^\lambda)$, $\text{crs} \leftarrow \text{ZK}.\text{CRS}(1^\lambda)$, and $(\text{ask}_i, \text{apk}_i) \leftarrow \mathbf{S}_{\text{acc}}.\text{KG}(1^\lambda)$ for every $i \in [n]$. It gives $\mathcal{A}$ the tuple $(\text{gpk}, \{\text{apk}_i\}_{i \in [n]})$ and answers $\mathbf{O}_{\text{AC}}$ and $\mathbf{O}_{\text{Cor}}$ honestly, which it can do since it knows every SK_i and every ask_i . When $\mathcal{A}$ outputs (m^*, σ^*) , $\mathcal{B}_0$ uses sk_E to check whether $\mathcal{A}$ wins and $\text{PK} \neq \text{PK}_{i^*}$; if so, it outputs $((i^*, \text{PK}), \text{cert})$, and otherwise it aborts. The simulation is perfect, and, as in the proof of Claim 2, (i^*, PK) was never queried to $\mathbf{O}_{\text{Sig}}$. Therefore

$$\Pr[\mathcal{A} \text{ wins} \wedge \text{PK} \neq \text{PK}_{i^*}] \leq \Pr[\text{ExpEUF}(\text{Sig}_{\mathcal{S}, \mathcal{B}_0})(\lambda) = 1] \leq \text{negl}(\lambda).$$

Case decomposition.

From here on, let Good_{PK} denote the event $\text{PK} = \text{PK}_{i^*}$. Whenever $\mathcal{A}$ wins and Good_{PK} holds, exactly one of the following events occurs.

Bad_a : $j^* \notin L_{\text{Co}}$.

Bad_{b1} : $j^* \in L_{\text{Co}}$ and $(i^*, m^*) \notin L_{\text{AC}}$.

Bad_{b2a} : $j^* \in L_{\text{Co}}$, $(i^*, m^*) \in L_{\text{AC}}$, and $c^* \in \mathcal{C}_{i^*}$.

Bad_{b2b} : $j^* \in L_{\text{Co}}$, $(i^*, m^*) \in L_{\text{AC}}$, and $c^* \notin \mathcal{C}_{i^*}$.

Here each event is understood to include that $\mathcal{A}$ wins and Good_{PK} holds. In the last three cases, $i^* \notin L_{\text{Co}}$, since $j^* \in L_{\text{Co}}$ and $\{i^*, j^*\} \not\subseteq L_{\text{Co}}$.

Bounding $\Pr[\text{Bad}_a]$.

We construct a reduction $\mathcal{B}_1$ against the EUF-CMA security of $\mathbf{S}_{\text{acc}}$. On receiving pk and oracle access to $\mathbf{O}_{\text{Sig}}$, the algorithm $\mathcal{B}_1$ chooses $k \leftarrow [n]$ uniformly, sets $\text{apk}_k := \text{pk}$, and generates $(\text{ask}_i, \text{apk}_i) \leftarrow \mathbf{S}_{\text{acc}}.\text{KG}(1^\lambda)$ for $i \neq k$. It runs GKG by itself, thereby obtaining sk_E , and gives $\mathcal{A}$ the tuple $(\text{gpk}, \{\text{apk}_i\}_{i \in [n]})$. It answers $\mathbf{O}_{\text{AC}}(i, m)$ with $\Pi.\text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, m)$ if $i \neq k$. If $i = k$, it obtains σ_{acc} by querying $\mathbf{O}_{\text{Sig}}$ on (k, m) and completes the remaining steps of $\Pi.\text{AccGSig}$ itself, which do not involve ask_k . It answers $\mathbf{O}_{\text{Cor}}(i)$ with $(\text{sk}_i, \text{ask}_i)$ if $i \neq k$, and aborts if $i = k$. When $\mathcal{A}$ outputs (m^*, σ^*) , $\mathcal{B}_1$ uses sk_E to compute i^* , J^* , and j^* , and aborts unless Bad_a occurs and $j^* = k$. Otherwise, it decrypts c^* and outputs $((i^*, m^*), \sigma_{\text{acc}}^*)$.

Until $\mathcal{B}_1$ aborts, the simulation is perfect, and k is independent of the view of $\mathcal{A}$. If Bad_a occurs and $j^* = k$, then $\mathcal{A}$ never queried $\mathbf{O}_{\text{Cor}}(k)$, so $\mathcal{B}_1$ does not abort before the end. By the structural property, $\mathbf{S}_{\text{acc}}.\text{Vrf}(\text{pk}, (i^*, m^*), \sigma_{\text{acc}}^*) = 1$. Every signing query of $\mathcal{B}_1$ has the form $(k, \cdot) = (j^*, \cdot)$, whereas $j^* \neq i^*$; hence (i^*, m^*) is fresh. Therefore

$$\Pr[\text{Bad}_a] \leq n \cdot \Pr[\text{ExpEUF}(\text{Sig}_{\text{acc}, \mathcal{B}_1})(\lambda) = 1] \leq \text{negl}(\lambda).$$

Bounding $\Pr[\text{Bad}_{b1}]$.

We construct a reduction $\mathcal{B}_2$ against the EUF-CMA security of $\mathcal{S}$. On receiving pk and oracle access to O_{Sig} , the algorithm $\mathcal{B}_2$ chooses $\hat{i} \leftarrow [n]$ uniformly and sets $\text{PK}_{\hat{i}} := \text{pk}$. It generates all remaining keys itself, including $\text{sk}_{\mathcal{E}}$, $\text{sk}_{\mathcal{S}}$, every cert_i , every SK_i with $i \neq \hat{i}$, and every $(\text{ask}_i, \text{apk}_i)$, and gives $\mathcal{A}$ the tuple $(\text{gpk}, \{\text{apk}_i\}_{i \in [n]})$. It answers $\text{O}_{\text{AC}}(i, \mathbf{m})$ with $\Pi.\text{AccGSig}(\text{gpk}, \text{sk}_i, \text{ask}_i, \mathbf{m})$ if $i \neq \hat{i}$. If $i = \hat{i}$, it obtains σ' by querying O_{Sig} on $\mathbf{m}$ and completes the remaining steps of $\Pi.\text{AccGSig}$ itself. It answers $\text{O}_{\text{Cor}}(i)$ honestly if $i \neq \hat{i}$, and aborts if $i = \hat{i}$. When $\mathcal{A}$ outputs $(\mathbf{m}^*, \sigma^*)$, $\mathcal{B}_2$ uses $\text{sk}_{\mathcal{E}}$ to check whether Bad_{b1} occurs and $i^* = \hat{i}$, and aborts if not. Otherwise, it decrypts c^* and outputs $(\mathbf{m}^*, \sigma')$.

Until $\mathcal{B}_2$ aborts, the simulation is perfect, and $\hat{i}$ is independent of the view of $\mathcal{A}$. If Bad_{b1} occurs and $i^* = \hat{i}$, then $i^* \notin \text{L}_{\text{Co}}$, so $\mathcal{B}_2$ does not abort on a corruption query, and $(i^*, \mathbf{m}^*) \notin \text{L}_{\text{AC}}$, so $\mathcal{B}_2$ never queried O_{Sig} on $\mathbf{m}^*$. By Good_{PK} and the structural property, $\mathcal{S}.\text{Vrf}(\text{pk}, \mathbf{m}^*, \sigma') = 1$. Therefore

$$\Pr[\text{Bad}_{b1}] \leq n \cdot \Pr[\text{ExpEUFSig}_{\mathcal{S}, \mathcal{B}_2}(\lambda) = 1] \leq \text{negl}(\lambda).$$

Bounding $\Pr[\text{Bad}_{b2a}]$.

Suppose that Bad_{b2a} occurs. Since $c^* \in \mathcal{C}_{i^*}$, there is a query $\text{O}_{\text{AC}}(i^*, \tilde{\mathbf{m}})$ whose response has ciphertext component c^* , so that

$$c^* = \text{E.Enc}(\text{pk}_{\mathcal{E}}, \tilde{\mathbf{M}}; \tilde{r}), \quad \tilde{\mathbf{M}} = (0^\lambda, \tilde{\mathbf{m}}, i^*, \text{PK}_{i^*}, \text{cert}_{i^*}, \tilde{\sigma}', \tilde{\sigma}_{\text{acc}}),$$

where $\tilde{\sigma}_{\text{acc}} \leftarrow \mathcal{S}_{\text{acc}}.\text{Sig}(\text{ask}_{i^*}, (i^*, \tilde{\mathbf{m}}))$. Under $\neg \text{Bad}_{\text{corr}}$, $\text{E.Dec}(\text{sk}_{\mathcal{E}}, c^*) = \tilde{\mathbf{M}}$. Since $\Pi.\text{Open}$ checks that the message in the plaintext equals $\mathbf{m}^*$, we have $\tilde{\mathbf{m}} = \mathbf{m}^*$, and $\sigma_{\text{acc}}^* = \tilde{\sigma}_{\text{acc}}$. By the correctness of $\mathcal{S}_{\text{acc}}$, $\mathcal{S}_{\text{acc}}.\text{Vrf}(\text{apk}_{i^*}, (i^*, \mathbf{m}^*), \tilde{\sigma}_{\text{acc}}) = 1$, except with negligible probability. By the structural property, $\mathcal{S}_{\text{acc}}.\text{Vrf}(\text{apk}_{j^*}, (i^*, \mathbf{m}^*), \tilde{\sigma}_{\text{acc}}) = 1$. Since $j^* \neq i^*$, the keys apk_{j^*} and apk_{i^*} are distinct, which contradicts the key-binding property of $\mathcal{S}_{\text{acc}}$. Hence $\Pr[\text{Bad}_{b2a}] \leq \Pr[\text{Bad}_{\text{corr}}] + \text{negl}(\lambda) \leq \text{negl}(\lambda)$.

Bounding $\Pr[\text{Bad}_{b2b}]$.

We use a sequence of hybrids. Each hybrid additionally samples $\hat{i} \leftarrow [n]$ uniformly and independently of everything else, and P_h denotes the probability of $\text{Bad}_{b2b} \wedge \hat{i} = i^*$ in Hybrid h . Let $Q = \text{poly}(\lambda)$ be an upper bound on the number of O_{AC} queries of $\mathcal{A}$.

Hybrid 0.

Hybrid 0 is $\text{ExpORCon}_{\Pi, \mathcal{A}}(\lambda)$ together with $\hat{i}$. Since $\hat{i}$ is independent of the experiment, $P_0 = \frac{1}{n} \Pr[\text{Bad}_{b2b}]$.

Hybrid 1.

Hybrid 1 is identical to Hybrid 0 except that $(\text{crs}, \tau) \leftarrow \text{ZK.SimCRS}(1^\lambda)$ is used and every proof returned by O_{AC} is generated as $\text{ZK.Sim}(\text{crs}, \cdot, \tau)$. A reduction $\mathcal{D}$ to the zero-knowledge property of ZK obtains crs from its challenger, generates all other keys itself, including $\text{sk}_{\mathcal{E}}$, and obtains every O_{AC} proof by querying its oracle on the corresponding statement–witness pair. At the end, it evaluates $\text{Bad}_{b2b} \wedge \hat{i} = i^*$ using $\text{sk}_{\mathcal{E}}$. It perfectly realizes Hybrid 0 or Hybrid 1 depending on its challenge bit, so $|P_0 - P_1| \leq \text{negl}(\lambda)$.

Hybrid 2.

Hybrid 2 is identical to Hybrid 1 except for the responses to $\text{O}_{\text{AC}}(i, \mathbf{m})$. For each such query, Hybrid 2 computes $c \leftarrow \text{E.Enc}(\text{pk}_{\mathcal{E}}, M_\perp; r)$ for $r \leftarrow \mathcal{R}$, where $M_\perp$ has the length of the

honest plaintext, and returns (π, c) with $\pi \leftarrow \text{ZK.Sim}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), \tau)$. All other oracle responses are as in Hybrid 1.

We show $|P_1 - P_2| \leq \text{negl}(\lambda)$ by a reduction $\mathcal{B}_3$ against the LoR-CCA security of E . On receiving pk , the algorithm $\mathcal{B}_3$ sets $\text{pk}_E := \text{pk}$, samples $\hat{i} \leftarrow [n]$, and generates all other keys itself: $(\text{crs}, \tau) \leftarrow \text{ZK.SimCRS}(1^\lambda)$, $(\text{sk}_S, \text{pk}_S)$, every $(\text{SK}_i, \text{PK}_i)$ and cert_i , and every $(\text{ask}_i, \text{apk}_i)$. It gives $\mathcal{A}$ the tuple $(\text{gpk}, \{\text{apk}_i\}_{i \in [n]})$ and answers the queries of $\mathcal{A}$ as follows.

$\text{O}_{\text{AC}}(i, m)$: $\mathcal{B}_3$ computes $\sigma' \leftarrow \text{S.Sig}(\text{SK}_i, m)$, $\sigma_{\text{acc}} \leftarrow \text{S}_{\text{acc}}.\text{Sig}(\text{ask}_i, (i, m))$, and $M := (0^\lambda, m, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}})$. If $i \neq \hat{i}$, it computes $c \leftarrow E.\text{Enc}(\text{pk}_E, M; r)$ for $r \leftarrow R$. If $i = \hat{i}$, it obtains c by querying O_{LR} on $(M, M_\perp)$. In either case, it computes $\pi \leftarrow \text{ZK.Sim}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), \tau)$, updates L_{AC} and $\mathcal{C}_i$, and returns (π, c) .

$\text{O}_{\text{Cor}}(i)$: $\mathcal{B}_3$ updates L_{Co} and returns $(\text{sk}_i, \text{ask}_i)$.

When $\mathcal{A}$ outputs $(m^*, (\pi^*, c^*))$, the algorithm $\mathcal{B}_3$ outputs 0 if $c^* \in \mathcal{C}_{\hat{i}}$. Otherwise, c^* was never returned by O_{LR} , so $\mathcal{B}_3$ may query O_{Dec} on c^* . From the resulting plaintext, $\mathcal{B}_3$ runs the remaining steps of $\Pi.\text{Open}$ and of $\Pi.\text{Reval}$ with respect to every apk_j , and thereby evaluates whether $\text{Bad}_{b2b} \wedge \hat{i} = i^*$ occurs; it outputs 1 if and only if it does.

If O_{LR} encrypts its left inputs, $\mathcal{B}_3$ perfectly realizes Hybrid 1, and if it encrypts its right inputs, $\mathcal{B}_3$ perfectly realizes Hybrid 2. The final test is exact because $\text{Bad}_{b2b} \wedge \hat{i} = i^*$ implies $c^* \notin \mathcal{C}_{i^*} = \mathcal{C}_{\hat{i}}$. Since $\mathcal{B}_3$ makes at most Q queries to O_{LR} , Fact 2 gives

$$|P_1 - P_2| = \text{Adv}_{E, \mathcal{B}_3}^{\text{lor}}(\lambda) \leq Q \cdot \text{Adv}_{E, \mathcal{B}_3'}^{\text{cca}}(\lambda) \leq \text{negl}(\lambda)$$

for some PPT adversary $\mathcal{B}_3'$.

Bounding P_2 .

We construct a reduction $\mathcal{B}_4$ against the EUF-CMA security of S . On receiving pk and oracle access to O_{Sig} , the algorithm $\mathcal{B}_4$ samples $\hat{i} \leftarrow [n]$, sets $\text{PK}_{\hat{i}} := \text{pk}$, and generates all other keys itself: $(\text{sk}_E, \text{pk}_E)$, $(\text{crs}, \tau) \leftarrow \text{ZK.SimCRS}(1^\lambda)$, $(\text{sk}_S, \text{pk}_S)$, every cert_i including $\text{cert}_{\hat{i}}$, every $(\text{SK}_i, \text{PK}_i)$ with $i \neq \hat{i}$, and every $(\text{ask}_i, \text{apk}_i)$. It gives $\mathcal{A}$ the tuple $(\text{gpk}, \{\text{apk}_i\}_{i \in [n]})$ and answers the queries of $\mathcal{A}$ as in Hybrid 2. In particular, every response to $\text{O}_{\text{AC}}(\hat{i}, \cdot)$ consists of an encryption of $M_\perp$ and a simulated proof, and therefore requires no signature under $\text{SK}_{\hat{i}}$. It answers $\text{O}_{\text{Cor}}(i)$ honestly if $i \neq \hat{i}$, and aborts if $i = \hat{i}$. When $\mathcal{A}$ outputs (m^*, σ^*) , $\mathcal{B}_4$ uses sk_E to check whether $\text{Bad}_{b2b} \wedge \hat{i} = i^*$ occurs, and aborts if not. Otherwise, it decrypts c^* and outputs (m^*, σ') .

Until $\mathcal{B}_4$ aborts, it perfectly realizes Hybrid 2. If $\text{Bad}_{b2b} \wedge \hat{i} = i^*$ occurs, then $\hat{i} \notin \text{L}_{\text{Co}}$, so $\mathcal{B}_4$ does not abort on a corruption query. By Good_{PK} and the structural property, $\text{S.Vrf}(\text{pk}, m^*, \sigma') = 1$. Since $\mathcal{B}_4$ never queries its signing oracle, (m^*, σ') is a valid fresh forgery. Therefore $P_2 \leq \Pr[\text{ExpEUF}(\text{Sig}_S, \mathcal{B}_4)(\lambda) = 1] \leq \text{negl}(\lambda)$.

Combining.

By the triangle inequality,

$$\Pr[\text{Bad}_{b2b}] = n \cdot P_0 \leq n(|P_0 - P_1| + |P_1 - P_2| + P_2) \leq \text{negl}(\lambda).$$

Conclusion.

Combining the bound on $\Pr[\neg \text{Good}_{\text{PK}}]$ with the four cases,

$$\begin{aligned} \Pr[\text{ExpORCon}_{\Pi, \mathcal{A}}(\lambda) = 1] &\leq \Pr[\mathcal{A} \text{ wins} \wedge \neg \text{Good}_{\text{PK}}] + \Pr[\text{Bad}_a] + \Pr[\text{Bad}_{b1}] \\ &\quad + \Pr[\text{Bad}_{b2a}] + \Pr[\text{Bad}_{b2b}] \leq \text{negl}(\lambda). \end{aligned}$$

Hence Π satisfies opening-revalidation consistency. $\square$

Lemma 5 (Indistinguishability) *The construction Π satisfies indistinguishability with respect to RefGSig if $\mathbf{E}$ is correct and IND-CCA2 secure, $\mathbf{S}$ is correct, and $\mathbf{ZK}$ is complete and zero-knowledge.*

Proof We verify the two properties required by Definition 15.

Property 1: Correctness of RefGSig .

Fix honestly generated keys, $i \in [n]$, and a message $\mathbf{m}$, and let $\sigma = (\pi, c) \leftarrow \text{RefGSig}(\text{gpk}, \text{sk}_i, \mathbf{m})$. By the correctness of $\mathbf{S}$, the pair consisting of $(\text{pk}_E, \text{pk}_S, \mathbf{m}, c)$ and $(i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_\perp, r)$ is in $R_{\mathcal{L}}$, so $\Pi.\text{Vrf}(\text{gpk}, \mathbf{m}, \sigma) = 1$ by the completeness of $\mathbf{ZK}$, except with negligible probability. By the correctness of $\mathbf{E}$, c decrypts to $(0^\lambda, \mathbf{m}, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_\perp)$. The checks of $\Pi.\text{Open}$ do not involve the last component, so they all succeed and $\Pi.\text{Open}$ returns i .

Property 2: Indistinguishability.

Let $\mathcal{A}$ be a PPT adversary in $\text{ExpIND}_{\Pi, \text{RefGSig}, \mathcal{A}}$. Recall that $\mathcal{A}$ receives $(\text{gpk}, \{\text{sk}_i\}_{i \in [n]})$ and access to $\mathbf{O}_b$ only; in particular, it has access to neither an opening nor a revalidation oracle. For a query $(i, \mathbf{m})$, the algorithms AccGSig and RefGSig differ only in the last plaintext component, which is $\sigma_{\text{acc}} \leftarrow \mathbf{S}_{\text{acc}}.\text{Sig}(\text{ask}_i, (i, \mathbf{m}))$ in the former and $\sigma_\perp$ in the latter; both have length ℓ_{acc} . Let $Q = \text{poly}(\lambda)$ be an upper bound on the number of oracle queries of $\mathcal{A}$, and, for each Hybrid h , let P_h denote the probability that $\mathcal{A}$ outputs 1.

Hybrid 0.

Hybrid 0 is ExpIND with $b = 1$, in which every query $(i, \mathbf{m})$ is answered by $\Pi.\text{AccGSig}$ with the keys of user i .

Hybrid 1.

Hybrid 1 is identical to Hybrid 0 except that $(\text{crs}, \tau) \leftarrow \mathbf{ZK}.\text{SimCRS}(1^\lambda)$ is used and every proof is generated as $\pi \leftarrow \mathbf{ZK}.\text{Sim}(\text{crs}, (\text{pk}_E, \text{pk}_S, \mathbf{m}, c), \tau)$. A reduction to the zero-knowledge property of $\mathbf{ZK}$ generates all other keys itself and obtains each proof by querying its oracle on the honest statement–witness pair; hence $|P_0 - P_1| \leq \text{negl}(\lambda)$.

Hybrid 2.

Hybrid 2 is identical to Hybrid 1 except that every response encrypts $(0^\lambda, \mathbf{m}, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_\perp)$ instead of $(0^\lambda, \mathbf{m}, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}})$; proofs are still simulated.

We show $|P_1 - P_2| \leq \text{negl}(\lambda)$ by a reduction $\mathcal{B}$ against the LoR-CCA security of $\mathbf{E}$.

Setup Phase.

On receiving pk , the algorithm $\mathcal{B}$ sets $\text{pk}_E := \text{pk}$ and generates $(\text{crs}, \tau) \leftarrow \mathbf{ZK}.\text{SimCRS}(1^\lambda)$, $(\text{sk}_S, \text{pk}_S) \leftarrow \mathbf{S}.\text{KG}(1^\lambda)$, and, for every $i \in [n]$, $(\text{SK}_i, \text{PK}_i) \leftarrow \mathbf{S}.\text{KG}(1^\lambda)$, $\text{cert}_i \leftarrow \mathbf{S}.\text{Sig}(\text{sk}_S, (i, \text{PK}_i))$, and $(\text{ask}_i, \text{apk}_i) \leftarrow \mathbf{S}_{\text{acc}}.\text{KG}(1^\lambda)$. It gives $(\text{gpk}, \{\text{sk}_i\}_{i \in [n]})$ to $\mathcal{A}$.

Challenge Phase.

For each query $(i, \mathbf{m})$, the algorithm $\mathcal{B}$ computes $\sigma' \leftarrow \mathbf{S}.\text{Sig}(\text{SK}_i, \mathbf{m})$ and $\sigma_{\text{acc}} \leftarrow \mathbf{S}_{\text{acc}}.\text{Sig}(\text{ask}_i, (i, \mathbf{m}))$, and queries $\mathbf{O}_{\text{LR}}$ on the pair

$$((0^\lambda, \mathbf{m}, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_{\text{acc}}), (0^\lambda, \mathbf{m}, i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_\perp)),$$

whose components have equal length, to obtain c . It returns (π, c) with $\pi \leftarrow \text{ZK.Sim}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), \tau)$. Finally, $\mathcal{B}$ outputs the bit output by $\mathcal{A}$. It never queries O_{Dec} .

Analysis.

If O_{LR} encrypts its left inputs, $\mathcal{B}$ perfectly realizes Hybrid 1, and if it encrypts its right inputs, $\mathcal{B}$ perfectly realizes Hybrid 2. By Fact 2, $|P_1 - P_2| = \text{Adv}_{E, \mathcal{B}}^{\text{lor}}(\lambda) \leq Q \cdot \text{Adv}_{E, \mathcal{B}'}^{\text{cca}}(\lambda) \leq \text{negl}(\lambda)$ for some PPT $\mathcal{B}'$. Since $\mathcal{B}$ makes no decryption query, IND-CPA security of E would in fact suffice for this step.

Hybrid 3.

Hybrid 3 is identical to Hybrid 2 except that $\text{crs} \leftarrow \text{ZK.CRS}(1^\lambda)$ is used and every proof is generated as $\pi \leftarrow \text{ZK.Priv}(\text{crs}, (\text{pk}_E, \text{pk}_S, m, c), (i, \text{PK}_i, \text{cert}_i, \sigma', \sigma_\perp, r))$. Since these statement-witness pairs are in $R_{\mathcal{L}}$, a reduction to the zero-knowledge property as for Hybrids 0 and 1 gives $|P_2 - P_3| \leq \text{negl}(\lambda)$. Hybrid 3 is exactly ExpIND with $b = 0$.

Hence

$$\left| \Pr[\text{ExpIND}_{\Pi, \text{RefGSig}, \mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| = \frac{1}{2} |P_0 - P_3| \leq \text{negl}(\lambda),$$

and Property 2 holds. $\square$

Lemma 6 (Accountable-Key Committing) *The construction Π satisfies accountable-key committing if E is correct and S_{acc} is correct and key-binding.*

Proof Let $\mathcal{A}$ be a PPT adversary in $\text{ExpAKC}_{\Pi, \mathcal{A}}$, and suppose that $\mathcal{A}$ wins with output $(i^*, m^*, \sigma^*, \text{apk}^*)$. By the winning condition, $(i^*, m^*, \sigma^*) \in L_{\text{AC}}^{\text{AKC}}$, $\text{apk}^* \neq \text{apk}_{i^*}$, and $\Pi.\text{Reval}(\text{gmsk}, \text{gpk}, \text{apk}^*, m^*, \sigma^*) = 1$.

Since $(i^*, m^*, \sigma^*) \in L_{\text{AC}}^{\text{AKC}}$, the signature $\sigma^* = (\pi^*, c^*)$ was returned by $\text{O}_{\text{AC}}^{\text{AKC}}$ on input (i^*, m^*) , and was therefore generated by $\Pi.\text{AccGSig}$ with the keys of user i^* . Thus

$$c^* = E.\text{Enc}(\text{pk}_E, (0^\lambda, m^*, i^*, \text{PK}_{i^*}, \text{cert}_{i^*}, \sigma', \sigma_{\text{acc}}^*); r)$$

for some σ' and r , where $\sigma_{\text{acc}}^* \leftarrow S_{\text{acc}}.\text{Sig}(\text{ask}_{i^*}, (i^*, m^*))$. Under $\neg \text{Bad}_{\text{corr}}$, $\Pi.\text{Reval}$ decrypts c^* to exactly this plaintext and verifies σ_{acc}^* with respect to (i^*, m^*) . Hence the winning condition implies $S_{\text{acc}}.\text{Vrf}(\text{apk}^*, (i^*, m^*), \sigma_{\text{acc}}^*) = 1$. On the other hand, the correctness of S_{acc} gives $S_{\text{acc}}.\text{Vrf}(\text{apk}_{i^*}, (i^*, m^*), \sigma_{\text{acc}}^*) = 1$, except with negligible probability. Since $\text{apk}^* \neq \text{apk}_{i^*}$, this contradicts the key-binding property of S_{acc} , which applies to arbitrary bit strings and hence also to the adversarially chosen apk^* . Therefore $\mathcal{A}$ wins only if Bad_{corr} occurs or the correctness of S_{acc} fails, and

$$\Pr[\text{ExpAKC}_{\Pi, \mathcal{A}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

$\square$

5 Conclusion and Future Work

In this paper, we introduced GSPCA, which extends post-compromise accountability to static group signatures. We formalized its syntax and security requirements, including opening-revalidation consistency, a requirement that arises only when an opening

mechanism and a revalidation mechanism coexist. We then presented a generic construction in the BMW framework and proved it secure in the standard model, assuming only an IND-CCA2 secure PKE scheme, EUF-CMA secure signature schemes, and a simulation-sound NIZK proof system. The construction places the accountable signature in the plaintext of the ciphertext that the BMW framework already addresses to the group manager. Since revalidation, like opening, is a group-manager operation in our model, this ciphertext hides the accountable signature from every other party, and opening and revalidation are evaluated on a single decryption. Even so, opening-revalidation consistency is not automatic: an adversary may attempt to transplant an accountable signature of a corrupted user into a ciphertext that opens to an honest user, and excluding this requires the non-malleability provided by IND-CCA2 security.

Several directions for future work remain. First, it is natural to extend GSPCA to dynamic group signatures. Such an extension would require formulating post-compromise security in the presence of dynamic enrollment and adaptive corruption, and clarifying its interaction with security properties such as non-frameability. Second, our model restricts revalidation to the group manager. Supporting revalidation by parties other than the group manager, as in the public accountability setting of SPCA, cannot rely on a ciphertext that only the group manager can decrypt, and would therefore require a different mechanism for hiding the accountable signature. Third, as discussed in Section 1.2, GSPCA, like SPCA, does not prevent a dishonest member from deliberately signing without its accountable key and later repudiating the signature; enforcing the use of the accountable signing procedure is an interesting open problem. Finally, our construction is generic, and its reductions to the security of S and S_{acc} lose a factor of n . Efficient concrete instantiations, for example in pairing-based or lattice-based settings, and tighter security reductions are left for future work.

Declarations

- **Funding.** This research was supported by JSPS KAKENHI Grant Number JP26H02489, Japan.
- **Competing interests.** The author has no competing interests to declare that are relevant to the content of this article.
- **Data availability.** Data sharing is not applicable to this article, as no datasets were generated or analysed during the current study.
- **Use of AI tools.** The initial research direction, the core idea of extending post-compromise accountability to group signatures, an initial GSPCA construction, and parts of the security analysis were developed by the author. During the subsequent development of the work, the author used ChatGPT (GPT-6) and Claude Opus 5.5 as interactive research and writing assistants to refine definitions and constructions, examine proof strategies and identify potential gaps, explore alternative technical approaches, and improve the presentation of the paper. The author critically reviewed all AI-generated suggestions and independently verified the correctness and originality of all definitions, constructions, claims, and proofs in the final manuscript.

References

- [1] Dayanikli, D., Lang, J., Lehmann, A.: Signatures with post-compromise accountability. In: Galdi, C., Siniscalchi, L. (eds.) *Security and Cryptography for Networks*, pp. 75–95. Springer, Cham (2027)
- [2] Malkin, T., Obana, S., Yung, M.: The hierarchy of key evolving signatures and a characterization of proxy signatures. In: Cachin, C., Camenisch, J. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 306–322. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_19
- [3] Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) *CRYPTO’87*. LNCS, vol. 293, pp. 120–127. Springer, Berlin, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_8
- [4] Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) *CRYPTO’89*. LNCS, vol. 435, pp. 307–315. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_28
- [5] Itakura, K.: A public-key cryptosystem suitable for digital multisignatures (1983). <https://api.semanticscholar.org/CorpusID:60170133>
- [6] Błażkiewicz, P., Kutylowski, M., Słowik, M.: Signature-in-signature: The last line of defence in case of signing key compromise. In: Bertino, E., Shulman, H., Waidner, M. (eds.) *ESORICS 2021, Part II*. LNCS, vol. 12973, pp. 777–782. Springer, Cham (2021). Poster paper
- [7] Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) *EUROCRYPT 2003*. LNCS, vol. 2656, pp. 614–629. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_38
- [8] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) *CT-RSA 2005*. LNCS, vol. 3376, pp. 136–153. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_11
- [9] Lin, X., Sun, X., Ho, P.-H., Shen, X.: Gsis: A secure and privacy-preserving protocol for vehicular communications. *IEEE Transactions on Vehicular Technology* **56**(6), 3442–3456 (2007) <https://doi.org/10.1109/TVT.2007.906878>
- [10] O’Flynn, C., d’Eon, G.: Power analysis and fault attacks against secure CAN: How safe are your keys? *SAE International Journal of Transportation Cybersecurity and Privacy* **1**(1), 3–17 (2018) <https://doi.org/10.4271/11-01-01-0001>
- [11] Pornin, T., Stern, J.P.: Digital signatures do not guarantee exclusive ownership.

- In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 138–150. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11496137_10
- [12] Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) CRYPTO’99. LNCS, vol. 1666, pp. 431–448. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_28
 - [13] Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong key-insulated signature schemes. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_10
 - [14] Itkis, G., Reyzin, L.: SiBIR: Signer-Base Intrusion-Resilient signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_32
 - [15] Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 168–177. ACM Press, New York, NY (2004). <https://doi.org/10.1145/1030083.1030106>
 - [16] Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
 - [17] Jaeger, J., Stracovsky, R.: Dictators? Friends? Forgers. - breaking and fixing unforgeability definitions for anamorphic signature schemes. In: Chung, K.-M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part II. LNCS, vol. 15485, pp. 105–137. Springer, Singapore (2024). https://doi.org/10.1007/978-981-96-0888-1_4
 - [18] Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: Anamorphic signatures: Secrecy from a dictator who only permits authentication! In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 759–790. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38545-2_25
 - [19] Fraser, A., Garms, L., Quaglia, E.A.: On the incoercibility of digital signatures. In: CSF 2023 Computer Security Foundations Symposium, pp. 153–168. IEEE Computer Society Press, Los Alamitos, CA (2023). <https://doi.org/10.1109/CSF57540.2023.00018>
 - [20] Menezes, A., Smart, N.P.: Security of signature schemes in a multi-user setting. DCC **33**(3), 261–274 (2004) <https://doi.org/10.1023/B:DESI.0000036250.18062.3f>
 - [21] Cremers, C., Düzl , S., Fiedler, R., Fischlin, M., Janson, C.: BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In: 2021 IEEE Symposium on Security and Privacy, pp. 1696–1714. IEEE Computer Society Press, Los Alamitos, CA (2021). <https://doi.org/10.1109/SP40001.2021>

00093

- [22] Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_18
- [23] Cachin, C., Camenisch, J. (eds.): EUROCRYPT 2004. LNCS, vol. 3027. Springer, Berlin, Heidelberg (2004)