IACR Communications in Cryptology
Vol. 0, No. 0, 151 pages.

Foundations of Dynamic Group Signatures: The Case of Malicious Openers and Issuers

Stephan Krenn[1], Kai Samelin[2] and Daniel Slamanig[3]

[1] AIT Austrian Institute of Technology, Vienna, Austria
[2] Independent, Germany
[3] Research Institute CODE, Universität der Bundeswehr München, Munich, Germany

Abstract. Dynamic group signatures are commonly formalized in the model of Bellare, Shi, and Zhang (BSZ, CT-RSA 2005), whose security notions are defined relative to specific trust assumptions and registration state.
We revisit these assumptions and systematically study which guarantees continue to hold when the trust assumptions on the issuer and opener are considered independently and no trusted registration structure is shared between them. In particular, we show that the meaning of traceability changes substantially in such settings. With a malicious issuer and an honest opener, the natural analogue of traceability only ensures that every valid signature admits an accepted opening; it does not ensure that the resulting identity obtained its membership through a valid join. With a malicious opener, classical accountability guarantees likewise fail to enforce this connection to group membership. We demonstrate this by constructing a scheme satisfying the BSZ-style security notions in which a malicious opener can produce valid signatures that are not associated with any membership established through the join algorithm, even though the issuer is honest.
To capture the missing relations between joining, signing, and opening, we introduce three unforgeability-type properties. Certificate-unforgeability binds valid signatures to issued membership certificates. Key-unforgeability requires signature generation to rely on a corresponding member signing key. Opening-unforgeability prevents accepted openings to identities for which no membership was issued. We prove that these properties are logically independent and give a construction satisfying the classical and additional guarantees under mild assumptions.
Our results identify precisely which aspects of accountability follow from trusted issuers, trusted openers, or shared registration state, and which must be required explicitly once these assumptions are removed.

Keywords: Group Signatures · Traceability · Security Models · Framework

1 Introduction

Group signatures, introduced by Chaum and van Heyst [CvH91], provide anonymous authentication with accountability. Members can sign on behalf of a group without revealing their identity, while a designated opening authority can later attribute a signature to its signer, for example to resolve a dispute. Dynamic group signatures extend this functionality by allowing users to join the group over time [BSZ05].

The intended semantics connect three distinct operations. An *issuer* admits users to the group, a *member* signs on behalf of the group, and an *opener* attributes valid signatures to identities. Security should therefore ensure not only anonymity and accountable opening,

E-mail: stephan.krenn@ait.ac.at (Stephan Krenn), kaispapers@gmail.com (Kai Samelin), daniel.slamanig@unibw.de (Daniel Slamanig)

This work is licensed under a "CC BY 4.0" license.

but also that signing and opening remain consistent with the memberships established by the prescribed join procedure.

Formal security notions do not capture these requirements in isolation. Their meaning also depends on the syntax, state, and trust assumptions built into the corresponding experiments. A relation between joining, signing, and opening may therefore be enforced by the surrounding experiment even when it is not stated as an independent security property. Once the surrounding structure is changed, the same named notion need not retain the same interpretation.

Motivation. This phenomenon already appears in the standard model of Bellare, Shi, and Zhang (BSZ) [BSZ05]. There, traceability is defined relative to a registration structure populated by join executions. A successful opening points to an entry in this structure, and every such entry arises from the prescribed join procedure. Consequently, the experiment itself establishes the connection between an opened identity and group membership.

Subsequent works weaken individual trust assumptions, especially on the opening side, considering malicious opening authorities; see, for example, Bootle et al. [BCC+16, BCC+20]. These formulations nevertheless retain an issuer-side registration mechanism against which accountability is evaluated. This raises a more fundamental question: which relations between joining, signing, and opening are guaranteed by the security notions themselves, and which follow only from the structural assumptions of the model?

Table 1 summarizes the structural assumptions underlying existing formulations. A common pattern is that accountability requires an honest issuer and a stateful registration mechanism, which implicitly enforces the relation between joining, signing, and opening.

Modeling choices. Our starting point is not the separation of issuer and opener by itself, which already appears in prior models, but the removal of the trusted mechanisms that coordinate their respective views of group membership. Our model combines four choices. First, issuer and opener are syntactically separate and their trust assumptions are considered independently. Second, authority and member keys can be reused across contexts rather than being tied to a single group instance. Third, the scheme algorithms do not receive or share trusted registration state. Security experiments retain challenger-maintained issuance and query logs solely to define winning conditions. Fourth, the relevant interfaces are non-interactive.

We refer to this as a *registry-less non-interactive issuer–opener setting*. Registry-less means that the scheme provides no trusted registration relation linking joining, signing, and opening. Non-interactive means that the scheme syntax consists entirely of local algorithms and contains no interactive protocol; in particular, joining is modeled as a single local issuer algorithm. This is an abstraction at the level of the cryptographic scheme interface and does not preclude the use of an external PKI, registration protocol, or other authentication mechanism in a concrete deployment. As with standard digital signatures, such mechanisms may be required to establish and authenticate the correspondence between parties and their public keys, but are kept outside the cryptographic interface considered here.

Challenger-maintained logs are used only to formulate issuance and freshness conditions. Membership semantics must instead be enforced by explicit security requirements. We also distinguish between exposure of an honestly generated authority key and malicious generation of authority keys; these represent different adversarial settings and are treated separately in the corresponding experiments.

Removing the registration structure changes what classical notions express. With a malicious issuer and an honest opener, the natural analogue of traceability can still require every valid signature to admit an accepted opening. It does not, however, imply that the opened identity obtained membership through the join algorithm. Thus, traceability

Table 1: Architectural and adversarial assumptions underlying representative traceability formulations. Joint denotes a single authority generating issuer and opener key material, whereas Separate denotes distinct roles. Here, H denotes an honestly generated authority, C a malicious or adversarially controlled authority, and $\mathsf{H}^{\mathsf{exp}}$ an honestly generated authority whose secret key is exposed. Yes means that traceability is evaluated relative to trusted membership information established at setup or through join executions, depending on the model. All prior formulations shown assume an honest issuer and such trusted membership information; our model removes both assumptions.

Reference	**Authority roles**	**Issuer**	**Opener**	**Registration**
Bellare et al. [BSZ05]	Separate	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Bichsel et al. [BCN+10]	Joint	H	H	Yes
Sakai et al. [SSE+12]	Separate	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Bootle et al. [BCC+16]	Separate	H	C	Yes
Bootle et al. [BCC+20]	Separate	$\mathsf{H}^{\mathsf{exp}}$	$\mathsf{H}^{\mathsf{exp}}$	Yes
Boneh and Shacham [BS04]	Joint	H	H	Yes
Backes et al. [BHS19]	Separate	H	C	Yes
Groth [Gro07]	Separate	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Benhamouda et al. [BCK+14]	Separate	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Delerablée and Pointcheval [DP06]	Separate	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Alamélou et al. [ABCG17]	Separate	H	H	Yes
Krenn et al. [KSS19]	Separate	H	H	Yes
Bellare et al. [BMW03]	Joint	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Derler and Slamanig [DS18]	Separate	H	$\mathsf{H}^{\mathsf{exp}}$	Yes
Ours	Separate	C	$\mathsf{H}^{\mathsf{exp}}$	No

becomes an opening-related guarantee rather than a guarantee of membership provenance.

This interpretation does not contradict the impossibility of traceability with a malicious issuer in the BSZ model. BSZ traceability is defined relative to a trusted registration mechanism and therefore expresses a stronger connection between opening and enrollment. Once this mechanism is absent, the corresponding opening guarantee has a different meaning.

The gap is also visible when the issuer is honest and the opener is malicious. We construct a scheme satisfying the classical BSZ-style security notions in which a malicious opener can nevertheless produce valid signatures that are not associated with any membership established through the join algorithm. Hence, successful verification and opening do not by themselves establish that a signature originates from a legitimately enrolled member.

Contribution. We provide a systematic decomposition of the trust and state assumptions underlying security for dynamic group signatures. This separates guarantees concerning anonymity and opening from guarantees concerning the provenance of group membership and makes explicit which relations were previously induced by the experiment structure.

We then formulate three additional unforgeability-type properties capturing the missing membership semantics. *Certificate-unforgeability* ensures that producing a fresh valid signature requires access to a valid membership certificate. *Key-unforgeability* analogously requires access to the corresponding member secret key. *Opening-unforgeability* prevents accepted openings to identities for which no membership was issued.

Together, these properties explicitly connect joining, signing, and opening without

Table 2: Security properties and their associated adversarial settings. Here, H denotes an honestly generated party, C an adversarially controlled party, and $\mathsf{H}^{\mathsf{exp}}$ an honestly generated party whose secret key is exposed. The classical notions govern anonymity and opening behavior, whereas the additional unforgeability properties explicitly enforce consistency with membership issuance. For certificate- and key-unforgeability, freshness is replay-based: the guarantee concerns fresh accepted transcripts and should not be read as a causal statement about how the underlying signature string was originally generated.

Property	**Issuer**	**Opener**	**Members**	**Guarantee**
(Strengthened) Classical Properties				
Anonymity	C	H	$\geq 2\,\mathsf{H}^{\mathsf{exp}}$	Hides the signer's identity
Non-frameability	C	C	$\geq 1\,\mathsf{H}$	Prevents false attribution
Traceability	C	$\mathsf{H}^{\mathsf{exp}}$	C	Every valid signature can be opened
Trace-soundness	C	C	C	At most one accepted attribution tuple
Non-claimability	C	C	$\geq 1\,\mathsf{H}^{\mathsf{exp}}$	Prevents re-attribution
Required Additional Properties				
Certificate-unforgeability	H	C	all $\mathsf{H}^{\mathsf{exp}}$	Producing a fresh valid signature requires a membership certificate
Key-unforgeability	H	C	all H	Producing a fresh valid signature requires the corresponding member secret key
Opening-unforgeability	H	C	C	Every accepted opening corresponds to an issued membership

relying on a trusted registration structure.

We establish the relevant independence relations between the classical and the additional properties. In particular, the new properties do not follow from the classical notions in our setting and cannot in general be collapsed into one another without adding further structural assumptions. Finally, we give a modular construction from standard cryptographic primitives and prove that it achieves the resulting set of guarantees.

Table 2 summarizes the decomposition. It records the adversarial setting associated with each property and separates classical opening-related guarantees from the additional guarantees needed to enforce membership semantics.

Related work. Early works on provable security in group signatures include Ateniese et al. [ACJT00], who first introduced the notion of coalition resistance. The modern rigorous formal treatment of group signatures begins with the static model of Bellare, Micciancio, and Warinschi [BMW03] and its extension to dynamic groups by Bellare, Shi, and Zhang [BSZ05]. The BSZ model gives rigorous definitions of anonymity, traceability, and non-frameability for groups in which users can join over time and provides the reference point for much subsequent work. Kiayias and Yung [KY06] give a formal model with dynamic joins and observe that join management and anonymity revocation can be separated, with anonymity holding even against an adversary controlling the join manager; see also [KY05]. For broader background on group and ring signatures, we refer to the survey of Perera et al. [PNH+22].

Many constructions instantiate or extend these models with additional functionality, including efficient dynamic enrollment, revocation, and stronger privacy guarantees; repre-

sentative examples are the works summarized in Table 1. These contributions primarily address how particular functionality and security goals can be realized efficiently. Our focus is instead on which relations are expressed by the experiments and which are supplied by their surrounding registration structure.

Sakai et al. [SSE+12] identify signature hijacking as a gap in earlier formulations and introduce opening soundness to prevent a party from successfully claiming another member's signature. Bootle et al. [BCC+16, BCC+20] develop a comprehensive treatment of fully dynamic group signatures and consider stronger behavior on the opening side. Backes et al. [BHS19] further study privacy in fully dynamic settings. More recently, Emura [Emu26] revisits the role of individual winning conditions in traceability, while Cao and Emura [CE26] give an additional interpretation of non-frameability in the context of anonymous auctions. These works are orthogonal to our setting: they refine or reinterpret the security meaning of classical experiments while retaining the surrounding registration structure. Although these works strengthen or clarify accountability and privacy in important ways, membership continues to be evaluated relative to issuer-maintained registration state.

Our work is complementary. We isolate the relation between joining, signing, and opening and show that this relation is not implied by the classical notions once the trusted registration structure is removed. The additional unforgeability properties make the missing issuance and membership semantics explicit.

Practical relevance. The distinction matters whenever issuance and opening are assigned to different administrative or legal entities. An issuer may maintain its own enrollment records, while an independent opening authority is deliberately prevented from obtaining or trusting the complete registration database. Such separation can arise from organizational separation of duties, data minimization requirements, or the use of independently operated services. Long-lived authority and member keys may additionally be reused across multiple contexts, while non-exportable keys held in hardware security modules prevent the parties from collapsing their roles through key sharing.

In these deployments, the parties need not share a single authoritative view of membership. It is nevertheless tempting to interpret a successful opening as proof that the identified party was legitimately enrolled. Our results show that this conclusion does not follow from classical traceability alone. If enrollment provenance is required, the relation must be captured explicitly by the security definition rather than inferred from the existence of an accepted opening.

Which additional properties are needed? We do not propose the three additional unforgeability notions as a mandatory replacement for the classical framework in every deployment. Their relevance depends on which structural guarantees are already enforced by the deployment. For example, if membership issuance and the corresponding registration state are fully trusted, certificate provenance may already be guaranteed outside the cryptographic security experiment. Likewise, assumptions on opener behavior or on admissible key generation may render some of the additional guarantees redundant. Our aim is instead to identify the guarantees that must be stated explicitly once these structural assumptions are removed. In the registry-less and non-interactive setting considered here, none of the three guarantees can in general be inferred from the remaining security notions.

A concrete example. Sakai et al. [SSE+12] attack the Furukawa–Imai scheme [FI05] with a corrupt opener and two corrupt members. Their result concerns *opening soundness*. We present an extension to an unjoined target that concerns our setting below. Specifically, consider an adaptation whose opening-verification algorithm accepts precisely when the

group signature verifies and the supplied opening proof verifies the relation

$$\exists w \in \mathbb{Z}_q^* : \quad Q = R/V^{1/w}.$$

This adapted verifier performs no additional membership-issuance check for Q. The argument below concerns this explicitly specified verifier, not the full verification interface of the original scheme.

Let $\mathbb{G}$ have prime order q, let $S = g^s$ with $s \in \mathbb{Z}_q^*$, and consider

$$(R, V) = (g^{x_i+r}, S^r), \qquad r \in \mathbb{Z}_q^*.$$

Let the issuer be honest and let the adversary obtain a membership certificate only for the controlled signer $Q_i = g^{x_i}$ under the chosen opener key. The corrupt opener and controlled signer know s, x_i, r. Choose $x_j \in \mathbb{Z}_q \setminus \{x_i, x_i + r\}$ and set $Q_j = g^{x_j}$. All exponent arithmetic is modulo q. Then

$$w = \frac{sr}{x_i + r - x_j} \in \mathbb{Z}_q^*, \qquad \frac{R}{V^{1/w}} = \frac{g^{x_i+r}}{g^{sr/w}} = g^{x_j} = Q_j.$$

Thus the adversary knows a witness for the alternative opening relation.

Using the witness w, the adversary runs the honest opening-proof algorithm to obtain O_j. Completeness of that proof system and validity of the original group signature imply acceptance by the adapted verifier. Since $Q_j \neq Q_i$ and the only join query was for Q_i, the pair (Q_j, opk) is absent from the join relation. Hence the output $(Q_j, \mathsf{opk}, m, \sigma_m, O_j)$ violates opening-unforgeability. While this particular attack also violates trace-soundness, trace-soundness alone does not enforce the missing connection to membership issuance: Theorem 32 exhibits a trace-sound scheme that nevertheless fails opening-unforgeability.

2 Preliminaries

2.1 Notation

Let $\lambda \in \mathbb{N}$ denote the security parameter. Unless stated otherwise, all algorithms take 1^λ as an implicit input. Whenever a security experiment generates public parameters, they are available as implicit input to the adversary and to every oracle and algorithm invoked in the experiment.

For a probabilistic algorithm A, we write

$$a \leftarrow_\$ A(x)$$

to denote that a is sampled according to the output distribution of A on input x. If A is deterministic, we write

$$a \leftarrow A(x).$$

If all randomness is supplied explicitly as ξ, we write

$$a \leftarrow A(x; \xi).$$

Whenever explicit randomness is supplied, ξ ranges over the valid random tapes of A on the given input; invalid random tapes result in $\perp$. Accordingly, statements quantified over all $a \leftarrow_\$ A(x)$ are understood pointwise over all outputs obtained from valid random tapes. In particular, if

$$a = A(x; \xi) \neq \perp$$

for valid ξ, then a is covered by such a correctness statement.

When both the output and the internally sampled randomness are relevant, we write

$$(a; \xi) \leftarrow_{\$} A(x).$$

An algorithm is called *efficient* if it runs in probabilistic polynomial time in λ and in the length of its explicit inputs. Unless stated otherwise, all adversaries and explicitly defined algorithms are assumed to be efficient.

Algorithms may output the distinguished symbol $\perp \notin \{0,1\}^*$ to indicate failure. All algorithms are interpreted as total algorithms on their syntactic input domains. Inputs that do not parse as required, have the wrong type, or fail an implicit domain check result in $\perp$; decision algorithms return 0.

For a nonempty finite set S, the notation

$$a \leftarrow_{\$} S$$

denotes uniform sampling of a from S.

We write $\mathcal{MS}$ for the relevant message space, which is assumed to be efficiently derivable from the corresponding public parameters or public key. All tuples and lists admit an injective and efficiently invertible encoding into $\{0,1\}^*$.

A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible* if, for every positive polynomial $p(\cdot)$, there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$\mathsf{negl}(n) \leq \frac{1}{p(n)}.$$

We use standard one-way functions and an existentially unforgeable digital signature scheme, together with public-key encryption and non-interactive proof systems. In addition to their standard security properties, our constructions require key-verifiability and decryption-soundness for encryption and statement-binding for non-interactive proofs.

These requirements play different roles. Key-verifiability and decryption-soundness are explicit interfaces for properties of encryption that are needed when adversarially generated keys occur in the security proofs; in particular, both are satisfied by the ElGamal instantiation considered below. Statement-binding is a syntactic strengthening that can be obtained generically from any proof system satisfying simulation-sound extractability (SSE) by binding the proof to its complete statement, as shown below. SSE itself is the substantive proof-system requirement used for witness extraction in our reductions. Thus, the additional terminology is primarily intended to expose the exact primitive interfaces used by the construction rather than to introduce new hardness assumptions.

2.2 One-Way Functions

One-way functions formalize the intuitive notion of a function that is easy to compute but hard to invert.

Let $\mathcal{F} = \{f_\lambda : C_\lambda \to D_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of efficiently computable functions, where C_λ and D_λ are efficiently describable domains. We require that uniform sampling from C_λ is efficient. We further assume that $f_\lambda(w) = \perp$ whenever $w \notin C_\lambda$.

Definition 1 (One-Way Function)**.** We call $\mathcal{F}$ a *one-way function family* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} x \leftarrow_{\$} C_\lambda;\ y \leftarrow f_\lambda(x); \\ x' \leftarrow_{\$} \mathcal{A}(1^\lambda, y) : x' \in C_\lambda \wedge f_\lambda(x') = y \end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.3 Digital Signature Schemes

A digital signature scheme allows a party to sign messages such that the authenticity of the resulting signature can be publicly verified. For the constructions below, its message space is $\{0,1\}^*$, so every injectively encoded certificate or auxiliary-signature message is admissible.

Framework. We use the standard syntax.

Definition 2 (Digital Signature Scheme)**.** A digital signature scheme Σ is a tuple of four PPT algorithms

$$(\mathsf{PP}_\Sigma, \mathsf{KG}_\Sigma, \mathsf{Sgn}_\Sigma, \mathsf{Vfy}_\Sigma)$$

defined as follows.

PP_Σ**.** On input 1^λ, output public parameters pp_Σ:

$$\mathsf{pp}_\Sigma \leftarrow_\$ \mathsf{PP}_\Sigma(1^\lambda).$$

We assume that pp_Σ is implicit input to all other algorithms.

KG_Σ**.** On input pp_Σ, output a signing key sk_Σ and a verification key pk_Σ:

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma).$$

Sgn_Σ**.** On input (sk_Σ, m), output a signature σ:

$$\sigma \leftarrow_\$ \mathsf{Sgn}_\Sigma(\mathsf{sk}_\Sigma, m).$$

Vfy_Σ**.** On input $(\mathsf{pk}_\Sigma, m, \sigma)$, output a decision bit $d \in \{0,1\}$:

$$d \leftarrow \mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, m, \sigma).$$

Definition 3 (Correctness)**.** A digital signature scheme Σ is *correct* if for all $\lambda \in \mathbb{N}$, all $\mathsf{pp}_\Sigma \leftarrow_\$ \mathsf{PP}_\Sigma(1^\lambda)$, all $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma)$, all messages $m \in \mathcal{MS}$, and all signatures $\sigma \leftarrow_\$ \mathsf{Sgn}_\Sigma(\mathsf{sk}_\Sigma, m)$, we have

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, m, \sigma) = 1.$$

2.3.1 Security

We use existential unforgeability under chosen-message attacks.

Unforgeability. Informally, even given adaptive oracle access to the signing algorithm, an adversary should not be able to produce a valid signature on a previously unsigned message.

Definition 4 (Existential Unforgeability under Chosen-Message Attacks)**.** A digital signature scheme Σ is *existentially unforgeable under chosen-message attacks* (EUF-CMA) if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\Sigma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 1.

$\mathbf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\Sigma}(\lambda)$:
$\mathsf{pp}_\Sigma \leftarrow_\$ \mathsf{PP}_\Sigma(1^\lambda)$
$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma)$
$\mathcal{Q} \leftarrow \emptyset$
$(m^*, \sigma^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}^{\mathsf{sk}_\Sigma}_{\mathsf{Sgn}_\Sigma}}(\mathsf{pk}_\Sigma)$
where oracle $\mathcal{O}^{\mathsf{sk}_\Sigma}_{\mathsf{Sgn}_\Sigma}(m)$:
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{m\}$
return $\sigma \leftarrow_\$ \mathsf{Sgn}_\Sigma(\mathsf{sk}_\Sigma, m)$
return 1 if $\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q}$
return 0

Figure 1: EUF-CMA experiment for digital signature scheme Σ.

2.4 Public-Key Encryption

We use public-key encryption with an explicit key-verification algorithm. Besides standard correctness and confidentiality, we require the key-verifiability and decryption-soundness properties defined below.

Framework. We use the following syntax.

Definition 5 (Public-Key Encryption Scheme)**.** A public-key encryption scheme Θ is a tuple of PPT algorithms

$$(\mathsf{PP}_\Theta, \mathsf{KG}_\Theta, \mathsf{Enc}_\Theta, \mathsf{Dec}_\Theta, \mathsf{KVfy}_\Theta)$$

defined as follows.

PP_Θ**.** On input 1^λ, output public parameters pp_Θ:

$$\mathsf{pp}_\Theta \leftarrow_\$ \mathsf{PP}_\Theta(1^\lambda).$$

We assume that pp_Θ is implicitly available to all remaining algorithms.

KG_Θ**.** On input pp_Θ, output a decryption key sk_Θ and an encryption key pk_Θ:

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta).$$

Enc_Θ**.** On input (pk_Θ, m), output a ciphertext c:

$$c \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m).$$

Dec_Θ**.** On input (sk_Θ, c), output a message m or $\perp$:

$$m \leftarrow \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c).$$

KVfy_Θ**.** On input $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$, output a decision bit $d \in \{0, 1\}$:

$$d \leftarrow \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta).$$

Definition 6 (Correctness)**.** A public-key encryption scheme Θ is *correct* if for all $\lambda \in \mathbb{N}$, all $\mathsf{pp}_\Theta \leftarrow_\$ \mathsf{PP}_\Theta(1^\lambda)$, all $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta)$, all messages $m \in \mathcal{MS}$, and all ciphertexts $c \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m)$, we have

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c) = m \qquad \text{and} \qquad \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1.$$

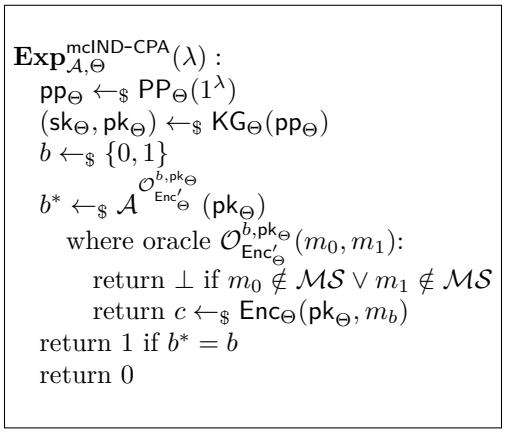


Figure 2: mcIND-CPA experiment for public-key encryption scheme Θ.

2.4.1 Security

For the encryption schemes used here, all messages in $\mathcal{MS}$ have a common publicly determined encoding length for each parameter instance. Thus every admissible left/right query below has equal-length messages. If variable-length message spaces are used instead, the oracle must additionally reject unequal lengths, and applications must pad their challenge plaintexts to a common length. We use multi-challenge indistinguishability under chosen-plaintext attacks. This is polynomially equivalent to the standard single-challenge formulation by a standard hybrid argument [BBM00].

mcIND-CPA security. Informally, an adversary should not be able to distinguish encryptions of adaptively chosen left/right message pairs.

Definition 7 (mcIND-CPA Security)**.** A public-key encryption scheme Θ is *mcIND-CPA-secure* if for every PPT adversary $\mathcal{A}$ whose output lies in $\{0,1\}$ there exists a negligible function negl such that

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{A},\Theta}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 2.

2.4.2 Key-Verifiability and Decryption-Soundness

In our construction, encryption keys and decryption keys may occur inside adversarially generated proofs. For this reason, standard correctness is not sufficient: correctness only covers honestly generated key pairs. We therefore require two additional properties.

Key-verifiability. Key-verifiability requires that the verification algorithm KVfy_Θ meaningfully pins down decryption behavior. Concretely, no efficient adversary should be able to produce two decryption keys that both pass verification for the same public key and yet decrypt some ciphertext differently.

Definition 8 (Key-Verifiability)**.** A PKE Θ is *key-verifiable* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{KeyVerifiability}}_{\mathcal{A},\Theta}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment is given in Figure 3.

$\mathbf{Exp}^{\mathsf{KeyVerifiability}}_{\mathcal{A},\Theta}(\lambda)$:
$\mathsf{pp}_\Theta \leftarrow_\$ \mathsf{PP}_\Theta(1^\lambda)$
$(\mathsf{sk}^*_{\Theta 0}, \mathsf{sk}^*_{\Theta 1}, \mathsf{pk}^*_\Theta, c^*) \leftarrow_\$ \mathcal{A}(\mathsf{pp}_\Theta)$
return 0 if $\mathsf{KVfy}_\Theta(\mathsf{sk}^*_{\Theta 0}, \mathsf{pk}^*_\Theta) \neq 1 \ \vee \ \mathsf{KVfy}_\Theta(\mathsf{sk}^*_{\Theta 1}, \mathsf{pk}^*_\Theta) \neq 1$
return 1 if $\mathsf{Dec}_\Theta(\mathsf{sk}^*_{\Theta 0}, c^*) \neq \mathsf{Dec}_\Theta(\mathsf{sk}^*_{\Theta 1}, c^*)$
return 0

Figure 3: Key-verifiability experiment for public-key encryption scheme Θ.

Decryption-soundness. Decryption-soundness strengthens correctness to adversarially generated key pairs that pass KVfy_Θ. Intuitively, once $(\mathsf{pk}_\Theta, \mathsf{sk}_\Theta)$ is accepted as a valid key pair, ciphertexts honestly generated under pk_Θ must decrypt consistently.

These properties are incomparable: key-verifiability controls ambiguity among accepted decryption keys for the same public key, whereas decryption-soundness rules out malformed but accepted key pairs that decrypt honestly generated ciphertexts incorrectly. This formalizes the distinction underlying the ideas of Derler et al. [DSS20].

$\mathbf{Exp}^{\mathsf{DecSound}}_{\mathcal{A},\Theta}(\lambda)$:
$\mathsf{pp}_\Theta \leftarrow_\$ \mathsf{PP}_\Theta(1^\lambda)$
$(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta, m^*, r^*) \leftarrow_\$ \mathcal{A}(\mathsf{pp}_\Theta)$
return 0 if $\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) \neq 1$
$c \leftarrow \mathsf{Enc}_\Theta(\mathsf{pk}^*_\Theta, m^*; r^*)$
return 0 if $c = \perp$
return 1 if $\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c) \neq m^*$
return 0

Figure 4: Decryption-soundness experiment for public-key encryption scheme Θ.

Definition 9 (Decryption-Soundness)**.** A PKE Θ is *decryption-sound* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{DecSound}}_{\mathcal{A},\Theta}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment is given in Figure 4.

Together with correctness, decryption-soundness extends the correctness guarantee from honestly generated key pairs to adversarially generated key pairs that pass KVfy_Θ.

By correctness, every honestly generated key pair is accepted by KVfy_Θ.

Schemes intentionally permitting multiple secret keys or multiple valid decryption behaviors arise in non-committing encryption, functional encryption, and related settings [BSW11, CLNS17, HRW16, Nie02].

We next instantiate these requirements with ElGamal and then record the corresponding properties.

2.4.3 ElGamal Instantiation

Decisional Diffie–Hellman Assumption

Definition 10 (Decisional Diffie–Hellman)**.** Let

$$(\mathbb{G}, q, g) \leftarrow_{\$} \mathsf{GGen}(1^\lambda)$$

and let $x, y, z \leftarrow_{\$} \mathbb{Z}_q$. We say that the DDH assumption holds for GGen if, for every PPT distinguisher $\mathcal{A}$,

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1]|$$

is negligible in λ.

Instantiation. We use standard ElGamal encryption over prime-order groups. Messages are assumed to be elements of $\mathbb{G}$.

$\underline{\mathsf{PP}_\Theta(1^\lambda)}$: Run

$$(\mathbb{G}, q, g) \leftarrow_{\$} \mathsf{GGen}(1^\lambda)$$

where $\mathbb{G}$ is a cyclic group of prime order q generated by g. Output

$$\mathsf{pp}_\Theta \leftarrow (\mathbb{G}, q, g).$$

$\underline{\mathsf{KG}_\Theta(\mathsf{pp}_\Theta)}$: Sample $x \leftarrow_{\$} \mathbb{Z}_q$. Set

$$\mathsf{sk}_\Theta \leftarrow x \qquad \text{and} \qquad \mathsf{pk}_\Theta \leftarrow g^x \in \mathbb{G}.$$

Return $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$.

$\underline{\mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m)}$: Sample $r \leftarrow_{\$} \mathbb{Z}_q$ and output

$$c \leftarrow (c_1, c_2) \leftarrow (g^r,\, m \cdot \mathsf{pk}_\Theta^r) \in \mathbb{G}^2.$$

$\underline{\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c)}$: Parse $c = (c_1, c_2)$ and output

$$m \leftarrow c_2 / c_1^{\mathsf{sk}_\Theta}.$$

$\underline{\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)}$: Return 1 if $\mathsf{pk}_\Theta = g^{\mathsf{sk}_\Theta}$, and 0 otherwise.

Construction 1: ElGamal Encryption over Prime-Order Groups

Theorem 1. *Let Θ be as in Construction 1. Then Θ is perfectly correct.*

Proof. Let $x, r \in \mathbb{Z}_q$, let $\mathsf{pk}_\Theta = g^x$, and let

$$c = (g^r, m\mathsf{pk}_\Theta^r) = (g^r, mg^{xr}).$$

Then

$$\mathsf{Dec}_\Theta(x, c) = \frac{mg^{xr}}{(g^r)^x} = m.$$

Moreover,

$$\mathsf{KVfy}_\Theta(x, g^x) = 1.$$

Hence ElGamal is perfectly correct. □

Theorem 2. *Let Θ be as in Construction 1. Then Θ is mcIND-CPA-secure under the DDH assumption.*

Proof. Standard ElGamal is IND-CPA secure under DDH. Indeed, for public key $\mathsf{pk}_\Theta = g^x$, an encryption of m_b has the form

$$(g^r, m_b g^{xr}).$$

By DDH,

$$(g^x, g^r, g^{xr})$$

is computationally indistinguishable from

$$(g^x, g^r, g^z)$$

for uniform $z \leftarrow_\$ \mathbb{Z}_q$. In the latter distribution, g^z is uniform in $\mathbb{G}$, and therefore $m_b g^z$ is uniform and independent of b. Thus ElGamal is IND-CPA secure under DDH.

Multi-challenge IND-CPA follows from single-challenge IND-CPA by the standard hybrid argument stated above. □

For the following statements, let Θ be the standard ElGamal encryption scheme above. These statements establish the additional encryption properties needed by our construction. The group-signature constructions below additionally require $D_\lambda \subseteq \mathcal{M}_{\Theta,\lambda}$, so outputs of the chosen one-way function are valid encryption plaintexts.

Theorem 3. Θ *is key-verifiable.*

Proof idea. In a prime-order group, the public key uniquely fixes the corresponding exponent modulo the group order, and thus uniquely determines decryption behavior.

Proof. Consider ElGamal over a cyclic group $\mathbb{G}$ of prime order q with generator g. Public keys are of the form

$$\mathsf{pk}_\Theta = g^{\mathsf{sk}_\Theta}$$

for $\mathsf{sk}_\Theta \in \mathbb{Z}_q$, and the key-verification algorithm accepts exactly the well-typed pairs $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$ satisfying this equality.

Suppose that an adversary outputs $(\mathsf{sk}_{\Theta 0}^*, \mathsf{sk}_{\Theta 1}^*, \mathsf{pk}_\Theta^*, c^*)$ in the key-verifiability experiment. If either secret key fails verification, the experiment returns 0. It therefore remains to consider the case

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta 0}^*, \mathsf{pk}_\Theta^*) = \mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta 1}^*, \mathsf{pk}_\Theta^*) = 1.$$

By the implicit domain checks, $\mathsf{sk}_{\Theta 0}^*, \mathsf{sk}_{\Theta 1}^* \in \mathbb{Z}_q$, and acceptance implies

$$g^{\mathsf{sk}_{\Theta 0}^*} = \mathsf{pk}_\Theta^* = g^{\mathsf{sk}_{\Theta 1}^*}.$$

Since g generates a group of order q, the map

$$x \longmapsto g^x$$

is injective on $\mathbb{Z}_q$. Hence $\mathsf{sk}_{\Theta 0}^* = \mathsf{sk}_{\Theta 1}^*$.

Because Dec_Θ is deterministic and total, it follows for every ciphertext c^* that

$$\mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) = \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*).$$

Thus the winning condition in Figure 3 never occurs, and the experiment returns 1 with probability 0. □

Theorem 4. Θ *is decryption-sound.*

Proof idea. Acceptance by KVfy_Θ forces $\mathsf{pk}_\Theta = g^{\mathsf{sk}_\Theta}$. Hence decrypting an honestly generated ElGamal ciphertext with any accepted secret key is exactly ElGamal correctness.

Proof. Consider an arbitrary output $(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta, m^*, r^*)$ of the adversary in the decryption-soundness experiment. If $\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) \neq 1$, the experiment returns 0. We may therefore assume

$$\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) = 1,$$

which, by definition of key verification, implies

$$\mathsf{pk}^*_\Theta = g^{\mathsf{sk}^*_\Theta}.$$

The experiment computes

$$c \leftarrow \mathsf{Enc}_\Theta(\mathsf{pk}^*_\Theta, m^*; r^*).$$

If this invocation returns $\perp$, the experiment again returns 0. Otherwise, the implicit domain checks ensure that $m^* \in \mathbb{G}$ and $r^* \in \mathbb{Z}_q$, and ElGamal encryption yields

$$c = (c_1, c_2) = \left(g^{r^*}, m^*(\mathsf{pk}^*_\Theta)^{r^*}\right).$$

Consequently,

$$\begin{aligned}\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c) &= \frac{c_2}{c_1^{\mathsf{sk}^*_\Theta}} \\ &= \frac{m^*(g^{\mathsf{sk}^*_\Theta})^{r^*}}{(g^{r^*})^{\mathsf{sk}^*_\Theta}} \\ &= m^*.\end{aligned}$$

Hence the winning condition in Figure 4 never occurs, and the experiment returns 1 with probability 0. □

2.4.4 Separating Key-Verifiability and Decryption-Soundness

Theorem 5 (Key-Verifiability $\not\Rightarrow$ Decryption-Soundness)**.** *There exists a correct and key-verifiable PKE that is not decryption-sound.*

Proof idea. One can augment the public key with auxiliary data that leaves key-verifiability intact while modifying decryption on honestly generated ciphertexts for accepted but malformed key pairs.

Proof. Let

$$\Theta_0 = (\mathsf{PP}_\Theta, \mathsf{KG}_\Theta, \mathsf{Enc}_\Theta, \mathsf{Dec}_\Theta, \mathsf{KVfy}_\Theta)$$

be any correct and key-verifiable PKE. We construct a correct and key-verifiable PKE

$$\Theta' = (\mathsf{PP}'_\Theta, \mathsf{KG}'_\Theta, \mathsf{Enc}'_\Theta, \mathsf{Dec}'_\Theta, \mathsf{KVfy}'_\Theta)$$

that is not decryption-sound.

Construction. The public parameters and message space of Θ' are those of Θ_0. The derived scheme attaches a bit to both the public key and the secret key. Honest key generation always uses the bit 0.

PP'_Θ. Run PP_Θ without modification.

KG'_Θ. On input pp_Θ, sample

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta)$$

and output

$$\mathsf{sk}'_\Theta \leftarrow (\mathsf{sk}_\Theta, 0) \qquad \text{and} \qquad \mathsf{pk}'_\Theta \leftarrow (\mathsf{pk}_\Theta, 0).$$

KVfy'_Θ**.** On input $((\mathsf{sk}_\Theta, b), (\mathsf{pk}_\Theta, b'))$, return 1 if and only if

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1 \qquad \text{and} \qquad b = b'.$$

Enc'_Θ**.** On input $((\mathsf{pk}_\Theta, b), m)$, ignore b and run $\mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m)$. With explicit randomness r, define

$$\mathsf{Enc}'_\Theta((\mathsf{pk}_\Theta, b), m; r) \leftarrow \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m; r).$$

Dec'_Θ**.** On input $((\mathsf{sk}_\Theta, b), c)$, return

$$\mathsf{Dec}'_\Theta((\mathsf{sk}_\Theta, b), c) \leftarrow \begin{cases} \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c), & b = 0, \\ \bot, & b = 1. \end{cases}$$

Correctness. Honest key generation always sets both key bits to 0. Key verification then accepts by correctness of Θ_0, encryption agrees with Enc_Θ, and decryption agrees with Dec_Θ. Thus Θ' inherits correctness from Θ_0.

Key-verifiability. Suppose an adversary wins the key-verifiability experiment against Θ'. Let

$$((\mathsf{sk}_{\Theta 0}, b_0), (\mathsf{sk}_{\Theta 1}, b_1), (\mathsf{pk}_\Theta, b'), c^*)$$

denote its output. Since both secret keys pass KVfy'_Θ for the same public key, we have

$$b_0 = b_1 = b'$$

and

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta 0}, \mathsf{pk}_\Theta) = \mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta 1}, \mathsf{pk}_\Theta) = 1.$$

If $b' = 1$, both derived decryption algorithms return $\bot$, so their outputs cannot differ. Therefore, every successful adversary must have $b' = 0$. In that case,

$$\mathsf{Dec}'_\Theta((\mathsf{sk}_{\Theta i}, 0), c^*) = \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta i}, c^*) \qquad \text{for } i \in \{0, 1\}.$$

Any difference between the two derived decryption outputs therefore yields a difference between the corresponding outputs in the key-verifiability experiment for Θ_0. Forwarding $(\mathsf{sk}_{\Theta 0}, \mathsf{sk}_{\Theta 1}, \mathsf{pk}_\Theta, c^*)$ gives an adversary against key-verifiability of Θ_0 with the same success probability. This probability is negligible, and hence Θ' is key-verifiable.

Failure of decryption-soundness. Consider the following adversary. It generates an honest base key pair

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta),$$

chooses any message m^* in the message space and valid encryption randomness r^*, and outputs

$$(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta, m^*, r^*) \leftarrow ((\mathsf{sk}_\Theta, 1), (\mathsf{pk}_\Theta, 1), m^*, r^*).$$

By correctness of the base key-verification algorithm,

$$\mathsf{KVfy}'_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) = 1.$$

Moreover,

$$c \leftarrow \mathsf{Enc}'_\Theta(\mathsf{pk}^*_\Theta, m^*; r^*)$$

is an honestly formed ciphertext of Θ_0, while

$$\mathsf{Dec}'_\Theta(\mathsf{sk}^*_\Theta, c) = \bot \neq m^*.$$

Thus the adversary wins the decryption-soundness experiment with probability 1, and Θ' is not decryption-sound. □

Theorem 6 (Decryption-Soundness $\not\Rightarrow$ Key-Verifiability)**.** *There exists a correct PKE Θ'' that is decryption-sound but not key-verifiable.*

Proof idea. One can encode auxiliary information into the secret key and ciphertext so that honest executions remain correct, while distinct accepted decryption keys still induce different outputs on some ciphertext.

Proof. Let

$$\Theta_0 = (\mathsf{PP}_\Theta, \mathsf{KG}_\Theta, \mathsf{Enc}_\Theta, \mathsf{Dec}_\Theta, \mathsf{KVfy}_\Theta)$$

be any correct and decryption-sound PKE. We construct a correct and decryption-sound PKE

$$\Theta'' = (\mathsf{PP}''_\Theta, \mathsf{KG}''_\Theta, \mathsf{Enc}''_\Theta, \mathsf{Dec}''_\Theta, \mathsf{KVfy}''_\Theta)$$

that is not key-verifiable.

Construction. The public parameters and message space of Θ'' are those of Θ_0. Secret keys are augmented by a bit b, and ciphertexts are augmented by a bit f. Honest encryption always sets $f = 0$, whereas decryption may behave differently on well-formed ciphertexts with $f = 1$.

PP''_Θ**.** Run PP_Θ without modification.

KG''_Θ**.** On input pp_Θ, sample

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta) \qquad \text{and} \qquad b \leftarrow_\$ \{0,1\},$$

and output

$$\mathsf{sk}''_\Theta \leftarrow (\mathsf{sk}_\Theta, b) \qquad \text{and} \qquad \mathsf{pk}''_\Theta \leftarrow \mathsf{pk}_\Theta.$$

KVfy''_Θ**.** On input $((\mathsf{sk}_\Theta, b), \mathsf{pk}_\Theta)$, ignore b and return

$$\mathsf{KVfy}''_\Theta((\mathsf{sk}_\Theta, b), \mathsf{pk}_\Theta) \leftarrow \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta).$$

Enc''_Θ**.** On input (pk_Θ, m), compute

$$c_0 \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m).$$

If $c_0 = \perp$, return $\perp$; otherwise return

$$c \leftarrow (0, c_0).$$

With explicit randomness r, the same procedure uses $c_0 \leftarrow \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m; r)$.

Dec''_Θ**.** On input $((\mathsf{sk}_\Theta, b), (f, c_0))$, return

$$\mathsf{Dec}''_\Theta((\mathsf{sk}_\Theta, b), (f, c_0)) \leftarrow \begin{cases} \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c_0), & f = 0, \\ \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c_0), & f = 1 \text{ and } b = 0, \\ \perp, & f = 1 \text{ and } b = 1. \end{cases}$$

Correctness. Every honestly generated ciphertext has the form $(0, c_0)$. Consequently, decryption under either value of the honestly generated key bit b agrees with $\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c_0)$. Key verification also agrees with the base scheme. Correctness of Θ'' therefore follows directly from correctness of Θ_0.

Decryption-soundness. Let $\mathcal{A}$ be an adversary against decryption-soundness of Θ''. We build an adversary $\mathcal{B}$ against decryption-soundness of Θ_0. On input pp_Θ, adversary $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}_\Theta)$ to obtain

$$((\mathsf{sk}^*_\Theta, b^*), \mathsf{pk}^*_\Theta, m^*, r^*)$$

and outputs

$$(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta, m^*, r^*)$$

to its own experiment.

The derived verification condition

$$\mathsf{KVfy}''_\Theta((\mathsf{sk}^*_\Theta, b^*), \mathsf{pk}^*_\Theta) = 1$$

is identical to

$$\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) = 1.$$

Moreover, if base encryption returns

$$c_0 \leftarrow \mathsf{Enc}_\Theta(\mathsf{pk}^*_\Theta, m^*; r^*),$$

then derived encryption returns $(0, c_0)$, and

$$\mathsf{Dec}''_\Theta((\mathsf{sk}^*_\Theta, b^*), (0, c_0)) = \mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c_0).$$

Thus $\mathcal{A}$ wins against decryption-soundness of Θ'' exactly when $\mathcal{B}$ wins against decryption-soundness of Θ_0. Since Θ_0 is decryption-sound, this probability is negligible. Hence Θ'' is decryption-sound.

Failure of key-verifiability. Consider the following adversary in the key-verifiability experiment. It generates an honest base key pair

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta),$$

chooses any message m in the message space, and computes

$$c_0 \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, m).$$

It then outputs

$$\mathsf{sk}_{\Theta 0}^* \leftarrow (\mathsf{sk}_\Theta, 0), \qquad \mathsf{sk}_{\Theta 1}^* \leftarrow (\mathsf{sk}_\Theta, 1), \qquad \mathsf{pk}^*_\Theta \leftarrow \mathsf{pk}_\Theta, \qquad c^* \leftarrow (1, c_0).$$

By correctness of base key verification, both derived secret keys satisfy

$$\mathsf{KVfy}''_\Theta(\mathsf{sk}_{\Theta i}^*, \mathsf{pk}^*_\Theta) = 1 \qquad \text{for } i \in \{0, 1\}.$$

However,

$$\mathsf{Dec}''_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) = \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c_0) = m,$$

whereas

$$\mathsf{Dec}''_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*) = \perp.$$

The two outputs differ, so the adversary wins with probability 1. Therefore Θ'' is not key-verifiable.

The separating ciphertext $c^* = (1, c_0)$ is well formed for Θ'' but lies outside the range of its honest encryption algorithm. This is precisely the gap exploited by the separation: key-verifiability quantifies over arbitrary well-formed ciphertexts, whereas decryption-soundness considers only ciphertexts produced by the prescribed encryption algorithm. □

2.5 Non-Interactive Proof Systems

Let $L \subseteq \{0,1\}^*$ be an NP-language with associated witness relation R. A non-interactive proof system allows proving membership $x \in L$ without interaction.

Parameterized languages. In many applications, the underlying relation depends on externally chosen public parameters. We model this as a family of languages $\{L_f\}_{f \in \mathcal{F}}$, where

$$L_f \;:=\; \{\, x \mid \exists w : R_f(x,w) = 1 \,\}.$$

Whenever useful, we view the statement as already containing the relevant public parameters. For readability, we suppress this dependence when it is clear from context.

For simplicity, we write $R(x,w)$ and assume that any parameters are encoded in x. In particular, the displayed statements for $L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta}$ and $L_2^{\mathsf{pp}_\Theta}$ in the constructions and reductions suppress, respectively, the fixed prefixes $(f, \mathsf{pp}_\Sigma, \mathsf{pp}_\Theta)$ and pp_Θ. These prefixes are part of the actual statement passed to every prover, simulator, verifier, and extractor. We use the corresponding universal NP relations, so their CRS setup does not require fixing these externally generated parameters first. All equality and freshness tests on statements refer to these full encodings. In particular, the full statement contains all public data relevant to the relation, such as public keys, ciphertexts, and the signed message.

Framework. We use the following syntax.

Definition 11 (Non-Interactive Proof System)**.** A non-interactive proof system Π for L consists of PPT algorithms

$$(\mathsf{PG}_\Pi, \mathsf{Prf}_\Pi, \mathsf{Vfy}_\Pi)$$

defined as follows.

PG_Π**.** On input 1^λ, output a common reference string crs_Π:

$$\mathsf{crs}_\Pi \leftarrow_\$ \mathsf{PG}_\Pi(1^\lambda).$$

Prf_Π**.** On input (crs_Π, x, w), output a proof π:

$$\pi \leftarrow_\$ \mathsf{Prf}_\Pi(\mathsf{crs}_\Pi, x, w).$$

We interpret the prover as a total algorithm and require

$$\mathsf{Prf}_\Pi(\mathsf{crs}_\Pi, x, w) = \bot$$

whenever $R(x,w) \neq 1$.

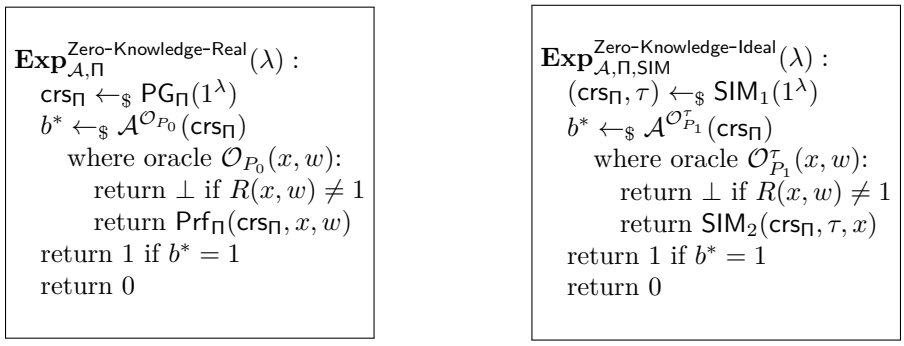
Vfy_Π**.** On input $(\mathsf{crs}_\Pi, x, \pi)$, output a decision bit $d \in \{0,1\}$:

$$d \leftarrow \mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi, x, \pi).$$

Definition 12 (Correctness)**.** A non-interactive proof system Π for L is *correct* if for all $\lambda \in \mathbb{N}$, all $\mathsf{crs}_\Pi \leftarrow_\$ \mathsf{PG}_\Pi(1^\lambda)$, all statements $x \in L$, all witnesses w with $R(x,w) = 1$, and all proofs $\pi \leftarrow_\$ \mathsf{Prf}_\Pi(\mathsf{crs}_\Pi, x, w)$, we have

$$\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi, x, \pi) = 1.$$

We additionally require zero-knowledge and simulation-sound extractability.



(a) Real world (b) Ideal world

Figure 5: Real and ideal zero-knowledge experiments for NIZK Π.

2.5.1 Security

Besides the real setup algorithm PG_Π, the following notions use a simulated setup SIM_1 and an extractable setup $\mathcal{E}_1$. The CRS distributions induced by SIM_1 and $\mathcal{E}_1$ are identical. The real and simulated experiments are computationally indistinguishable as required by zero-knowledge.

Zero-knowledge. Informally, simulated proofs should be indistinguishable from real proofs.

Definition 13 (Zero-Knowledge)**.** A non-interactive proof system Π for L is *zero-knowledge* if there exists a PPT simulator $\mathsf{SIM} = (\mathsf{SIM}_1, \mathsf{SIM}_2)$ such that for every PPT adversary $\mathcal{A}$ there exists a negligible function negl with

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{Zero\text{-}Knowledge\text{-}Real}}_{\mathcal{A},\Pi}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{Zero\text{-}Knowledge\text{-}Ideal}}_{\mathcal{A},\Pi,\mathsf{SIM}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

The corresponding experiments are given in Figures 5a and 5b.

We do not require composable zero-knowledge in the sense of Groth [Gro07].

Simulation-sound extractability. Even after seeing simulated proofs for adaptively chosen statements, any accepting proof not previously returned by the simulator should admit extraction of a valid witness.

Definition 14 (Simulation-Sound Extractability)**.** A non-interactive proof system Π for L is *simulation-sound extractable* if there exists a PPT extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ and a simulator $\mathsf{SIM} = (\mathsf{SIM}_1, \mathsf{SIM}_2)$ as in Definition 13 such that:

(i) the joint distribution of (crs_Π, τ) generated by $\mathsf{SIM}_1(1^\lambda)$ is identical to the marginal distribution of (crs_Π, τ) generated as part of

$$(\mathsf{crs}_\Pi, \tau, \zeta) \leftarrow_\$ \mathcal{E}_1(1^\lambda);$$

(ii) for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{SimSoundExt}}_{\mathcal{A},\Pi,\mathcal{E}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment is given in Figure 6.

$\mathbf{Exp}^{\mathsf{SimSoundExt}}_{\mathcal{A},\Pi,\mathcal{E}}(\lambda)$:
$(\mathsf{crs}_\Pi, \tau, \zeta) \leftarrow_\$ \mathcal{E}_1(1^\lambda)$
$\mathcal{Q} \leftarrow \emptyset$
$(x^*, \pi^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}^\tau_{\mathsf{SIM}}}(\mathsf{crs}_\Pi)$
where oracle $\mathcal{O}^\tau_{\mathsf{SIM}}(x)$:
$\pi \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}_\Pi, \tau, x)$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x, \pi)\}$
return π
$w^* \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi, \zeta, x^*, \pi^*)$
return 1 if $\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi, x^*, \pi^*) = 1 \ \wedge \ R(x^*, w^*) = 0 \ \wedge \ (x^*, \pi^*) \notin \mathcal{Q}$
return 0

Figure 6: Simulation-sound extractability experiment for NIZK Π.

2.5.2 Statement-Binding

Statement-binding ties a proof to its complete statement: a proof π should not verify for two distinct statements even when the adversary knows the simulation and extraction trapdoors. A standard way to enforce this is to append the statement to the proof string [FKMV12]. This property is used later in our strengthened trace-soundness notion; see Section 5.2.4.

Fix an extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ witnessing simulation-sound extractability of Π.

$\mathbf{Exp}^{\mathsf{StatementBinding}}_{\mathcal{A},\Pi,\mathcal{E}}(\lambda)$:
$(\mathsf{crs}_\Pi, \tau, \zeta) \leftarrow_\$ \mathcal{E}_1(1^\lambda)$
$(x_0^*, x_1^*, \pi^*) \leftarrow_\$ \mathcal{A}(\mathsf{crs}_\Pi, \tau, \zeta)$
return 0 if $x_0^* = x_1^*$
return 1 if $\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi, x_0^*, \pi^*) = 1 \wedge \mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi, x_1^*, \pi^*) = 1$
return 0

Figure 7: Statement-binding experiment for NIZK Π.

Definition 15 (Statement-Binding)**.** We say that Π is *statement-binding relative to* $\mathcal{E}$ if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{StatementBinding}}_{\mathcal{A},\Pi,\mathcal{E}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Since the adversary is given both the simulation and extraction auxiliary information, statement-binding must hold even under exposure of the setup trapdoors. The role of $\mathcal{E}_1$ is only to generate the CRS and this auxiliary setup information; statement-binding adds no extraction requirement beyond SSE and only rules out cross-statement proof reuse.

2.5.3 Construction of Statement-Binding Πs

We now show that statement-binding can be enforced generically.

Let Π be a zero-knowledge and simulation-sound extractable proof system. We define a derived proof system Π' by appending the statement to the proof and checking consistency during verification; see Construction 2. This is the form used later in our construction.

PG'_Π. On input 1^λ, run

$$\mathsf{crs}_\Pi \leftarrow_\$ \mathsf{PG}_\Pi(1^\lambda)$$

and output crs_Π.

Prf'_Π. On input (crs_Π, x, w), compute

$$\pi' \leftarrow_\$ \mathsf{Prf}_\Pi(\mathsf{crs}_\Pi, x, w)$$

if $\pi' = \bot$, return $\bot$, else return (π', x).

Vfy'_Π. On input $(\mathsf{crs}_\Pi, x, \pi)$, parse $\pi = (\pi', x')$. If $x \neq x'$, return 0. Otherwise, return $\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi, x, \pi')$.

Construction 2: Statement-Binding Π'

We record the resulting guarantees for later use.

Theorem 7 (Zero-Knowledge of Π')**.** *If Π is zero-knowledge, then Π' is zero-knowledge.*

Proof. Let $\mathsf{SIM} = (\mathsf{SIM}_1, \mathsf{SIM}_2)$ be the simulator for Π. Define $\mathsf{SIM}'_1 \leftarrow \mathsf{SIM}_1$. On input $(\mathsf{crs}_\Pi, \tau, x)$, SIM'_2 computes

$$\pi' \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}_\Pi, \tau, x).$$

If $\pi' = \bot$, it returns $\bot$; otherwise, it returns

$$(\pi', x).$$

Since the transformed proof simply appends the public statement x in both the real and simulated worlds, any distinguisher for Π' immediately yields a distinguisher for Π. Hence, the advantage remains negligible. □

Theorem 8 (Simulation-Sound Extractability of Π')**.** *If Π is simulation-sound extractable, then Π' is simulation-sound extractable.*

Proof. Let

$$\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2) \qquad \text{and} \qquad \mathsf{SIM} = (\mathsf{SIM}_1, \mathsf{SIM}_2)$$

witness simulation-sound extractability of Π.

Define the simulator $\mathsf{SIM}' = (\mathsf{SIM}'_1, \mathsf{SIM}'_2)$ by

$$\mathsf{SIM}'_1 \leftarrow \mathsf{SIM}_1.$$

On input $(\mathsf{crs}_\Pi, \tau, x)$, SIM'_2 computes

$$\pi' \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}_\Pi, \tau, x).$$

If $\pi' = \bot$, it returns $\bot$; otherwise, it returns (π', x).

Define the extractor

$$\mathcal{E}'_1 \leftarrow \mathcal{E}_1.$$

On input $(\mathsf{crs}_\Pi, \zeta, x, (\pi', x'))$, $\mathcal{E}'_2$ returns $\bot$ if $x \neq x'$, and otherwise returns

$$\mathcal{E}_2(\mathsf{crs}_\Pi, \zeta, x, \pi').$$

The required setup-distribution condition follows immediately from that of $\mathcal{E}$ and SIM.

Now consider an accepting proof (π', x') for statement x that is fresh in the SSE experiment for Π'. Verification implies $x = x'$. Moreover, every simulated proof returned by SIM_2' on statement x is exactly of the form (π, x), where π is the proof returned by SIM_2 on the same statement. Hence

$$(x, (\pi', x)) \notin \mathcal{Q}' \quad \Longrightarrow \quad (x, \pi') \notin \mathcal{Q},$$

where $\mathcal{Q}'$ and $\mathcal{Q}$ denote the simulator-query logs of the derived and base experiments, respectively.

Thus, any accepting fresh proof for Π' from which $\mathcal{E}_2'$ fails to extract a valid witness yields an accepting fresh proof for Π from which $\mathcal{E}_2$ fails to extract a valid witness. This contradicts SSE of Π except with negligible probability. □

Theorem 9 (Statement-Binding of Π')**.** Π' *satisfies statement-binding.*

Proof. Suppose a proof $\pi = (\pi', x')$ verifies for two statements x_0 and x_1. By definition of Vfy_Π', acceptance for x_0 implies $x_0 = x'$, and acceptance for x_1 implies $x_1 = x'$. Hence $x_0 = x_1$, contradicting the requirement $x_0 \neq x_1$ in Figure 7. □

The transformation increases the proof size by $|x|$. We use it as a generic, assumption-free method for obtaining statement-binding and do not claim compactness of the resulting proof representation.

3 BSZ-Style Dynamic Group Signatures

We use a registry-based baseline inspired by the dynamic group signature framework of Bellare, Shi, and Zhang (BSZ) [BSZ05] and its later refinements, including Sakai et al. [SSE+12] and Bootle et al. [BCC+16, BCC+20]. The purpose of this baseline is to isolate the effect of trusted registration state. We therefore retain the separated and non-interactive interfaces used in our later model, but equip the issuer with state updated through successful join executions. Opening and opening verification are evaluated relative to this registry.

Remark 1 (Scope of the Abstraction)*.* Our syntax is not a verbatim restatement of the original BSZ framework. In particular, we abstract from the interactive join protocol and model only completed join executions as state transitions. We also make member, issuer, and opener key generation explicit and separate. This presentation preserves the registry-based relation relevant to our analysis while allowing a direct comparison with our model introduced later.

3.1 Framework

A BSZ-style group signature scheme consists of algorithms for parameter generation, member key generation, issuer key generation, opener key generation, joining, signing, verification, opening, and opening verification. Throughout this section, member public keys serve as member identities.

A registry REG is scheme-dependent state maintained by the issuer and updated through join executions. We intentionally impose no public format or membership predicate on this state: depending on the scheme, it may contain full join transcripts, compressed auxiliary information, or no semantically relevant data at all. Security experiments that must determine whether a public key was submitted to a successful join record this fact in a separate challenger-maintained query log. The initial registry state is denoted by $\emptyset$.

Definition 16 (Group Signature Scheme)**.** A group signature scheme Γ is a tuple of PPT algorithms

$$(\mathsf{PG}_\Gamma, \mathsf{KgM}_\Gamma, \mathsf{KgI}_\Gamma, \mathsf{KgO}_\Gamma, \mathsf{Join}_\Gamma, \mathsf{Sgn}_\Gamma, \mathsf{Vfy}_\Gamma, \mathsf{Opn}_\Gamma, \mathsf{Jdg}_\Gamma),$$

defined as follows.

PG_Γ**.** On input 1^λ, output public parameters pp_Γ:

$$\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda).$$

We display pp_Γ explicitly as input to the key-generation algorithms and treat it as implicit input to all remaining algorithms.

KgM_Γ**.** On input pp_Γ, output a member secret/public key pair $(\mathsf{usk}, \mathsf{upk})$:

$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma).$$

KgI_Γ**.** On input pp_Γ, output issuer keys $(\mathsf{isk}, \mathsf{ipk})$:

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma).$$

KgO_Γ**.** On input pp_Γ, output opener keys $(\mathsf{osk}, \mathsf{opk})$:

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma).$$

Join_Γ**.** On input $(\mathsf{isk}, \mathsf{ipk}, \mathsf{REG}, \mathsf{upk}, \mathsf{opk})$, output either a membership certificate σ_s and an updated registry REG', or $\perp$:

$$(\sigma_s, \mathsf{REG}') \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{REG}, \mathsf{upk}, \mathsf{opk}).$$

Sgn_Γ**.** On input $(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$ with $m \in \mathcal{MS}$, output either a group signature σ_m or $\perp$:

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m).$$

Vfy_Γ**.** On input $(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)$, output a decision bit $d \in \{0, 1\}$:

$$d \leftarrow \mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m).$$

Opn_Γ**.** On input $(\mathsf{osk}, \mathsf{opk}, \mathsf{REG}, \mathsf{ipk}, m, \sigma_m)$, output either a member public key upk and opening information O, or $\perp$:

$$(\mathsf{upk}, O) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{REG}, \mathsf{ipk}, m, \sigma_m).$$

Jdg_Γ**.** On input $(\mathsf{ipk}, \mathsf{opk}, \mathsf{REG}, m, \sigma_m, \mathsf{upk}, O)$, output a decision bit $d \in \{0, 1\}$:

$$d \leftarrow \mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \mathsf{REG}, m, \sigma_m, \mathsf{upk}, O).$$

The registry is globally readable. In an honest execution, it is modified only by Join_Γ. Security experiments involving a corrupted issuer explicitly specify the adversary's access to or control over the registry.

Definition 17 (Correctness)**.** A group signature scheme Γ is *perfectly correct* if the following holds for all $\lambda \in \mathbb{N}$, all $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$, all issuer keys $(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$, all opener keys $(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$, all $n \in \mathbb{N}$, and all sequences

$$\{(\mathsf{usk}_i, \mathsf{upk}_i)\}_{i=1}^n \quad \text{with} \quad (\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma) \quad \text{for all } i.$$

Let $\mathsf{REG}_0 \leftarrow \emptyset$, and for each $i \in \{1, \ldots, n\}$ let

$$(\sigma_s^i, \mathsf{REG}_i) \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{REG}_{i-1}, \mathsf{upk}_i, \mathsf{opk}).$$

Every such join invocation must succeed, i.e., $(\sigma_s^i, \mathsf{REG}_i) \neq \bot$.

Then, for every $i \in \{1, \ldots, n\}$, every message $m \in \mathcal{MS}$, and every

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}_i, \mathsf{upk}_i, \sigma_s^i, \mathsf{ipk}, \mathsf{opk}, m),$$

we require $\sigma_m \neq \bot$ and

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m) = 1.$$

Moreover, for every

$$(\mathsf{upk}', O) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{REG}_n, \mathsf{ipk}, m, \sigma_m),$$

we require $(\mathsf{upk}', O) \neq \bot$ and

$$\mathsf{upk}' = \mathsf{upk}_i \qquad \text{and} \qquad \mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \mathsf{REG}_n, m, \sigma_m, \mathsf{upk}', O) = 1.$$

Correctness is evaluated relative to the final registry state obtained after the entire sequence of joins. It therefore requires subsequent join executions neither to invalidate an earlier member's signatures nor to change the identity to which they open. It does not require the registry itself to expose a public membership relation.

3.2 Security

We next formulate the five BSZ-style properties that serve as our registry-based baseline: *anonymity*, *traceability*, *non-frameability*, *trace-soundness*, and *non-claimability*. The experiments make three distinctions explicit: whether a key pair is generated honestly, whether an honestly generated secret key is subsequently leaked, and whether the corresponding role is fully controlled by the adversary. These distinctions will become important when we later remove the registry.

At a high level, the properties provide the following guarantees.

- *Anonymity* hides which of two honestly generated member key pairs was used to produce a signature. The issuer may be malicious and both challenge member secret keys are leaked, whereas the opener is honest.
- *Traceability* requires every accepted signature under honestly generated issuer and opener keys to admit a valid opening, even when the opener secret key is leaked to the adversary.
- *Non-frameability* prevents an adversary controlling both the issuer and the opener from producing an accepted opening that falsely attributes a signature to an honestly generated member key pair.
- *Trace-soundness* requires that an accepted signature can be attributed at most once.
- *Non-claimability* prevents an honestly generated signature from subsequently being re-attributed to a different member public key.

These informal descriptions only summarize the intended guarantees. The experiments below determine precisely which keys, registries, and auxiliary values are controlled by the adversary. In particular, an honestly generated member key pair is not by itself evidence of membership: membership is created only by a successful join execution and its corresponding registry entry.

3.2.1 Anonymity

Anonymity is defined against a malicious issuer and under exposure of both challenge member secret keys, but with an honest opener. The challenger generates the opener key pair and two member key pairs honestly, samples a single hidden bit b, and gives the adversary both member secret keys. The two key pairs are therefore honestly generated *challenge key pairs*; since the experiment performs no honest join, they need not correspond to registered members.

The adversary controls the issuer side of the experiment. On each left-or-right signing query, it supplies an issuer public key ipk and one candidate certificate for each challenge public key. The oracle signs the same message under both candidate inputs and returns the signature selected by b. It returns $\perp$ unless both candidate signatures are accepted by Vfy_Γ. Consequently, adversarially chosen certificates cannot reveal b merely by making signing or verification succeed for only one branch.

The adversary also has access to an opening oracle. Because the issuer is malicious, the adversary supplies both the issuer public key and the registry used in each opening query. The oracle refuses every query whose verification statement $(\mathsf{ipk}, m, \sigma_m)$ was returned by the left-or-right signing oracle. The registry is deliberately not part of the recorded statement, so the adversary cannot open a challenge signature simply by resubmitting it relative to a different registry.

$\mathbf{Exp}^{\mathsf{Anon}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$
  $(\mathsf{usk}_0, \mathsf{upk}_0) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
  $(\mathsf{usk}_1, \mathsf{upk}_1) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
  $b \leftarrow_\$ \{0,1\}$
  $\mathcal{Q}_\sigma \leftarrow \emptyset$
  $b^* \leftarrow_\$ \mathcal{A}^{\mathcal{O}^b_{\mathsf{LoRSign}}, \mathcal{O}_{\mathsf{Open}}}(\mathsf{pp}_\Gamma, \mathsf{opk}, (\mathsf{usk}_0, \mathsf{upk}_0), (\mathsf{usk}_1, \mathsf{upk}_1))$
    where oracle $\mathcal{O}^b_{\mathsf{LoRSign}}(m, \mathsf{ipk}, \sigma^0_s, \sigma^1_s)$:
      $\sigma^0_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}_0, \mathsf{upk}_0, \sigma^0_s, \mathsf{ipk}, \mathsf{opk}, m)$
      $\sigma^1_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}_1, \mathsf{upk}_1, \sigma^1_s, \mathsf{ipk}, \mathsf{opk}, m)$
      return $\perp$ if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma^0_m) \neq 1$
      return $\perp$ if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma^1_m) \neq 1$
      $\mathcal{Q}_\sigma \leftarrow \mathcal{Q}_\sigma \cup \{(\mathsf{ipk}, m, \sigma^b_m)\}$
      return σ^b_m
    where oracle $\mathcal{O}_{\mathsf{Open}}(\mathsf{ipk}, \mathsf{REG}, m, \sigma_m)$:
      return $\perp$ if $(\mathsf{ipk}, m, \sigma_m) \in \mathcal{Q}_\sigma$
      return $\mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{REG}, \mathsf{ipk}, m, \sigma_m)$
  return 1 if $b^* = b$; else return 0

Figure 8: Anonymity experiment for group signature scheme Γ.

Definition 18 (Anonymity)**.** A group signature scheme Γ satisfies *anonymity* if for every PPT adversary $\mathcal{A}$ whose output lies in $\{0,1\}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{Anon}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] - 1/2\right| \leq \mathsf{negl}(\lambda).$$

The experiment is given in Figure 8.

The experiment uses one hidden bit across all signing queries and therefore captures multi-challenge left-or-right anonymity. An analogous notion that gave the adversary the opener secret key while retaining a registry from which the challenge signatures can be opened would be incompatible with opening correctness: the adversary could run Opn_Γ on a challenge signature and recover the corresponding member public key and, whenever the two challenge public keys are distinct, determine the hidden bit. This is why anonymity requires an honest, non-corrupted opener.

3.2.2 Traceability

Traceability is defined with an honest issuer and an honestly generated opener key pair whose secret key is exposed to the adversary. This follows the registry-based structure of the BSZ traceability notion [BSZ05], adapted to our separated key-generation and non-interactive join interfaces. The issuer secret key remains hidden, whereas the adversary receives $(\mathsf{osk}, \mathsf{opk})$ and may therefore use the opener secret key when constructing its forgery.

The challenger initializes an empty registry and a separate join log $\mathcal{Q}_{\mathsf{join}}$. On input a member public key upk, the oracle invokes the honest issuer's join algorithm. If joining fails, the oracle returns $\perp$ without changing either object. Otherwise, it updates the registry, records upk in $\mathcal{Q}_{\mathsf{join}}$, and returns the membership certificate together with the updated, globally readable registry.

At the end of the experiment, the adversary outputs a message and a candidate group signature. The adversary wins if the signature is accepted but the prescribed opening algorithm either fails, produces an opening rejected by Jdg_Γ, or identifies a member public key that was not submitted to any successful join invocation. The last condition is checked against $\mathcal{Q}_{\mathsf{join}}$ rather than by inspecting the scheme-dependent registry representation.

$\mathbf{Exp}^{\mathsf{Traceability}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$
  $(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$
  $\mathsf{REG} \leftarrow \emptyset$
  $\mathcal{Q}_{\mathsf{join}} \leftarrow \emptyset$
  $(m^*, \sigma_m^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}_{\mathsf{Join}}}(\mathsf{pp}_\Gamma, \mathsf{ipk}, \mathsf{osk}, \mathsf{opk})$
    where oracle $\mathcal{O}_{\mathsf{Join}}(\mathsf{upk})$:
      $z \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{REG}, \mathsf{upk}, \mathsf{opk})$
      return $\perp$ if $z = \perp$
      parse z as $(\sigma_s, \mathsf{REG}')$
      $\mathsf{REG} \leftarrow \mathsf{REG}'$
      $\mathcal{Q}_{\mathsf{join}} \leftarrow \mathcal{Q}_{\mathsf{join}} \cup \{\mathsf{upk}\}$
      return (σ_s, REG)
  return 0 if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m^*, \sigma_m^*) \neq 1$
  $z^* \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{REG}, \mathsf{ipk}, m^*, \sigma_m^*)$
  return 1 if $z^* = \perp$
  parse z^* as (upk^*, O)
  return 1 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \mathsf{REG}, m^*, \sigma_m^*, \mathsf{upk}^*, O) \neq 1$
  return 1 if $\mathsf{upk}^* \notin \mathcal{Q}_{\mathsf{join}}$
  return 0

Figure 9: Traceability experiment for group signature scheme Γ.

Definition 19 (Traceability)**.** A group signature scheme Γ satisfies *traceability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbf{Exp}^{\mathsf{Traceability}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The experiment is given in Figure 9.

This definition relies essentially on trusted registration bookkeeping. Because the challenger controls the issuer key and records every successful join, the condition $\mathsf{upk}^* \in \mathcal{Q}_{\mathsf{join}}$ certifies that the opened identity was submitted to a successful join execution, independently of how REG is represented. Once this trusted log is removed or the issuer is allowed to control the relevant bookkeeping maliciously, the same conclusion no longer follows merely from the existence of a valid opening.

3.2.3 Non-Frameability

Non-frameability protects an honestly generated member identity against false attribution. The challenger generates a single target key pair $(\mathsf{usk}, \mathsf{upk})$ and keeps usk hidden. In contrast, the adversary fully controls the issuer and opener contexts: it may choose arbitrary issuer and opener public keys, arbitrary certificates, and the registry used for the final adjudication.

The adversary has access to a signing oracle for the target secret key. Each query specifies the issuer public key, opener public key, certificate, and message under which the target is to sign. Thus, non-frameability does not prevent the adversary from obtaining signatures that the target was explicitly asked to produce. It prevents the adversary from producing a *fresh* verification statement that is publicly accepted as an attribution to upk.

The experiment contains no honest join execution. Accordingly, $(\mathsf{usk}, \mathsf{upk})$ is an honestly generated target key pair, but its public key is not necessarily a legitimately registered member identity. This distinction is intentional: because the issuer and the final registry are adversarially controlled, the protection is tied to possession of the target secret key, not to trusted membership state.

$\mathbf{Exp}^{\mathsf{NonFrame}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
  $\mathcal{Q}_\sigma \leftarrow \emptyset$
  $(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, m^*, \sigma^*_m, O^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}_\Gamma, \mathsf{upk})$
    where oracle $\mathcal{O}_{\mathsf{Sign}}(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$:
      $\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$
      return $\perp$ if $\sigma_m = \perp$
      $\mathcal{Q}_\sigma \leftarrow \mathcal{Q}_\sigma \cup \{(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)\}$
      return σ_m
  return 0 if $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m) \in \mathcal{Q}_\sigma$
  return 1 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, m^*, \sigma^*_m, \mathsf{upk}, O^*) = 1$
  return 0

Figure 10: Non-frameability experiment for group signature scheme Γ.

Definition 20 (Non-Frameability)**.** A group signature scheme Γ satisfies *non-frameability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every

$\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbf{Exp}^{\mathsf{NonFrame}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The experiment is given in Figure 10.

The freshness test records the complete verification statement

$$(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m),$$

but not the certificate or registry. Hence an oracle-generated signature does not become fresh merely because the adversary supplies a different certificate or asks for its attribution relative to a different registry. Conversely, reusing the same signature string in a different issuer, opener, or message context constitutes a distinct statement whose attribution must also be unforgeable.

Acceptance by Jdg_Γ is itself the public assertion that the target member signed. The experiment therefore does not separately require

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m) = 1.$$

A scheme whose adjudication algorithm accepts a false attribution for an otherwise invalid signature is still frameable.

3.2.4 Trace-Soundness

Trace-soundness rules out ambiguous attribution. The adversary chooses the issuer and opener public keys, a message and signature, and two candidate opening claims. Both claims refer to the same verification statement

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m),$$

but they may be evaluated relative to different adversarially chosen registries. The adversary wins if both claims are accepted by Jdg_Γ and identify different member public keys.

Allowing REG^*_0 and REG^*_1 to differ is intentional. The resulting notion requires *cross-registry* consistency: changing the registration state must not permit the same signature to be attributed to a different identity. This is stronger than uniqueness relative to one fixed registry, which would only exclude two distinct accepted identities for $\mathsf{REG}^*_0 = \mathsf{REG}^*_1$.

$\mathbf{Exp}^{\mathsf{TraceSound}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $z^* \leftarrow_\$ \mathcal{A}(\mathsf{pp}_\Gamma)$
  return 0 if z^* does not parse as
    $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m, \mathsf{upk}^*_0, O^*_0, \mathsf{REG}^*_0, \mathsf{upk}^*_1, O^*_1, \mathsf{REG}^*_1)$
  return 0 if $\mathsf{upk}^*_0 = \mathsf{upk}^*_1$
  return 0 if
    $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*_0, m^*, \sigma^*_m, \mathsf{upk}^*_0, O^*_0) \neq 1$
  return 0 if
    $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*_1, m^*, \sigma^*_m, \mathsf{upk}^*_1, O^*_1) \neq 1$
  return 1

Figure 11: Trace-soundness experiment for group signature scheme Γ.

Definition 21 (Trace-Soundness)**.** A group signature scheme Γ satisfies *trace-soundness* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbf{Exp}^{\mathsf{TraceSound}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The experiment is given in Figure 11.

As in non-frameability, acceptance by Jdg_Γ is the relevant public adjudication event; a separate call to Vfy_Γ is therefore unnecessary. Trace-soundness does not require an accepted opening to exist. It only requires that all accepted openings, if any, identify the same member public key.

Thus, traceability and trace-soundness are informally existence and uniqueness requirements, respectively. This analogy should not be read as a formal duality: traceability above is defined relative to honestly generated issuer and opener keys and challenger-maintained registration state, whereas trace-soundness quantifies over adversarially chosen public keys and registries.

3.2.5 Non-Claimability

Non-claimability protects signatures produced with an honestly generated member secret key against re-attribution. The challenger generates a target key pair $(\mathsf{usk}, \mathsf{upk})$, keeps usk secret, and gives the adversary access to a signing oracle for that key. Thus, the adversary can obtain signatures from the target but cannot use the target secret key directly.

The adversary controls the issuer and opener contexts and may query the signing oracle using arbitrary issuer public keys, opener public keys, certificates, and messages. The oracle records each non-failing output together with its verification context. The adversary wins if it selects one of these recorded statements and produces an opening, relative to an arbitrary registry, that is accepted as an attribution to a public key different from upk.

As in the non-frameability experiment, no honest join is performed. The target is therefore an honestly generated key pair, but not necessarily a member whose admission is certified by trusted registration state. Non-claimability binds the protected signatures to the public key used by the signing oracle rather than to an independently established membership record.

Definition 22 (Non-Claimability)**.** A group signature scheme Γ satisfies *non-claimability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbf{Exp}^{\mathsf{NonClaimability}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The experiment is given in Figure 12.

The signing oracle defines the set of protected statements. Requiring the candidate statement to occur in $\mathcal{Q}_\sigma$ separates re-attribution from ordinary signature forgery: only signatures actually returned by the target's signing oracle are eligible.

Non-claimability and trace-soundness address different forms of ambiguity. Trace-soundness requires two accepted openings to different identities, but applies to arbitrary adversarially generated statements. Non-claimability is restricted to signing-oracle outputs generated with the honest target key pair, but it is violated by a single accepted opening to any identity other than upk; an accepted opening to upk is not required.

4 A BSZ-Style Secure yet Intuitively Forgeable Construction

We construct a group signature scheme that satisfies anonymity, traceability, non-frameability, trace-soundness, and non-claimability as defined in Section 3. Nevertheless,

$\mathbf{Exp}^{\mathsf{NonClaimability}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
  $\mathcal{Q}_\sigma \leftarrow \emptyset$
  $z^* \leftarrow_\$ \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}_\Gamma, \mathsf{upk})$
    where oracle $\mathcal{O}_{\mathsf{Sign}}(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$:
      $\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$
      return $\perp$ if $\sigma_m = \perp$
      $\mathcal{Q}_\sigma \leftarrow \mathcal{Q}_\sigma \cup \{(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)\}$
      return σ_m
  return 0 if z^* does not parse as
    $(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, \mathsf{upk}^*, m^*, \sigma^*_m, O^*)$
  return 0 if $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m) \notin \mathcal{Q}_\sigma$
  return 0 if $\mathsf{upk}^* = \mathsf{upk}$
  return 1 if
    $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, m^*, \sigma^*_m, \mathsf{upk}^*, O^*) = 1$
  return 0

Figure 12: Non-claimability experiment for group signature scheme Γ.

a malicious opener can produce verifying signatures that are not backed by an issuer-generated membership certificate and do not arise from any join execution, even when the issuer is honest.

The construction is intentionally artificial. Its purpose is not to provide a deployable scheme, but to isolate a semantic guarantee that is absent from the preceding experiments: those experiments do not require every verifying signature, or every accepted attribution under a malicious opener key, to be induced by issuer-certified membership.

Dummy registry. Throughout this section, the registry is the fixed dummy state

$$\mathsf{REG} \leftarrow \emptyset.$$

Joining leaves this state unchanged. Every algorithm that takes a registry as input returns $\perp$, or 0 in the case of a decision algorithm, whenever $\mathsf{REG} \neq \emptyset$. The scheme-dependent registry therefore carries no membership information. This does not make the traceability experiment vacuous: that experiment separately records successful join queries in the challenger-maintained set $\mathcal{Q}_{\mathsf{join}}$.

4.1 Overview

The standard branch follows the standard encrypt-then-prove paradigm. The issuer signs a membership certificate for the member public key. To sign, the member encrypts its public key under the opener key and proves knowledge of the corresponding member secret key, certificate, and encryption randomness. The opener decrypts the embedded identity and proves that the decryption was performed with a valid decryption key.

The contrived feature is an auxiliary branch in Vfy_Γ. The opener public key contains an additional digital-signature verification key pk'_Σ. Honest opener key generation erases the corresponding signing key sk'_Σ. A malicious opener may instead retain this key and use it to authenticate an arbitrary signature tuple directly. This auxiliary branch requires neither a membership certificate nor a join execution.

The opening-verification algorithm contains a matching auxiliary branch. It accepts an attribution to y' when the adversary supplies a preimage of y' under a fixed one-way function. Consequently, a malicious opener can create both a verifying group signature and an accepted attribution to an identity of its choice, without involving the issuer.

4.2 Building Blocks and Languages

We use the primitives introduced in Section 2. In particular, let

$$\mathcal{F} = \{f_\lambda : C_\lambda \to D_\lambda\}_{\lambda\in\mathbb{N}}$$

be a fixed one-way function family. We assume that C_λ is efficiently samplable, that the description of f_λ is efficiently obtainable from 1^λ, and that

$$D_\lambda \subseteq \mathcal{M}_{\Theta,\lambda}.$$

As usual, $f_\lambda(w) = \bot$ for $w \notin C_\lambda$.

In the relations below, $w \in C_\lambda$, $c \neq \bot$, and the claimed plaintext in L_2 satisfies $y \neq \bot$. Encryption randomness must be a valid random tape. These are part of the relation's domain checks, not additional hardness assumptions. In particular, two failing algorithm evaluations do not establish an encryption or decryption relation merely because both return $\bot$.

The construction uses two parameterized NP languages. Their statements contain the complete public signature context, including the issuer key, opener key, ciphertext, and message. Some of these values do not occur in the algebraic conditions below; including them nevertheless binds each proof to the full verification statement.

$$\begin{aligned} L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta} \coloneqq \{\ & (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m) \mid \\ & \exists (r, \sigma_s, w) : \mathsf{Vfy}_\Sigma\big(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, f(w)), \sigma_s\big) = 1 \\ & \wedge\, c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w); r)\ \}, \end{aligned}$$

and

$$\begin{aligned} L_2^{\mathsf{pp}_\Theta} \coloneqq \{\ & (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m, y) \mid \\ & \exists \mathsf{sk}_\Theta : \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1 \\ & \wedge\, y = \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c)\ \}. \end{aligned}$$

4.3 Construction

The construction is given in Construction 3. Boxes mark the auxiliary opener-controlled branch; deleting the boxed components yields the intended certificate-based core.

We slightly overload notation and use $\bot$ inside encoded tuples as a distinguished syntactic placeholder for the empty string.



Construction 3: Contrived group signature scheme Γ. Boxes mark the auxiliary opener-controlled branch, which is removed in the final construction.

Theorem 10 (Correctness)**.** *The group signature scheme Γ is perfectly correct.*

Proof. Consider honestly generated parameters, issuer and opener keys, and any sequence of honestly generated member key pairs. Every invocation of Join_Γ receives $\mathsf{REG} = \emptyset$ and returns the same registry state, so all joins succeed. For a member pair (w, y), we have $y = f(w)$, and correctness of Σ implies that the issued certificate σ_s verifies for $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y)$.

For every honestly generated ciphertext

$$c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y; r),$$

the tuple (r, σ_s, w) is therefore a valid witness for the corresponding statement in $L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta}$. Correctness of the proof system implies that Vfy_Γ accepts the resulting standard-branch signature.

By correctness of Θ,

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c) = y \qquad \text{and} \qquad \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1.$$

Hence sk_Θ is a valid witness for the opening statement in $L_2^{\mathsf{pp}_\Theta}$. Correctness of the proof system implies that Jdg_Γ accepts the output of Opn_Γ and that the recovered public key equals the signer's public key y. Honest signing never invokes the auxiliary branch, which proves the claim. □

4.4 The Membership Bypass

We now describe the attack explicitly. Let the issuer public key pk_Σ be honestly generated. A malicious opener generates

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta) \qquad \text{and} \qquad (\mathsf{sk}'_\Sigma, \mathsf{pk}'_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma),$$

publishes $\mathsf{opk} = (\mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma)$, and retains both secret keys. It then:

1. samples $O^* \leftarrow_\$ C_\lambda$ and sets $y^* \leftarrow f(O^*)$;
2. chooses a message $m^* \in \mathcal{MS}$, a well-formed ciphertext c^*, and an arbitrary syntactically valid string π^*;
3. computes
$$\sigma'^* \leftarrow_\$ \mathsf{Sgn}_\Sigma\big(\mathsf{sk}'_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y^*, m^*, c^*, \pi^*)\big);$$
4. outputs
$$\sigma^*_m = (c^*, \pi^*, \sigma'^*, y^*).$$

Correctness of Σ gives

$$\mathsf{Vfy}_\Gamma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma), m^*, \sigma^*_m) = 1.$$

Moreover, for the dummy registry $\emptyset$,

$$\mathsf{Jdg}_\Gamma\big(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma), \emptyset, m^*, \sigma^*_m, y^*, O^*\big) = 1$$

because $f(O^*) = y^*$. No membership certificate, issuer secret key, or join execution is used at any point. Thus, even with an honest issuer, a malicious opener can create a verifying signature and an accepted attribution that have no issuer-certified membership semantics.

The attack does not contradict correctness or any of the five security notions from Section 3. Honest opener key generation erases sk'_Σ, while traceability uses an honestly generated opener key. In the experiments that allow a malicious opener key, the remaining properties constrain ambiguity and false attribution, but do not require every accepted auxiliary-branch identity to occur in the trusted join log.

4.5 Security

Table 3 summarizes the assumptions used for each property. The entries give the corresponding reduction loss; an empty entry means that the assumption is not used.

We now record the resulting guarantees.

Theorem 11 (Trace-Soundness)**.** Γ *is trace-sound.*

Proof idea. For an auxiliary-branch signature, every accepted opening must identify the fixed value y' contained in the signature. For a standard-branch signature, simulation-sound extraction yields valid decryption keys for both accepted opening proofs. If the extracted openings identify distinct values, the two keys pass KVfy_Θ for the same public key but decrypt the same ciphertext differently, contradicting key-verifiability. Zero-knowledge accounts for the preceding CRS and proof hybrids.

The proof is given in Section 4.8.1.

Table 3: Assumptions used in the displayed proof of each security property of the contrived construction Γ. Entries denote the corresponding loss; an empty cell means that the assumption is not used in that proof. The parameter q_o denotes the relevant opening-query bound. KV denotes key-verifiability, and DS denotes decryption-soundness.

Property	**ZK**	**SSE**	**KV**	**OWF**	**Σ-UNF**	**DS**	**mcIND-CPA**
Anonymity	2	q_o					2
Non-frameability	2	2		2		1	
Traceability	1	1			2		
Trace-soundness	1	2	1				
Non-claimability	1	1				1	

Theorem 12 (Non-Frameability)**.** Γ *is non-frameable.*

Proof idea. In the auxiliary branch, an accepted attribution to the honest target public key directly yields a preimage under f. In the standard branch, extraction recovers a certified value encrypted in c and a valid decryption key from the opening proof. Decryption-soundness forces the extracted plaintext to equal the attributed target key, again yielding a preimage of that key. Freshness of the framed signature permits simulation-sound extraction, and zero-knowledge supports the required proof hybrids.

The proof is given in Section 4.8.2.

Theorem 13 (Traceability)**.** Γ *is traceable.*

Proof idea. The opener key is honestly generated in the traceability experiment, so the adversary does not obtain sk'_{Σ}. An accepted auxiliary-branch signature therefore yields a forgery under pk'_{Σ}. From an accepted standard-branch signature, extraction recovers an issuer certificate and the encrypted member public key. Either that certificate originates from a successful join query, in which case the opened key occurs in $\mathcal{Q}_{\mathsf{join}}$, or it yields a forgery under the honest issuer verification key pk_{Σ}.

The proof is given in Section 4.8.3.

Theorem 14 (Anonymity)**.** Γ *is anonymous.*

Proof idea. Challenge signatures always use the standard branch. Zero-knowledge permits replacement of the certificate proofs by simulated proofs. The mcIND-CPA property hides which challenge public key is encrypted. SSE supports consistent simulation of adaptive opening queries. It also prevents a proof produced outside the simulation from evading the intended relation.

The proof is given in Section 4.8.4.

Theorem 15 (Non-Claimability)**.** Γ *is non-claimable.*

Proof idea. Every signing-oracle output uses the standard branch and contains an honestly generated encryption of the target public key under the adversarially supplied encryption key. Extraction from an accepted re-attribution yields a decryption key accepted by KVfy_{Θ}. Decryption-soundness forces that key to recover the original target public key, contradicting the requirement that the claimed identity be different. Zero-knowledge and simulation-sound extractability justify the proof-system hybrids and extraction step.

The proof is given in Section 4.8.5.

4.6 Relation to Existing Definitions

Bootle et al. [BCC+16, BCC+20] consider both monolithic and separated management roles and incorporate protection against maliciously chosen keys where possible. Backes et

al. [BHS19] build on this fully dynamic line of work. The feature relevant here is structural: traceability is coupled to honest admission and challenger-maintained registration bookkeeping, while the remaining opening-based notions constrain the validity or consistency of attribution.

Our distilled BSZ-style experiments preserve this structural feature. They do not contain a separate requirement that the verification relation be exclusive to join-generated certificates, nor do they require every accepted attribution under a malicious opener key to identify an entry in the trusted join log. The auxiliary branch exploits precisely this gap.

This construction should therefore be read relative to the exact experiments in Section 3, rather than as a claim that our simplified syntax is a verbatim reproduction of every cited model. We remove the structural restriction in our framework and allow the adversary to supply candidate openings directly; see Section 5.

4.7 Conclusion

The five classical properties ensure anonymity, opening existence under the traceability trust assumptions, resistance to framing, uniqueness of accepted openings, and resistance to re-attribution. They do not, by themselves, require every verifying signature or accepted attribution under a malicious opener key to originate from issuer-certified membership.

The auxiliary verification key makes this distinction explicit. Honest opener key generation erases the corresponding signing key, so all classical experiments remain secure under the assumptions summarized above. A malicious opener can retain that key, bypass the certificate relation, and create both a verifying signature and an accepted opening for an identity that never joined. The missing guarantee is therefore not ordinary signature validity or opening consistency, but an explicit binding between acceptance and membership issuance.

4.8 Proofs for the Contrived Construction

Challenge-parameter convention. Whenever a reduction embeds a primitive challenge, it uses the public parameters supplied with that challenge throughout the simulated experiment. In particular, an EUF-CMA reduction embeds the challenger's pp_Σ together with its verification key; it does not resample pp_Σ. Any other signature keys needed by the simulation are generated under that same pp_Σ. Generating the "remaining public parameters" means generating only the other, independently sampled components, with relation-dependent setups using the resulting public parameters.

4.8.1 Proof of Theorem 11 (Trace-Soundness)

Proof strategy. We first observe that the auxiliary branch cannot yield two distinct accepted identities, since every accepted auxiliary opening is fixed by the value carried in the signature. It therefore remains to consider the standard branch. We switch the L_2 setup to extractable mode, extract a valid decryption key from each accepted opening, and finally use key-verifiability to show that both keys decrypt the common ciphertext identically.

- **Game 0 (Original experiment).** Exclude a successful auxiliary-branch output directly and reduce the remaining case to two accepted L_2 openings.
- **Game 1 (Simulated L_2 CRS).** Replace the real L_2 CRS by a simulated CRS. Since the experiment generates no L_2 proof, zero-knowledge is used only for the setup transition.
- **Game 2 (Extractable L_2 CRS).** Switch to the extractable setup; the public CRS distribution is identical by setup compatibility.

- **Game 3 (Extract both opening witnesses).** Extract a valid decryption key from each accepted opening proof. Invalid extraction violates SSE for L_2.
- **Game 4 (Consistent decryption).** Use key-verifiability to rule out different decryptions of the common ciphertext under two keys accepted for the same public key.

Proof. Let $\mathcal{A}$ be a PPT adversary in the trace-soundness experiment. We proceed through a sequence of games. For every i, let S_i denote the event that $\mathcal{A}$ wins Game i.

Game 0: Original experiment. This is the experiment from Figure 11. Suppose that S_0 occurs, and write the adversary's output as

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{REG}_0^*, \mathsf{upk}_1^*, O_1^*, \mathsf{REG}_1^*).$$

Since both opening verifications accept, the public keys and the signature parse as

$$\mathsf{ipk}^* = \mathsf{pk}_\Sigma^*, \qquad \mathsf{opk}^* = (\mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*), \qquad \sigma_m^* = (c^*, \pi^*, \sigma'^*, y'^*),$$

and $\mathsf{REG}_0^* = \mathsf{REG}_1^* = \emptyset$. Moreover, σ'^* and y'^* are either both $\perp$ or both different from $\perp$.

Assume first that $\sigma'^* \neq \perp$ and $y'^* \neq \perp$. In this case, both invocations of Jdg_Γ use the auxiliary branch. Acceptance therefore implies

$$f(O_0^*) = y'^* = \mathsf{upk}_0^* \qquad \text{and} \qquad f(O_1^*) = y'^* = \mathsf{upk}_1^*.$$

Thus $\mathsf{upk}_0^* = \mathsf{upk}_1^*$, contradicting the winning condition. Hence every successful execution necessarily satisfies

$$\sigma_m^* = (c^*, \pi^*, \perp, \perp),$$

and both accepted openings use the $L_2^{\mathsf{pp}_\Theta}$ branch.

Game 1: Simulated CRS for $L_2^{\mathsf{pp}_\Theta}$. Instead of generating crs_Π^2 with the real setup algorithm, generate

$$(\mathsf{crs}_\Pi^2, \tau^2) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda).$$

All remaining parameters are generated as in Game 0. In particular, the trace-soundness experiment has no prover oracle and the challenger generates no $L_2^{\mathsf{pp}_\Theta}$ proofs.

Consider the ZK distinguisher that embeds its input CRS as crs_Π^2, generates all remaining public parameters honestly, runs $\mathcal{A}$, and outputs 1 exactly if $\mathcal{A}$ wins the resulting trace-soundness experiment. This distinguisher makes no prover-oracle queries. Zero-knowledge therefore gives a negligible function $\varepsilon_{\mathsf{zk}}$ such that

$$\big|\Pr[S_1] - \Pr[S_0]\big| \leq \varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk}}(1^\lambda)}$

1. Receive crs_Π^2 from the zero-knowledge challenger and embed it as the CRS for $L_2^{\mathsf{pp}_\Theta}$.
2. Generate all remaining public parameters honestly.
3. Run $\mathcal{A}$ in the trace-soundness experiment.
4. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 2: Extractable CRS for $L_2^{\mathsf{pp}_\Theta}$. Replace the simulated setup by

$$(\mathsf{crs}_\Pi^2, \tau^2, \zeta^2) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

The trapdoor ζ^2 is kept by the challenger and is not given to $\mathcal{A}$. By the assumed identity of the CRS distributions output by SIM_1 and $\mathcal{E}_1$, the adversary's view is identically distributed in Games 1 and 2. Consequently,

$$\Pr[S_2] = \Pr[S_1].$$

Game 3: Extract the opening witnesses. Game 3 is identical to Game 2, except that after a successful output the challenger extracts a witness from each accepted opening proof. For $i \in \{0,1\}$, define

$$x^*_{o,i} = (\mathsf{pk}^*_\Sigma, \mathsf{pk}^*_\Theta, \mathsf{pk}'^*_\Sigma, c^*, m^*, \mathsf{upk}^*_i)$$

and compute

$$\mathsf{sk}_{\Theta i}^* \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}^2_\Pi, \zeta^2, x^*_{o,i}, O^*_i).$$

Let $\mathsf{Bad}_{\mathsf{ext},i}$ be the event that S_2 occurs but the extracted value is not a valid witness for $x^*_{o,i}$, i.e.,

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta i}^*, \mathsf{pk}^*_\Theta) \neq 1 \quad \text{or} \quad \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta i}^*, c^*) \neq \mathsf{upk}^*_i.$$

Set

$$\mathsf{Bad}_{\mathsf{ext}} = \mathsf{Bad}_{\mathsf{ext},0} \vee \mathsf{Bad}_{\mathsf{ext},1},$$

and let Game 3 abort whenever $\mathsf{Bad}_{\mathsf{ext}}$ occurs.

For each fixed $i \in \{0,1\}$, construct an SSE adversary $\mathcal{B}_{\mathsf{sse},i}$ as follows. It embeds the CRS received from the SSE experiment as crs^2_Π, generates all remaining public parameters honestly, and runs $\mathcal{A}$. If $\mathcal{A}$ produces a successful standard-branch output, $\mathcal{B}_{\mathsf{sse},i}$ returns $(x^*_{o,i}, O^*_i)$; otherwise, it returns an arbitrary rejecting pair. The reduction makes no simulation-oracle queries. Hence $(x^*_{o,i}, O^*_i)$ is fresh in the SSE experiment. Furthermore, on S_2 we have

$$\mathsf{Vfy}_\Pi(\mathsf{crs}^2_\Pi, x^*_{o,i}, O^*_i) = 1$$

because the corresponding invocation of Jdg_Γ accepts. Thus, $\mathsf{Bad}_{\mathsf{ext},i}$ makes $\mathcal{B}_{\mathsf{sse},i}$ win the SSE experiment.

Let $\varepsilon_{\mathsf{sse},i}$ denote the advantage of $\mathcal{B}_{\mathsf{sse},i}$. By simulation-sound extractability and a union bound,

$$\begin{aligned}\big|\Pr[S_3] - \Pr[S_2]\big| &\leq \Pr[\mathsf{Bad}_{\mathsf{ext}}] \\ &\leq \varepsilon_{\mathsf{sse},0}(\lambda) + \varepsilon_{\mathsf{sse},1}(\lambda).\end{aligned}$$

$\underline{\mathcal{B}_{\mathsf{sse},i}(1^\lambda)}$

1. Receive crs^2_Π from the SSE challenger and embed it as the CRS for $L_2^{\mathsf{pp}_\Theta}$.
2. Generate all remaining public parameters honestly and run $\mathcal{A}$.
3. If $\mathcal{A}$ does not produce a successful standard-branch output, return an arbitrary rejecting statement–proof pair.
4. For the fixed $i \in \{0,1\}$, define
$$x^*_{o,i} = (\mathsf{pk}^*_\Sigma, \mathsf{pk}^*_\Theta, \mathsf{pk}'^*_\Sigma, c^*, m^*, \mathsf{upk}^*_i).$$
5. Output $(x^*_{o,i}, O^*_i)$.

Whenever S_3 occurs, both extracted values therefore satisfy

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta i}^*, \mathsf{pk}^*_\Theta) = 1 \quad \text{and} \quad \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta i}^*, c^*) = \mathsf{upk}^*_i \qquad \text{for } i \in \{0,1\}. \tag{1}$$

Game 4: Key-verifiability. Game 4 is identical to Game 3, except that it aborts if

$$\mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) \neq \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*).$$

By (1), both extracted keys pass KVfy_Θ for the same public key pk^*_Θ. The abort event therefore violates key-verifiability.

More precisely, a key-verifiability adversary $\mathcal{B}_{\mathsf{kv}}$ embeds the public encryption parameters received from its challenger into pp_Γ, generates crs^2_Π together with its extraction trapdoor using $\mathcal{E}_1$, and runs $\mathcal{A}$. It then extracts $\mathsf{sk}_{\Theta 0}^*$ and $\mathsf{sk}_{\Theta 1}^*$ as in Game 3. If both witnesses are valid, it outputs

$$(\mathsf{sk}_{\Theta 0}^*, \mathsf{sk}_{\Theta 1}^*, \mathsf{pk}^*_\Theta, c^*)$$

to the key-verifiability experiment; otherwise, it outputs an arbitrary non-winning tuple. Thus, for a negligible function $\varepsilon_{\mathsf{kv}}$,

$$\left|\Pr[S_4] - \Pr[S_3]\right| \leq \varepsilon_{\mathsf{kv}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{kv}}(1^\lambda)}$

1. Receive pp_Θ from the key-verifiability challenger and embed it into pp_Γ.
2. Generate crs_Π^2 with $\mathcal{E}_1$, retain ζ^2, and generate all remaining public parameters honestly.
3. Run $\mathcal{A}$ and extract $\mathsf{sk}_{\Theta 0}^*$ and $\mathsf{sk}_{\Theta 1}^*$ from the two accepted opening proofs as in Game 3.
4. If either extraction is invalid, output an arbitrary non-winning tuple.
5. Otherwise output
$$(\mathsf{sk}_{\Theta 0}^*, \mathsf{sk}_{\Theta 1}^*, \mathsf{pk}_\Theta^*, c^*).$$

It remains to analyze Game 4. If S_4 occurred, then Equation (1) and the absence of the key-verifiability abort would give

$$\mathsf{upk}_0^* = \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) = \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*) = \mathsf{upk}_1^*,$$

contradicting the trace-soundness winning condition. Hence

$$\Pr[S_4] = 0.$$

Combining the game transitions yields

$$\Pr[S_0] \leq \varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{sse},0}(\lambda) + \varepsilon_{\mathsf{sse},1}(\lambda) + \varepsilon_{\mathsf{kv}}(\lambda),$$

which is negligible. Therefore Γ is trace-sound. □

4.8.2 Proof of Theorem 12 (Non-Frameability)

Proof strategy. We treat the two verification branches separately. A successful attribution through the auxiliary branch immediately reveals a preimage of the honest member public key. For the standard branch, we simulate signing-oracle proofs, extract the certified plaintext and the opening key, and invoke decryption-soundness to identify the extracted plaintext with the framed member public key. The resulting preimage again contradicts one-wayness.

- **Game 0 (Original experiment).** This is the non-frameability experiment.
- **Game 1 (Simulated L_1 proofs).** Simulate every L_1 proof returned by the signing oracle. This is justified by zero-knowledge.
- **Game 2 (Extractable L_2 setup).** Switch the L_2 CRS to extractable mode through the simulated setup.
- **Game 3 (Exclude the auxiliary branch).** An accepted auxiliary attribution yields a preimage of the honest target key, violating one-wayness of f.
- **Game 4 (Extract the L_1 witness).** Extract the encrypted certified identity from the fresh framed signature. Invalid extraction violates SSE for L_1.
- **Game 5 (Extract the L_2 witness).** Extract a valid decryption key from the accepted opening proof. Invalid extraction violates SSE for L_2.
- **Game 6 (Enforce correct decryption).** Decryption-soundness forces the extracted key to recover the identity certified by the L_1 witness. The remaining successful output yields a preimage of the honest target key.

Proof. Let $\mathcal{A}$ be a PPT adversary in the non-frameability experiment, and let

$$(w^\diamond, y^\diamond) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$$

be the honest member key pair. Thus $y^\diamond = f(w^\diamond)$. We use a sequence of games and write S_i for the event that $\mathcal{A}$ wins Game i.

Game 0: Original experiment. This is the experiment from Figure 10. Let $\mathcal{Q}_\sigma$ be the set of signing-oracle outputs, and write the adversary's final output as

$$(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, m^*, \sigma_m^*, O^*).$$

The event S_0 occurs if

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*) \notin \mathcal{Q}_\sigma$$

and

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, m^*, \sigma_m^*, y^\diamond, O^*) = 1.$$

Game 1: Simulate the L_1 proofs. Generate the L_1 setup as

$$(\mathsf{crs}_\Pi^1, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda)$$

and use SIM_2 to generate the L_1 proofs returned by the signing oracle. More precisely, on a query $(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$, the oracle parses

$$\mathsf{ipk} = \mathsf{pk}_\Sigma \qquad \text{and} \qquad \mathsf{opk} = (\mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma).$$

If parsing fails or

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y^\diamond), \sigma_s) \neq 1,$$

it sets $\sigma_m \leftarrow \bot$. Otherwise, it samples

$$(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y^\diamond).$$

If $c = \bot$, it again sets $\sigma_m \leftarrow \bot$. If $c \neq \bot$, it defines

$$x = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m), \qquad \pi \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}_\Pi^1, \tau^1, x),$$

and sets

$$\sigma_m \leftarrow \begin{cases} (c, \pi, \bot, \bot), & \pi \neq \bot, \\ \bot, & \pi = \bot. \end{cases}$$

If $\sigma_m \neq \bot$, the oracle records $(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)$ in $\mathcal{Q}_\sigma$. It returns σ_m in either case.

For an honestly generated member pair, every query that passes the certificate and encryption checks supplies a valid L_1 statement with witness $(r, \sigma_s, w^\diamond)$. Hence the real and simulated signing oracles are computationally indistinguishable by zero-knowledge. Here we use the setup-compatibility clause that the joint distribution $(\mathsf{crs}_\Pi^1, \tau^1)$ output by SIM_1 is identical to the first two components output by $\mathcal{E}_1$. Therefore, for a negligible function $\varepsilon_{\mathsf{zk},1}$,

$$\big|\Pr[S_1] - \Pr[S_0]\big| \leq \varepsilon_{\mathsf{zk},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},1}(1^\lambda)}$

1. Receive crs_Π^1 and access to the zero-knowledge proof oracle for L_1.
2. Generate the honest target member pair and all remaining public parameters as in the non-frameability experiment.
3. On a signing query, perform the parsing and certificate checks from Game 1 and sample $(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y^\diamond)$.
4. For every non-failing query, define

$$x = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m) \quad \text{and} \quad w_{L_1} = (r, \sigma_s, w^\diamond),$$

   obtain π from the proof oracle, and apply the same logging and return rules as Game 1.
5. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 2: Extractable setup for L_2. Replace the real L_2 setup by

$$(\mathsf{crs}_{\Pi}^{2}, \tau^{2}, \zeta^{2}) \leftarrow_{\$} \mathcal{E}_1(1^{\lambda}).$$

The non-frameability challenger generates no L_2 proofs. A ZK distinguisher can therefore compare the real CRS with a simulated CRS while making no prover-oracle queries; the simulated and extractable CRS distributions are identical by setup compatibility. Thus, for a negligible function $\varepsilon_{\mathsf{zk},2}$,

$$\left|\Pr[S_2] - \Pr[S_1]\right| \leq \varepsilon_{\mathsf{zk},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},2}(1^{\lambda})}$

1. Receive crs_{Π}^{2} from the zero-knowledge challenger and embed it as the CRS for L_2.
2. Generate crs_{Π}^{1} and its simulation information as in Game 1, and generate all remaining public parameters honestly.
3. Run $\mathcal{A}$, answering signing queries exactly as in Game 1.
4. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 3: Exclude the auxiliary branch. Game 3 is identical to Game 2, except that it aborts if a successful output is accepted through the auxiliary branch of Jdg_{Γ}. Let $\mathsf{Bad}_{\mathsf{aux}}$ denote this event.

If $\mathsf{Bad}_{\mathsf{aux}}$ occurs, then the final signature parses as

$$\sigma_m^{*} = (c^{*}, \pi^{*}, \sigma'^{*}, y'^{*}) \quad \text{with} \quad \sigma'^{*} \neq \perp \neq y'^{*},$$

and acceptance as an attribution to $y^{\diamond}$ implies

$$f(O^{*}) = y'^{*} = y^{\diamond}.$$

This immediately yields an inverter for f: given an OWF challenge $y^{\diamond}$, the inverter uses it as the honest member public key, generates both setups with $\mathcal{E}_1$, and answers signing queries with the simulated oracle from Game 1. This simulation does not require the unknown preimage of $y^{\diamond}$. If $\mathsf{Bad}_{\mathsf{aux}}$ occurs, the inverter outputs O^{*}.

The OWF challenge has exactly the distribution of an honestly generated member public key. Hence, for a negligible function $\varepsilon_{\mathsf{ow},\mathsf{aux}}$,

$$\left|\Pr[S_3] - \Pr[S_2]\right| \leq \Pr[\mathsf{Bad}_{\mathsf{aux}}] \leq \varepsilon_{\mathsf{ow},\mathsf{aux}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{ow},\mathsf{aux}}(1^{\lambda})}$

1. Receive a one-wayness challenge $y^{\diamond}$ and use it as the honest target member public key.
2. Generate both proof-system setups with $\mathcal{E}_1$ and generate all remaining public parameters honestly.
3. Run $\mathcal{A}$ and answer signing queries using the simulated oracle from Game 1, which requires no preimage of $y^{\diamond}$.
4. If $\mathcal{A}$ wins through the auxiliary branch, output O^{*}; otherwise, output an arbitrary element of the one-way-function domain.

In every successful execution of Game 3, the auxiliary branch is not taken. Since Jdg_{Γ} rejects if exactly one of σ'^{*} and y'^{*} equals $\perp$, every successful output therefore has the form

$$\mathsf{ipk}^{*} = \mathsf{pk}_{\Sigma}^{*}, \qquad \mathsf{opk}^{*} = (\mathsf{pk}_{\Theta}^{*}, \mathsf{pk}_{\Sigma}'^{*}), \qquad \sigma_m^{*} = (c^{*}, \pi^{*}, \perp, \perp), \qquad O^{*} = \pi_o^{*},$$

with $\mathsf{REG}^{*} = \emptyset$.

Game 4: Extract the L_1 witness. Define

$$x_1^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*, m^*)$$

and extract

$$(r^*, \sigma_s^*, w^*) \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^1, \zeta^1, x_1^*, \pi^*).$$

Let Bad_{L_1} be the event that S_3 occurs but the extracted tuple is not a valid witness for x_1^*.

The pair (x_1^*, π^*) is fresh relative to the L_1 simulation oracle. Indeed, suppose the same pair had previously been returned in response to a signing query. Equality of the complete statements fixes $\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*$, and m^*; equality of the proofs then fixes the entire standard-branch signature. The tuple $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*)$ would consequently belong to $\mathcal{Q}_\sigma$, contradicting the non-frameability freshness condition.

An SSE adversary can therefore simulate the signing oracle using its simulation oracle and output (x_1^*, π^*). If Bad_{L_1} occurs, this adversary wins the SSE experiment for L_1. Consequently, for a negligible function $\varepsilon_{\mathsf{sse},1}$,

$$\big|\Pr[S_4] - \Pr[S_3]\big| \leq \Pr[\mathsf{Bad}_{L_1}] \leq \varepsilon_{\mathsf{sse},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},1}(1^\lambda)}$

1. Receive crs_Π^1 and access to the SSE simulation oracle for L_1.
2. Generate crs_Π^2 with $\mathcal{E}_1$, generate the target member pair and all remaining public parameters honestly, and run $\mathcal{A}$.
3. Answer signing queries as in Game 1, obtaining every L_1 proof from the simulation oracle and maintaining $\mathcal{Q}_\sigma$.
4. If $\mathcal{A}$ produces a successful fresh standard-branch output, define

$$x_1^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*, m^*)$$

and output (x_1^*, π^*); otherwise, output an arbitrary rejecting pair.

Whenever S_4 occurs, the extracted witness satisfies

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, f(w^*); r^*), \tag{2}$$

$$\mathsf{Vfy}_\Sigma\big(\mathsf{pk}_\Sigma^*, (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, f(w^*)), \sigma_s^*\big) = 1. \tag{3}$$

Game 5: Extract the L_2 witness. Define

$$x_2^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*, m^*, y^\diamond)$$

and extract

$$\mathsf{sk}_\Theta^* \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^2, \zeta^2, x_2^*, \pi_o^*).$$

Let Bad_{L_2} be the event that S_4 occurs but sk_Θ^* is not a valid witness for x_2^*. The challenger never generates or simulates an L_2 proof, so the corresponding simulation-query set is empty and (x_2^*, π_o^*) is fresh. Simulation-sound extractability for L_2 gives a negligible function $\varepsilon_{\mathsf{sse},2}$ such that

$$\big|\Pr[S_5] - \Pr[S_4]\big| \leq \Pr[\mathsf{Bad}_{L_2}] \leq \varepsilon_{\mathsf{sse},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},2}(1^\lambda)}$

1. Receive crs_Π^2 from the SSE challenger and embed it as the CRS for L_2.
2. Generate crs_Π^1 with $\mathcal{E}_1$, retain ζ^1, and generate the target member pair and all remaining public parameters.
3. Run $\mathcal{A}$, answer signing queries as in Game 1, and perform the Game 4 extraction and abort checks for L_1.
4. On a successful standard-branch output, define

$$x_2^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*, m^*, y^\diamond)$$

and output (x_2^*, π_o^*); otherwise, output an arbitrary rejecting pair.

Whenever S_5 occurs, the extracted witness satisfies

$$\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) = 1, \tag{4}$$
$$\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) = y^\diamond. \tag{5}$$

Game 6: Decryption-soundness. Game 6 is identical to Game 5, except that it aborts if

$$\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) \neq f(w^*).$$

By (2), c^* is an honestly formed encryption of $f(w^*)$ under pk^*_Θ using randomness r^*, and by (4), sk^*_Θ passes key verification for pk^*_Θ. The abort event therefore gives the tuple

$$(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta, f(w^*), r^*)$$

that wins the decryption-soundness experiment. Hence, for a negligible function $\varepsilon_{\mathsf{ds}}$,

$$\big|\Pr[S_6] - \Pr[S_5]\big| \leq \varepsilon_{\mathsf{ds}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{ds}}(1^\lambda)}$

1. Receive pp_Θ from the decryption-soundness challenger and embed it into pp_Γ.
2. Generate both proof-system setups with $\mathcal{E}_1$, retain the extraction trapdoors, and generate the target member pair and all remaining public parameters honestly.
3. Run $\mathcal{A}$, answering signing queries as in Game 1, and extract the witnesses (r^*, σ_s^*, w^*) and sk^*_Θ from the final accepted transcript as in Games 4 and 5.
4. If all checks succeed, output
$$(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta, f(w^*), r^*);$$
otherwise, output an arbitrary non-winning tuple.

Finally, suppose that S_6 occurs. Equations (5) and the absence of the decryption-soundness abort imply

$$y^\diamond = \mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) = f(w^*).$$

This yields a second OWF inverter. Given a challenge $y^\diamond$, it runs Game 6 using the simulated signing oracle, extracts w^* from the fresh L_1 proof, and outputs w^*. Thus, for a negligible function $\varepsilon_{\mathsf{ow,std}}$,

$$\Pr[S_6] \leq \varepsilon_{\mathsf{ow,std}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{ow,std}}(1^\lambda)}$

1. Receive a one-wayness challenge $y^\diamond$ and use it as the honest target member public key.
2. Generate both proof-system setups with $\mathcal{E}_1$ and generate all remaining public parameters honestly.
3. Run $\mathcal{A}$, answer signing queries using the simulated oracle from Game 1, and perform all extraction and abort checks through Game 6.
4. If $\mathcal{A}$ wins Game 6, output the extracted value w^*; otherwise, output an arbitrary element of the one-way-function domain.

Combining the transitions gives

$$\Pr[S_0] \leq \varepsilon_{\mathsf{zk},1}(\lambda) + \varepsilon_{\mathsf{zk},2}(\lambda) + \varepsilon_{\mathsf{sse},1}(\lambda) + \varepsilon_{\mathsf{sse},2}(\lambda) + \varepsilon_{\mathsf{ds}}(\lambda) + \varepsilon_{\mathsf{ow,aux}}(\lambda) + \varepsilon_{\mathsf{ow,std}}(\lambda),$$

which is negligible. Equivalently, using a common upper bound for each assumption, the loss is

$$2\varepsilon_{\mathsf{zk}} + 2\varepsilon_{\mathsf{sse}} + \varepsilon_{\mathsf{ds}} + 2\varepsilon_{\mathsf{ow}}.$$

Therefore Γ is non-frameable. □

4.8.3 Proof of Theorem 13 (Traceability)

Proof strategy. We first move the L_1 CRS to extractable mode. A valid auxiliary-branch signature would forge a signature under the auxiliary verification key. For a standard-branch signature, extraction yields both the encrypted member identity and an issuer certificate for that identity. If the identity was not submitted to the join oracle, this certificate is an issuer-signature forgery; otherwise, correctness of encryption and opening gives an accepted opening to a registered member.

- **Game 0 (Original experiment).** This is the registry-based traceability experiment.
- **Game 1 (Extractable L_1 setup).** Switch the L_1 CRS from real to simulated and then to extractable mode.
- **Game 2 (Exclude the auxiliary branch).** A successful auxiliary-branch signature violates EUF-CMA security of the auxiliary signature key.
- **Game 3 (Extract the L_1 witness).** Extract the encrypted member identity and its issuer certificate. Invalid extraction violates SSE for L_1.
- **Game 4 (Enforce registered membership).** A certificate for an identity absent from the join log violates EUF-CMA security of the issuer signature. Otherwise, the prescribed opening algorithm returns that registered identity.

Proof. Let $\mathcal{A}$ be a PPT adversary in the traceability experiment. We use a sequence of games and write S_i for the event that $\mathcal{A}$ wins Game i.

Game 0: Original experiment. The challenger generates

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma), \qquad (\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta),$$

and, as part of opener key generation,

$$(\mathsf{sk}'_\Sigma, \mathsf{pk}'_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma).$$

It erases sk'_Σ and sets

$$\mathsf{ipk} = \mathsf{pk}_\Sigma, \qquad \mathsf{osk} = \mathsf{sk}_\Theta, \qquad \mathsf{opk} = (\mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma).$$

The adversary receives $(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk})$. In particular, it receives the honest decryption key sk_Θ, but not the erased auxiliary signing key sk'_Σ.

The registry is initialized as $\mathsf{REG} \leftarrow \emptyset$. On a join query y, the challenger computes

$$\sigma_s \leftarrow_\$ \mathsf{Sgn}_\Sigma(\mathsf{sk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y)),$$

adds y to the challenger-maintained log $\mathcal{Q}_{\mathsf{join}}$, and returns $(\sigma_s, \emptyset)$. Thus, the registry supplied to the scheme algorithms remains $\emptyset$ throughout the experiment; the join log is separate state used only in the traceability winning condition.

Eventually, $\mathcal{A}$ outputs (m^*, σ^*_m). If the signature verifies, the challenger computes

$$(\mathsf{upk}^*, O) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{sk}_\Theta, \mathsf{opk}, \emptyset, \mathsf{ipk}, m^*, \sigma^*_m).$$

The event S_0 occurs if the signature verifies but either the resulting opening is rejected or $\mathsf{upk}^* \notin \mathcal{Q}_{\mathsf{join}}$.

Game 1: Extractable setup for L_1. Replace the real setup of crs_{Π}^{1} by

$$(\mathsf{crs}_{\Pi}^{1}, \tau^1, \zeta^1) \leftarrow_{\$} \mathcal{E}_1(1^\lambda).$$

To justify the transition, first replace the real CRS by a CRS generated with SIM_1. The traceability challenger generates no L_1 proofs, so the ZK distinguisher makes no prover-oracle queries. Next replace the simulated CRS by the extractable CRS; their public distributions are identical by setup compatibility. Hence, for a negligible function $\varepsilon_{\mathsf{zk}}$,

$$\left|\Pr[S_1] - \Pr[S_0]\right| \le \varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk}}(1^\lambda)}$

1. Receive crs_{Π}^{1} from the zero-knowledge challenger and embed it as the CRS for L_1.
2. Generate the issuer and opener key material and all remaining public parameters honestly.
3. Run $\mathcal{A}$, answering every join query exactly as in the traceability experiment.
4. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 2: Exclude the auxiliary branch. Game 2 is identical to Game 1, except that it aborts if a successful signature is accepted through the auxiliary branch of Vfy_{Γ}. Let $\mathsf{Bad}_{\mathsf{aux}}$ denote this event.

If $\mathsf{Bad}_{\mathsf{aux}}$ occurs, then the signature parses as

$$\sigma_m^* = (c^*, \pi^*, \sigma'^*, y'^*) \quad \text{with} \quad \sigma'^* \neq \bot \neq y'^*,$$

and

$$\mathsf{Vfy}_{\Sigma}(\mathsf{pk}'_{\Sigma}, M^*_{\mathsf{aux}}, \sigma'^*) = 1,$$

where

$$M^*_{\mathsf{aux}} = (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, \mathsf{pk}'_{\Sigma}, y'^*, m^*, c^*, \pi^*).$$

No algorithm or oracle in the traceability experiment signs a message under pk'_{Σ}. Therefore $(M^*_{\mathsf{aux}}, \sigma'^*)$ is an EUF-CMA forgery. The corresponding reduction embeds the EUF-CMA challenge key as pk'_{Σ} and generates the issuer and encryption keys honestly; it never queries its signing oracle. Consequently, for a negligible function $\varepsilon_{\mathsf{sig,aux}}$,

$$\left|\Pr[S_2] - \Pr[S_1]\right| \le \Pr[\mathsf{Bad}_{\mathsf{aux}}] \le \varepsilon_{\mathsf{sig,aux}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sig,aux}}(1^\lambda)}$

1. Receive the EUF-CMA challenge verification key and set pk'_{Σ} equal to that key.
2. Generate the issuer signature key pair, the encryption key pair, the extractable L_1 setup, and all remaining public parameters honestly.
3. Run $\mathcal{A}$, answering join queries with the locally generated issuer signing key; make no query to the EUF-CMA signing oracle.
4. If $\mathcal{A}$ produces a successful auxiliary-branch signature, output

$$(M^*_{\mathsf{aux}}, \sigma'^*).$$

Otherwise, output an arbitrary non-winning pair.

Whenever S_2 occurs, the signature must instead use the standard branch:

$$\sigma_m^* = (c^*, \pi^*, \bot, \bot).$$

Since Vfy_{Γ} accepts, the proof π^* verifies for

$$x_1^* = (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, \mathsf{pk}'_{\Sigma}, c^*, m^*).$$

Game 3: Extract the L_1 witness. Extract

$$(r^*, \sigma_s^*, w^*) \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^1, \zeta^1, x_1^*, \pi^*).$$

Let Bad_{L_1} be the event that S_2 occurs but the extracted tuple is not a valid witness for x_1^*, and let Game 3 abort on this event.

The challenger has generated no simulated L_1 proof, so the simulation query set in the SSE experiment is empty. Thus (x_1^*, π^*) is fresh, and Bad_{L_1} directly yields an SSE adversary for L_1. For a negligible function $\varepsilon_{\mathsf{sse}}$,

$$\big|\Pr[S_3] - \Pr[S_2]\big| \le \Pr[\mathsf{Bad}_{L_1}] \le \varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse}}(1^\lambda)}$

1. Receive crs_Π^1 from the SSE challenger and embed it as the CRS for L_1.
2. Generate the issuer and opener key material and all remaining public parameters honestly.
3. Run $\mathcal{A}$, answering join queries exactly as in the traceability experiment.
4. If $\mathcal{A}$ produces a successful standard-branch signature, define

$$x_1^* = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c^*, m^*)$$

and output (x_1^*, π^*); otherwise, output an arbitrary rejecting pair.

Whenever S_3 occurs, define $y^* \leftarrow f(w^*)$. The extracted witness satisfies

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y^*; r^*), \tag{6}$$

$$\mathsf{Vfy}_\Sigma\big(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y^*), \sigma_s^*\big) = 1. \tag{7}$$

Game 4: Enforce membership in the join log. Game 4 is identical to Game 3, except that it aborts if

$$y^* \notin \mathcal{Q}_{\mathsf{join}}.$$

Call this event $\mathsf{Bad}_{\mathsf{join}}$. If it occurs, then Equation (7) gives a valid issuer signature on

$$M_{\mathsf{cert}}^* = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y^*).$$

The reduction to EUF-CMA embeds the challenge verification key as pk_Σ. For every join query y, it requests a signature on $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y)$ from its signing oracle, records y in $\mathcal{Q}_{\mathsf{join}}$, and returns $(\sigma_s, \emptyset)$ to $\mathcal{A}$. Because $y^* \notin \mathcal{Q}_{\mathsf{join}}$, the message M_{cert}^* was never queried. Thus $(M_{\mathsf{cert}}^*, \sigma_s^*)$ is an EUF-CMA forgery. For a negligible function $\varepsilon_{\mathsf{sig,cert}}$,

$$\big|\Pr[S_4] - \Pr[S_3]\big| \le \Pr[\mathsf{Bad}_{\mathsf{join}}] \le \varepsilon_{\mathsf{sig,cert}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sig,cert}}(1^\lambda)}$

1. Receive the EUF-CMA challenge verification key and set pk_Σ equal to that key.
2. Generate the encryption key pair, the auxiliary signature key pair, the extractable L_1 setup, and all remaining public parameters.
3. On every join query y, query the EUF-CMA signing oracle on

$$(\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y),$$

record y in $\mathcal{Q}_{\mathsf{join}}$, and return $(\sigma_s, \emptyset)$.
4. Run $\mathcal{A}$ and extract (r^*, σ_s^*, w^*) from a successful standard-branch output as in Game 3; set $y^* = f(w^*)$.
5. If $y^* \notin \mathcal{Q}_{\mathsf{join}}$, output

$$((\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y^*), \sigma_s^*).$$

Otherwise, output an arbitrary non-winning pair.

It remains to analyze Game 4. Suppose that S_4 occurs. By Equation (6) and correctness of Θ for the honestly generated key pair $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$,

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c^*) = y^*.$$

The prescribed opening algorithm receives the unchanged registry $\emptyset$, confirms that the signature uses the standard branch and verifies, decrypts c^* to y^*, and generates an L_2 proof using sk_Θ. Since

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1 \qquad \text{and} \qquad \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c^*) = y^*,$$

correctness of the proof system implies that the resulting opening is accepted by Jdg_Γ. Moreover, the absence of $\mathsf{Bad}_{\mathsf{join}}$ gives

$$\mathsf{upk}^* = y^* \in \mathcal{Q}_{\mathsf{join}}.$$

Thus neither traceability winning condition holds, a contradiction. Hence

$$\Pr[S_4] = 0.$$

Combining the game transitions yields

$$\begin{aligned}\Pr[S_0] \le\ & \varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{sse}}(\lambda) \\ & + \varepsilon_{\mathsf{sig,aux}}(\lambda) + \varepsilon_{\mathsf{sig,cert}}(\lambda),\end{aligned}$$

which is negligible. Equivalently, using a common upper bound for the two signature reductions, the loss is

$$\varepsilon_{\mathsf{zk}} + \varepsilon_{\mathsf{sse}} + 2\varepsilon_{\mathsf{sig}}.$$

Therefore Γ is traceable. □

4.8.4 Proof of Theorem 14 (Anonymity)

Proof strategy. We simulate the L_1 proofs returned by the left-or-right signing oracle and the L_2 proofs returned by the opening oracle. We then answer admissible opening queries by extracting the plaintext certified by their L_1 proofs, which removes the need for the opener secret key. Finally, mcIND-CPA security allows us to replace every left-or-right ciphertext by an encryption of an independently sampled dummy identity. In the resulting game, the complete adaptive view is independent of the anonymity bit.

- **Game 0 (Original experiment).** This is the anonymity experiment with an equivalent implementation of the left-or-right signing oracle.
- **Game 1 (Simulated L_1 proofs).** Generate the L_1 setup in extractable mode and simulate the single L_1 proof contained in each returned left-or-right signature.
- **Game 2 (Simulated L_2 proofs).** Simulate every L_2 proof returned by the opening oracle.
- **Game 3 (Extraction in the opening oracle).** Extract the plaintext certified by each admissible opening query. Invalid extraction violates SSE for L_1.
- **Game 4 (Independent ciphertexts).** Replace all left-or-right ciphertexts by encryptions of independently sampled dummy identities. This is justified by mcIND-CPA security.

Proof. Let $\mathcal{A}$ be a PPT adversary in the anonymity experiment. The challenger generates

$$(w_0, y_0), (w_1, y_1) \leftarrow_{\$} \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma), \qquad b \leftarrow_{\$} \{0,1\},$$

where $y_j = f(w_j)$ for $j \in \{0,1\}$. It also generates the honest opener public key

$$\mathsf{opk} = (\mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma).$$

We give $\mathcal{A}$ exactly the challenge-key information prescribed by the anonymity experiment and use a sequence of games. Let S_i denote the event that $\mathcal{A}$ outputs the hidden bit in Game i.

We first record an equivalent implementation of the left-or-right signing oracle. On a query

$$(m, \mathsf{ipk}, \sigma_s^0, \sigma_s^1),$$

parse $\mathsf{ipk} = \mathsf{pk}_\Sigma$ and, for $j \in \{0,1\}$, test

$$\mathsf{Vfy}_\Sigma\big(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, y_j), \sigma_s^j\big) = 1.$$

If either test fails, return $\perp$. Otherwise, generate only the signature for member b. This does not change the oracle's distribution. Indeed, for an honest member pair and the honestly generated encryption key, a valid certificate supplies a valid L_1 witness and hence yields an accepted group signature by correctness; an invalid certificate makes the witness supplied to the prover invalid, so the total-prover convention makes Sgn_Γ return $\perp$. The unreturned candidate signature therefore affects only the public certificate test.

Game 0: Original experiment. This is the experiment from Figure 8, using the equivalent signing-oracle implementation above. The oracle records every returned challenge signature as

$$(\mathsf{ipk}, m, \sigma_m^b) \in \mathcal{Q}_\sigma.$$

An opening query has the form $(\mathsf{ipk}, \mathsf{REG}, m, \sigma_m)$ and is rejected when this complete triple belongs to $\mathcal{Q}_\sigma$.

Game 1: Simulate the L_1 proofs. Generate the L_1 setup as

$$(\mathsf{crs}_\Pi^1, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

After both certificate tests of a left-or-right query have succeeded, compute

$$(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y_b), \qquad x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m),$$

and replace the real L_1 proof by

$$\pi \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}_\Pi^1, \tau^1, x_1).$$

As in the original experiment, return $\perp$ if the resulting signature fails verification. Otherwise, return

$$\sigma_m = (c, \pi, \perp, \perp)$$

and add $(\mathsf{pk}_\Sigma, m, \sigma_m)$ to $\mathcal{Q}_\sigma$.

The statement x_1 is true with witness (r, σ_s^b, w_b). Zero-knowledge, with polynomially many prover queries, therefore justifies replacing all returned proofs by simulated proofs. We also use the setup-compatibility clause that the joint distribution $(\mathsf{crs}_\Pi^1, \tau^1)$ output by SIM_1 is identical to the first two components output by $\mathcal{E}_1$. Thus, for a negligible function $\varepsilon_{\mathsf{zk},1}$,

$$\big|\Pr[S_1] - \Pr[S_0]\big| \le \varepsilon_{\mathsf{zk},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},1}(1^\lambda)}$

1. Receive crs_Π^1 and access to the zero-knowledge proof oracle for L_1.
2. Generate the honest opener key pair, both challenge member pairs, the remaining public parameters, and $b \leftarrow_\$ \{0,1\}$ honestly.
3. On every successful left-or-right query, sample $(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y_b)$, define

$$x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m),$$

   and query the proof oracle on $(x_1, (r, \sigma_s^b, w_b))$.
4. Apply all verification, $\perp$, logging, and return rules of Game 1, and answer opening queries with the honest opener key subject to the challenge-signature restriction.
5. Let b^* be the bit returned by $\mathcal{A}$. Output 1 if $b^* = b$ and 0 otherwise.

Notice that exactly one L_1 proof is simulated for every successful left-or-right query, namely the proof contained in the returned challenge signature. No proof for the unreturned branch is generated.

Game 2: Simulate the L_2 opening proofs. Generate

$$(\mathsf{crs}^2_{\Pi}, \tau^2) \leftarrow_{\$} \mathsf{SIM}_1(1^\lambda)$$

and modify the opening oracle as follows. On input $(\mathsf{ipk}, \mathsf{REG}, m, \sigma_m)$, first apply the challenge-signature restriction using $(\mathsf{ipk}, m, \sigma_m)$. Then run Opn_Γ up to the generation of its L_2 proof. Thus, the oracle returns $\perp$ on a parsing failure, if $\mathsf{REG} \neq \emptyset$, if the signature uses the auxiliary branch, or if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m) \neq 1$. Otherwise, parse

$$\mathsf{ipk} = \mathsf{pk}_\Sigma, \qquad \sigma_m = (c, \pi, \perp, \perp),$$

compute $y \leftarrow \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c)$, return $\perp$ if $y = \perp$, and define

$$x_2 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m, y),$$

and return (y, π_o), where

$$\pi_o \leftarrow_{\$} \mathsf{SIM}_2(\mathsf{crs}^2_{\Pi}, \tau^2, x_2).$$

If $\pi_o = \perp$, the oracle returns $\perp$, as does Opn_Γ.

Since $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$ is honestly generated and $y = \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c)$, every simulated L_2 proof replaces a proof for a true statement with witness sk_Θ. Zero-knowledge gives a negligible function $\varepsilon_{\mathsf{zk},2}$ such that

$$\big|\Pr[S_2] - \Pr[S_1]\big| \leq \varepsilon_{\mathsf{zk},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},2}(1^\lambda)}$

1. Receive crs^2_{Π} and access to the zero-knowledge proof oracle for L_2.
2. Generate crs^1_{Π} with $\mathcal{E}_1$, generate the honest opener and challenge member pairs, and simulate left-or-right responses as in Game 1.
3. On each admissible opening query, run Opn_Γ up to L_2 proof generation, define

   $$x_2 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m, y),$$

   and query the proof oracle on $(x_2, \mathsf{sk}_\Theta)$.
4. Apply all $\perp$ and return rules of Opn_Γ. Let b^* be the bit returned by $\mathcal{A}$; output 1 if $b^* = b$ and 0 otherwise.

Game 3: Extract in the opening oracle. Game 3 is identical to Game 2, except that an admissible opening query is answered using an extracted L_1 witness. After the challenge restriction and the public checks above have succeeded, define

$$x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m)$$

and extract

$$(r, \sigma_s, w) \leftarrow_{\$} \mathcal{E}_2(\mathsf{crs}^1_{\Pi}, \zeta^1, x_1, \pi).$$

Let $\mathsf{Bad}_{\mathsf{sse}}$ occur if the extracted tuple is not a valid witness for x_1. On this event, the opening oracle returns $\perp$ for the current query. Otherwise, it sets

$$y \leftarrow f(w), \qquad x_2 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m, y),$$

simulates π_o under crs^2_{Π}, and returns (y, π_o).

Every simulated L_1 pair (x_1, π) that verifies is contained in the standard-branch signature returned by the corresponding left-or-right query, and that signature is recorded together with the same pk_Σ and m in $\mathcal{Q}_\sigma$. Consequently, if an opening query containing such

a pair were submitted, the challenge-signature restriction would reject it before extraction. Every pair that reaches extraction is therefore fresh relative to the L_1 simulation oracle.

The SSE reduction need not know the extraction trapdoor. It generates the honest opener key pair itself and answers earlier opening queries as in Game 2, using decryption and the L_2 simulator. For the analysis only, instrument this Game 2 execution with unobserved extractions under the extractable setup. Couple these extractions and all other randomness with Game 3. Before the first invalid extraction, encryption correctness makes the two executions identical, including the adversary's adaptive queries. Consequently, the probability of a first invalid extraction is the same in this instrumented Game 2 and in Game 3.

Let $q_o \geq 1$ bound the total number of opening queries. The reduction chooses a uniform index $j \in \{1, \ldots, q_o\}$, simulates Game 2 up to that query, and outputs its fresh accepting pair (x_1, π) to the SSE challenger. If the query is not reached or fails the public admissibility checks, it outputs a rejecting pair. The challenger, rather than the reduction, performs the extraction. The simulation before query j does not depend on j, and the challenger uses the same extraction distribution as the instrumented execution. Guessing the first failing query therefore succeeds with probability at least $\Pr[\mathsf{Bad}_{\mathsf{sse}}]/q_o$. Thus

$$\Pr[\mathsf{Bad}_{\mathsf{sse}}] \leq q_o \varepsilon_{\mathsf{sse}}(\lambda)$$

for a negligible function $\varepsilon_{\mathsf{sse}}$. Moreover, conditioned on $\neg\mathsf{Bad}_{\mathsf{sse}}$, the extracted witness gives

$$c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w); r).$$

Correctness for the honestly generated encryption key implies

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c) = f(w).$$

Thus Game 2 and Game 3 return the same member public key and simulate an L_2 proof for the same statement. It follows that

$$\left|\Pr[S_3] - \Pr[S_2]\right| \leq q_o \varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse}}(1^\lambda)}$

1. Receive $\mathsf{crs}_{\mathsf{\Pi}}^1$ and access to the SSE simulation oracle for L_1.
2. Guess $j \leftarrow_\$ \{1, \ldots, q_o\}$, generate the L_2 simulation setup, the honest opener and challenge member pairs, the remaining public parameters, and $b \leftarrow_\$ \{0, 1\}$.
3. Answer left-or-right queries as in Game 1, obtaining each returned L_1 proof from the simulation oracle and maintaining $\mathcal{Q}_\sigma$.
4. Answer opening queries before the j-th query as in Game 2 using the honest opener secret key. On the j-th opening query, first apply the challenge restriction and public checks. If they succeed, parse $\sigma_m = (c, \pi, \bot, \bot)$ and define
$$x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m).$$
5. Output (x_1, π) if the selected query passes these checks and the pair is fresh relative to the simulation log. If the selected query is not reached or any check fails, output a rejecting pair.

Game 4: Independent ciphertexts. Modify each successful left-or-right signing query as follows. Sample

$$\widetilde{w} \leftarrow_\$ C_\lambda, \qquad \widetilde{y} \leftarrow f(\widetilde{w}),$$

and replace

$$c \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y_b)$$

by

$$c \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, \widetilde{y}).$$

All subsequent operations remain unchanged. In particular, the oracle forms

$$x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma, c, m),$$

computes

$$\pi \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}^1_\Pi, \tau^1, x_1),$$

performs the prescribed verification check, and returns $\perp$ if that check fails. Only a non-failing response is entered into $\mathcal{Q}_\sigma$.

To justify the transition, consider an mcIND-CPA reduction that embeds the challenge encryption key as pk_Θ, generates the remaining public parameters and both member pairs honestly, and samples $b \leftarrow_\$ \{0,1\}$. For every left-or-right signing query that passes both certificate checks, it samples an independent $\widetilde{y}$ as above and submits

$$(y_b, \widetilde{y})$$

to its mcIND-CPA oracle. It uses the returned ciphertext to form x_1, runs SIM_2, and retains all verification checks and possible $\perp$ responses. Opening queries are answered as in Game 3 using the L_1 extraction trapdoor and the L_2 simulation trapdoor; the reduction therefore does not need the challenge decryption key.

If the mcIND-CPA oracle encrypts the left messages, the reduction simulates Game 3. If it encrypts the right messages, it simulates Game 4. After the ciphertext switch, x_1 need not belong to L_1. This causes no problem: the game invokes SIM_2 directly, and its output, the subsequent verification result, and any resulting $\perp$ response are efficient post-processing of the mcIND-CPA ciphertext. No real prover is invoked and no zero-knowledge claim is made for such a statement.

Let β denote the hidden bit of the mcIND-CPA experiment and let A denote the event $b^* = b$. The reduction outputs 0 if A occurs and 1 otherwise. Consequently,

$$\begin{aligned}\mathrm{Adv}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{B}_{\mathsf{cpa}}}(\lambda) &= \left|\Pr[\beta^* = \beta] - \frac{1}{2}\right| \\ &= \frac{1}{2}\big|\Pr[S_3] - \Pr[S_4]\big|.\end{aligned}$$

Therefore,

$$\big|\Pr[S_4] - \Pr[S_3]\big| \le 2\,\mathrm{Adv}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{B}_{\mathsf{cpa}}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{cpa}}(1^\lambda)}$

1. Receive $(\mathsf{pp}_\Theta, \mathsf{pk}_\Theta)$ and access to the mcIND-CPA left-or-right encryption oracle.
2. Generate crs^1_Π with $\mathcal{E}_1$, generate the L_2 simulation setup, generate the remaining public parameters and both challenge member pairs honestly, and sample $b \leftarrow_\$ \{0,1\}$.
3. On every left-or-right signing query that passes both certificate checks, sample $\widetilde{w} \leftarrow_\$ C_\lambda$, set $\widetilde{y} = f(\widetilde{w})$, and query the encryption oracle on $(y_b, \widetilde{y})$.
4. Use the returned ciphertext to form x_1, simulate its L_1 proof, and apply all verification, $\perp$, logging, and return rules of the game.
5. Answer opening queries as in Game 3 using the L_1 extraction trapdoor and the L_2 simulation trapdoor.
6. Let A be the event that $\mathcal{A}$ outputs b. Output 0 if A occurs and 1 otherwise.

It remains to analyze Game 4. Both certificate candidates are checked before the oracle produces a response, so this check is independent of b. Furthermore, $\widetilde{y}$ and c are sampled independently of b, and SIM_2, verification, the possible $\perp$ response, and the update of $\mathcal{Q}_\sigma$ depend only on these b-independent values. The opening oracle is determined by its submitted query, the resulting challenge log, extraction, the local $\perp$ rule, and fresh simulation randomness, and is therefore independent of b as well. By induction over the

adversary's adaptive queries, its entire view in Game 4 is independent of b. Since the adversary outputs a bit,

$$\Pr[S_4] = \frac{1}{2}.$$

Combining the transitions gives

$$\left|\Pr[S_0] - \frac{1}{2}\right| \leq \varepsilon_{\mathsf{zk},1}(\lambda) + \varepsilon_{\mathsf{zk},2}(\lambda) + q_o\varepsilon_{\mathsf{sse}}(\lambda) + 2\,\mathrm{Adv}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{B}_{\mathsf{cpa}}}(\lambda),$$

which is negligible. Therefore Γ is anonymous. □

4.8.5 Proof of Theorem 15 (Non-Claimability)

Proof strategy. Every protected signing-oracle transcript contains an honestly formed encryption of the honest member public key, even though the surrounding group context is chosen adversarially. We switch the L_2 setup to extractable mode, extract a valid decryption key from the adversary's accepted re-attribution, and use decryption-soundness to show that this key must recover the original member public key rather than the claimed different identity.

- **Game 0 (Original experiment).** Record the encryption randomness used in every protected signing transcript; this instrumentation does not change the adversary's view.
- **Game 1 (Extractable L_2 setup).** Switch the L_2 CRS from real to simulated and then to extractable mode.
- **Game 2 (Extract the opening witness).** Extract a valid decryption key from the accepted re-attribution. Invalid extraction violates SSE for L_2.
- **Game 3 (Enforce correct decryption).** Decryption-soundness forces the extracted key to decrypt the protected ciphertext to the original member public key, contradicting successful re-attribution.

Proof. Let $\mathcal{A}$ be a PPT adversary in the non-claimability experiment. The challenger generates

$$(\mathsf{usk}, \mathsf{upk}) = (w, y) \leftarrow_{\$} \mathsf{KgM}_{\Gamma}(\mathsf{pp}_{\Gamma}),$$

so $y = f(w)$, and gives y to $\mathcal{A}$. We use a sequence of games and let S_i denote the event that $\mathcal{A}$ wins Game i.

Game 0: Original experiment. This is the experiment from Figure 12. On every signing query

$$(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m),$$

the challenger runs Sgn_{Γ}. If the result is $\perp$, it returns $\perp$ without modifying $\mathcal{Q}_\sigma$. Otherwise, it records

$$(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m) \in \mathcal{Q}_\sigma$$

and returns σ_m.

For the analysis, the challenger additionally instruments every such non-failing execution of Sgn_{Γ}. A returned signature has the standard-branch form

$$\mathsf{ipk} = \mathsf{pk}_{\Sigma}, \qquad \mathsf{opk} = (\mathsf{pk}_{\Theta}, \mathsf{pk}'_{\Sigma}), \qquad \sigma_m = (c, \pi, \perp, \perp),$$

and the challenger records in an internal transcript $\mathcal{T}_\sigma$ the randomness r used by the encryption step. Thus the corresponding entry satisfies

$$c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y; r).$$

Every entry in $\mathcal{Q}_\sigma$ therefore has a corresponding entry in $\mathcal{T}_\sigma$. This instrumentation does not change the adversary's view.

Eventually, $\mathcal{A}$ outputs

$$(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, \mathsf{upk}^*, m^*, \sigma_m^*, O^*).$$

For S_0 to occur, the tuple $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*)$ must belong to $\mathcal{Q}_\sigma$, the claimed identity must satisfy $\mathsf{upk}^* \neq y$, and

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{REG}^*, m^*, \sigma_m^*, \mathsf{upk}^*, O^*) = 1.$$

Since Jdg_Γ accepts, σ_m^* cannot equal $\perp$. Because every non-$\perp$ signing-oracle output uses the standard branch, it parses as

$$\mathsf{ipk}^* = \mathsf{pk}_\Sigma^*, \qquad \mathsf{opk}^* = (\mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*), \qquad \sigma_m^* = (c^*, \pi^*, \perp, \perp),$$

and has a matching entry in $\mathcal{T}_\sigma$. Consequently, for some recorded r^*,

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, y; r^*). \tag{8}$$

Game 1: Extractable setup for L_2. Replace the real setup of crs_Π^2 by

$$(\mathsf{crs}_\Pi^2, \tau^2, \zeta^2) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

To justify the transition, first replace the real CRS by one generated with SIM_1. The challenger generates no L_2 proof in the non-claimability experiment, so the ZK distinguisher makes no prover-oracle query. The simulated and extractable public CRS distributions are identical by setup compatibility. Hence, for a negligible function $\varepsilon_{\mathsf{zk}}$,

$$\big|\Pr[S_1] - \Pr[S_0]\big| \leq \varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk}}(1^\lambda)}$

1. Receive crs_Π^2 from the zero-knowledge challenger and embed it as the CRS for L_2.
2. Generate the honest target member pair and all remaining public parameters as in the non-claimability experiment.
3. Run $\mathcal{A}$, answer signing queries honestly, and maintain $\mathcal{Q}_\sigma$ and the internal transcript $\mathcal{T}_\sigma$.
4. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 2: Extract the opening witness. After a successful adversarial output, define

$$x_2^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*, m^*, \mathsf{upk}^*)$$

and extract

$$\mathsf{sk}_\Theta^* \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^2, \zeta^2, x_2^*, O^*).$$

Let Bad_{L_2} be the event that S_1 occurs but sk_Θ^* is not a valid witness for x_2^*, and let Game 2 abort on this event.

Acceptance by Jdg_Γ implies

$$\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi^2, x_2^*, O^*) = 1.$$

Moreover, the challenger has generated or simulated no L_2 proof. Hence the SSE simulation log is empty and (x_2^*, O^*) is fresh. Therefore Bad_{L_2} directly yields an SSE adversary for L_2. For a negligible function $\varepsilon_{\mathsf{sse}}$,

$$\big|\Pr[S_2] - \Pr[S_1]\big| \leq \Pr[\mathsf{Bad}_{L_2}] \leq \varepsilon_{\mathsf{sse}}(\lambda).$$

$\mathcal{B}_{\mathsf{sse}}(1^\lambda)$

1. Receive crs_Π^2 from the SSE challenger and embed it as the CRS for L_2.
2. Generate the honest target member pair and all remaining public parameters as in Game 1.
3. Run $\mathcal{A}$, answer signing queries honestly, and maintain $\mathcal{Q}_\sigma$ and $\mathcal{T}_\sigma$.
4. If $\mathcal{A}$ produces a successful re-attribution, define

$$x_2^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, \mathsf{pk}_\Sigma'^*, c^*, m^*, \mathsf{upk}^*)$$

and output (x_2^*, O^*); otherwise, output an arbitrary rejecting pair.

Whenever S_2 occurs, extraction yields

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*) = 1, \tag{9}$$

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = \mathsf{upk}^*. \tag{10}$$

Game 3: Decryption-soundness. Game 3 is identical to Game 2, except that it aborts if

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) \neq y.$$

By (8), the ciphertext is an honestly formed encryption of y under the adversarially supplied public key pk_Θ^*, using the recorded randomness r^*. By (9), the extracted key sk_Θ^* passes verification for the same public key. Hence the abort event produces

$$(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*, y, r^*)$$

that wins the decryption-soundness experiment.

The reduction receives pp_Θ from the decryption-soundness challenger, embeds it into pp_Γ, generates the L_2 setup with $\mathcal{E}_1$, and runs $\mathcal{A}$ while maintaining $\mathcal{Q}_\sigma$ and $\mathcal{T}_\sigma$. It extracts sk_Θ^* from the final accepted opening and obtains r^* from the matching transcript entry. Thus, for a negligible function $\varepsilon_{\mathsf{ds}}$,

$$\left|\Pr[S_3] - \Pr[S_2]\right| \leq \varepsilon_{\mathsf{ds}}(\lambda).$$

$\mathcal{B}_{\mathsf{ds}}(1^\lambda)$

1. Receive pp_Θ from the decryption-soundness challenger and embed it into pp_Γ.
2. Generate the L_2 setup with $\mathcal{E}_1$, retain ζ^2, and generate the honest target member pair and all remaining parameters.
3. Run $\mathcal{A}$, answering signing queries while maintaining $\mathcal{Q}_\sigma$ and $\mathcal{T}_\sigma$.
4. From a successful final output, extract sk_Θ^* from O^* and obtain the recorded randomness r^* from the matching transcript entry.
5. If all checks succeed, output

$$(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*, y, r^*);$$

otherwise, output an arbitrary non-winning tuple.

Finally, if S_3 occurred, Equation (10) and the absence of the decryption-soundness abort would imply

$$\mathsf{upk}^* = \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = y = \mathsf{upk},$$

contradicting the winning condition $\mathsf{upk}^* \neq \mathsf{upk}$. Hence

$$\Pr[S_3] = 0.$$

Combining the game transitions gives

$$\Pr[S_0] \leq \varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{sse}}(\lambda) + \varepsilon_{\mathsf{ds}}(\lambda),$$

which is negligible. Therefore Γ is non-claimable. □

5 Registry-Less Non-Interactive Group Signatures

We now define *registry-less non-interactive group signatures*. In the remainder of the paper, we simply refer to them as group signatures.

Our syntax follows the algorithmic interface of BSZ-style group signatures, but removes the issuer-maintained registry and any other shared state between issuer and opener. For fixed public parameters pp_Γ, a group is therefore determined solely by the public key pair $(\mathsf{ipk}, \mathsf{opk})$.

In our model, member identities are represented directly by their public keys. The issuer produces membership certificates, members produce group signatures, and the opener must open using only its secret key together with public inputs. There is no shared registration structure in the definition.

We also distinguish explicitly between *membership certificates* σ_s and *group signatures* σ_m. This distinction is essential for the additional unforgeability notions introduced below.

5.1 Framework

Definition 23 (Group Signature Scheme)**.** A group signature scheme Γ is a tuple of PPT algorithms

$$(\mathsf{PG}_\Gamma, \mathsf{KgM}_\Gamma, \mathsf{KgI}_\Gamma, \mathsf{KgO}_\Gamma, \mathsf{Join}_\Gamma, \mathsf{Sgn}_\Gamma, \mathsf{Vfy}_\Gamma, \mathsf{Opn}_\Gamma, \mathsf{Jdg}_\Gamma),$$

with the following interfaces, where σ_s denotes a membership certificate and σ_m denotes a group signature.

PG_Γ**.** On input 1^λ, output public parameters pp_Γ:

$$\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda).$$

> We display pp_Γ explicitly as input to the key-generation algorithms and treat it as an implicit input to all remaining algorithms.

KgM_Γ**.** On input pp_Γ, output a member secret/public key pair $(\mathsf{usk}, \mathsf{upk})$:

$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma).$$

KgI_Γ**.** On input pp_Γ, output issuer keys $(\mathsf{isk}, \mathsf{ipk})$:

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma).$$

KgO_Γ**.** On input pp_Γ, output opener keys $(\mathsf{osk}, \mathsf{opk})$:

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma).$$

Join_Γ**.** On input $(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk})$, output a membership certificate σ_s:

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk}).$$

Sgn_Γ**.** On input $(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$ with $m \in \mathcal{MS}$, output a group signature σ_m:

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m).$$

Vfy_Γ**.** On input $(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)$, output a decision bit:

$$d \leftarrow \mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m).$$

Opn_Γ**.** On input $(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, m, \sigma_m)$, output a member public key upk and opening information O:

$$(\mathsf{upk}, O) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, m, \sigma_m).$$

Jdg_Γ**.** On input $(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m, \mathsf{upk}, O)$, output a decision bit:

$$d \leftarrow \mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m, \mathsf{upk}, O).$$

Definition 24 (Correctness)**.** A group signature scheme Γ is correct if for all $\lambda \in \mathbb{N}$, all $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$, all issuer keys $(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$, all opener keys $(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$, all member keys $(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$, all certificates

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk}),$$

all messages $m \in \mathcal{MS}$, and all signatures

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m),$$

we require $\sigma_s \neq \bot$ and $\sigma_m \neq \bot$, and we have

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m) = 1.$$

Moreover, for every

$$z \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, m, \sigma_m),$$

it holds that $z \neq \bot$ and, writing $z = (\mathsf{upk}', O)$,

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m, \mathsf{upk}', O) = 1 \qquad \text{and} \qquad \mathsf{upk}' = \mathsf{upk}.$$

Our interface does not require the issuer to keep state. In particular, multiple certificates for the same upk may exist. All security notions are defined without relying on any global registration state.

Security experiments may maintain internal query logs in order to define issuance and freshness conditions. These logs are maintained solely by the challenger: they are neither inputs to the scheme algorithms nor shared registration state.

5.2 Security

In our setting, we require the classical properties *anonymity*, *traceability*, *non-frameability*, *trace-soundness*, and *non-claimability*, but now without any registration list or issuer-maintained state. In addition, we introduce three further unforgeability properties that make membership soundness explicit.

These notions are intentionally phrased at the level of accepted algorithmic behavior. In particular, traceability requires that every valid signature can be opened by the prescribed opening algorithm, trace-soundness enforces uniqueness of openings, even across distinct group public-key pairs, and non-claimability protects attribution of honestly generated transcripts even when the honest signer's secret key is exposed. The additional unforgeability notions then capture the membership semantics that are no longer induced by a shared registry.

Informally, the classical properties require:

- anonymity hides the signer identity against a malicious issuer and exposed challenge-member signing keys, but not against a corrupt opener;
- traceability requires that every valid signature be openable by the prescribed opening algorithm, even under a malicious issuer and leaked opener secret key;

- non-frameability prevents false attribution to an honest member even if issuer and opener are malicious;
- trace-soundness requires uniqueness of accepted attribution tuples, even across distinct group public-key pairs; and
- non-claimability prevents re-attribution of an honestly generated transcript, even when the honest signer's secret key is known to the adversary.

The three additional unforgeability properties will be introduced below. They explicitly capture the relation between accepted signatures, membership certificates, and issuer-authorized membership.

5.2.1 Anonymity

Anonymity is defined against a malicious issuer and honestly generated but key-exposed challenge members, while the opener remains honest. The challenger generates an honest opener key pair and two honest member key pairs and gives both member secret keys to the adversary. Thus, anonymity does not rely on the secrecy of the challenge members' keys. We restrict the adversary to output a bit $b^* \in \{0,1\}$.

The adversary chooses the issuer public key and the two membership certificates for every left-or-right signing query. The experiment imposes no external consistency condition between ipk, upk_i, and σ_s^i. Any required consistency must therefore be enforced by the combined behavior of Sgn_Γ and Vfy_Γ.

For each query, the oracle generates and verifies both candidate signatures before returning the one selected by the fixed challenge bit. The opening oracle refuses to open exact challenge transcripts recorded in $\mathcal{Q}_\sigma$, but remains available on all other inputs, including modified challenge signatures.

Anonymity against a corrupt opener is incompatible with correctness whenever the two challenge public keys are distinct: the opener can run Opn_Γ on the challenge signature and compare the recovered public key with upk_0 and upk_1.

Definition 25 (Anonymity)**.** A group signature scheme Γ satisfies *anonymity* if for every PPT adversary $\mathcal{A}$ whose output lies in $\{0,1\}$ there exists a negligible function negl such that

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{Anon}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] - 1/2\right| \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 13.

5.2.2 Traceability

Traceability is formulated here as a deliberately *functional* notion: every verifying signature under a group public-key pair $(\mathsf{ipk}, \mathsf{opk})$ must admit a valid opening under the prescribed opening algorithm. In particular, this definition does *not* by itself assert that the opened identity corresponds to any honest join-derived membership relation. That missing semantic dimension is captured separately by the unforgeability notions introduced later.

Figure 14 formalizes this requirement in the presence of a malicious issuer. The challenger generates an honest opener key pair and gives the adversary the opener secret key, so the opening key is leaked even though the opening algorithm itself remains the honest one. The adversary then chooses ipk^*, m^*, and σ_m^*. It wins if σ_m^* verifies but the prescribed opening procedure fails to yield a valid opening.

The definition only requires that the prescribed opening algorithm return an opening accepted by Jdg_Γ, without imposing any further semantic constraints on the returned member public key. No additional oracles are provided, as traceability only concerns the behavior of the opening algorithm on adversarially generated inputs.

$\mathbf{Exp}^{\mathsf{Anon}}_{\mathcal{A},\Gamma}(\lambda)$:
$\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
$(\mathsf{osk},\mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$
$(\mathsf{usk}_0,\mathsf{upk}_0) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
$(\mathsf{usk}_1,\mathsf{upk}_1) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
$b \leftarrow_\$ \{0,1\}$
$\mathcal{Q}_\sigma \leftarrow \emptyset$
$b^* \leftarrow_\$ \mathcal{A}^{\mathcal{O}^{b,\mathsf{usk}_0,\mathsf{usk}_1}_{\mathsf{LoRSign}},\mathcal{O}^{\mathsf{osk}}_{\mathsf{Opn}'_\Gamma}}(\mathsf{pp}_\Gamma,\mathsf{opk},(\mathsf{upk}_0,\mathsf{usk}_0),(\mathsf{upk}_1,\mathsf{usk}_1))$
where oracle $\mathcal{O}^{b,\mathsf{usk}_0,\mathsf{usk}_1}_{\mathsf{LoRSign}}(m,\mathsf{ipk},\sigma^0_s,\sigma^1_s)$:
$\sigma^0_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}_0,\mathsf{upk}_0,\sigma^0_s,\mathsf{ipk},\mathsf{opk},m)$
$\sigma^1_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}_1,\mathsf{upk}_1,\sigma^1_s,\mathsf{ipk},\mathsf{opk},m)$
return $\bot$ if $\mathsf{Vfy}_\Gamma(\mathsf{ipk},\mathsf{opk},m,\sigma^0_m) \neq 1$
return $\bot$ if $\mathsf{Vfy}_\Gamma(\mathsf{ipk},\mathsf{opk},m,\sigma^1_m) \neq 1$
$\mathcal{Q}_\sigma \leftarrow \mathcal{Q}_\sigma \cup \{(\sigma^b_m,m,\mathsf{ipk})\}$
return σ^b_m
where oracle $\mathcal{O}^{\mathsf{osk}}_{\mathsf{Opn}'_\Gamma}(\mathsf{ipk},m,\sigma_m)$:
return $\bot$ if $(\sigma_m,m,\mathsf{ipk}) \in \mathcal{Q}_\sigma$
return $\mathsf{Opn}_\Gamma(\mathsf{osk},\mathsf{opk},\mathsf{ipk},m,\sigma_m)$
return 1 if $b^* = b$; else 0

Figure 13: Anonymity experiment for group signature scheme Γ.

$\mathbf{Exp}^{\mathsf{Traceability}}_{\mathcal{A},\Gamma}(\lambda)$:
$\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
$(\mathsf{osk},\mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$
$(\mathsf{ipk}^*,m^*,\sigma^*_m) \leftarrow_\$ \mathcal{A}(\mathsf{pp}_\Gamma,(\mathsf{opk},\mathsf{osk}))$
return 0 if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}^*,\mathsf{opk},m^*,\sigma^*_m) \neq 1$
$z^* \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk},\mathsf{opk},\mathsf{ipk}^*,m^*,\sigma^*_m)$
return 1 if $z^* = \bot$
return 1 if z^* does not parse as (upk^*,O^*)
return 1 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*,\mathsf{opk},m^*,\sigma^*_m,\mathsf{upk}^*,O^*) \neq 1$
return 0

Figure 14: Traceability experiment for group signature scheme Γ.

Definition 26 (Traceability)**.** A group signature scheme Γ satisfies *traceability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{Traceability}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 14.

5.2.3 Non-Frameability

Non-frameability protects an honest member against false attribution. Figure 15 fixes one honestly generated member key pair $(\mathsf{usk},\mathsf{upk})$. The adversary chooses arbitrary issuer

and opener public keys, arbitrary certificates, and may query a signing oracle under these adversarially chosen contexts.

The adversary wins if it outputs a fresh transcript together with opening information that causes Jdg_Γ to accept an attribution to the honest member public key upk. Hence, even under malicious issuer and opener contexts, a fixed honest member must not be falsely implicated.

The adversary is not required to provide an opening produced by Opn_Γ, but only one accepted by Jdg_Γ. We do not impose a separate condition $\mathsf{Vfy}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*) = 1$, since Jdg_Γ is the attribution predicate in this notion. Any required consistency and validity checks must therefore be enforced by Jdg_Γ itself.

$\mathbf{Exp}^{\mathsf{NonFrame}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
  $\mathcal{Q}_\sigma \leftarrow \emptyset$
  $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, O^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}^{\mathsf{usk}}_{\mathsf{Sgn}_\Gamma}}(\mathsf{pp}_\Gamma, \mathsf{upk})$
    where oracle $\mathcal{O}^{\mathsf{usk}}_{\mathsf{Sgn}_\Gamma}(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$:
      $\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$
      return $\perp$ if $\sigma_m = \perp$
      $\mathcal{Q}_\sigma \leftarrow \mathcal{Q}_\sigma \cup \{(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)\}$
      return σ_m
  return 0 if $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*) \in \mathcal{Q}_\sigma$
  return 1 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, \mathsf{upk}, O^*) = 1$
  return 0

Figure 15: Non-frameability experiment for group signature scheme Γ.

Definition 27 (Non-Frameability)**.** A group signature scheme Γ satisfies *non-frameability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{NonFrame}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 15.

5.2.4 Trace-Soundness

Trace-soundness is intentionally strengthened beyond the local classical form. We require uniqueness not merely of the opened member under a fixed group context, but of the entire accepted attribution tuple. Computationally, no efficient adversary should be able to find two distinct tuples

$$(\mathsf{upk}, \mathsf{ipk}, \mathsf{opk})$$

for the same message-signature pair such that both corresponding openings are accepted by Jdg_Γ. Thus, the notion explicitly rules out cross-context equivocation.

Figure 16 formalizes this strengthened requirement. The adversary outputs one message-signature pair together with two candidate attributions under possibly different group public-key pairs. It wins if both attributions are accepted by Jdg_Γ, but

$$(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) \neq (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*).$$

The opening-information strings O_0^* and O_1^* need not be unique; only the accepted attribution tuple is required to be unique.

Trace-soundness is defined directly at the level of the accepted attribution relation induced by Jdg_Γ. No separate Vfy_Γ condition is imposed; any required transcript validity and consistency checks must be enforced by Jdg_Γ.

Member public keys may occur in multiple group instances. Accordingly, an attribution is identified by the complete tuple

$$(\mathsf{upk}, \mathsf{ipk}, \mathsf{opk}),$$

rather than by upk alone.

$\mathbf{Exp}^{\mathsf{TraceSound}}_{\mathcal{A},\Gamma}(\lambda)$:
$\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
$z^* \leftarrow_\$ \mathcal{A}(\mathsf{pp}_\Gamma)$
return 0 if z^* does not parse as
$(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*)$
return 0 if $(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) = (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*)$
return 0 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*) \neq 1$
return 0 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_1^*, O_1^*) \neq 1$
return 1

Figure 16: Trace-soundness experiment for group signature scheme Γ.

Definition 28 (Trace-Soundness)**.** A group signature scheme Γ satisfies *trace-soundness* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{TraceSound}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 16.

5.2.5 Non-Claimability

Non-claimability protects outputs of the honest signing algorithm against re-attribution. Unlike non-frameability, the protected transcript is produced by an honest execution of Sgn_Γ using a fixed honestly generated member key pair. The surrounding issuer key, opener key, and membership certificate may nevertheless be adversarially chosen.

The adversary is additionally given the honest member secret key. This strengthening is intentional: the property does not rely on secrecy of the member secret key, but protects the attribution of concrete transcripts previously returned by the signing oracle. Since the adversary already knows usk, the signing oracle does not provide an otherwise unavailable signing capability. Its purpose is to designate the concrete honestly generated transcripts protected by the experiment.

Figure 17 formalizes this setting. The challenger generates an honest member key pair $(\mathsf{usk}, \mathsf{upk})$, gives both keys to the adversary, and provides a signing oracle. The adversary wins if it selects a transcript previously returned by that oracle and supplies opening information that is accepted as an attribution to a different member public key.

A protected transcript consists of the issuer public key, opener public key, message, and group signature string. Consequently, non-claimability prevents reassignment within the original group context. Cross-context uniqueness of accepted attributions is addressed

separately by strengthened trace-soundness. In particular, whenever a protected transcript has an accepted attribution in its original context, trace-soundness rules out a distinct accepted attribution under another group context.

No separate Vfy_Γ condition is imposed, since Jdg_Γ is the attribution predicate and must enforce any required transcript validity and consistency checks.

$\mathbf{Exp}^{\mathsf{NonClaimability}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$
  $\mathcal{Q}_\sigma \leftarrow \emptyset$
  $z^* \leftarrow_\$ \mathcal{A}^{\mathcal{O}^{\mathsf{usk}}_{\mathsf{Sgn}_\Gamma}}(\mathsf{pp}_\Gamma, \mathsf{usk}, \mathsf{upk})$
    where oracle $\mathcal{O}^{\mathsf{usk}}_{\mathsf{Sgn}_\Gamma}(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$:
      $\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$
      return $\perp$ if $\sigma_m = \perp$
      $\mathcal{Q}_\sigma \leftarrow \mathcal{Q}_\sigma \cup \{(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)\}$
      return σ_m
  return 0 if z^* does not parse as
    $(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{upk}^*, m^*, \sigma^*_m, O^*)$
  return 0 if $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m) \notin \mathcal{Q}_\sigma$
  return 0 if $\mathsf{upk}^* = \mathsf{upk}$
  return 1 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m, \mathsf{upk}^*, O^*) = 1$
  return 0

Figure 17: Non-claimability experiment for group signature scheme Γ.

Definition 29 (Non-Claimability)**.** A group signature scheme Γ satisfies *non-claimability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathbf{Exp}^{\mathsf{NonClaimability}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 17.

5.2.6 Certificate-Unforgeability

The next three notions formalize complementary aspects of membership soundness under an honest issuer. They deliberately separate certificate issuance, member-key ownership, and accepted attribution.

The first two notions use replay-based freshness. An accepting signature is fresh unless the complete relevant transcript was previously returned by the corresponding signing oracle. Opening-unforgeability instead uses relation-based freshness: an attribution is fresh if the corresponding member–opener pair was not submitted to the honest join oracle.

Certificate-unforgeability exposes honestly generated member secret keys while withholding the corresponding membership certificates behind opaque handles. It requires that the adversary cannot produce a fresh verifying group signature under the honest issuer key unless that exact signature was returned by the signing oracle using a certificate generated by the honest join oracle.

The adversary may choose arbitrary opener public keys and is given oracle access to member generation, join under the honest issuer, and signing. The member generation oracle returns the complete member key pair. The join oracle generates and internally

stores a membership certificate and returns only an opaque certificate handle. The signing oracle is restricted to these internally stored certificate instances.

The adversary succeeds if it outputs a signature σ_m^*, message m^*, and opener public key opk^* such that

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}^*, m^*, \sigma_m^*) = 1,$$

and

$$(\sigma_m^*, m^*, \mathsf{opk}^*) \notin \mathcal{Q}_\sigma.$$

Intuition. Certificate-unforgeability rules out signatures that bypass honest certificate issuance even when the corresponding member secret keys are exposed. Since freshness is defined by exact transcript replay, the notion also rules out public rerandomization of signing-oracle outputs. It does not yet ensure that a certificate cannot be transferred to a different member secret key; that binding is captured separately by key-unforgeability.

Definition 30 (Certificate-Unforgeability)**.** A group signature scheme Γ satisfies *certificate-unforgeability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}_{\mathcal{A},\Gamma}^{\mathsf{CertUnforgeability}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 18a.

5.2.7 Key-Unforgeability

Key-unforgeability exposes membership certificates while withholding the challenger-generated member secret keys. It requires that the adversary cannot produce a fresh verifying group signature under the honest issuer key unless that exact signature was returned by the signing oracle using one of the challenger-generated member secret keys.

The adversary interacts with an honest issuer and may choose arbitrary opener public keys. It has access to member generation, join, and signing oracles. The member-generation oracle returns only the member index and public key. The join oracle returns membership certificates generated by the honest issuer.

In contrast to certificate-unforgeability, the signing oracle is not gated by certificate handles. Instead, the adversary selects a challenger-generated member key through its index j and supplies an arbitrary certificate σ_s, issuer public key ipk', opener public key opk, and message m. The challenger then executes Sgn_Γ using the secret key associated with j.

The adversary succeeds if it outputs a signature σ_m^*, message m^*, and opener public key opk^* such that

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}^*, m^*, \sigma_m^*) = 1,$$

and

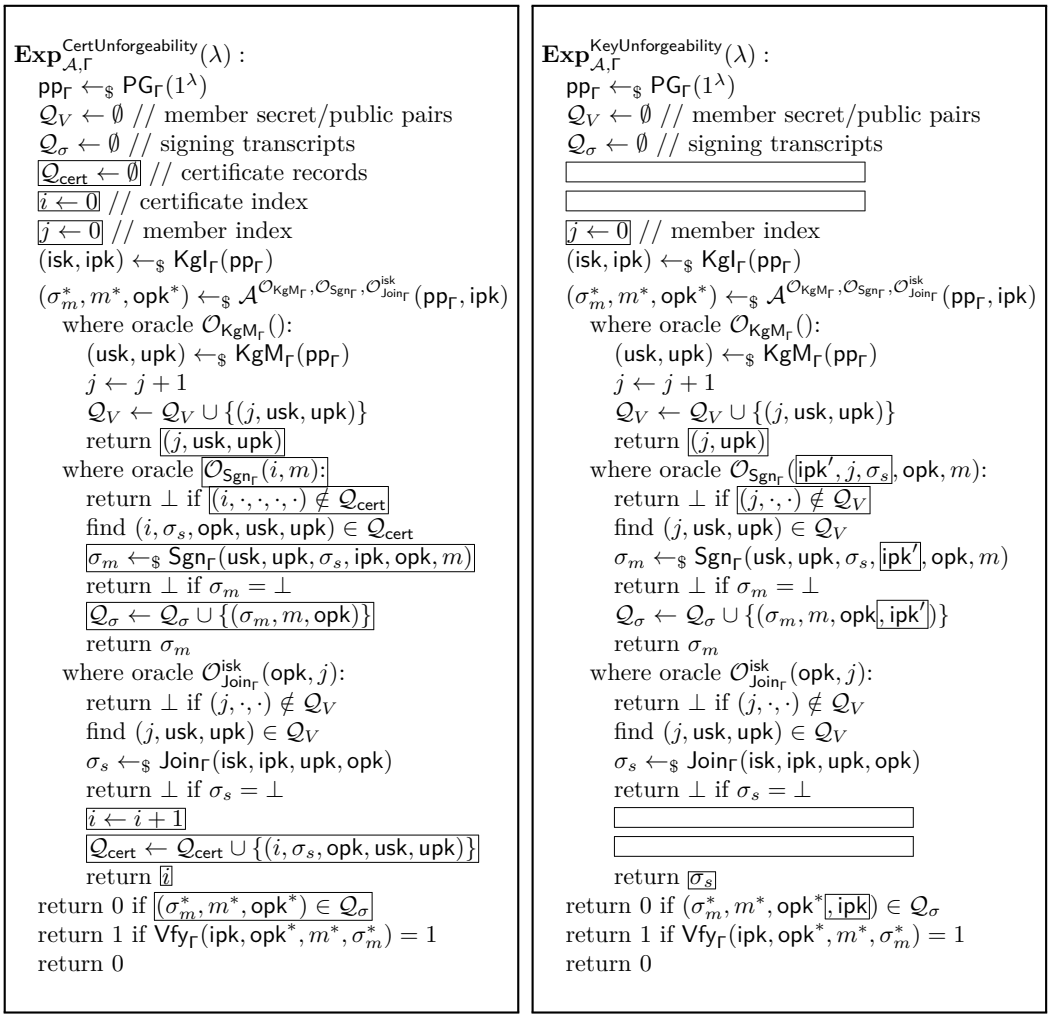
$$(\sigma_m^*, m^*, \mathsf{opk}^*, \mathsf{ipk}) \notin \mathcal{Q}_\sigma.$$

Intuition. Where certificate-unforgeability excludes fresh signatures generated without access to an honest certificate instance, key-unforgeability prevents certificates issued for challenger-generated member public keys from being transferred into fresh valid signatures generated with a different, adversary-controlled member secret key. Since freshness is defined by exact transcript replay, the notion also rules out public rerandomization of signing-oracle outputs.

Definition 31 (Key-Unforgeability)**.** A group signature scheme Γ satisfies *key-unforgeability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}_{\mathcal{A},\Gamma}^{\mathsf{KeyUnforgeability}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 18b.



(a) Certificate-Unforgeability of group signature Γ.

(b) Key-Unforgeability of group signature Γ.

Figure 18: Side-by-side view of certificate-unforgeability and key-unforgeability. In the certificate-unforgeability experiment, the join oracle stores certificates internally and returns opaque certificate handles; the signing oracle can use only these stored certificate instances. In the key-unforgeability experiment, the adversary receives certificates from the join oracle and supplies an arbitrary certificate to the signing oracle, while the member secret key is selected through a challenger-maintained member index. The key-unforgeability replay filter additionally records the issuer public key because its signing oracle accepts adversarially selected issuer contexts. In the certificate-unforgeability experiment, the issuer key is fixed and need not be stored explicitly. Empty boxes indicate components present in the certificate-unforgeability experiment but absent from the key-unforgeability experiment.

5.2.8 Opening-Unforgeability

Opening-unforgeability is the attribution-side counterpart of certificate-unforgeability and key-unforgeability. It requires that no efficient adversary can produce an accepted attribution under the honest issuer key to a member–opener pair for which the honest issuer never successfully issued a membership certificate.

$\mathbf{Exp}^{\mathsf{OpeningUnforgeability}}_{\mathcal{A},\Gamma}(\lambda)$:
  $\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda)$
  $(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$
  $\mathcal{R}_{\mathsf{join}} \leftarrow \emptyset$
  $(\mathsf{upk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, O^*) \leftarrow_\$ \mathcal{A}^{\mathcal{O}^{\mathsf{isk}}_{\mathsf{Join}_\Gamma}}(\mathsf{pp}_\Gamma, \mathsf{ipk})$
    where oracle $\mathcal{O}^{\mathsf{isk}}_{\mathsf{Join}_\Gamma}(\mathsf{upk}, \mathsf{opk})$:
      $\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk})$
      return $\perp$ if $\sigma_s = \perp$
      $\mathcal{R}_{\mathsf{join}} \leftarrow \mathcal{R}_{\mathsf{join}} \cup \{(\mathsf{upk}, \mathsf{opk})\}$
      return σ_s
  return 0 if $(\mathsf{upk}^*, \mathsf{opk}^*) \in \mathcal{R}_{\mathsf{join}}$
  return 0 if $\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}^*, m^*, \sigma_m^*, \mathsf{upk}^*, O^*) \neq 1$
  return 1

Figure 19: Opening-unforgeability experiment for group signature scheme Γ.

Unlike the two signature-unforgeability notions, opening-unforgeability does not use replay-based transcript freshness. Instead, freshness is defined by the join relation maintained by the challenger. Whenever the honest join oracle successfully issues a certificate for a pair

$$(\mathsf{upk}, \mathsf{opk}),$$

the challenger records that pair in $\mathcal{R}_{\mathsf{join}}$.

The adversary may request certificates for arbitrary member–opener pairs. No signing oracle is required: it may locally generate member and opener key pairs, request certificates for the corresponding public keys, and execute Sgn_Γ itself.

The adversary wins if it outputs

$$(\mathsf{upk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, O^*)$$

such that

$$(\mathsf{upk}^*, \mathsf{opk}^*) \notin \mathcal{R}_{\mathsf{join}},$$

but

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}^*, m^*, \sigma_m^*, \mathsf{upk}^*, O^*) = 1.$$

Intuition. Opening-unforgeability rules out phantom attributions to member–opener pairs that were never certified by the honest issuer. It is orthogonal to non-frameability, which protects a fixed honest member against false attribution, and complements trace-soundness, which enforces uniqueness but not membership legitimacy of accepted attributions.

No separate Vfy_Γ condition is imposed, since Jdg_Γ is the attribution predicate and must enforce any required transcript validity and consistency checks.

Definition 32 (Opening-Unforgeability)**.** A group signature scheme Γ satisfies *opening-unforgeability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathbf{Exp}^{\mathsf{OpeningUnforgeability}}_{\mathcal{A},\Gamma}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

The corresponding experiment is given in Figure 19.

5.3 Secure Construction

We now present a canonical construction satisfying all eight properties under the building-block assumptions stated in Section 2. It is obtained from the contrived construction by removing the auxiliary bypass entirely. The resulting scheme follows the standard BSZ intuition: the issuer certifies a member identity, the signer proves knowledge of a valid certificate while encrypting its identity to the opener, and the opener proves correct decryption.

Let

$$\mathcal{F} = \{f_\lambda : C_\lambda \to D_\lambda\}_{\lambda \in \mathbb{N}}$$

be a fixed one-way function family. We assume that the description of f_λ is efficiently obtainable from 1^λ and that

$$D_\lambda \subseteq \mathcal{M}_{\Theta,\lambda}.$$

5.3.1 Languages

In the relations below, $w \in C_\lambda$, $c \neq \bot$, and the claimed plaintext in L_2 satisfies $y \neq \bot$. Encryption randomness must be a valid random tape. These are part of the relation's domain checks, not additional hardness assumptions. In particular, two failing algorithm evaluations do not establish an encryption or decryption relation merely because both return $\bot$.

As before, we include the full signature context in the statement, in particular the message m, so that proofs are bound to the complete verification context.

$$\begin{aligned} L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta} := \big\{ (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m) \;\big|\; & \exists (r, \sigma_s, w) : \\ & \mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, f(w)), \sigma_s) = 1 \\ & \wedge\ c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w); r) \big\} \end{aligned}$$

$$\begin{aligned} L_2^{\mathsf{pp}_\Theta} := \big\{ (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m, y) \;\big|\; & \exists \mathsf{sk}_\Theta : \\ & \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1 \\ & \wedge\ \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c) = y \big\}. \end{aligned}$$

5.3.2 Construction

Construction 4 gives the scheme.

Theorem 16. *Γ is correct.*

Proof. Let (w, y) be an honestly generated member key pair. Then $y = f(w)$. Correctness of Σ implies that the certificate returned by Join_Γ satisfies

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, f(w)), \sigma_s) = 1.$$

Moreover, Sgn_Γ computes $c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w); r)$ for the sampled randomness r. Hence (r, σ_s, w) is a valid witness for the resulting statement in $L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta}$, and correctness of the proof system implies that Vfy_Γ accepts.

For honestly generated opener keys, correctness of Θ gives

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c) = f(w) = y \qquad \text{and} \qquad \mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1.$$

Thus sk_Θ is a valid witness for the statement in $L_2^{\mathsf{pp}_\Theta}$ generated by Opn_Γ. Correctness of the proof system implies that Jdg_Γ accepts the resulting opening, and the opened member public key is exactly $y = \mathsf{upk}$. □

$\mathsf{PG}_\Gamma(1^\lambda)$:

- Let $f \leftarrow f_\lambda$ for the fixed one-way function family $\mathcal{F} = \{f_\lambda : C_\lambda \to D_\lambda\}_{\lambda\in\mathbb{N}}$,
- $\mathsf{pp}_\Sigma \leftarrow_\$ \mathsf{PP}_\Sigma(1^\lambda)$,
- $\mathsf{pp}_\Theta \leftarrow_\$ \mathsf{PP}_\Theta(1^\lambda)$,
- $\mathsf{crs}^1_\Pi \leftarrow_\$ \mathsf{PG}_\Pi(1^\lambda)$ for $L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta}$,
- $\mathsf{crs}^2_\Pi \leftarrow_\$ \mathsf{PG}_\Pi(1^\lambda)$ for $L_2^{\mathsf{pp}_\Theta}$,
- Return $\mathsf{pp}_\Gamma = (\mathsf{crs}^1_\Pi, \mathsf{crs}^2_\Pi, \mathsf{pp}_\Sigma, \mathsf{pp}_\Theta, f)$.

$\mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$:

- Parse $\mathsf{pp}_\Gamma = (\mathsf{crs}^1_\Pi, \mathsf{crs}^2_\Pi, \mathsf{pp}_\Sigma, \mathsf{pp}_\Theta, f)$,
- Sample $w \leftarrow_\$ C_\lambda$,
- Set $y \leftarrow f(w)$,
- Return $(\mathsf{usk}, \mathsf{upk}) = (w, y)$.

$\mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$:

- Parse pp_Γ as before,
- $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow_\$ \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma)$,
- Return $(\mathsf{isk}, \mathsf{ipk}) = (\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma)$.

$\mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$:

- Parse pp_Γ as before,
- $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta)$,
- Return $(\mathsf{osk}, \mathsf{opk}) = (\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$.

$\mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk})$:

- Parse $\mathsf{sk}_\Sigma = \mathsf{isk}$, $\mathsf{pk}_\Sigma = \mathsf{ipk}$, $\mathsf{upk} = y$, $\mathsf{opk} = \mathsf{pk}_\Theta$,
- Return $\mathsf{Sgn}_\Sigma(\mathsf{sk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y))$.

$\mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)$:

- Parse $\mathsf{usk} = w$, $\mathsf{upk} = y$, $\mathsf{ipk} = \mathsf{pk}_\Sigma$, $\mathsf{opk} = \mathsf{pk}_\Theta$,
- $(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y)$,
- Return $\perp$ if $c = \perp$,
- $\pi \leftarrow_\$ \mathsf{Prf}_\Pi(\mathsf{crs}^1_\Pi, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m), (r, \sigma_s, w))$ for $L_1^{f,\mathsf{pp}_\Sigma,\mathsf{pp}_\Theta}$,
- Return $\perp$ if $\pi = \perp$,
- Return $\sigma_m = (c, \pi)$.

$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m)$:

- Parse $(c, \pi) = \sigma_m$, $\mathsf{opk} = \mathsf{pk}_\Theta$, $\mathsf{ipk} = \mathsf{pk}_\Sigma$,
- Return $\mathsf{Vfy}_\Pi(\mathsf{crs}^1_\Pi, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m), \pi)$.

$\mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, m, \sigma_m)$:

- Parse $\mathsf{osk} = \mathsf{sk}_\Theta$, $\sigma_m = (c, \pi)$, $\mathsf{ipk} = \mathsf{pk}_\Sigma$, $\mathsf{opk} = \mathsf{pk}_\Theta$,
- Return $\perp$ if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m) \neq 1$,
- Decrypt $y \leftarrow \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c)$,
- Return $\perp$ if $y = \perp$,
- Generate $\pi_o \leftarrow_\$ \mathsf{Prf}_\Pi(\mathsf{crs}^2_\Pi, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m, y), \mathsf{sk}_\Theta)$ for $L_2^{\mathsf{pp}_\Theta}$,
- Return $\perp$ if $\pi_o = \perp$,
- Return $(\mathsf{upk}, O) = (y, \pi_o)$.

$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m, \mathsf{upk}, O)$:

- Parse $O = \pi_o$, $\mathsf{upk} = y$, $\mathsf{opk} = \mathsf{pk}_\Theta$, $(c, \pi) = \sigma_m$, $\mathsf{ipk} = \mathsf{pk}_\Sigma$,
- Return 0 if $\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m) \neq 1$,
- Return $\mathsf{Vfy}_\Pi(\mathsf{crs}^2_\Pi, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m, y), \pi_o)$.

Construction 4: Secure Group Signature Scheme Γ

Table 4: Assumptions used in the displayed proof of each security property of Γ. Entries denote the corresponding loss; an empty cell means the assumption is not used in that proof. The parameters q_o and q_m denote, respectively, the adversary's bounds on opening-oracle and member-generation-oracle queries. KV is key-verifiability, SB statement-binding, and DS decryption-soundness.

Property	**ZK**	**SSE**	**SB**	**KV**	**OWF**	**Σ**-UNF	**DS**	**mcIND -CPA**
Anonymity	2	q_o						2
Non-frameability	2	2			1		1	
Traceability	1	1						
Trace-soundness	2	2	1	1				
Non-claimability	1	1					1	
Certificate-unforgeability	1	1				1		
Key-unforgeability	1	1			q_m	1		
Opening-unforgeability	2	2				1	1	

5.3.3 Security

We prove each property separately. Table 4 summarizes the required assumptions. For anonymity, traceability, non-frameability, and non-claimability, the proof intuition follows the contrived construction with the auxiliary bypass removed. The strengthened trace-soundness proof additionally uses statement-binding to show that two accepted attributions of the same message-signature pair must refer to the same issuer and opener public keys.

Theorem 17 (Trace-Soundness)**.** Γ *is trace-sound.*

The proof is given in Section 5.4.1.

Theorem 18 (Non-Frameability)**.** Γ *is non-frameable.*

The proof is given in Section 5.4.2.

Theorem 19 (Traceability)**.** Γ *is traceable.*

The proof is given in Section 5.4.3.

Theorem 20 (Anonymity)**.** Γ *is anonymous.*

The proof is given in Section 5.4.4.

Theorem 21 (Non-Claimability)**.** Γ *is non-claimable.*

The proof is given in Section 5.4.5.

Theorem 22 (Certificate-Unforgeability)**.** Γ *is certificate-unforgeable.*

Intuition. The join oracle reveals only opaque certificate handles. After replacing the signing-oracle proofs by simulated proofs, the reduction need not generate any issuer certificates for the adversary's join queries. Simulation-sound extractability applied to a fresh accepting signature then yields a valid certificate under the honest issuer key. Since no corresponding issuer signature was previously issued, this certificate constitutes a Σ-forgery.

The proof is given in Section 5.4.6.

Theorem 23 (Key-Unforgeability)**.** Γ *is key-unforgeable.*

Intuition. Simulation-sound extractability yields

$$(r^*, \sigma_s^*, w^*)$$

from a fresh accepting signature. If σ_s^* certifies an issuer-message that was not previously submitted to the honest join oracle, then it yields a Σ-forgery. Otherwise, the certificate refers to one of the challenger-generated member public keys y_j. Since

$$f(w^*) = y_j,$$

the extracted witness inverts the one-way-function challenge embedded into that member key.

The proof is given in Section 5.4.7.

Theorem 24 (Opening-Unforgeability)**.** Γ *is opening-unforgeable.*

Intuition. Simulation-sound extractability yields a valid issuer certificate and the plaintext encrypted in the group signature, as well as a valid decryption key from the opening proof. Decryption-soundness forces the claimed member public key to equal the identity certified and encrypted in the signature. If the resulting pair $(\mathsf{upk}^*, \mathsf{opk}^*)$ was not submitted to the honest join oracle, the extracted certificate therefore yields a Σ-forgery.

The proof is given in Section 5.4.8.

5.4 Proofs for the Secure Scheme

Challenge-parameter convention. Whenever a reduction embeds a primitive challenge, it uses the public parameters supplied with that challenge throughout the simulated experiment. In particular, an EUF-CMA reduction embeds the challenger's pp_Σ together with its verification key; it does not resample pp_Σ. Any other signature keys needed by the simulation are generated under that same pp_Σ. Generating the "remaining public parameters" means generating only the other, independently sampled components, with relation-dependent setups using the resulting public parameters.

5.4.1 Proof of Theorem 17 (Trace-Soundness)

Proof strategy. We gradually transform the trace-soundness experiment into one where all openings are enforced by extraction and consistency checks. Each hop either changes the CRS distribution, activates extraction, or aborts if the adversary violates a binding property. Any adversary that distinguishes along the way would break one of the underlying assumptions. At the end, two different valid openings for the same (m^*, σ_m^*) are impossible.

- **Game 0 (Original experiment).** Baseline trace-soundness experiment.
- **Game 1 (Simulated CRSs).** Replace both crs_Π^1 and crs_Π^2 by simulated CRSs. Indistinguishability follows from two zero-knowledge hybrids.
- **Game 2 (Extractable CRSs).** Switch both CRSs to their extractable setups. The distributions are identical to the simulated setups.
- **Game 3 (Extraction from L_2).** Extract witnesses from both accepting L_2 proofs and check consistency. Otherwise, SSE is broken.
- **Game 4 (Context consistency via L_1 binding).** Abort if the same signature verifies under two different $(\mathsf{ipk}, \mathsf{opk})$ contexts. Otherwise statement-binding of L_1 is violated.

- **Game 5 (Key-verifiability).** Abort if two accepted decryption keys for the same pk_Θ decrypt the challenge ciphertext differently. Otherwise key-verifiability is broken.

Proof. Let $\mathcal{A}$ be a PPT adversary against trace-soundness. Let S_i denote the success event in Game i.

Game 0: Original experiment. The standard trace-soundness experiment (Figure 16). The adversary outputs

$$(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*)$$

with

$$\sigma_m^* = (c^*, \pi^*) \qquad \text{and} \qquad O_i^* = \pi_{o,i}^* \text{ for } i \in \{0,1\}.$$

Let S_0 be the event that both openings are accepting, i.e.,

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*) = 1$$

and

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_1^*, O_1^*) = 1,$$

and

$$(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) \neq (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*).$$

Game 1: Simulate both CRSs. For each $j \in \{1,2\}$, generate

$$(\mathsf{crs}_\Pi^j, \tau^j) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$$

for the proof system associated with L_j, and embed both simulated CRSs into the public parameters. All other components remain unchanged.

Transition.

We pass through two hybrids. The first replaces crs_Π^1 by a simulated CRS while leaving crs_Π^2 real. The second additionally replaces crs_Π^2 by a simulated CRS. The experiment generates no proofs under either CRS. Hence a distinguisher for either hop yields a zero-knowledge distinguisher for the corresponding proof system.

$$|\Pr[S_1] - \Pr[S_0]| \leq 2\varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},j}(1^\lambda)}$

1. For the j-th hybrid, receive crs_Π from the ZK challenger and set $\mathsf{crs}_\Pi^j \leftarrow \mathsf{crs}_\Pi$.
2. Generate the other CRS according to the corresponding adjacent hybrid: use a real crs_Π^2 when $j = 1$, and a simulated crs_Π^1 when $j = 2$. Generate all remaining public parameters as in the prior game.
3. Run $\mathcal{A}$ and obtain $(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*)$.
4. Parse $\sigma_m^* = (c^*, \pi^*)$ and $O_i^* = \pi_{o,i}^*$ for $i \in \{0,1\}$.
5. Output 1 if $\mathcal{A}$ wins the experiment and 0 otherwise.

Game 2: Extractable CRSs. For each $j \in \{1,2\}$, replace the simulated setup by an extractable setup

$$(\mathsf{crs}_\Pi^j, \tau^j, \zeta^j) \leftarrow_\$ \mathcal{E}_1(1^\lambda)$$

for the proof system associated with L_j. Store the setup information for later use. All other components remain unchanged.

Transition.

By definition, the CRS distributions produced by the simulated and extractable setups are identical. Therefore,

$$|\Pr[S_2] - \Pr[S_1]| = 0.$$

Game 3: Extraction from L_2. For each accepting opening $(\mathsf{upk}_i^*, O_i^*)$ with $O_i^* = \pi_{o,i}^*$, define

$$y_i^* \leftarrow \mathsf{upk}_i^*.$$

Parse

$$\mathsf{pk}_{\Sigma i}^* \leftarrow \mathsf{ipk}_i^* \qquad \text{and} \qquad \mathsf{pk}_{\Theta i}^* \leftarrow \mathsf{opk}_i^*.$$

Define

$$x_{o,i}^* = (\mathsf{pk}_{\Sigma i}^*, \mathsf{pk}_{\Theta i}^*, c^*, m^*, y_i^*).$$

Using ζ^2, compute

$$\mathsf{sk}_{\Theta i}^* \leftarrow_{\$} \mathcal{E}_2(\mathsf{crs}_{\Pi}^2, \zeta^2, x_{o,i}^*, \pi_{o,i}^*) \qquad \text{for } i \in \{0,1\}.$$

Define $\mathsf{Bad}_{\mathsf{ext}}$ if, for some $i \in \{0,1\}$, the extracted value is not a valid witness, that is, if

$$\mathsf{KVfy}_{\Theta}(\mathsf{sk}_{\Theta i}^*, \mathsf{pk}_{\Theta i}^*) \neq 1 \quad \text{or} \quad \mathsf{Dec}_{\Theta}(\mathsf{sk}_{\Theta i}^*, c^*) \neq y_i^*.$$

Abort on $\mathsf{Bad}_{\mathsf{ext}}$.

Transition.

Since the experiment neither generates nor simulates any L_2 proof, the simulation log in the SSE experiment is empty. Hence both submitted statement–proof pairs are fresh. An SSE reduction guesses the index of an invalid extraction and outputs the corresponding pair. Therefore, by a union bound over the two extractions,

$$|\Pr[S_3] - \Pr[S_2]| \leq 2\varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse}}(1^\lambda)}$

1. Receive crs_{Π}^2 from the challenger.
2. Set crs_{Π}^2 into the public parameters. Set up the rest as in the prior game.
3. Run $\mathcal{A}$ and obtain $(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*)$.
4. Parse $\sigma_m^* = (c^*, \pi^*)$ and $O_i^* = \pi_{o,i}^*$ for $i \in \{0,1\}$.
5. Sample $i \leftarrow_{\$} \{0,1\}$.
6. Let $y_i^* = \mathsf{upk}_i^*$.
7. Parse
$$\mathsf{pk}_{\Sigma i}^* \leftarrow \mathsf{ipk}_i^* \qquad \text{and} \qquad \mathsf{pk}_{\Theta i}^* \leftarrow \mathsf{opk}_i^*.$$
8. Define
$$x_{o,i}^* = (\mathsf{pk}_{\Sigma i}^*, \mathsf{pk}_{\Theta i}^*, c^*, m^*, y_i^*).$$
9. Output $(x_{o,i}^*, \pi_{o,i}^*)$.

Game 4: Context consistency via L_1 binding. Parse the adversary output as

$$(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*),$$

and parse

$$\sigma_m^* = (c^*, \pi^*).$$

For $i \in \{0,1\}$, define

$$\mathsf{pk}_{\Sigma i}^* \leftarrow \mathsf{ipk}_i^* \qquad \text{and} \qquad \mathsf{pk}_{\Theta i}^* \leftarrow \mathsf{opk}_i^*.$$

Then define

$$x_0^* = (\mathsf{pk}_{\Sigma 0}^*, \mathsf{pk}_{\Theta 0}^*, c^*, m^*) \qquad \text{and} \qquad x_1^* = (\mathsf{pk}_{\Sigma 1}^*, \mathsf{pk}_{\Theta 1}^*, c^*, m^*).$$

Define $\mathsf{Bad}_{\mathsf{sb}}$ if

$$x_0^* \neq x_1^*.$$

Abort on $\mathsf{Bad}_{\mathsf{sb}}$.

Transition.
In Game 3, the event S_3 implies that both openings are accepted, that is,

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}_i^*, \mathsf{opk}_i^*, m^*, \sigma_m^*, \mathsf{upk}_i^*, O_i^*) = 1 \qquad \text{for } i \in \{0,1\}.$$

By definition of Jdg_Γ, acceptance implies

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}_i^*, \mathsf{opk}_i^*, m^*, \sigma_m^*) = 1 \qquad \text{for } i \in \{0,1\}.$$

Since $\sigma_m^* = (c^*, \pi^*)$ and Vfy_Γ runs

$$\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi^1, (\mathsf{ipk}_i^*, \mathsf{opk}_i^*, c^*, m^*), \pi^*),$$

it follows that

$$\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi^1, x_0^*, \pi^*) = 1 \qquad \text{and} \qquad \mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi^1, x_1^*, \pi^*) = 1.$$

Thus, if $\mathsf{Bad}_{\mathsf{sb}}$ occurs, then the same proof π^* verifies for two distinct statements $x_0^* \neq x_1^*$, violating statement-binding of L_1. Hence,

$$\Pr[S_3 \wedge \mathsf{Bad}_{\mathsf{sb}}] \leq \varepsilon_{\mathsf{sb}}(\lambda).$$

Therefore,

$$|\Pr[S_4] - \Pr[S_3]| \leq \varepsilon_{\mathsf{sb}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sb}}(1^\lambda)}$

1. Receive $(\mathsf{crs}_\Pi^1, \tau^1, \zeta^1)$ from the statement-binding challenger.
2. Embed crs_Π^1 into the public parameters and generate all remaining components as in the prior game.
3. Run $\mathcal{A}$ and obtain
$$(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*).$$
4. Parse $\sigma_m^* = (c^*, \pi^*)$.
5. Define
$$\mathsf{pk}_{\Sigma i}^* \leftarrow \mathsf{ipk}_i^* \qquad \text{and} \qquad \mathsf{pk}_{\Theta i}^* \leftarrow \mathsf{opk}_i^* \quad \text{for } i \in \{0,1\}.$$
6. Define
$$x_i^* \leftarrow (\mathsf{pk}_{\Sigma i}^*, \mathsf{pk}_{\Theta i}^*, c^*, m^*) \quad \text{for } i \in \{0,1\}.$$
7. Output (x_0^*, x_1^*, π^*).

Game 5: Key-verifiability. From Game 4, we have

$$\mathsf{pk}_{\Theta 0}^* = \mathsf{pk}_{\Theta 1}^*.$$

Moreover, from Game 4 we have

$$(\mathsf{pk}_{\Sigma 0}^*, \mathsf{pk}_{\Theta 0}^*) = (\mathsf{pk}_{\Sigma 1}^*, \mathsf{pk}_{\Theta 1}^*).$$

Since

$$(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) \neq (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*),$$

it follows that the tuples differ only in the member component, and hence

$$y_0^* \neq y_1^*.$$

Define

$$\mathsf{pk}_\Theta^* \leftarrow \mathsf{pk}_{\Theta 0}^* = \mathsf{pk}_{\Theta 1}^*.$$

Define $\mathsf{Bad}_{\mathsf{kv}}$ if

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta i}^*, \mathsf{pk}_\Theta^*) = 1 \quad \text{for } i \in \{0,1\}$$

and

$$\mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) \neq \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*).$$

Abort on $\mathsf{Bad}_{\mathsf{kv}}$.

Transition.

From Game 3, extraction yields decryption keys $\mathsf{sk}_{\Theta i}^*$ such that

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_{\Theta i}^*, \mathsf{pk}_{\Theta i}^*) = 1 \quad \text{and} \quad \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta i}^*, c^*) = y_i^* \qquad \text{for } i \in \{0,1\}.$$

In Game 4, we have $\mathsf{pk}_{\Theta 0}^* = \mathsf{pk}_{\Theta 1}^* = \mathsf{pk}_\Theta^*$. Moreover, since

$$(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) \neq (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*)$$

and the public keys coincide, it follows that

$$y_0^* \neq y_1^*.$$

Thus,

$$\mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) \neq \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*),$$

and both keys verify under pk_Θ^*.

Whenever $S_4 \wedge \mathsf{Bad}_{\mathsf{kv}}$ occurs, the two extracted keys, the common public key, and c^* satisfy exactly the winning condition of the key-verifiability experiment. Consequently,

$$\Pr[S_4 \wedge \mathsf{Bad}_{\mathsf{kv}}] \leq \varepsilon_{\mathsf{kv}}(\lambda).$$

Since Game 5 differs only by aborting on this event,

$$|\Pr[S_5] - \Pr[S_4]| \leq \varepsilon_{\mathsf{kv}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{kv}}(1^\lambda)}$

1. Receive pp_Θ from the challenger.
2. Embed pp_Θ into the public parameters. In particular, generate $(\mathsf{crs}_\Pi^2, \tau^2, \zeta^2) \leftarrow_\$ \mathcal{E}_1(1^\lambda)$ and generate all remaining components as in the prior game.
3. Run $\mathcal{A}$ and obtain
$$(\mathsf{ipk}_0^*, \mathsf{opk}_0^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*, m^*, \sigma_m^*, \mathsf{upk}_0^*, O_0^*, \mathsf{upk}_1^*, O_1^*).$$
4. Parse $\sigma_m^* = (c^*, \pi^*)$ and $O_i^* = \pi_{o,i}^*$ for $i \in \{0,1\}$.
5. For $i \in \{0,1\}$, define
$$x_{o,i}^* \leftarrow (\mathsf{ipk}_i^*, \mathsf{opk}_i^*, c^*, m^*, \mathsf{upk}_i^*)$$
and compute
$$\mathsf{sk}_{\Theta i}^* \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^2, \zeta^2, x_{o,i}^*, \pi_{o,i}^*).$$
6. Define
$$\mathsf{pk}_\Theta^* \leftarrow \mathsf{opk}_0^*.$$
7. Output $(\mathsf{sk}_{\Theta 0}^*, \mathsf{sk}_{\Theta 1}^*, \mathsf{pk}_\Theta^*, c^*)$.

Final step. Suppose that S_5 occurs. Since Game 5 does not abort, $\mathsf{Bad}_{\mathsf{kv}}$ does not occur.

From Game 4, we have

$$(\mathsf{pk}_{\Sigma 0}^*, \mathsf{pk}_{\Theta 0}^*) = (\mathsf{pk}_{\Sigma 1}^*, \mathsf{pk}_{\Theta 1}^*).$$

In particular,

$$\mathsf{pk}_{\Theta 0}^* = \mathsf{pk}_{\Theta 1}^* =: \mathsf{pk}_\Theta^*.$$

From Game 3, extraction yields for $i \in \{0,1\}$ that

$$\mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta i}^*, c^*) = y_i^* \qquad \text{and} \qquad y_i^* = \mathsf{upk}_i^*.$$

Since $\mathsf{Bad}_{\mathsf{kv}}$ does not occur in Game 5 and both extracted keys are accepted for pk^*_Θ, their decryption results on c^* coincide. Combining this with the extracted relations, we obtain

$$\mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 0}^*, c^*) = y_0^* \qquad \text{and} \qquad \mathsf{Dec}_\Theta(\mathsf{sk}_{\Theta 1}^*, c^*) = y_1^*,$$

and hence

$$y_0^* = y_1^*.$$

Therefore,

$$\mathsf{upk}_0^* = \mathsf{upk}_1^*.$$

Together with $(\mathsf{ipk}_0^*, \mathsf{opk}_0^*) = (\mathsf{ipk}_1^*, \mathsf{opk}_1^*)$, this contradicts

$$(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) \neq (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*).$$

Hence $\Pr[S_5] = 0$.

Advantage analysis. By a union bound over all transitions, we obtain

$$\Pr[S_0] \leq 2\varepsilon_{\mathsf{zk}}(\lambda) + 2\varepsilon_{\mathsf{sse}}(\lambda) + \varepsilon_{\mathsf{sb}}(\lambda) + \varepsilon_{\mathsf{kv}}(\lambda).$$

□

Conclusion. Any successful adversary breaks zero-knowledge, simulation-sound extractability, statement-binding, or key-verifiability. Hence Γ satisfies trace-soundness.

5.4.2 Proof of Theorem 18 (Non-Frameability)

Proof strategy. We gradually transform the non-frameability experiment into one where every accepting opening yields a consistent and extractable witness. Each hop either changes the CRS distribution, activates extraction, or aborts if the adversary violates a correctness or consistency property. Any non-negligible change in success probability yields an adversary against one of the underlying assumptions. At the end, every successful forgery yields a valid preimage for the one-way function.

- **Game 0 (Original experiment).** Baseline non-frameability experiment.
- **Game 1 (Simulated CRS for L_1).** Replace crs_Π^1 by a simulated CRS. Indistinguishable by zero-knowledge.
- **Game 2 (Simulated CRS for L_2).** Replace crs_Π^2 by a simulated CRS. Indistinguishable by zero-knowledge.
- **Game 3 (Extractable CRS for L_1).** Switch crs_Π^1 to an extractable setup. The joint distribution of the CRS and simulation trapdoor is identical by setup compatibility.
- **Game 4 (Extractable CRS for L_2).** Switch crs_Π^2 to an extractable setup. The public CRS distribution is identical by setup compatibility.
- **Game 5 (Extraction from L_1).** Extract witnesses from the L_1 proof and enforce correctness of the ciphertext and signature. Otherwise, SSE for L_1 is broken.
- **Game 6 (Extraction from L_2).** Extract the decryption key from the L_2 proof and enforce correctness of the opening. Otherwise, SSE for L_2 is broken.
- **Game 7 (Decryption consistency).** Enforce that the extracted key decrypts the ciphertext to the value fixed by the L_1 witness. Otherwise, decryption-soundness is violated.

- **Game 8 (OWF inversion).** The extracted witness yields a preimage of the target value. Otherwise, a contradiction is obtained.

Proof. Let $\mathcal{A}$ be a PPT adversary. Let S_i denote the success event in Game i.

Game 0: Original experiment. The challenger samples all public parameters honestly and samples the honest target member as

$$(\mathsf{usk}^\diamond, \mathsf{upk}^\diamond) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma).$$

The adversary receives pp_Γ and the target public key $\mathsf{upk}^\diamond$.

Let $\mathcal{Q}_\sigma$ denote the set of signatures returned by the signing oracle in the non-frameability experiment.

The adversary outputs

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, O^*)$$

with

$$\sigma_m^* = (c^*, \pi^*), \qquad O^* = \pi_o^*.$$

Define

$$y^\diamond \leftarrow \mathsf{upk}^\diamond.$$

Let S_0 be the event that

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*) \notin \mathcal{Q}_\sigma$$

and

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, \mathsf{upk}^\diamond, O^*) = 1.$$

Game 1: Simulate crs_Π^1. Generate $(\mathsf{crs}_\Pi^1, \tau^1) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$ and embed crs_Π^1 into the public parameters.

Modify the signing oracle as follows: whenever it would generate an L_1 proof for a statement

$$x = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m)$$

with witness

$$w_{L_1} = (r, \sigma_s, w),$$

the proof is obtained from the zero-knowledge oracle instead of being generated honestly.

All other components remain unchanged.

Transition.

Game 1 differs from Game 0 in two aspects: the CRS crs_Π^1 is generated by SIM_1 instead of PG_Π, and every L_1 proof returned through the signing oracle is obtained from the zero-knowledge oracle instead of Prf_Π.

Thus, if $\mathcal{A}$ distinguishes, we obtain a distinguisher against zero-knowledge of L_1.

$$|\Pr[S_1] - \Pr[S_0]| \leq \varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},1}(1^\lambda)}$

1. Receive crs_Π from the zero-knowledge challenger.
2. Set $\mathsf{crs}_\Pi^1 \leftarrow \mathsf{crs}_\Pi$ and generate all remaining public parameters honestly.
3. Run $\mathcal{A}$ and simulate the non-frameability experiment.
4. Whenever the signing oracle is invoked on input $(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$:
   (a) parse $\mathsf{ipk} = \mathsf{pk}_\Sigma$ and $\mathsf{opk} = \mathsf{pk}_\Theta$,
   (b) let $(\mathsf{usk}, \mathsf{upk}) = (w, y)$ be the honest target member key,
   (c) compute $(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y)$,
   (d) return $\perp$ without changing $\mathcal{Q}_\sigma$ if $c = \perp$,
   (e) define
   $$x \leftarrow (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m) \qquad \text{and} \qquad w_{L_1} \leftarrow (r, \sigma_s, w),$$
   (f) query the zero-knowledge oracle on (x, w_{L_1}) to obtain π,
   (g) return $\perp$ without changing $\mathcal{Q}_\sigma$ if $\pi = \perp$; otherwise set $\sigma_m \leftarrow (c, \pi)$, record the complete transcript in $\mathcal{Q}_\sigma$, and return σ_m.
5. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 2: Simulate crs_Π^2. Generate $(\mathsf{crs}_\Pi^2, \tau^2) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$ and embed crs_Π^2 into the public parameters.

All other components remain unchanged.

Transition.

The only change is the distribution of crs_Π^2. The challenger never produces any L_2 proofs; all L_2 proofs are provided by the adversary.

Thus, only the CRS distribution changes.

If $\mathcal{A}$ distinguishes, we obtain a distinguisher against zero-knowledge of L_2.

$$|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},2}(1^\lambda)}$

1. Receive crs_Π from the ZK challenger.
2. Set $\mathsf{crs}_\Pi^2 \leftarrow \mathsf{crs}_\Pi$ and generate all remaining public parameters as in the prior game.
3. Run $\mathcal{A}$ in the resulting experiment.
4. Output 1 if $\mathcal{A}$ wins and 0 otherwise.

Game 3: Extractable crs_Π^1. Replace $(\mathsf{crs}_\Pi^1, \tau^1)$ by

$$(\mathsf{crs}_\Pi^1, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda),$$

and embed crs_Π^1 into the public parameters.

All other components remain unchanged.

Transition.

By setup compatibility, the joint distribution $(\mathsf{crs}_\Pi^1, \tau^1)$ produced by SIM_1 is identical to the first two components produced by $\mathcal{E}_1$. Since the signing oracle invokes the same simulator SIM_2 with τ^1 in both games, its complete response distribution is unchanged. Hence,

$$|\Pr[S_3] - \Pr[S_2]| = 0.$$

Game 4: Extractable crs_Π^2. Replace $(\mathsf{crs}_\Pi^2, \tau^2)$ by

$$(\mathsf{crs}_\Pi^2, \tau^2, \zeta^2) \leftarrow_\$ \mathcal{E}_1(1^\lambda),$$

and embed crs_Π^2 into the public parameters.

All other components remain unchanged.

Transition.

Again, the public CRS distributions induced by SIM_1 and $\mathcal{E}_1$ are identical by setup compatibility. Hence,

$$|\Pr[S_4] - \Pr[S_3]| = 0.$$

Game 5: Extraction from L_1. Define

$$\mathsf{pk}^*_{\Sigma} \leftarrow \mathsf{ipk}^* \qquad \text{and} \qquad \mathsf{pk}^*_{\Theta} \leftarrow \mathsf{opk}^*.$$

Define the L_1 statement

$$x^* = (\mathsf{pk}^*_{\Sigma}, \mathsf{pk}^*_{\Theta}, c^*, m^*).$$

Using ζ^1, compute

$$(r^*, \sigma^*_s, w^*) \leftarrow_{\$} \mathcal{E}_2(\mathsf{crs}^1_{\Pi}, \zeta^1, x^*, \pi^*).$$

Define $\mathsf{Bad}_{\mathsf{ext1}}$ if extraction fails or

$$c^* \neq \mathsf{Enc}_{\Theta}(\mathsf{pk}^*_{\Theta}, f(w^*); r^*)$$

or

$$\mathsf{Vfy}_{\Sigma}(\mathsf{pk}^*_{\Sigma}, (\mathsf{pk}^*_{\Sigma}, \mathsf{pk}^*_{\Theta}, f(w^*)), \sigma^*_s) \neq 1.$$

Abort on $\mathsf{Bad}_{\mathsf{ext1}}$.

Transition.

Every non-$\perp$ L_1 proof simulated in response to a signing query is returned as part of a transcript

$$(\mathsf{ipk}, \mathsf{opk}, m, (c, \pi))$$

recorded in $\mathcal{Q}_\sigma$. Since x^* contains the same issuer key, opener key, ciphertext, and message, equality of (x^*, π^*) with a pair in the SSE simulation log would imply

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m) \in \mathcal{Q}_\sigma,$$

contrary to the winning condition. Thus (x^*, π^*) is fresh relative to the SSE simulation log.

Moreover, by the winning condition, π^* is accepting for x^*.

Therefore, any failure of extraction yields a violation of simulation-sound extractability.

$$|\Pr[S_5] - \Pr[S_4]| \leq \varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},1}(1^\lambda)}$

1. Receive crs^1_{Π} from the challenger and obtain access to its simulation oracle $\mathcal{O}^{\tau^1}_{\mathsf{SIM}}$.
2. Embed crs^1_{Π} into the public parameters, generate $(\mathsf{crs}^2_{\Pi}, \tau^2, \zeta^2) \leftarrow_{\$} \mathcal{E}_1(1^\lambda)$, and generate the honest target pair $(w^\diamond, y^\diamond) \leftarrow_{\$} \mathsf{KgM}_{\Gamma}(\mathsf{pp}_{\Gamma})$.
3. Run $\mathcal{A}$ and answer every signing query $(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$ as follows. Parse $\mathsf{ipk} = \mathsf{pk}_{\Sigma}$ and $\mathsf{opk} = \mathsf{pk}_{\Theta}$, sample $(c; r) \leftarrow_{\$} \mathsf{Enc}_{\Theta}(\mathsf{pk}_{\Theta}, y^\diamond)$, and return $\perp$ without changing $\mathcal{Q}_\sigma$ if $c = \perp$ or if
$$\mathsf{Vfy}_{\Sigma}(\mathsf{pk}_{\Sigma}, (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, y^\diamond), \sigma_s) \neq 1.$$
Otherwise define $x = (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, c, m)$, obtain $\pi \leftarrow_{\$} \mathcal{O}^{\tau^1}_{\mathsf{SIM}}(x)$, and return $\perp$ without changing $\mathcal{Q}_\sigma$ if $\pi = \perp$. Otherwise record $(\mathsf{ipk}, \mathsf{opk}, m, (c, \pi))$ in $\mathcal{Q}_\sigma$ and return (c, π). Failed parsing is handled exactly as by the real signing oracle and does not change $\mathcal{Q}_\sigma$.
4. Obtain $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m, O^*)$ from $\mathcal{A}$.
5. Parse $\sigma^*_m = (c^*, \pi^*)$.
6. Set $\mathsf{pk}^*_{\Sigma} \leftarrow \mathsf{ipk}^*$ and $\mathsf{pk}^*_{\Theta} \leftarrow \mathsf{opk}^*$.
7. Define $x^* = (\mathsf{pk}^*_{\Sigma}, \mathsf{pk}^*_{\Theta}, c^*, m^*)$.
8. Output (x^*, π^*).

Game 6: Extraction from L_2. Define

$$\mathsf{pk}^*_{\Sigma} \leftarrow \mathsf{ipk}^* \qquad \text{and} \qquad \mathsf{pk}^*_{\Theta} \leftarrow \mathsf{opk}^*.$$

Define

$$x^*_o = (\mathsf{pk}^*_{\Sigma}, \mathsf{pk}^*_{\Theta}, c^*, m^*, y^\diamond).$$

Using ζ^2, compute

$$\mathsf{sk}^*_\Theta \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}^2_\Pi, \zeta^2, x^*_o, \pi^*_o).$$

Define $\mathsf{Bad}_{\mathsf{ext2}}$ if extraction fails or

$$\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) \neq 1 \quad \text{or} \quad \mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) \neq y^\diamond.$$

Abort on $\mathsf{Bad}_{\mathsf{ext2}}$.

Transition.

Since the experiment neither generates nor simulates any L_2 proof, the SSE simulation log for L_2 is empty. Hence (x^*_o, π^*_o) is fresh. Moreover, by the winning condition,

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m, y^\diamond, \pi^*_o) = 1,$$

and thus π^*_o is accepting for x^*_o.

Therefore, any failure of extraction yields a violation of simulation-sound extractability.

$$|\Pr[S_6] - \Pr[S_5]| \leq \varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},2}(1^\lambda)}$

1. Receive crs^2_Π from the SSE challenger. Its simulation oracle is not queried.
2. Embed crs^2_Π into the public parameters, generate $(\mathsf{crs}^1_\Pi, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda)$, and generate all remaining parameters as in the prior game.
3. Run $\mathcal{A}$, answering its signing queries with simulated L_1 proofs exactly as in Game 5. Apply the extraction and abort rule of Game 5 using ζ^1, and obtain $(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m, O^*)$.
4. Parse $\sigma^*_m = (c^*, \pi^*)$ and $O^* = \pi^*_o$.
5. Set $\mathsf{pk}^*_\Sigma \leftarrow \mathsf{ipk}^*$ and $\mathsf{pk}^*_\Theta \leftarrow \mathsf{opk}^*$.
6. Define
$$x^*_o = (\mathsf{pk}^*_\Sigma, \mathsf{pk}^*_\Theta, c^*, m^*, y^\diamond).$$
7. Output (x^*_o, π^*_o).

Game 7: Decryption-soundness. Define the bad event $\mathsf{Bad}_{\mathsf{ds}}$ to occur if

$$\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) \neq f(w^*).$$

Abort if $\mathsf{Bad}_{\mathsf{ds}}$ occurs.

Transition.

If $\mathsf{Bad}_{\mathsf{ds}}$ occurs, then by Game 5 we have

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}^*_\Theta, f(w^*); r^*),$$

while by Game 6 the extracted key satisfies

$$\mathsf{KVfy}_\Theta(\mathsf{sk}^*_\Theta, \mathsf{pk}^*_\Theta) = 1,$$

but

$$\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) \neq f(w^*).$$

Thus, we obtain a valid decryption key sk^*_Θ for pk^*_Θ such that a correctly generated ciphertext c^* decrypts to a value different from $f(w^*)$, violating decryption-soundness.

$$|\Pr[S_7] - \Pr[S_6]| \leq \varepsilon_{\mathsf{ds}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{ds}}(1^\lambda)}$

1. Receive pp_Θ from the decryption-soundness challenger.
2. Embed pp_Θ into pp_Γ, generate both proof-system setups with $\mathcal{E}_1$, retain ζ^1 and ζ^2, and generate the honest target member as in Game 6.
3. Run $\mathcal{A}$ on input $(\mathsf{pp}_\Gamma, y^\diamond)$, answering all signing queries with simulated L_1 proofs as in Game 6, and obtain
$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, O^*).$$
4. Parse
$$\sigma_m^* = (c^*, \pi^*) \qquad \text{and} \qquad O^* = \pi_o^*.$$
5. Set
$$\mathsf{pk}_\Sigma^* \leftarrow \mathsf{ipk}^* \qquad \text{and} \qquad \mathsf{pk}_\Theta^* \leftarrow \mathsf{opk}^*.$$
6. Extract (r^*, σ_s^*, w^*) from π^* using ζ^1 and sk_Θ^* from π_o^* using ζ^2. Apply all verification, freshness, and witness checks from Games 5 and 6; if any check fails, terminate without winning.
7. Define
$$m^\dagger \leftarrow f(w^*).$$
8. Output
$$(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*, m^\dagger, r^*)$$
to the decryption-soundness challenger.

Game 8: OWF inversion. Define the bad event $\mathsf{Bad}_{\mathsf{owf}}$ to occur if

$$f(w^*) = y^\diamond.$$

Abort if $\mathsf{Bad}_{\mathsf{owf}}$ occurs.

Transition.

If $\mathsf{Bad}_{\mathsf{owf}}$ occurs, then

$$f(w^*) = y^\diamond,$$

and thus w^* is a valid preimage of the OWF challenge.

Hence, any adversary causing $\mathsf{Bad}_{\mathsf{owf}}$ yields an OWF inverter.

Therefore, there exists $\mathcal{B}_{\mathsf{owf}}$ such that

$$\Pr[S_7 \wedge \mathsf{Bad}_{\mathsf{owf}}] \leq \varepsilon_{\mathsf{owf}}(\lambda).$$

Thus,

$$|\Pr[S_8] - \Pr[S_7]| \leq \varepsilon_{\mathsf{owf}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{owf}}(1^\lambda)}$

1. Receive an OWF challenge
$$y$$
from the challenger.
2. Set
$$y^\diamond \leftarrow y$$
and embed this value as the target public key in the non-frameability experiment. This has exactly the distribution of an honestly generated member public key, because the OWF challenger samples a uniform preimage from C_λ before computing y.
3. Generate all remaining public parameters as in Game 7. In particular, generate
$$(\mathsf{crs}_\Pi^j, \tau^j, \zeta^j) \leftarrow_\$ \mathcal{E}_1(1^\lambda) \qquad \text{for } j \in \{1,2\}.$$
4. Run $\mathcal{A}$ and simulate its signing oracle without knowing a preimage of $y^\diamond$. On a query $(\mathsf{ipk}, \mathsf{opk}, \sigma_s, m)$, parse $\mathsf{ipk} = \mathsf{pk}_\Sigma$ and $\mathsf{opk} = \mathsf{pk}_\Theta$ and first check
$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y^\diamond), \sigma_s) = 1.$$
If the check fails, return $\perp$ without changing $\mathcal{Q}_\sigma$, exactly as in the real experiment. Otherwise sample
$$(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y^\diamond),$$
return $\perp$ without changing $\mathcal{Q}_\sigma$ if $c = \perp$, define $x = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m)$, compute
$$\pi \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}_\Pi^1, \tau^1, x),$$
return $\perp$ without changing $\mathcal{Q}_\sigma$ if $\pi = \perp$, and otherwise return and record (c, π) exactly as in Game 7.
This simulation does not require the hidden target secret key. If the certificate check succeeds, the statement is true because the OWF challenge guarantees the existence of some $w^\diamond \in C_\lambda$ with $f(w^\diamond) = y^\diamond$. If it fails, the total-prover convention makes the real signing algorithm return $\perp$ as well.
5. Obtain the adversary output
$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, O^*)$$
and parse
$$\sigma_m^* = (c^*, \pi^*) \qquad \text{and} \qquad O^* = \pi_o^*.$$
6. Set
$$\mathsf{pk}_\Sigma^* \leftarrow \mathsf{ipk}^* \qquad \text{and} \qquad \mathsf{pk}_\Theta^* \leftarrow \mathsf{opk}^*.$$
7. Using ζ^1 and ζ^2, perform the extractions and all checks from Games 5–7. If the adversary does not win or one of these checks fails, terminate without winning. Let (r^*, σ_s^*, w^*) be the resulting L_1 witness.
8. If $f(w^*) = y^\diamond$, output w^* to the OWF challenger; otherwise terminate without winning.

Final step. We condition on the event that none of the bad events $\mathsf{Bad}_{\mathsf{ext1}}$, $\mathsf{Bad}_{\mathsf{ext2}}$, $\mathsf{Bad}_{\mathsf{ds}}$, and $\mathsf{Bad}_{\mathsf{owf}}$ occur.

From Game 5 we have
$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, f(w^*); r^*).$$
From Game 6 we have
$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = y^\diamond.$$
From Game 7 (no $\mathsf{Bad}_{\mathsf{ds}}$) we obtain
$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = f(w^*).$$
Thus,
$$f(w^*) = y^\diamond.$$
However, since $\mathsf{Bad}_{\mathsf{owf}}$ did not occur, we have
$$f(w^*) \neq y^\diamond,$$
which is a contradiction.

Advantage analysis.

$$\Pr[S_0] \leq 2\varepsilon_{\mathsf{zk}}(\lambda) + 2\varepsilon_{\mathsf{sse}}(\lambda) + \varepsilon_{\mathsf{ds}}(\lambda) + \varepsilon_{\mathsf{owf}}(\lambda).$$

□

Conclusion. Any successful adversary breaks zero-knowledge, simulation-sound extractability, decryption-soundness, or the one-wayness of f. Hence Γ is non-frameable.

5.4.3 Proof of Theorem 19 (Traceability)

Proof strategy. We gradually transform the traceability experiment into one in which every verifying signature yields an extractable L_1 witness. Since the opener key pair is honestly generated, the extracted ciphertext relation and correctness of the encryption and proof systems then ensure that the prescribed opening algorithm returns an accepted opening.

- **Game 0 (Original experiment).** This is the traceability experiment from Figure 14.
- **Game 1 (Simulated CRS for L_1).** Replace crs^1_Π by a simulated CRS. Since the experiment generates no L_1 proof, only the public CRS distribution changes.
- **Game 2 (Extractable CRS for L_1).** Replace the simulated setup by the extractable setup. The public CRS distributions are identical by setup compatibility.
- **Game 3 (Extraction from L_1).** Extract a witness from the accepting L_1 proof. An invalid extraction violates simulation-sound extractability.

Proof. Let $\mathcal{A}$ be a PPT adversary against traceability, and let S_i denote its success event in Game i.

Game 0: Original experiment. The challenger generates

$$\mathsf{pp}_\Gamma \leftarrow_\$ \mathsf{PG}_\Gamma(1^\lambda) \qquad \text{and} \qquad (\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma).$$

Write

$$\mathsf{osk} = \mathsf{sk}_\Theta \qquad \text{and} \qquad \mathsf{opk} = \mathsf{pk}_\Theta.$$

The adversary, which receives $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$, outputs

$$(\mathsf{ipk}^*, m^*, \sigma_m^*) \qquad \text{with} \qquad \sigma_m^* = (c^*, \pi^*).$$

Let S_0 be the event that

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}, m^*, \sigma_m^*) = 1$$

and that, for

$$(\mathsf{upk}^*, O^*) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}^*, m^*, \sigma_m^*),$$

either $(\mathsf{upk}^*, O^*) = \perp$ or

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}, m^*, \sigma_m^*, \mathsf{upk}^*, O^*) \neq 1.$$

Game 1: Simulated crs_Π^1. Generate

$$(\mathsf{crs}_\Pi^1, \tau^1) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$$

and embed crs_Π^1 into the public parameters. All other components remain unchanged.

Transition. The traceability experiment provides no signing or proof-generation oracle; the adversary produces π^* itself. Hence the transition changes only the public CRS distribution. A distinguisher between Games 0 and 1 therefore yields a zero-knowledge distinguisher for the proof system associated with L_1, making no prover-oracle query. Thus

$$\big|\Pr[S_1] - \Pr[S_0]\big| \le \varepsilon_{\mathsf{zk}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk}}(1^\lambda)}$

1. Receive crs_Π from the zero-knowledge challenger and set $\mathsf{crs}_\Pi^1 \leftarrow \mathsf{crs}_\Pi$.
2. Generate all remaining public parameters and the opener key pair honestly, and give $(\mathsf{osk}, \mathsf{opk})$ to $\mathcal{A}$.
3. Obtain $(\mathsf{ipk}^*, m^*, \sigma_m^*)$ from $\mathcal{A}$ and evaluate the traceability winning predicate using the honestly generated opener key.
4. Output 1 exactly when that predicate holds.

Game 2: Extractable crs_Π^1. Replace the simulated setup by

$$(\mathsf{crs}_\Pi^1, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

All other components remain unchanged.

Transition. By setup compatibility, the public CRS distributions induced by SIM_1 and $\mathcal{E}_1$ are identical. Since no proof is generated or simulated by the experiment, no property of the trapdoor distribution is needed in this transition. Consequently,

$$\big|\Pr[S_2] - \Pr[S_1]\big| = 0.$$

Game 3: Extraction from L_1. On a successful adversarial output, parse

$$\mathsf{ipk}^* = \mathsf{pk}_\Sigma^*, \qquad \mathsf{opk} = \mathsf{pk}_\Theta, \qquad \sigma_m^* = (c^*, \pi^*),$$

and define

$$x^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta, c^*, m^*).$$

Compute

$$(r^*, \sigma_s^*, w^*) \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^1, \zeta^1, x^*, \pi^*).$$

Let $\mathsf{Bad}_{\mathsf{ext}}$ occur if the extracted value is not a valid L_1 witness, that is, if

$$c^* \neq \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w^*); r^*)$$

or

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma^*, (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta, f(w^*)), \sigma_s^*) \neq 1.$$

Game 3 aborts on $\mathsf{Bad}_{\mathsf{ext}}$.

Transition. Success in Game 2 implies

$$\mathsf{Vfy}_\Pi(\mathsf{crs}_\Pi^1, x^*, \pi^*) = 1.$$

Moreover, the SSE reduction makes no query to its simulation oracle, so its simulation log is empty and (x^*, π^*) is fresh. Therefore an accepting proof with an invalid extracted witness directly violates simulation-sound extractability. Hence

$$\big|\Pr[S_3] - \Pr[S_2]\big| \le \varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse}}(1^\lambda)}$

1. Receive crs_Π^1 from the SSE challenger. The simulation oracle is not queried.
2. Embed crs_Π^1 into the public parameters, generate all remaining parameters and the opener key pair honestly, and give $(\mathsf{osk}, \mathsf{opk})$ to $\mathcal{A}$.
3. Obtain $(\mathsf{ipk}^*, m^*, \sigma_m^*)$ from $\mathcal{A}$ and parse $\mathsf{ipk}^* = \mathsf{pk}_\Sigma^*$, $\mathsf{opk} = \mathsf{pk}_\Theta$, and $\sigma_m^* = (c^*, \pi^*)$.
4. Define $x^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta, c^*, m^*)$ and output (x^*, π^*).

Final step. Suppose that S_3 occurs. The extracted witness satisfies

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w^*); r^*).$$

Set

$$y^* \leftarrow f(w^*).$$

Because $(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta)$ was honestly generated, correctness of Θ gives

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) = 1 \qquad \text{and} \qquad \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c^*) = y^*.$$

Now consider the prescribed execution of

$$\mathsf{Opn}_\Gamma(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta, \mathsf{pk}^*_\Sigma, m^*, \sigma^*_m).$$

Its initial Vfy_Γ check succeeds by the event S_3, and its decryption step returns y^*. Define

$$x^*_o = (\mathsf{pk}^*_\Sigma, \mathsf{pk}_\Theta, c^*, m^*, y^*).$$

The honestly generated key sk_Θ is a valid witness for $x^*_o \in L_2$. Since crs^2_Π was generated honestly and was never modified in the game sequence, correctness of the proof system implies that every

$$\pi^*_o \leftarrow_\$ \mathsf{Prf}_\Pi(\mathsf{crs}^2_\Pi, x^*_o, \mathsf{sk}_\Theta)$$

is accepted. Hence Opn_Γ returns (y^*, π^*_o), and

$$\mathsf{Jdg}_\Gamma(\mathsf{pk}^*_\Sigma, \mathsf{pk}_\Theta, m^*, \sigma^*_m, y^*, \pi^*_o) = 1.$$

Thus the prescribed opening algorithm neither fails nor returns an invalid opening, contradicting S_3. Therefore $\Pr[S_3] = 0$.

Advantage analysis. Combining the transitions yields

$$\Pr[S_0] \leq \varepsilon_{\mathsf{zk}}(\lambda) + \varepsilon_{\mathsf{sse}}(\lambda).$$

□

Conclusion. Any successful traceability adversary breaks zero-knowledge or simulation-sound extractability. Hence Γ satisfies traceability.

5.4.4 Proof of Theorem 20 (Anonymity)

Proof strategy. We first simulate the L_1 proofs contained in left-or-right signing-oracle responses and the L_2 proofs returned by the opening oracle. We then answer admissible opening queries by extracting an L_1 witness rather than by using the opener secret key. This makes the opening oracle simulatable when the honest encryption key is embedded from an mcIND-CPA experiment. Finally, all left-or-right ciphertexts are switched to encryptions of independently sampled dummy member keys. In the resulting game, the complete adversarial view is independent of the anonymity bit.

- **Game 0 (Original experiment).** This is the anonymity experiment from Figure 13.
- **Game 1 (Simulated L_1 proofs).** Generate crs^1_Π in extractable mode and simulate every L_1 proof returned by the left-or-right signing oracle.
- **Game 2 (Simulated L_2 proofs).** Generate crs^2_Π in simulated mode and simulate every L_2 proof returned by the opening oracle.

- **Game 3 (Extraction in the opening oracle).** Extract the plaintext certified by every admissible opening query. Invalid extraction violates simulation-sound extractability.
- **Game 4 (Challenge-ciphertext switch).** Replace each left-or-right ciphertext by an encryption of an independently sampled dummy member key. This is justified by mcIND-CPA security.

Proof. Let $\mathcal{A}$ be a PPT anonymity adversary. Let $q_o = q_o(\lambda) \geq 1$ be a polynomial upper bound on its number of opening queries, and let S_i denote the event that $\mathcal{A}$ outputs the hidden anonymity bit in Game i.

Equivalent left-or-right oracle. We use the following equivalent implementation throughout the proof. On a query

$$(m, \mathsf{ipk}, \sigma_s^0, \sigma_s^1),$$

parse $\mathsf{ipk} = \mathsf{pk}_\Sigma$, $\mathsf{opk} = \mathsf{pk}_\Theta$, and $(\mathsf{usk}_i, \mathsf{upk}_i) = (w_i, y_i)$ for $i \in \{0,1\}$. First check

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y_i), \sigma_s^i) = 1 \qquad \text{for both } i \in \{0,1\}.$$

If either check fails, return $\perp$. Otherwise generate only the candidate selected by the fixed bit b: sample

$$(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y_b),$$

form the corresponding L_1 proof, verify the resulting signature, and return and record it exactly as in the original experiment.

This implementation is equivalent to the oracle in Figure 13. Indeed, for an honest challenge-member pair, a certificate passing the public test supplies a valid L_1 witness, so the corresponding candidate signature verifies by correctness. If the test fails, the total-prover convention makes the corresponding call to Sgn_Γ return $\perp$, and the original oracle returns $\perp$ at its verification check. Hence generating the unreturned candidate has no effect on the response distribution.

Game 0: Original experiment. This is the original anonymity experiment using the equivalent oracle above. The challenger generates an honest opener pair

$$(\mathsf{sk}_\Theta, \mathsf{pk}_\Theta) \leftarrow_\$ \mathsf{KG}_\Theta(\mathsf{pp}_\Theta),$$

two honest member pairs

$$(w_i, y_i) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma) \qquad \text{for } i \in \{0,1\},$$

and $b \leftarrow_\$ \{0,1\}$. The adversary receives

$$\mathsf{pk}_\Theta, (w_0, y_0), (w_1, y_1)$$

and adaptive access to the left-or-right signing oracle and the restricted opening oracle. Every returned challenge transcript is recorded as

$$(\sigma_m, m, \mathsf{pk}_\Sigma) \in \mathcal{Q}_\sigma,$$

and the opening oracle rejects exactly these transcripts.

Game 1: Simulated L_1 proofs. Generate

$$(\mathsf{crs}_{\Pi}^{1}, \tau^1, \zeta^1) \leftarrow_{\$} \mathcal{E}_1(1^\lambda)$$

for L_1. For every successful left-or-right query, form

$$x_1 = (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, c, m)$$

and replace the real proof by

$$\pi \leftarrow_{\$} \mathsf{SIM}_2(\mathsf{crs}_{\Pi}^{1}, \tau^1, x_1).$$

Return $\perp$ if $c = \perp$, $\pi = \perp$, or the resulting signature does not verify. Otherwise return $\sigma_m = (c, \pi)$ and record the complete challenge transcript in $\mathcal{Q}_\sigma$.

Transition. First replace the real L_1 setup and all returned real proofs by the ideal zero-knowledge setup and proof oracle. Every proof-oracle query is made only after both certificate checks have succeeded and is supplied with the valid witness

$$(r, \sigma_s^b, w_b).$$

Zero-knowledge therefore justifies this replacement for polynomially many adaptive left-or-right queries. By setup compatibility, the joint distribution $(\mathsf{crs}_{\Pi}^{1}, \tau^1)$ output by SIM_1 is identical to the first two components output by $\mathcal{E}_1$. Hence the subsequent switch to the extractable setup does not change either the public CRS or the simulated-proof distribution. Thus

$$\big|\Pr[S_1] - \Pr[S_0]\big| \leq \varepsilon_{\mathsf{zk},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},1}(1^\lambda)}$

1. Receive crs_{Π}^{1} and access to the zero-knowledge proof oracle for L_1.
2. Generate the remaining parameters, the honest opener pair, the two member pairs, and b honestly.
3. On a left-or-right query, perform both certificate checks. If they succeed, sample $(c; r) \leftarrow_{\$} \mathsf{Enc}_{\Theta}(\mathsf{pk}_{\Theta}, y_b)$, define $x_1 = (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, c, m)$, and query the proof oracle on

$$(x_1, (r, \sigma_s^b, w_b)).$$

   Apply all $\perp$, verification, logging, and return rules of the equivalent oracle.
4. Answer opening queries using the honestly generated opener key, subject to the challenge-transcript restriction.
5. Let b^* be the bit returned by $\mathcal{A}$. Output 1 if $b^* = b$ and 0 otherwise.

Game 2: Simulated L_2 proofs. Generate

$$(\mathsf{crs}_{\Pi}^{2}, \tau^2) \leftarrow_{\$} \mathsf{SIM}_1(1^\lambda)$$

for L_2. On an opening query $(\mathsf{pk}_{\Sigma}, m, \sigma_m)$, first apply the challenge-transcript restriction and all parsing and Vfy_{Γ} checks of Opn_{Γ}. If these checks succeed, parse $\sigma_m = (c, \pi)$, compute

$$y \leftarrow \mathsf{Dec}_{\Theta}(\mathsf{sk}_{\Theta}, c),$$

return $\perp$ if $y = \perp$, define

$$x_2 = (\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, c, m, y),$$

and return (y, π_o) with

$$\pi_o \leftarrow_{\$} \mathsf{SIM}_2(\mathsf{crs}_{\Pi}^{2}, \tau^2, x_2).$$

The oracle returns $\perp$ if $\pi_o = \perp$.

Transition. The opener key pair is honestly generated, and

$$\mathsf{KVfy}_{\Theta}(\mathsf{sk}_{\Theta}, \mathsf{pk}_{\Theta}) = 1 \qquad \text{and} \qquad y = \mathsf{Dec}_{\Theta}(\mathsf{sk}_{\Theta}, c).$$

Thus every proof replaced in an admissible opening response has the valid L_2 witness sk_Θ. Zero-knowledge, with polynomially many adaptive opening queries, gives

$$\left|\Pr[S_2] - \Pr[S_1]\right| \leq \varepsilon_{\mathsf{zk},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},2}(1^\lambda)}$

1. Receive crs_Π^2 and access to the zero-knowledge proof oracle for L_2.
2. Generate crs_Π^1 with $\mathcal{E}_1$, generate the honest opener and member pairs, and simulate left-or-right responses as in Game 1.
3. On every admissible opening query, run Opn_Γ up to proof generation, define x_2 as above, and query the proof oracle on $(x_2, \mathsf{sk}_\Theta)$. Apply all $\perp$ and return rules of Opn_Γ.
4. Let b^* be the bit returned by $\mathcal{A}$. Output 1 if $b^* = b$ and 0 otherwise.

Game 3: Extract in the opening oracle. Modify every admissible opening query as follows. After the challenge-transcript restriction and the public verification checks have succeeded, parse

$$\sigma_m = (c, \pi), \qquad x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m),$$

and compute

$$(r, \sigma_s, w) \leftarrow_\$ \mathcal{E}_2(\mathsf{crs}_\Pi^1, \zeta^1, x_1, \pi).$$

Let $\mathsf{Bad}_{\mathsf{sse}}$ occur if the extracted tuple is not a valid L_1 witness, that is, if

$$c \neq \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w); r)$$

or

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, f(w)), \sigma_s) \neq 1.$$

On $\mathsf{Bad}_{\mathsf{sse}}$, the opening oracle returns $\perp$ for this query. Otherwise set

$$y \leftarrow f(w), \qquad x_2 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m, y),$$

simulate π_o under crs_Π^2, and return (y, π_o) with the same $\perp$ handling as in Game 2.

Transition. Every accepting L_1 statement–proof pair simulated by the experiment occurs in a left-or-right signature returned to $\mathcal{A}$, and the corresponding complete transcript is recorded in $\mathcal{Q}_\sigma$. If an opening query contained the same statement–proof pair, then, because x_1 contains the issuer key, opener key, ciphertext, and message, the query would be the exact recorded challenge transcript and would be rejected before extraction. Hence every pair reaching extraction is fresh relative to the SSE simulation log.

The SSE reduction need not know the extraction trapdoor. It generates the honest opener key pair itself and answers earlier opening queries as in Game 2, using decryption and the L_2 simulator. For the analysis only, instrument this Game 2 execution with unobserved extractions under the extractable setup. Couple these extractions and all other randomness with Game 3. Before the first invalid extraction, encryption correctness makes the two executions identical, including the adversary's adaptive queries. Consequently, the probability of a first invalid extraction is the same in this instrumented Game 2 and in Game 3.

Let $q_o \geq 1$ bound the total number of opening queries. The reduction chooses a uniform index $j \in \{1, \ldots, q_o\}$, simulates Game 2 up to that query, and outputs its fresh accepting pair (x_1, π) to the SSE challenger. If the query is not reached or fails the public admissibility checks, it outputs a rejecting pair. The challenger, rather than the reduction, performs the extraction. The simulation before query j does not depend on j, and the challenger uses the same extraction distribution as the instrumented execution. Guessing the first failing query therefore succeeds with probability at least $\Pr[\mathsf{Bad}_{\mathsf{sse}}]/q_o$. Thus

$$\Pr[\mathsf{Bad}_{\mathsf{sse}}] \leq q_o \varepsilon_{\mathsf{sse}}(\lambda).$$

Conditioned on the complement of this event, extraction gives

$$c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, f(w); r).$$

Since pk_Θ belongs to the honestly generated opener pair, correctness of Θ implies

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta, c) = f(w).$$

Thus Games 2 and 3 use the same value y, simulate an L_2 proof for the same statement x_2, and return identically distributed responses. Therefore

$$\big|\Pr[S_3] - \Pr[S_2]\big| \leq q_o \varepsilon_{\mathsf{sse}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse}}(1^\lambda)}$

1. Receive crs_Π^1 and access to the SSE simulation oracle for L_1. Sample $j \leftarrow_\$ \{1, \ldots, q_o\}$.
2. Generate crs_Π^2 with SIM_1, the honest opener and member pairs, and b.
3. Answer left-or-right queries as in Game 1, using the SSE simulation oracle for every returned L_1 proof. Maintain $\mathcal{Q}_\sigma$.
4. Answer opening queries before the j-th query as in Game 2 using the honest opener key. Reject recorded challenge transcripts first.
5. On the j-th opening query, apply the challenge restriction and public verification checks. If they succeed, define (x_1, π) as above and output (x_1, π) to the SSE challenger. If this query is not reached or any check fails, output a rejecting pair.

Game 4: Switch the left-or-right ciphertexts. For every successful left-or-right query, independently sample

$$\widetilde{w} \leftarrow_\$ C_\lambda, \qquad \widetilde{y} \leftarrow f(\widetilde{w}),$$

and replace

$$c \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y_b)$$

by

$$c \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, \widetilde{y}).$$

Continue to simulate the L_1 proof on $x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m)$, retain the verification and $\perp$ checks, and answer opening queries by extraction as in Game 3.

Transition. An mcIND-CPA reduction embeds its challenge public key as pk_Θ. For every left-or-right query that passes both certificate checks, it samples $\widetilde{y}$ as above and submits

$$(y_b, \widetilde{y})$$

to its multi-challenge encryption oracle. It uses the returned ciphertext to form x_1 and simulates the L_1 proof. Opening queries are answered using the L_1 extraction trapdoor and the L_2 simulation trapdoor, so the reduction never needs the challenge decryption key.

If the encryption oracle selects the left messages, the reduction simulates Game 3; if it selects the right messages, it simulates Game 4. In the latter case, x_1 need not belong to L_1, because the certificate is for y_b rather than $\widetilde{y}$. This causes no problem: the game invokes SIM_2 directly, and the simulated proof, its verification result, and any resulting $\perp$ response are efficient post-processing of the challenge ciphertext. No real prover is invoked and no zero-knowledge claim is made for the possibly false statement.

Let β denote the hidden bit of the mcIND-CPA experiment and let A denote the event $b^* = b$. The reduction outputs 0 if A occurs and 1 otherwise. Consequently,

$$\begin{aligned}\mathrm{Adv}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{B}_{\mathsf{cpa}}}(\lambda) &= \left|\Pr[\beta^* = \beta] - \frac{1}{2}\right| \\ &= \frac{1}{2}\big|\Pr[S_3] - \Pr[S_4]\big|.\end{aligned}$$

Therefore,

$$\big|\Pr[S_4] - \Pr[S_3]\big| \leq 2\,\mathrm{Adv}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{B}_{\mathsf{cpa}}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{cpa}}(1^\lambda)}$

1. Receive $(\mathsf{pp}_\Theta, \mathsf{pk}_\Theta)$ and adaptive access to the mcIND-CPA left-or-right encryption oracle.
2. Generate crs_Π^1 with $\mathcal{E}_1$, generate crs_Π^2 with SIM_1, generate the remaining public parameters and member pairs, and sample $b \leftarrow_\$ \{0,1\}$.
3. On each left-or-right signing query, perform both certificate checks, sample $\widetilde{y}$, query the encryption oracle on $(y_b, \widetilde{y})$, simulate the L_1 proof, and apply all verification, logging, and $\perp$ rules of Game 3.
4. Answer admissible opening queries by extracting the L_1 witness with ζ^1 and simulating the corresponding L_2 proof with τ^2. Return $\perp$ whenever extraction fails to produce a valid L_1 witness, exactly as in Game 3. This is a local response rule in both ciphertext hybrids; no separate SSE bound after the ciphertext switch is needed for this reduction.
5. Output 0 if $b^* = b$, and output 1 otherwise.

Final step. In Game 4, both certificate candidates are checked before a response is generated, so the success or failure of these checks is independent of b. Every non-$\perp$ left-or-right response contains an encryption of an independently sampled dummy member key and a simulated proof whose public statement no longer depends on b. The challenge-transcript log, and hence the restriction applied by the opening oracle, is therefore independent of b as well. All remaining opening-oracle behavior depends only on the query and this b-independent state. Consequently, the adversary's complete adaptive view is independent of b, and

$$\Pr[S_4] = \frac{1}{2}.$$

Advantage analysis. Combining the transitions gives

$$\left|\Pr[S_0] - \frac{1}{2}\right| \leq \varepsilon_{\mathsf{zk},1}(\lambda) + \varepsilon_{\mathsf{zk},2}(\lambda) + q_o \varepsilon_{\mathsf{sse}}(\lambda) + 2\,\mathrm{Adv}^{\mathsf{mcIND\text{-}CPA}}_{\mathcal{B}_{\mathsf{cpa}}}(\lambda).$$

□

Conclusion. Any successful anonymity adversary breaks zero-knowledge, simulation-sound extractability, or mcIND-CPA security. Hence Γ satisfies anonymity.

5.4.5 Proof of Theorem 21 (Non-Claimability)

Proof strategy. We extract the decryption key certified by the adversary's accepted opening. Because the challenge signature is an exact output of the signing oracle, its ciphertext was honestly generated as an encryption of the fixed honest member identity. Decryption-soundness then forces every valid opening of that ciphertext to recover this identity, contradicting re-attribution to a different member public key.

- **Game 0 (Original experiment).** This is the non-claimability experiment from Figure 17, augmented only with an internal log of the signing randomness.
- **Game 1 (Simulated CRS for L_2).** Replace crs_Π^2 by a simulated CRS.
- **Game 2 (Extractable CRS for L_2).** Switch crs_Π^2 to the compatible extractable setup.
- **Game 3 (L_2 extraction).** Extract the decryption key from the accepted opening proof. Invalid extraction violates simulation-sound extractability.
- **Game 4 (Decryption consistency).** Use decryption-soundness to equate the claimed identity with the identity encrypted by the signing oracle.

Proof. Let $\mathcal{A}$ be a PPT adversary, and let S_i denote its success event in Game i.

Game 0: Original experiment and internal instrumentation.
This is $\mathbf{Exp}^{\mathsf{NonClaimability}}_{\mathcal{A},\Gamma}(\lambda)$. Write the honest member pair as

$$(\mathsf{usk}, \mathsf{upk}) = (w, y), \qquad y = f(w).$$

In addition to the public transcript log $\mathcal{Q}_\sigma$, the challenger maintains an internal table $\mathcal{T}_\sigma$. Whenever a signing query returns a non-$\perp$ signature

$$\sigma_m = (c, \pi),$$

the challenger records

$$(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m, r, y) \in \mathcal{T}_\sigma,$$

where r is the randomness used in

$$c = \mathsf{Enc}_\Theta(\mathsf{opk}, y; r).$$

This table is not shown to $\mathcal{A}$ and therefore does not change its view.

The event S_0 occurs if $\mathcal{A}$ outputs

$$(\mathsf{ipk}^*, \mathsf{opk}^*, \mathsf{upk}^*, m^*, \sigma^*_m, O^*)$$

such that

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m) \in \mathcal{Q}_\sigma, \qquad \mathsf{upk}^* \neq y,$$

and

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma^*_m, \mathsf{upk}^*, O^*) = 1.$$

Because only non-$\perp$ signing-oracle responses are logged, every successful transcript has a matching entry in $\mathcal{T}_\sigma$.

Game 1: Simulated crs^2_Π. Generate

$$(\mathsf{crs}^2_\Pi, \tau^2) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$$

for L_2 and use this CRS in the public parameters. No L_2 proof is generated by the challenger in the non-claimability experiment; only the CRS distribution changes. Hence zero-knowledge implies

$$\big|\Pr[S_1] - \Pr[S_0]\big| \leq \varepsilon_{\mathsf{zk},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},2}(1^\lambda)}$

1. Receive the challenge CRS crs^2_Π.
2. Generate all remaining parameters and the honest member pair, and run the non-claimability experiment with the instrumented signing oracle.
3. Output 1 exactly when $\mathcal{A}$ wins.

Game 2: Extractable crs^2_Π. Replace the simulated setup by

$$(\mathsf{crs}^2_\Pi, \tau^2, \zeta^2) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

By setup compatibility, the joint distribution of $(\mathsf{crs}^2_\Pi, \tau^2)$ is identical in the simulated and extractable setups. The trapdoor ζ^2 is kept internal, so

$$\Pr[S_2] = \Pr[S_1].$$

Game 3: Extract from the accepted L_2 proof. On a candidate winning output, parse

$$\sigma_m^* = (c^*, \pi^*), \qquad O^* = \pi_o^*,$$

set

$$\mathsf{pk}_\Sigma^* \leftarrow \mathsf{ipk}^*, \qquad \mathsf{pk}_\Theta^* \leftarrow \mathsf{opk}^*, \qquad y^* \leftarrow \mathsf{upk}^*,$$

and define

$$x_o^* = (\mathsf{pk}_\Sigma^*, \mathsf{pk}_\Theta^*, c^*, m^*, y^*).$$

Use ζ^2 to extract sk_Θ^* from (x_o^*, π_o^*). Let Bad_{L_2} occur if extraction fails or the extracted value does not satisfy

$$\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*) = 1 \qquad \text{and} \qquad \mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = y^*.$$

Game 3 aborts on Bad_{L_2}.

Transition. Acceptance by Jdg_Γ implies that π_o^* verifies for x_o^*. The challenger never generates or simulates an L_2 proof in this experiment. Thus (x_o^*, π_o^*) is fresh in the SSE experiment, and an invalid extraction is an SSE violation. Consequently,

$$\big|\Pr[S_3] - \Pr[S_2]\big| \le \varepsilon_{\mathsf{sse},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},2}(1^\lambda)}$
1. Receive the extractable CRS crs_Π^2 and embed it into the public parameters.
2. Simulate the experiment and obtain the final output of $\mathcal{A}$.
3. If the output is winning, form (x_o^*, π_o^*) as above and submit it to the SSE challenger.

Game 4: Decryption consistency. Since the final transcript belongs to $\mathcal{Q}_\sigma$, find a corresponding generation record

$$(\mathsf{ipk}^*, \mathsf{opk}^*, m^*, \sigma_m^*, r^*, y) \in \mathcal{T}_\sigma.$$

In particular,

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, y; r^*).$$

Let $\mathsf{Bad}_{\mathsf{ds}}$ occur if

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) \neq y.$$

Game 4 aborts on this event.

Transition. Outside Bad_{L_2}, the extracted key satisfies $\mathsf{KVfy}_\Theta(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*) = 1$. Together with the recorded equality

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, y; r^*),$$

the event $\mathsf{Bad}_{\mathsf{ds}}$ is exactly a violation of decryption-soundness. Therefore

$$\big|\Pr[S_4] - \Pr[S_3]\big| \le \varepsilon_{\mathsf{ds}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{ds}}(1^\lambda)}$
1. Receive pp_Θ, generate the L_2 CRS with $\mathcal{E}_1$ and retain ζ^2. Generate the remaining parameters as in Game 3 and simulate the instrumented experiment, giving both w and y to $\mathcal{A}$.
2. On a winning output, extract sk_Θ^* as in Game 3 and retrieve (r^*, y) from $\mathcal{T}_\sigma$.
3. Output $(\mathsf{sk}_\Theta^*, \mathsf{pk}_\Theta^*, y, r^*)$.

Final step. Outside the two bad events, Game 3 gives

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = y^* = \mathsf{upk}^*,$$

whereas Game 4 gives

$$\mathsf{Dec}_\Theta(\mathsf{sk}_\Theta^*, c^*) = y = \mathsf{upk}.$$

Hence $\mathsf{upk}^* = \mathsf{upk}$, contradicting the winning condition. Thus

$$\Pr[S_0] \le \varepsilon_{\mathsf{zk},2}(\lambda) + \varepsilon_{\mathsf{sse},2}(\lambda) + \varepsilon_{\mathsf{ds}}(\lambda),$$

which is negligible. □

Conclusion. Any successful adversary breaks zero-knowledge, simulation-sound extractability, or decryption-soundness. Hence Γ satisfies non-claimability.

5.4.6 Proof of Theorem 22 (Certificate-Unforgeability)

Proof strategy. We first replace every L_1 proof returned by the signing oracle with a simulated proof. Because certificates are hidden behind opaque handles, the join oracle can then stop generating issuer signatures altogether. We switch to the compatible extractable setup and extract a certificate from the adversary's fresh accepting group signature. Since the modified experiment never requested an issuer signature, the extracted certificate is a fresh Σ forgery.

- **Game 0 (Original experiment).** This is certificate-unforgeability from Figure 18a.
- **Game 1 (Simulated signing proofs).** Simulate all L_1 proofs produced by the signing oracle.
- **Game 2 (Certificate suppression).** Replace certificates stored behind handles by dummy internal records; no Σ signature is generated.
- **Game 3 (Extractable setup).** Switch to the compatible extractable setup for L_1.
- **Game 4 (Extraction and Σ forgery).** Extract a valid certificate from the fresh accepting forgery.

Proof. Let $\mathcal{A}$ be a PPT adversary, and let S_i denote its success event in Game i.

Game 0: Original experiment. This is $\mathbf{Exp}^{\mathsf{CertUnforgeability}}_{\mathcal{A},\Gamma}(\lambda)$. The honest issuer key is written as

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) = (\mathsf{isk}, \mathsf{ipk}).$$

The event S_0 occurs if $\mathcal{A}$ outputs

$$(\sigma^*_m, m^*, \mathsf{pk}^*_\Theta) \qquad \text{with} \qquad \sigma^*_m = (c^*, \pi^*)$$

such that

$$\mathsf{Vfy}_\Gamma(\mathsf{pk}_\Sigma, \mathsf{pk}^*_\Theta, m^*, \sigma^*_m) = 1$$

and

$$(\sigma^*_m, m^*, \mathsf{pk}^*_\Theta) \notin \mathcal{Q}_\sigma.$$

Notice that the replay filter does not contain pk_Σ because the issuer key is fixed by the experiment.

Game 1: Simulated L_1 proofs. Generate

$$(\mathsf{crs}^1_\Pi, \tau^1) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$$

for L_1. Join queries are answered exactly as in Game 0 and the resulting certificates remain stored behind their opaque handles. On a signing query (i, m), retrieve

$$(i, \sigma_s, \mathsf{pk}_\Theta, w, y) \in \mathcal{Q}_{\mathsf{cert}},$$

sample

$$(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y),$$

define

$$x_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, c, m),$$

and replace the real proof by

$$\pi \leftarrow_{\$} \mathsf{SIM}_2(\mathsf{crs}_{\Pi}^1, \tau^1, x_1).$$

The oracle rejects an invalid handle or malformed input, and returns $\perp$ if encryption fails, before invoking the simulator. It returns $\perp$ if the simulator returns $\perp$; otherwise it records and returns (c, π). There is no additional verification test: the original signing oracle records every non-$\perp$ output.

Transition. For every signing query that passes the handle, parsing, and encryption checks,

$$(r, \sigma_s, w)$$

is a valid witness for x_1: the stored certificate was generated on $(\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, y)$ and $y = f(w)$. Hence adaptive zero-knowledge gives

$$\big|\Pr[S_1] - \Pr[S_0]\big| \le \varepsilon_{\mathsf{zk},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},1}(1^\lambda)}$

1. Receive crs_{Π}^1 and access to the zero-knowledge proof oracle for L_1.
2. Generate the issuer key and all remaining parameters honestly.
3. Answer member-generation and join queries exactly as in Game 0.
4. On a signing query, construct $(x_1, (r, \sigma_s, w))$ as above, obtain the proof from the proof oracle, and apply all original $\perp$, logging, and return rules.
5. Output the bit indicating whether $\mathcal{A}$ wins.

Game 2: Suppress certificate generation. Modify the join oracle as follows. On a query $(\mathsf{pk}_{\Theta}, j)$ that passes exactly the public parsing and domain checks of Join_{Γ} and has an existing member index, retrieve

$$(j, w, y) \in \mathcal{Q}_V,$$

allocate the next handle i, and store only

$$(i, \mathsf{pk}_{\Theta}, w, y)$$

in an internal table $\widetilde{\mathcal{Q}}_{\mathsf{cert}}$. Return i without invoking Sgn_{Σ}. On a signing query (i, m), retrieve the record, sample $(c; r) \leftarrow_{\$} \mathsf{Enc}_{\Theta}(\mathsf{pk}_{\Theta}, y)$, simulate the L_1 proof for $(\mathsf{pk}_{\Sigma}, \mathsf{pk}_{\Theta}, c, m)$, and apply the same $\perp$, logging, and return rules as in Game 1. In particular, abort before simulation if encryption returns $\perp$.

Transition. The certificate itself is never revealed: the join oracle returns only its handle. In Game 1 it is used only as a witness to a proof that is already simulated. Replacing it by no stored value therefore leaves every public value and every oracle response identically distributed. Thus

$$\Pr[S_2] = \Pr[S_1].$$

Game 3: Extractable L_1 setup. Replace the simulated setup by

$$(\mathsf{crs}_{\Pi}^1, \tau^1, \zeta^1) \leftarrow_{\$} \mathcal{E}_1(1^\lambda),$$

and continue to generate all signing-oracle proofs with SIM_2 and τ^1. By setup compatibility, the joint distribution of $(\mathsf{crs}_{\Pi}^1, \tau^1)$ is unchanged. Hence

$$\Pr[S_3] = \Pr[S_2].$$

Game 4: Extract the forged certificate. For a candidate winning forgery, define

$$x_1^* = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta^*, c^*, m^*)$$

and use ζ^1 to extract

$$(r^*, \sigma_s^*, w^*)$$

from (x_1^*, π^*). Let Bad_{L_1} occur if extraction fails or the extracted tuple does not satisfy

$$c^* = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, f(w^*); r^*)$$

and

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta^*, f(w^*)), \sigma_s^*) = 1.$$

Game 4 aborts on Bad_{L_1}.

Freshness and extraction. Every non-$\perp$ proof simulated by the challenger appears in a signing-oracle response

$$\sigma_m = (c, \pi)$$

and is logged as $(\sigma_m, m, \mathsf{pk}_\Theta) \in \mathcal{Q}_\sigma$. If (x_1^*, π^*) were one of these simulated statement–proof pairs, then the fixed issuer key and equality of x_1^* would give the same pk_Θ^*, m^*, and c^*; equality of the proof gives the same complete signature. Thus

$$(\sigma_m^*, m^*, \mathsf{pk}_\Theta^*) \in \mathcal{Q}_\sigma,$$

contradicting the winning condition. Hence the final pair is fresh in the SSE experiment, and

$$\Pr[\mathsf{Bad}_{L_1}] \le \varepsilon_{\mathsf{sse},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},1}(1^\lambda)}$

1. Receive crs_Π^1 and access to the SSE simulation oracle for L_1.
2. Simulate Game 2, obtaining every signing-oracle proof from the simulation oracle and recording its statement–proof pair.
3. On a winning output, form (x_1^*, π^*) and submit it to the SSE challenger.

Final reduction to Σ unforgeability. Outside Bad_{L_1}, the extracted value σ_s^* is a valid Σ signature on

$$\mu^* = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta^*, f(w^*)).$$

Game 2 and all subsequent games never invoke Sgn_Σ. Consequently no message was submitted to the Σ signing oracle, and (μ^*, σ_s^*) is a fresh valid forgery.

$\underline{\mathcal{B}_{\mathsf{dsig}}(1^\lambda)}$

1. Receive the verification key pk_Σ from the Σ challenger and embed it as the honest issuer public key. Do not query the signing oracle.
2. Generate crs_Π^1 with $\mathcal{E}_1$ and simulate member, join, and signing queries exactly as in Game 3.
3. On a winning output, extract (r^*, σ_s^*, w^*).
4. Output
$$((\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta^*, f(w^*)), \sigma_s^*).$$

Combining the transitions yields

$$\Pr[S_0] \le \varepsilon_{\mathsf{zk},1}(\lambda) + \varepsilon_{\mathsf{sse},1}(\lambda) + \varepsilon_{\mathsf{dsigunf}}(\lambda),$$

which is negligible. □

Conclusion. Any successful adversary breaks zero-knowledge, simulation-sound extractability, or Σ unforgeability. Hence Γ satisfies certificate-unforgeability.

5.4.7 Proof of Theorem 23 (Key-Unforgeability)

Proof strategy. We simulate the L_1 proofs returned by the signing oracle, which removes their dependence on the hidden member preimages. We then extract a certificate and a preimage from the adversary's fresh accepting signature. If the extracted certificate message was never submitted to the honest join oracle, the certificate is a Σ forgery. Otherwise the extracted preimage inverts one of the challenger-generated member public keys. Guessing that member-generation query incurs the factor q_m.

- **Game 0 (Original experiment).** This is key-unforgeability from Figure 18b.
- **Game 1 (Simulated signing proofs).** Replace every L_1 proof returned by the signing oracle by a simulated proof.
- **Game 2 (Extractable setup).** Switch to the compatible extractable setup for L_1.
- **Game 3 (Erase member preimages).** Retain only the public member identities after member generation.
- **Game 4 (L_1 extraction).** Extract a valid certificate and a preimage from the fresh forgery.
- **Final case split.** A non-joined certificate message yields a Σ forgery; a joined message yields an inverter for f.

Proof. Let $\mathcal{A}$ be a PPT adversary. Let $q_m = q_m(\lambda) \geq 1$ be a polynomial upper bound on its number of member-generation queries, and let S_i denote its success event in Game i.

Game 0: Original experiment. This is $\mathbf{Exp}^{\mathsf{KeyUnforgeability}}_{\mathcal{A},\Gamma}(\lambda)$. Write the honest issuer pair as

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) = (\mathsf{isk}, \mathsf{ipk}).$$

For the proof, the challenger additionally maintains an internal join log

$$\mathcal{T}_{\mathsf{join}} = \{(j, \mathsf{pk}_\Theta, y_j, \mu_j)\}, \qquad \mu_j = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y_j),$$

containing one record for every successful join query. This instrumentation does not alter the experiment.

The event S_0 occurs if $\mathcal{A}$ outputs

$$(\sigma^*_m, m^*, \mathsf{pk}^*_\Theta) \qquad \text{with} \qquad \sigma^*_m = (c^*, \pi^*)$$

such that

$$\mathsf{Vfy}_\Gamma(\mathsf{pk}_\Sigma, \mathsf{pk}^*_\Theta, m^*, \sigma^*_m) = 1$$

and

$$(\sigma^*_m, m^*, \mathsf{pk}^*_\Theta, \mathsf{pk}_\Sigma) \notin \mathcal{Q}_\sigma.$$

Equivalent signing oracle. We use the following equivalent implementation. On a query

$$(\mathsf{pk}'_\Sigma, j, \sigma_s, \mathsf{pk}_\Theta, m),$$

retrieve $(j, w_j, y_j) \in \mathcal{Q}_V$ and first check

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}'_\Sigma, (\mathsf{pk}'_\Sigma, \mathsf{pk}_\Theta, y_j), \sigma_s) = 1.$$

If the check fails, return $\perp$. Otherwise run the prescribed signing algorithm and retain all of the original $\perp$, logging, and return rules. For an honest stored pair, the public check is precisely the missing certificate condition in the L_1 witness. Hence this implementation has the same response distribution as the original signing oracle under the total prover convention.

Game 1: Simulated L_1 proofs. Generate

$$(\mathsf{crs}^1_\Pi, \tau^1) \leftarrow_\$ \mathsf{SIM}_1(1^\lambda)$$

for L_1. On a signing query that passes the certificate check, sample

$$(c; r) \leftarrow_\$ \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y_j),$$

define

$$x_1 = (\mathsf{pk}'_\Sigma, \mathsf{pk}_\Theta, c, m),$$

and replace the real proof by

$$\pi \leftarrow_\$ \mathsf{SIM}_2(\mathsf{crs}^1_\Pi, \tau^1, x_1).$$

Return $\perp$ before proof simulation if $c = \perp$. After simulation, return $\perp$ if $\pi = \perp$; otherwise record and return (c, π) as in Game 0, without an additional verification test.

Transition. Whenever the certificate check succeeds and $c \neq \perp$,

$$(r, \sigma_s, w_j)$$

is a valid witness for x_1. Adaptive zero-knowledge therefore gives

$$\big|\Pr[S_1] - \Pr[S_0]\big| \leq \varepsilon_{\mathsf{zk},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},1}(1^\lambda)}$

1. Receive crs^1_Π and access to the zero-knowledge proof oracle for L_1.
2. Generate the issuer key and all remaining parameters honestly.
3. Answer member-generation and join queries as in Game 0.
4. On a valid signing query, construct $(x_1, (r, \sigma_s, w_j))$, obtain the proof from the proof oracle, and apply the original logging and return rules.
5. Output the bit indicating whether $\mathcal{A}$ wins.

Game 2: Extractable L_1 setup. Replace the simulated setup by

$$(\mathsf{crs}^1_\Pi, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda),$$

and continue to generate signing-oracle proofs with SIM_2 and τ^1. Setup compatibility leaves the joint distribution of $(\mathsf{crs}^1_\Pi, \tau^1)$ unchanged, so

$$\Pr[S_2] = \Pr[S_1].$$

Game 3: Erase the member preimages. On the j-th member-generation query, continue to sample

$$w_j \leftarrow_\$ C_\lambda, \qquad y_j \leftarrow f(w_j),$$

and return (j, y_j), but erase w_j immediately after computing y_j. Join queries need only y_j. Signing queries are now answered by first performing the public certificate check and then encrypting y_j and simulating the L_1 proof, so they also do not need w_j.

This is only an erasure of unused internal state. The public identities are still distributed as $f(w_j)$ for independent uniform $w_j \in C_\lambda$, and every oracle response is unchanged. Therefore

$$\Pr[S_3] = \Pr[S_2].$$

Game 4: Extract from the final L_1 proof. Define

$$x_1^* = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta^*, c^*, m^*)$$

and use ζ^1 to extract

$$(r^*, \sigma_s^*, w^*)$$

from (x_1^*, π^*). Set

$$y^* \leftarrow f(w^*) \qquad \text{and} \qquad \mu^* \leftarrow (\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta^*, y^*).$$

Let Bad_{L_1} occur if extraction fails or

$$c^* \neq \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta^*, y^*; r^*)$$

or

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, \mu^*, \sigma_s^*) \neq 1.$$

Game 4 aborts on Bad_{L_1}.

Freshness and extraction. Every non-$\perp$ simulated L_1 proof generated by the experiment is returned as part of a signing-oracle signature and logged together with $(m, \mathsf{pk}_\Theta, \mathsf{pk}'_\Sigma)$. If (x_1^*, π^*) were such a simulated pair, equality of the statement would imply $\mathsf{pk}'_\Sigma = \mathsf{pk}_\Sigma$, the same opener key, message, and ciphertext; equality of the proof would imply the same complete signature. The final transcript would therefore belong to $\mathcal{Q}_\sigma$, contrary to freshness. Thus (x_1^*, π^*) is fresh in the SSE experiment, and

$$\Pr[\mathsf{Bad}_{L_1}] \leq \varepsilon_{\mathsf{sse},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},1}(1^\lambda)}$

1. Receive crs_Π^1 and access to the SSE simulation oracle for L_1.
2. Simulate Game 3, obtaining every signing-oracle proof from the simulation oracle and recording the returned pairs.
3. On a winning output, form (x_1^*, π^*) and submit it to the SSE challenger.

Case 1: The certificate message was not joined. Suppose

$$\mu^* \notin \{\mu : (\cdot, \cdot, \cdot, \mu) \in \mathcal{T}_{\mathsf{join}}\}.$$

Outside Bad_{L_1}, the pair (μ^*, σ_s^*) is a valid Σ signature, and the displayed condition says that μ^* was never submitted to the Σ signing oracle used to answer joins. Hence it is a valid Σ forgery.

$\underline{\mathcal{B}_{\mathsf{dsig}}(1^\lambda)}$

1. Receive pk_Σ and access to the Σ signing oracle.
2. Generate crs_Π^1 with $\mathcal{E}_1$ and simulate member and signing queries as in Game 3.
3. Answer a join query (pk_Θ, j) by querying the Σ oracle on $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y_j)$, return the resulting certificate, and record the message in $\mathcal{T}_{\mathsf{join}}$.
4. Extract (r^*, σ_s^*, w^*) from a winning output and, if μ^* is absent from the join log, output (μ^*, σ_s^*).

Thus the probability of this case is at most $\varepsilon_{\mathsf{dsigunf}}(\lambda)$.

Case 2: The certificate message was joined. Suppose instead that μ^* occurs in $\mathcal{T}_{\mathsf{join}}$. Then there is at least one member index j for which a successful join query used pk_Θ^* and

$$y_j = y^* = f(w^*).$$

Fix the least such index and call it $j^\dagger$.

An OWF inverter guesses $j^\diamond \leftarrow_\$ \{1, \ldots, q_m\}$ and embeds its challenge $y^\diamond$ as the public key returned on the $j^\diamond$-th member-generation query. All other public member keys are sampled as independent images of f. The simulation is perfect because an OWF challenge has exactly the distribution $f(w)$ for uniform $w \in C_\lambda$, and Game 3 never uses the erased preimages. If $j^\diamond = j^\dagger$, then

$$y^\diamond = y_{j^\dagger} = y^* = f(w^*),$$

so w^* is a valid preimage of the OWF challenge.

$\underline{\mathcal{B}_{\mathsf{owf}}(1^\lambda)}$

1. Receive $y^\diamond$ and sample $j^\diamond \leftarrow_\$ \{1, \ldots, q_m\}$.
2. Generate the issuer key, the extractable L_1 setup, and all remaining parameters.
3. On member-generation query j, store $y_j = y^\diamond$ if $j = j^\diamond$; otherwise sample w_j, store $y_j = f(w_j)$, and erase w_j. Return (j, y_j) in either case.
4. Answer join queries with the known issuer secret key and signing queries with simulated L_1 proofs as in Game 3.
5. On a winning output, extract w^* and determine $j^\dagger$ from the join log. If $j^\diamond = j^\dagger$, output w^*; otherwise abort.

The guess is correct with probability at least $1/q_m$, and therefore the probability of Case 2 is bounded by

$$q_m \varepsilon_{\mathsf{owf}}(\lambda).$$

Combining extraction with the two exhaustive cases gives

$$\Pr[S_0] \leq \varepsilon_{\mathsf{zk},1}(\lambda) + \varepsilon_{\mathsf{sse},1}(\lambda) + \varepsilon_{\mathsf{dsigunf}}(\lambda) + q_m \varepsilon_{\mathsf{owf}}(\lambda),$$

which is negligible. □

Conclusion. Any successful adversary breaks zero-knowledge, simulation-sound extractability, Σ unforgeability, or the one-wayness of f. Hence Γ satisfies key-unforgeability.

5.4.8 Proof of Theorem 24 (Opening-Unforgeability)

Proof strategy. An accepted attribution contains two proofs with different roles. Extraction from the L_1 proof yields an issuer certificate and the identity encrypted in the group signature. Extraction from the L_2 proof yields a verified decryption key and the identity claimed by the opening. Decryption-soundness forces these two identities to coincide. The winning condition then says that the corresponding member–opener pair was never joined, so the extracted certificate is a fresh Σ forgery.

- **Game 0 (Original experiment).** This is opening-unforgeability from Figure 19.
- **Game 1 (Extractable L_1 setup).** Replace the real L_1 CRS by the compatible extractable CRS.
- **Game 2 (Extractable L_2 setup).** Replace the real L_2 CRS by the compatible extractable CRS.
- **Game 3 (L_1 extraction).** Extract the certified-and-encrypted identity.
- **Game 4 (L_2 extraction).** Extract the decryption key certifying the claimed identity.
- **Game 5 (Decryption consistency).** Use decryption-soundness to identify the two extracted plaintexts.
- **Final reduction.** Convert every remaining opening forgery into a Σ forgery.

Proof. Let $\mathcal{A}$ be a PPT adversary, and let S_i denote its success event in Game i.

Game 0: Original experiment. This is $\mathbf{Exp}^{\mathsf{OpeningUnforgeability}}_{\mathcal{A},\Gamma}(\lambda)$. Write the honest issuer pair as

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) = (\mathsf{isk}, \mathsf{ipk}).$$

The challenger maintains the join relation in the order used by the definition:

$$\mathcal{R}_{\mathsf{join}} \subseteq \{(\mathsf{upk}, \mathsf{opk})\}.$$

The event S_0 occurs if $\mathcal{A}$ outputs

$$(y^*, \mathsf{pk}^*_\Theta, m^*, \sigma^*_m, O^*)$$

such that

$$(y^*, \mathsf{pk}^*_\Theta) \notin \mathcal{R}_{\mathsf{join}}$$

and

$$\mathsf{Jdg}_\Gamma(\mathsf{pk}_\Sigma, \mathsf{pk}^*_\Theta, m^*, \sigma^*_m, y^*, O^*) = 1.$$

Game 1: Extractable L_1 setup. Replace the real CRS for L_1 by

$$(\mathsf{crs}^1_\Pi, \tau^1, \zeta^1) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

Formally, first replace the real CRS by the simulated CRS, which is justified by zero-knowledge even though the challenger generates no L_1 proofs in this experiment, and then use setup compatibility to replace the simulated setup by the extractable setup without changing the public distribution. Thus

$$\left|\Pr[S_1] - \Pr[S_0]\right| \le \varepsilon_{\mathsf{zk},1}(\lambda).$$

Game 2: Extractable L_2 setup. Likewise, replace the real CRS for L_2 by

$$(\mathsf{crs}^2_\Pi, \tau^2, \zeta^2) \leftarrow_\$ \mathcal{E}_1(1^\lambda).$$

The real-to-simulated CRS transition is justified by zero-knowledge, and the simulated-to-extractable transition is exact by setup compatibility. Hence

$$\left|\Pr[S_2] - \Pr[S_1]\right| \le \varepsilon_{\mathsf{zk},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{zk},i}(1^\lambda)}$

1. For $i \in \{1, 2\}$, receive the challenge CRS for L_i.
2. Generate the other CRS and all remaining parameters as required by the adjacent game, and simulate the join oracle honestly.
3. Run $\mathcal{A}$ and output 1 exactly when it wins.

Game 3: Extract from the accepted L_1 proof. On a candidate winning output, parse

$$\sigma^*_m = (c^*, \pi^*)$$

and define

$$x^*_1 = (\mathsf{pk}_\Sigma, \mathsf{pk}^*_\Theta, c^*, m^*).$$

Use ζ^1 to extract

$$(r^*, \sigma^*_s, w^*)$$

from (x^*_1, π^*), and set

$$\widehat{y} \leftarrow f(w^*).$$

Let Bad_{L_1} occur if extraction fails or

$$c^* \neq \mathsf{Enc}_\Theta(\mathsf{pk}^*_\Theta, \widehat{y}; r^*)$$

or

$$\mathsf{Vfy}_{\mathbf{\Sigma}}(\mathsf{pk}_{\mathbf{\Sigma}}, (\mathsf{pk}_{\mathbf{\Sigma}}, \mathsf{pk}^*_{\Theta}, \widehat{y}), \sigma^*_s) \neq 1.$$

Game 3 aborts on Bad_{L_1}.

Transition. Acceptance by $\mathsf{Jdg}_{\mathbf{\Gamma}}$ implies that $\mathsf{Vfy}_{\mathbf{\Gamma}}$ accepts σ^*_m, and hence that π^* verifies for x^*_1. The challenger never generates or simulates an L_1 proof in this experiment, so the final statement–proof pair is fresh in the SSE game. Therefore

$$\big|\Pr[S_3] - \Pr[S_2]\big| \leq \varepsilon_{\mathsf{sse},1}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},1}(1^\lambda)}$
1. Receive the extractable CRS for L_1 and embed it into the public parameters.
2. Generate all remaining parameters, answer join queries honestly, and run $\mathcal{A}$.
3. On a winning output, form (x^*_1, π^*) and submit it to the SSE challenger.

Game 4: Extract from the accepted L_2 proof. Parse

$$O^* = \pi^*_o$$

and define the opening statement with the identity claimed by the adversary:

$$x^*_2 = (\mathsf{pk}_{\mathbf{\Sigma}}, \mathsf{pk}^*_{\Theta}, c^*, m^*, y^*).$$

Use ζ^2 to extract sk^*_{Θ} from (x^*_2, π^*_o). Let Bad_{L_2} occur if extraction fails or

$$\mathsf{KVfy}_{\Theta}(\mathsf{sk}^*_{\Theta}, \mathsf{pk}^*_{\Theta}) \neq 1$$

or

$$\mathsf{Dec}_{\Theta}(\mathsf{sk}^*_{\Theta}, c^*) \neq y^*.$$

Game 4 aborts on Bad_{L_2}.

Transition. The accepted opening implies that π^*_o verifies for x^*_2. The challenger never generates or simulates an L_2 proof, so this pair is fresh in the SSE game. Consequently,

$$\big|\Pr[S_4] - \Pr[S_3]\big| \leq \varepsilon_{\mathsf{sse},2}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{sse},2}(1^\lambda)}$
1. Receive the extractable CRS for L_2 and embed it into the public parameters.
2. Generate all remaining parameters, answer join queries honestly, and run $\mathcal{A}$.
3. On a winning output, form (x^*_2, π^*_o) and submit it to the SSE challenger.

Game 5: Decryption consistency. Outside Bad_{L_1}, extraction gives

$$c^* = \mathsf{Enc}_{\Theta}(\mathsf{pk}^*_{\Theta}, \widehat{y}; r^*).$$

Outside Bad_{L_2}, extraction gives a key satisfying

$$\mathsf{KVfy}_{\Theta}(\mathsf{sk}^*_{\Theta}, \mathsf{pk}^*_{\Theta}) = 1.$$

Let $\mathsf{Bad}_{\mathsf{ds}}$ occur if

$$\mathsf{Dec}_{\Theta}(\mathsf{sk}^*_{\Theta}, c^*) \neq \widehat{y}.$$

Game 5 aborts on this event.

Transition. The values

$$(\mathsf{sk}^*_{\Theta}, \mathsf{pk}^*_{\Theta}, \widehat{y}, r^*)$$

constitute a decryption-soundness violation whenever $\mathsf{Bad}_{\mathsf{ds}}$ occurs. Therefore

$$\big|\Pr[S_5] - \Pr[S_4]\big| \leq \varepsilon_{\mathsf{ds}}(\lambda).$$

$\underline{\mathcal{B}_{\mathsf{ds}}(1^\lambda)}$
1. Receive pp_{Θ}, generate both CRSs with $\mathcal{E}_1$, and retain ζ^1, ζ^2. Generate the remaining parameters as in Game 4 and simulate the opening-unforgeability experiment.
2. On a winning output, extract (r^*, σ^*_s, w^*) and sk^*_{Θ} as in Games 3 and 4.
3. Set $\widehat{y} = f(w^*)$ and output $(\mathsf{sk}^*_{\Theta}, \mathsf{pk}^*_{\Theta}, \widehat{y}, r^*)$.

Final reduction to Σ unforgeability. Outside all bad events, Game 4 gives

$$\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) = y^*,$$

whereas Game 5 gives

$$\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c^*) = \widehat{y}.$$

Hence

$$y^* = \widehat{y}.$$

The winning condition now implies

$$(\widehat{y}, \mathsf{pk}^*_\Theta) = (y^*, \mathsf{pk}^*_\Theta) \notin \mathcal{R}_{\mathsf{join}}.$$

At the same time, L_1 extraction gives a valid certificate

$$\mathsf{Vfy}_\Sigma(\mathsf{pk}_\Sigma, (\mathsf{pk}_\Sigma, \mathsf{pk}^*_\Theta, \widehat{y}), \sigma^*_s) = 1.$$

Every message submitted to the Σ signing oracle while simulating a successful join has the form

$$(\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y)$$

and causes (y, pk_Θ) to be inserted into $\mathcal{R}_{\mathsf{join}}$. Thus the displayed winning condition guarantees that the extracted message was never queried, and σ^*_s is a fresh Σ forgery.

$\underline{\mathcal{B}_{\mathsf{dsig}}(1^\lambda)}$

1. Receive pk_Σ and access to the Σ signing oracle.
2. Generate both CRSs with their extractable setups and embed pk_Σ as the honest issuer public key.
3. On a valid join query (y, pk_Θ), request a signature on $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Theta, y)$, return it, and record (y, pk_Θ) in $\mathcal{R}_{\mathsf{join}}$.
4. On a winning output, extract (r^*, σ^*_s, w^*) and sk^*_Θ. If the extraction and decryption-consistency checks pass, output
$$((\mathsf{pk}_\Sigma, \mathsf{pk}^*_\Theta, f(w^*)), \sigma^*_s).$$

Combining the transitions yields

$$\begin{aligned}\Pr[S_0] \le\ & \varepsilon_{\mathsf{zk},1}(\lambda) + \varepsilon_{\mathsf{zk},2}(\lambda) \\ & + \varepsilon_{\mathsf{sse},1}(\lambda) + \varepsilon_{\mathsf{sse},2}(\lambda) \\ & + \varepsilon_{\mathsf{ds}}(\lambda) + \varepsilon_{\mathsf{dsigunf}}(\lambda),\end{aligned}$$

which is negligible. □

Conclusion. Any successful adversary breaks zero-knowledge, simulation-sound extractability, decryption-soundness, or Σ unforgeability. Hence Γ satisfies opening-unforgeability.

5.5 On the Role of Decryption-Soundness and Key-Verifiability

Two properties of the encryption primitive are used in different parts of the security analysis: decryption-soundness and key-verifiability.

Decryption-soundness. Decryption-soundness ensures that every ciphertext formed by the prescribed encryption algorithm decrypts to its intended plaintext under every decryption key that passes KVfy_Θ. In our proofs, extraction establishes an equality of the form

$$c = \mathsf{Enc}_\Theta(\mathsf{pk}_\Theta, y; r).$$

Decryption-soundness then forces every extracted opening key sk^*_Θ that is accepted for pk_Θ to satisfy $\mathsf{Dec}_\Theta(\mathsf{sk}^*_\Theta, c) = y$. This is the point needed in non-frameability, non-claimability, and opening-unforgeability.

Key-verifiability. Key-verifiability ensures functional consistency among accepted decryption keys for a fixed public key. Thus, two accepted decryption keys for the same pk_Θ cannot lead to different decryption results on the same ciphertext. This is the point needed in trace-soundness, where two different valid openings would otherwise correspond to two inconsistent decryption behaviors.

Why the displayed proofs use both. The two assumptions apply in different regimes. Decryption-soundness relates a ciphertext formed by the encryption algorithm to its intended plaintext and guarantees correct decryption under every secret key accepted for the corresponding public key. It does not provide a general uniqueness guarantee for arbitrary ciphertexts. Key-verifiability, by contrast, guarantees that any two accepted secret keys for the same public key behave identically on the same ciphertext, but does not identify that common output with the plaintext used to form an honest encryption. This explains their different uses in the displayed proofs; it is not a minimality claim for the complete construction. Indeed, an alternative trace-soundness proof can use statement-binding to fix the group context, extract one L_1 witness and both L_2 witnesses, and apply decryption-soundness twice to the ciphertext certified by L_1. Both claimed plaintexts then equal $f(w)$, without invoking key-verifiability. This alternative uses one additional L_1 extraction and two decryption-soundness reductions in place of the key-verifiability reduction.

6 Separations of the Definitions

We now show that the eight security notions are logically independent. For each property P, we give a black-box transformation $\mathcal{T}_\mathsf{P}$ such that, given a base scheme Γ satisfying all eight properties, the derived scheme

$$\Gamma^{\neg\mathsf{P}} := \mathcal{T}_\mathsf{P}(\Gamma)$$

satisfies all remaining seven properties but violates P.

Throughout this section, we instantiate the baseline scheme with message space $\mathcal{MS} = \{0,1\}^*$ and fix an efficiently computable, efficiently decodable injective encoding of polynomial-length tuples into $\{0,1\}^*$. We write tuples directly for their encodings. Hence every tuple used internally by the transformations below is a valid baseline message. This entails no restriction on the existential separation results proved here.

Lemma 1. *The existence of a correct certificate-unforgeable group signature scheme with message space $\{0,1\}^*$ implies the existence of a correct EUF-CMA-secure digital signature scheme with message space $\{0,1\}^*$.*

Proof. Let Γ be such a group signature scheme. We construct a digital signature scheme $\widehat{\Sigma}$ as follows. Its parameter-generation algorithm is PG_Γ. On input public parameters pp_Γ, key generation samples

$$(\mathsf{isk},\mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma), \qquad (\mathsf{osk},\mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma),$$

and

$$(\mathsf{usk},\mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma),$$

and then computes

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk},\mathsf{ipk},\mathsf{upk},\mathsf{opk}).$$

It outputs

$$\mathsf{sk}_\Sigma = (\mathsf{usk},\mathsf{upk},\sigma_s,\mathsf{ipk},\mathsf{opk}) \qquad \text{and} \qquad \mathsf{pk}_\Sigma = (\mathsf{ipk},\mathsf{opk}).$$

To sign $m \in \{0,1\}^*$, compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m),$$

and verify by evaluating

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma_m).$$

Correctness of $\widehat{\Sigma}$ follows directly from correctness of Γ.

Suppose that a PPT adversary $\mathcal{A}$ breaks EUF-CMA security of $\widehat{\Sigma}$. We construct a PPT adversary $\mathcal{B}$ against certificate-unforgeability of Γ. The certificate-unforgeability challenger gives $\mathcal{B}$ $(\mathsf{pp}_\Gamma, \mathsf{ipk})$. The reduction samples

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma),$$

queries $\mathcal{O}_{\mathsf{KgM}_\Gamma}$ once to obtain $(j, \mathsf{usk}, \mathsf{upk})$, and then queries

$$i \leftarrow_\$ \mathcal{O}^{\mathsf{isk}}_{\mathsf{Join}_\Gamma}(\mathsf{opk}, j).$$

By correctness, this honest join succeeds. The reduction gives

$$\mathsf{pk}_\Sigma = (\mathsf{ipk}, \mathsf{opk})$$

to $\mathcal{A}$. For every signing query m made by $\mathcal{A}$, it returns

$$\sigma_m \leftarrow_\$ \mathcal{O}_{\mathsf{Sgn}_\Gamma}(i, m).$$

This perfectly simulates the EUF-CMA experiment for $\widehat{\Sigma}$.

Let (m^*, σ^*_m) be a successful EUF-CMA forgery. Then

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m^*, \sigma^*_m) = 1.$$

Moreover, EUF-CMA freshness gives that m^* was never submitted to the signing oracle. Since every transcript inserted into $\mathcal{Q}_\sigma$ by the simulation has the form $(\sigma_m, m, \mathsf{opk})$ for a queried message m, we have

$$(\sigma^*_m, m^*, \mathsf{opk}) \notin \mathcal{Q}_\sigma.$$

Thus $(\sigma^*_m, m^*, \mathsf{opk})$ is a successful certificate-unforgeability forgery for Γ. Therefore $\widehat{\Sigma}$ is EUF-CMA secure. □

Lemma 2. *The existence of a correct EUF-CMA-secure digital signature scheme with message space $\{0,1\}^*$ implies the existence of a one-way function family.*

Proof. Let Σ be such a digital signature scheme. Fix polynomial upper bounds $\ell_{\mathsf{pp}}(\lambda)$ and $\ell_{\mathsf{kg}}(\lambda)$ on the random tapes used by PP_Σ and KG_Σ, respectively, padding unused random bits if necessary. Define

$$C_\lambda = \{0,1\}^{\ell_{\mathsf{pp}}(\lambda)} \times \{0,1\}^{\ell_{\mathsf{kg}}(\lambda)}.$$

For $(\rho_{\mathsf{pp}}, \rho_{\mathsf{kg}}) \in C_\lambda$, let

$$\mathsf{pp}_\Sigma \leftarrow \mathsf{PP}_\Sigma(1^\lambda; \rho_{\mathsf{pp}}), \qquad (\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma; \rho_{\mathsf{kg}}),$$

and define the efficiently computable function

$$f_\lambda(\rho_{\mathsf{pp}}, \rho_{\mathsf{kg}}) \coloneqq (\mathsf{pp}_\Sigma, \mathsf{pk}_\Sigma).$$

Uniform sampling from C_λ induces exactly the public-parameter and verification-key distribution of the EUF-CMA experiment.

Assume toward contradiction that $\{f_\lambda\}_{\lambda\in\mathbb{N}}$ is not one-way. Then there exists a PPT inverter $\mathcal{A}$ that, on a non-negligible fraction of honestly distributed pairs $(\mathsf{pp}_\Sigma, \mathsf{pk}_\Sigma)$, outputs $(\rho'_{\mathsf{pp}}, \rho'_{\mathsf{kg}}) \in C_\lambda$ satisfying

$$f_\lambda(\rho'_{\mathsf{pp}}, \rho'_{\mathsf{kg}}) = (\mathsf{pp}_\Sigma, \mathsf{pk}_\Sigma).$$

An EUF-CMA adversary, given $(\mathsf{pp}_\Sigma, \mathsf{pk}_\Sigma)$, runs this inverter. Whenever it succeeds, the adversary recomputes

$$(\mathsf{sk}'_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \mathsf{KG}_\Sigma(\mathsf{pp}_\Sigma; \rho'_{\mathsf{kg}})$$

and, without making any signing-oracle query, outputs

$$\big(0, \mathsf{Sgn}_\Sigma(\mathsf{sk}'_\Sigma, 0)\big).$$

Because $0 \in \{0,1\}^*$ and $(\mathsf{sk}'_\Sigma, \mathsf{pk}_\Sigma)$ is a valid output of key generation under pp_Σ, correctness implies that this signature verifies. The message 0 is fresh because no signing-oracle query was made. Hence the EUF-CMA adversary succeeds whenever the inverter succeeds, contradicting EUF-CMA security. Therefore $\{f_\lambda\}_{\lambda\in\mathbb{N}}$ is a one-way function family. If a fixed-length bitstring codomain is needed, encode $(\mathsf{pp}_\Sigma, \mathsf{pk}_\Sigma)$ injectively into a publicly fixed polynomial output length, including length delimiters and padding. Inverting the resulting encoding still recovers random tapes producing exactly the challenge $(\mathsf{pp}_\Sigma, \mathsf{pk}_\Sigma)$, so the same reduction applies. The two random tapes in C_λ can likewise be concatenated into one fixed-length bitstring. □

All transformations preserve the external interface of the scheme. The certificate-unforgeability and key-unforgeability separations syntactically use a fixed one-way function family. This introduces no additional cryptographic primitive-existence assumption. By Lemma 1, the existence of a correct certificate-unforgeable group signature scheme with message space $\{0,1\}^*$ already implies the existence of a correct EUF-CMA-secure digital signature scheme. By Lemma 2, the latter implies the existence of a one-way function family. Consequently, the one-way function used by these two wrappers does not strengthen the primitive-existence assumption underlying the separation results, and all eight transformations establish that none of the seven remaining properties implies the omitted property.

Our goal is a decomposition. The independence results should therefore be read relative to this syntax: we do not claim that every separation survives additional syntactic restrictions. For example, mandatory key validation, canonical key generation, restrictions on corruption, or trusted registration state may enforce parts of the relations isolated by our notions directly at the model level and can therefore collapse individual dimensions of the decomposition. Determining the minimal set of notions under each such restricted syntax is orthogonal to our goal; our results identify what cannot be inferred when these additional structural guarantees are absent.

Theorem 25 (Independence of Anonymity)**.** *There exists a group signature scheme* $\Gamma^{\neg\mathsf{ANON}}$ *that is non-frameable, traceable, trace-sound, non-claimable, certificate-unforgeable, key-unforgeable, and opening-unforgeable, but not anonymous.*

Intuition. We destroy anonymity by leaking the signer's public identity in the clear, while delegating all validity checks to the baseline scheme. Concretely, the wrapper has the baseline signer authenticate the encoded tuple (m, upk). Thus, the signer identity becomes explicit in every transcript, while the remaining seven properties continue to reduce directly to the baseline.

The proof is given in Section 6.1.1.

Theorem 26 (Independence of Non-Frameability)**.** *There exists a group signature scheme* $\Gamma^{\neg\mathsf{NF}}$ *that satisfies anonymity, traceability, trace-soundness, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability, but is not non-frameable.*

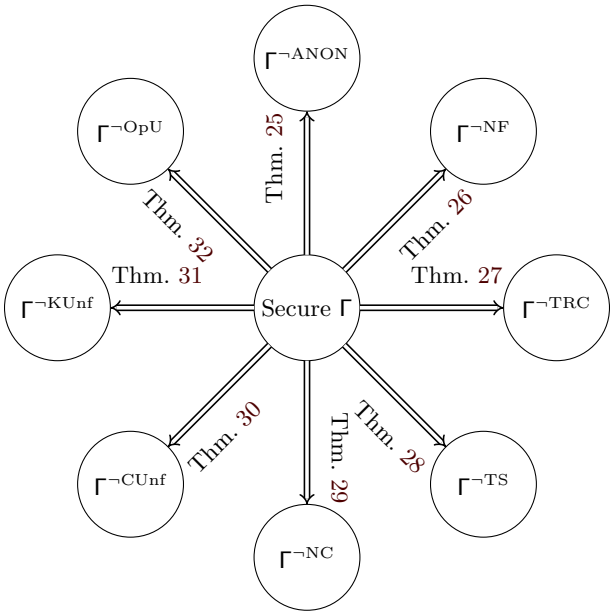


Figure 20: Logical independence of the eight security properties. Each arrow corresponds to a black-box transformation from a fully secure scheme to a scheme violating exactly one property.

Intuition. Honest signing is enabled only when both authority anchors are canonical; in that case it signs messages of the form $(m, 0)$. If issuer and opener collude, they may jointly publish the same noncanonical anchor identity $y^\star$ in their public keys. The modified opening-verification algorithm then reroutes all signatures on $(m, 1)$ to $y^\star$, independently of the actual witness. By choosing $y^\star$ to be the public key of an honest member, the colluding authorities can frame that member. The gadget branch is disabled as soon as one authority key is honestly generated. The ordinary branch delegates to the baseline only under two canonical authority keys and otherwise rejects, so the remaining seven properties persist.

The proof is given in Section 6.1.2.

Theorem 27 (Independence of Traceability)**.** *There exists a group signature scheme* $\Gamma^{\neg\mathsf{TRC}}$ *that satisfies anonymity, non-frameability, trace-soundness, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability, but is not traceable.*

Intuition. We augment only the issuer public key with a flag bit $b_{\mathsf{trc}} \in \{0, 1\}$. If the issuer is honest, it publishes $b_{\mathsf{trc}} = 0$ and the wrapper reduces to the baseline. A corrupt issuer may instead set $b_{\mathsf{trc}} = 1$. Verification is left unchanged, but whenever $b_{\mathsf{trc}} = 1$ and a signature verifies, the opening interfaces are hardwired to fail: Opn'_Γ returns $\perp$ and Jdg'_Γ returns 0. Hence, valid but unopenable signatures exist, violating traceability, while the remaining seven properties are inherited from the baseline.

The proof is given in Section 6.1.3.

Theorem 28 (Independence of Trace-Soundness)**.** *There exists a group signature scheme* $\Gamma^{\neg\mathsf{TS}}$ *that satisfies anonymity, non-frameability, traceability, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability, but is not trace-sound.*

Intuition. We augment both the issuer and the opener public keys with a public anchor. Honestly generated authority keys use the distinguished canonical anchor hon. If both authorities instead use the same noncanonical anchor $t \neq \mathsf{hon}$, the wrapper enables a special transcript that verifies unconditionally and admits two different valid openings to two distinguished dummy identities. These dummy identities are chosen outside the honest member-key space by tagging, so they cannot collide with honestly generated users.

Whenever at least one authority is honest, its anchor is hon, which disables the special branch. The ordinary branch delegates to the baseline only when both authority keys are canonical; all remaining contexts are rejected. Hence trace-soundness fails when both authorities collude on a common noncanonical anchor, while the remaining seven properties are preserved.

The proof is given in Section 6.1.4.

Theorem 29 (Independence of Non-Claimability)**.** *There exists a group signature scheme $\Gamma^{\neg\mathsf{NC}}$ that satisfies anonymity, non-frameability, traceability, trace-soundness, certificate-unforgeability, key-unforgeability, and opening-unforgeability, but is not non-claimable. Moreover, if either authority uses an honestly generated public key, the wrapper satisfies non-claimability under the baseline security assumptions.*

Intuition. We use disjoint encodings for ordinary member identities $(0, \mathsf{upk})$ and gadget identities $(1, t)$. Honestly generated issuer and opener public keys carry the canonical anchor hon. A corrupt issuer and opener may instead choose the same noncanonical anchor

$$a_G = a_O = t \neq \mathsf{hon}.$$

In this context, the wrapper signing algorithm still produces a valid baseline signature for the honest member. The gadget branch of Jdg'_Γ then accepts an embedded baseline-valid opening of this signature as an attribution to the distinct identity $(1, t)$. Consequently, the colluding authorities can re-attribute an exact signing-oracle transcript from $(0, \mathsf{upk})$ to $(1, t)$, violating non-claimability. If either authority uses its honestly generated key, the matching noncanonical anchors required by the gadget branch are unavailable, and the wrapper reduces to the baseline behavior.

The proof is given in Section 6.1.5.

Theorem 30 (Independence of Certificate-Unforgeability)**.** *There exists a group signature scheme $\Gamma^{\neg\mathsf{CUnf}}$ that satisfies anonymity, non-frameability, traceability, trace-soundness, non-claimability, key-unforgeability, and opening-unforgeability, but is not certificate-unforgeable.*

Intuition. The wrapper augments every member secret key with an auxiliary value x and uses a dedicated issuer component when the opener public key contains the switch bit $b_{\mathsf{cunf}} = 1$. In this branch, the signing oracle returns an authenticated mode-1 signature whose underlying message contains

$$h = H(x).$$

The certificate-unforgeability experiment exposes the complete member secret key and therefore reveals x, although the membership certificate itself remains hidden behind an opaque handle. The adversary can consequently transform the mode-1 signing-oracle output into a distinct mode-2 signature by appending x. Verification accepts because $H(x) = h$, while the resulting wrapper transcript is fresh. This violates certificate-unforgeability. One-wayness of H prevents parties that do not receive the auxiliary member secret from activating the same transformation in the remaining security experiments.

The proof is given in Section 6.1.6.

Theorem 31 (Independence of Key-Unforgeability)**.** *There exists a group signature scheme $\Gamma^{\neg\mathsf{KUnf}}$ that satisfies anonymity, non-frameability, traceability, trace-soundness, certificate-unforgeability, non-claimability, and opening-unforgeability, but is not key-unforgeable.*

Intuition. The wrapper augments every membership certificate with an independently sampled auxiliary value x. When the opener public key contains the switch bit $b_{\mathsf{kunf}} = 1$, the signing oracle returns an authenticated mode-1 signature whose underlying message contains

$$h = H(x).$$

In the key-unforgeability experiment, the complete wrapper certificate (σ_s, x) is exposed, whereas the corresponding member secret key remains hidden. The adversary can therefore transform the mode-1 signing-oracle output into a distinct mode-2 signature by revealing x. The transformed signature verifies because $H(x) = h$ and is fresh because its wrapper encoding differs from the oracle output. Thus, the adversary obtains a fresh valid signature without learning the challenger-generated member secret key, violating key-unforgeability.

The proof is given in Section 6.1.7.

Theorem 32 (Independence of Opening-Unforgeability)**.** *There exists a group signature scheme* $\Gamma^{\neg\mathsf{OpU}}$ *that satisfies anonymity, non-frameability, traceability, trace-soundness, non-claimability, key-unforgeability, and certificate-unforgeability, but is not opening-unforgeable.*

Intuition. The wrapper uses disjoint encodings $(0, \mathsf{upk})$ for ordinary member identities and $(1, t)$ for gadget identities. Honestly generated opener keys use the canonical tag hon, whereas a malicious opener may choose a noncanonical tag $t \neq \mathsf{hon}$. The adversary locally generates an ordinary member key pair, submits the ordinary identity $(0, \mathsf{upk})$ together with the malicious opener key to the honest join oracle, and obtains a valid membership certificate. Using the corresponding member secret key and this certificate, it can locally create a valid mode-1 baseline signature. The gadget branch of Jdg'_Γ accepts this signature as an attribution to $(1, t)$. However, the join oracle recorded only the ordinary identity $(0, \mathsf{upk})$, not the gadget identity $(1, t)$. The resulting accepted attribution therefore lies outside the honest join relation and violates opening-unforgeability.

The proof is given in Section 6.1.8.

6.1 Proofs of Separation

Challenge-parameter convention. Whenever a reduction embeds a primitive challenge, it uses the public parameters supplied with that challenge throughout the simulated experiment. In particular, an EUF-CMA reduction embeds the challenger's pp_Σ together with its verification key; it does not resample pp_Σ. Any other signature keys needed by the simulation are generated under that same pp_Σ. Generating the "remaining public parameters" means generating only the other, independently sampled components, with relation-dependent setups using the resulting public parameters.

Conventions. Let Γ be a baseline group signature scheme satisfying all eight security notions defined in Section 5. Throughout this section, we assume that its message space supports an efficiently computable injective encoding of polynomial-length tuples; in particular, one may take $\mathcal{MS} = \{0,1\}^*$. This restriction is sufficient for the existential separation results below. For each separation, we construct a derived scheme as a black-box wrapper around Γ.

All wrapper keys have the fixed tuple structure prescribed by the respective construction. Whenever such a tuple contains an *anchor*, that component ranges over the tagged domain

$$\mathsf{Anchor} := \{\mathsf{hon}\} \uplus \{0,1\}^*.$$

Honest anchor generation uses the distinguished value hon, whereas adversarially generated keys may contain arbitrary values from Anchor. We suppress the canonical injection of bitstrings into the second summand. In particular, hon is syntactically distinct from every

adversarial bitstring payload while still admitting an injective encoding into $\{0,1\}^*$. Inputs that do not parse according to the prescribed wrapper-key format are rejected according to the global parsing convention. Using hon rather than $\perp$ avoids confusing an honest anchor value with algorithmic failure. Boolean branch selectors used in some constructions are not anchors and retain their explicitly specified honest value 0.

Every message passed to an underlying signing, verification, opening, or opening-verification algorithm is the complete derived message defined locally for that wrapper. Tuple notation denotes the fixed efficiently computable injective encoding enc from Section 2. Each such message contains the original message, the complete wrapper-level issuer and opener public keys (including anchors), and any locally specified branch or member data. In particular, m_{upk}, $\widehat{m}$, $\widehat{m}_b$, and $\widehat{m}_1(h)$ below are actual baseline messages, not inputs to an additional implicit encoding. The same local definition is used in all four underlying algorithms and in the reductions. By the message-space convention above, these encodings are valid baseline messages.

Finally, certificate-unforgeability implies that independently generated issuer public keys collide only with negligible probability. Indeed, suppose that

$$(\mathsf{isk}_0, \mathsf{ipk}_0), (\mathsf{isk}_1, \mathsf{ipk}_1) \leftarrow_{\$} \mathsf{KgI}_{\Gamma}(\mathsf{pp}_{\Gamma})$$

satisfy $\mathsf{ipk}_0 = \mathsf{ipk}_1$ with non-negligible probability. An adversary in the certificate-unforgeability experiment can independently generate $(\mathsf{isk}_1, \mathsf{ipk}_1)$ and, whenever ipk_1 equals the challenge issuer public key ipk_0, use isk_1 to issue its own membership certificate and produce a fresh verifying group signature under ipk_0. By correctness, this attack succeeds whenever the two public keys coincide, contradicting certificate-unforgeability. Hence issuer-public-key collision probability is necessarily negligible and need not be imposed as an additional assumption.

6.1.1 Proof of Theorem 25 (Anonymity)

Proof. We provide a counterexample via the following black-box wrapper of Γ.

We first note that independently generated member public keys coincide only with negligible probability. This follows from non-frameability of Γ. Indeed, suppose that two independent executions of KgM_{Γ} produced the same public key with non-negligible probability. An adversary in the non-frameability experiment could independently generate

$$(\mathsf{usk}', \mathsf{upk}') \leftarrow_{\$} \mathsf{KgM}_{\Gamma}(\mathsf{pp}_{\Gamma})$$

and, whenever $\mathsf{upk}' = \mathsf{upk}$, generate its own issuer and opener key pairs, issue a certificate for upk', and produce a signature using usk'. By correctness, the resulting signature can be opened to $\mathsf{upk}' = \mathsf{upk}$ and the opening is accepted. Since the adversary makes no signing-oracle query, the transcript is fresh, contradicting non-frameability. Hence the collision probability is negligible.

Correctness. Consider an honestly generated wrapper signature. Its cleartext member-key component equals the signer public key, and its embedded issuer and opener public keys equal the external group context. The underlying signature is generated and verified on the same derived message

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}).$$

Likewise, opening and opening verification invoke the underlying algorithms on the same derived message and group context. Correctness of the wrapper therefore follows directly from correctness of Γ.

All unlisted algorithms are inherited unchanged from Γ. Define

$$m_{\mathsf{upk}} := (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}).$$

This is the complete baseline message, including the group context, according to the global convention.

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m)}$**:**
Compute

$$\sigma^\star_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, m_{\mathsf{upk}}).$$

Return $\perp$ if $\sigma^\star_m = \perp$. Otherwise return

$$\sigma'_m \leftarrow (\sigma^\star_m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}).$$

$\underline{\mathsf{Vfy}'_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma'_m)}$**:**
Parse

$$\sigma'_m = (\sigma^\star_m, \mathsf{upk}, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}).$$

Return 0 if

$$(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}) \neq (\mathsf{ipk}, \mathsf{opk}).$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m_{\mathsf{upk}}, \sigma^\star_m).$$

$\underline{\mathsf{Opn}'_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, m, \sigma'_m)}$**:**
Parse

$$\sigma'_m = (\sigma^\star_m, \mathsf{upk}, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}).$$

Return $\perp$ if

$$(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}) \neq (\mathsf{ipk}, \mathsf{opk}).$$

Otherwise return

$$\mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, m_{\mathsf{upk}}, \sigma^\star_m).$$

$\underline{\mathsf{Jdg}'_\Gamma(\mathsf{ipk}, \mathsf{opk}, m, \sigma'_m, \mathsf{upk}^*, O)}$**:**
Parse

$$\sigma'_m = (\sigma^\star_m, \mathsf{upk}, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}).$$

Return 0 if

$$(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}) \neq (\mathsf{ipk}, \mathsf{opk}).$$

Otherwise return

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, m_{\mathsf{upk}}, \sigma^\star_m, \mathsf{upk}^*, O).$$

Construction 5: Wrapper $\Gamma^{\neg\mathsf{ANON}}$.

Why anonymity fails. The anonymity adversary generates an issuer key pair

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{Kgl}_\Gamma(\mathsf{pp}_\Gamma)$$

and, using the honest opener public key opk supplied by the experiment, computes

$$\sigma^i_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}_i, \mathsf{opk}) \qquad \text{for } i \in \{0, 1\}.$$

It submits

$$(m, \mathsf{ipk}, \sigma^0_s, \sigma^1_s)$$

to the left-or-right signing oracle.

Both candidate wrapper signatures are valid, and the oracle returns

$$\sigma'_m = (\sigma^\star_m, \mathsf{upk}_b, \mathsf{ipk}, \mathsf{opk})$$

for the hidden challenge bit $b \in \{0, 1\}$. If

$$\mathsf{upk}_0 \neq \mathsf{upk}_1,$$

the adversary determines b by comparing the cleartext member-key component of the challenge signature with upk_0 and upk_1.

Since independently generated member public keys collide only with negligible probability, the adversary determines b with overwhelming probability. Hence $\Gamma^{\neg\mathsf{ANON}}$ is not anonymous.

Why the other seven properties hold. The wrapper appends the signer public key and the complete group public-key context to the signature and consistently binds the signer public key into the message supplied to the underlying scheme. Every accepting wrapper transcript therefore determines a unique accepting baseline transcript on the derived message

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}).$$

- **Traceability (TRC).** Suppose that

$$\sigma'_m = (\sigma^\star_m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk})$$

is a verifying wrapper signature for which the prescribed wrapper opening algorithm fails to return an accepted opening.

The context checks imply that the external issuer and opener keys equal the keys embedded in σ'_m. Consequently, $\sigma^\star_m$ is a verifying baseline signature on

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk})$$

for which the prescribed baseline opening algorithm fails to return an accepted opening. This contradicts traceability of Γ.

- **Trace-Soundness (TS).** Suppose that one fixed wrapper message-signature pair

$$(m, \sigma'_m)$$

admits two different accepted attribution tuples.

Since both invocations of Jdg'_Γ accept the same wrapper signature, the explicit context checks imply

$$\mathsf{ipk}^*_0 = \mathsf{ipk}^*_1 \qquad \text{and} \qquad \mathsf{opk}^*_0 = \mathsf{opk}^*_1.$$

The cleartext member-key component carried in σ'_m is also fixed. Hence both wrapper openings invoke the baseline opening-verification algorithm on the same baseline message

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}),$$

the same baseline signature $\sigma^\star_m$, and the same group public-key pair.

Since the two wrapper attribution tuples differ, their claimed member public keys must differ. They therefore yield two different accepted baseline attributions for the same message-signature pair, contradicting trace-soundness of Γ.

- **Non-Frameability (NF).** A successful wrapper adversary outputs a fresh wrapper signature and opening information accepted as an attribution to the fixed honest member.

Removing the cleartext wrapper components yields an accepted baseline attribution to the same honest member on the derived message

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}).$$

Freshness is preserved. If the corresponding baseline transcript had previously been returned by the baseline signing oracle, then the deterministic wrapper extension by

$$(\mathsf{upk}, \mathsf{ipk}, \mathsf{opk})$$

would imply that the complete wrapper transcript had also previously been returned. Thus, a successful wrapper frame yields a successful frame against Γ.

- **Non-Claimability (NC).** Every wrapper signing-oracle output is obtained by deterministically extending a baseline signing-oracle output with the honest signer public key and the relevant group context. Hence protected wrapper transcripts are in one-to-one correspondence with protected baseline transcripts.

Any accepted reassignment of a logged wrapper transcript to a different member public key therefore induces an accepted reassignment of the corresponding logged baseline transcript, contradicting non-claimability of Γ.

- **Certificate-Unforgeability (CUnf).** Let

$$\sigma'_m = (\sigma^\star_m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk})$$

be a fresh verifying wrapper forgery under the honest issuer key.

Then $\sigma^\star_m$ is a verifying baseline signature on

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk})$$

under the same opener key.

The reduction forwards member-generation and join queries to its baseline certificate-unforgeability experiment and records, for every certificate handle, the corresponding member public key and opener public key. On a wrapper signing query for a handle associated with upk, it queries the baseline signing oracle on m_{upk} and deterministically extends the returned baseline signature.

If this baseline transcript had previously been returned by the baseline signing oracle, the corresponding wrapper transcript would also have been recorded, contradicting wrapper freshness. Thus, the wrapper forgery yields a fresh baseline forgery, contradicting certificate-unforgeability of Γ.

- **Key-Unforgeability (KUnf).** The same argument applies to key-unforgeability.

The reduction forwards member-generation and join queries and records the public key associated with every member index. On a wrapper signing query using index j, it looks up the corresponding public key upk_j, queries the baseline signing oracle on m_{upk_j}, and deterministically extends the returned baseline signature.

A fresh verifying wrapper signature therefore yields a verifying baseline signature under the same issuer–opener context on the derived message

$$m' = (m, \mathsf{upk}, \mathsf{ipk}, \mathsf{opk}).$$

Replay of the baseline transcript would imply replay of the deterministically extended wrapper transcript. Hence wrapper freshness implies baseline freshness, and a successful wrapper forgery contradicts key-unforgeability of Γ.

- **Opening-Unforgeability (OpU).** The wrapper does not modify the join oracle or the join relation.

An accepted wrapper opening to a pair

$$(\mathsf{upk}^*, \mathsf{opk}^*)$$

that is not contained in the wrapper join relation therefore yields an accepted baseline opening to the same non-joined pair, on the derived message determined by the wrapper signature. This contradicts opening-unforgeability of Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{ANON}}$ breaks anonymity by explicitly leaking the signer public key in the signature. At the same time, it preserves traceability, trace-soundness, non-frameability, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability by black-box reduction to Γ. Hence anonymity is independent of the other seven notions.

6.1.2 Proof of Theorem 26 (Non-Frameability)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 6–7.

Extended issuer and opener public keys have the forms

$$\mathsf{ipk}' = (\mathsf{ipk}, y_I^\star) \qquad \text{and} \qquad \mathsf{opk}' = (\mathsf{opk}, y_O^\star).$$

Honestly generated keys use the canonical anchors

$$y_I^\star = y_O^\star = \mathsf{hon}.$$

An adversary controlling a public-key context may instead supply arbitrary well-formed anchors.

The wrapper uses disjoint encodings for ordinary and gadget signatures:

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}') \qquad \text{and} \qquad (1, \sigma_m, \mathsf{ipk}', \mathsf{opk}'),$$

respectively. Ordinary wrapper signatures are generated on the derived message

$$\widehat{m}_0 := (m, 0, \mathsf{ipk}', \mathsf{opk}'),$$

whereas gadget signatures are baseline signatures on

$$\widehat{m}_1 := (m, 1, \mathsf{ipk}', \mathsf{opk}').$$

These are the complete baseline messages under the global convention; no additional context wrapping is applied.

The ordinary branch is enabled only for canonical wrapper group keys. The gadget branch is enabled only if the issuer and opener anchors coincide and are different from hon.

Correctness. Honestly generated issuer and opener keys have canonical anchors:

$$y_I^\star = y_O^\star = \mathsf{hon}.$$

Consequently, Join'_Γ and Sgn'_Γ use the ordinary branch and consistently invoke the underlying algorithms on

$$\widehat{m}_0 = (m, 0, \mathsf{ipk}', \mathsf{opk}').$$

Honest signatures have the form

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}'),$$

and both opening algorithms invoke their baseline counterparts on the same derived message and public-key context. Correctness therefore follows directly from correctness of Γ.

Why non-frameability fails. Let

$$(\mathsf{usk}^{\mathsf{hon}}, \mathsf{upk}^{\mathsf{hon}})$$

be the honest member key pair generated by the non-frameability experiment.

The adversary independently generates baseline issuer and opener key pairs

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_{\$} \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$$

and

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_{\$} \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma),$$

All unlisted algorithms are inherited unchanged from Γ. Malformed inputs are rejected according to the global parsing convention. For

$$\mathsf{ipk}' = (\mathsf{ipk}, y_I^\star), \qquad \mathsf{opk}' = (\mathsf{opk}, y_O^\star),$$

define

$$\begin{aligned} \mathsf{Hon} &:= [y_I^\star = y_O^\star = \mathsf{hon}], \\ \mathsf{Gad} &:= [y_I^\star = y_O^\star \neq \mathsf{hon}], \\ \widehat{m}_b &:= (m, b, \mathsf{ipk}', \mathsf{opk}'), \\ \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') &:= [(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') = (\mathsf{ipk}', \mathsf{opk}')]. \end{aligned}$$

$\underline{\mathsf{KgI}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Run

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{isk}', \mathsf{ipk}') \leftarrow (\mathsf{isk}, (\mathsf{ipk}, \mathsf{hon})).$$

$\underline{\mathsf{KgO}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Run

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{osk}', \mathsf{opk}') \leftarrow (\mathsf{osk}, (\mathsf{opk}, \mathsf{hon})).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}', \mathsf{ipk}', \mathsf{upk}, \mathsf{opk}')}$**:**
Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, y_I^\star), \qquad \mathsf{opk}' = (\mathsf{opk}, y_O^\star),$$

and set $\mathsf{isk} \leftarrow \mathsf{isk}'$.
Return $\perp$ unless Hon. Otherwise compute

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk})$$

and return σ_s.

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}', \mathsf{opk}', m)}$**:**
Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, y_I^\star), \qquad \mathsf{opk}' = (\mathsf{opk}, y_O^\star).$$

Return $\perp$ unless Hon.
Compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}_0).$$

Return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$\sigma'_m \leftarrow (0, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

$\underline{\mathsf{Vfy}'_\Gamma(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m)}$**:**
Parse

$$\begin{aligned} \mathsf{ipk}' &= (\mathsf{ipk}, y_I^\star), \\ \mathsf{opk}' &= (\mathsf{opk}, y_O^\star), \\ \sigma'_m &= (b, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'). \end{aligned}$$

Return 0 unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

If $b = 0$, return 0 unless Hon; otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_0, \sigma_m).$$

If $b = 1$, return 0 unless Gad; otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_1, \sigma_m).$$

Return 0.

Construction 6: Wrapper $\Gamma^{\neg\mathsf{NF}}$ (Part I of II)

as well as an adversarial member key pair

$$(\widetilde{\mathsf{usk}}, \widetilde{\mathsf{upk}}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma).$$

It defines

$$\mathsf{ipk}' = (\mathsf{ipk}, \mathsf{upk}^{\mathsf{hon}}) \qquad \text{and} \qquad \mathsf{opk}' = (\mathsf{opk}, \mathsf{upk}^{\mathsf{hon}}).$$

Using the underlying scheme, it computes

$$\widetilde{\sigma}_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \widetilde{\mathsf{upk}}, \mathsf{opk}).$$

For any message m, set

$$\widehat{m}_1 = (m, 1, \mathsf{ipk}', \mathsf{opk}')$$

and compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\widetilde{\mathsf{usk}}, \widetilde{\mathsf{upk}}, \widetilde{\sigma}_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}_1).$$

The adversary aborts if any of these algorithms returns $\perp$. Otherwise it outputs the wrapper signature

$$\sigma'_m = (1, \sigma_m, \mathsf{ipk}', \mathsf{opk}')$$

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Opn}'_{\Gamma}(\mathsf{osk}', \mathsf{opk}', \mathsf{ipk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, y_I^\star),$$
$$\mathsf{opk}' = (\mathsf{opk}, y_O^\star),$$
$$\sigma'_m = (b, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

and set $\mathsf{osk} \leftarrow \mathsf{osk}'$.

Return $\perp$ unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') \wedge (b = 0) \wedge \mathsf{Hon}.$$

Compute

$$(\mathsf{upk}, O) \leftarrow_{\$} \mathsf{Opn}_{\Gamma}(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, \widehat{m}_0, \sigma_m).$$

Return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$(\mathsf{upk}, O).$$

$\underline{\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, y_I^\star),$$
$$\mathsf{opk}' = (\mathsf{opk}, y_O^\star),$$
$$\sigma'_m = (b, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Return 0 unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

If $b = 1$, return 0 unless Gad. Return 0 unless

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_1, \sigma_m) = 1.$$

Return 1 iff

$$\mathsf{upk}^* = y_I^\star.$$

If $b = 0$, return 0 unless Hon. Otherwise return

$$\mathsf{Jdg}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_0, \sigma_m, \mathsf{upk}^*, O).$$

Return 0.

Construction 7: Wrapper $\Gamma^{\neg\mathsf{NF}}$ (Part II of II)

and opening information

$$O^* = 0.$$

By correctness of the underlying scheme,

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_1, \sigma_m) = 1.$$

Since

$$y_I^\star = y_O^\star = \mathsf{upk}^{\mathsf{hon}} \neq \mathsf{hon},$$

we obtain

$$\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m) = 1$$

and

$$\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^{\mathsf{hon}}, O^*) = 1.$$

The wrapper signing oracle never returns a signature with mode bit 1. Therefore the output is fresh. Nevertheless, it is accepted as an attribution to the fixed honest member public key. Hence $\Gamma^{\neg\mathsf{NF}}$ is not non-frameable.

Why the other seven properties hold.

- **Anonymity (ANON).** The opener key is honestly generated and therefore satisfies

  $$y_O^\star = \mathsf{hon}.$$

  Hence the gadget branch is disabled. Every accepting challenge signature must use mode 0, which additionally requires

  $$y_I^\star = \mathsf{hon}.$$

  For canonical issuer and opener contexts, the challenge-signing and opening oracles coincide with the corresponding baseline oracles on the derived message

  $$\widehat{m}_0 = (m, 0, \mathsf{ipk}', \mathsf{opk}').$$

  For noncanonical contexts, the wrapper signing oracle returns $\perp$ on both sides, independently of the challenge bit. The public context embedded in the wrapper signature carries no member-dependent information. Thus, any anonymity adversary against the wrapper yields one against Γ.

- **Traceability (TRC).** The opener key is honestly generated, so

$$y_O^\star = \mathsf{hon}.$$

A mode-1 signature cannot verify, because the gadget branch requires matching noncanonical anchors. A mode-0 signature verifies only if the issuer key is also canonical:

$$y_I^\star = \mathsf{hon}.$$

Every verifying wrapper signature therefore corresponds to a verifying baseline signature on $\widehat{m}_0$, and the prescribed wrapper opening algorithm invokes the prescribed baseline opening algorithm on that same transcript. A successful wrapper traceability attack would thus contradict traceability of Γ.

- **Trace-Soundness (TS).** Consider a fixed wrapper message-signature pair admitting two accepted attribution tuples. Since the wrapper signature explicitly contains its complete public-key context, both accepted interpretations must use the same issuer and opener public keys. They must also use the same mode bit.

If the mode bit is (1), the gadget branch accepts only the unique member public key

$$\mathsf{upk}^* = y_I^\star = y_O^\star.$$

Hence the two attribution tuples cannot differ.

If the mode bit is (0), both keys are canonical and both openings reduce to baseline openings for the same message $\widehat{m}_0$ and the same baseline signature. Two different accepted wrapper attributions would therefore contradict trace-soundness of Γ.

- **Non-Claimability (NC).** Every non-bottom wrapper signing-oracle output has mode bit 0 and is generated under canonical issuer and opener keys. A successful non-claimability adversary must reuse an exact logged wrapper transcript, so it cannot replace that mode bit or its embedded group context.

Every accepted reassignment must therefore occur through the ordinary branch and yields an accepted reassignment of the corresponding logged baseline transcript on $\widehat{m}_0$. This contradicts non-claimability of Γ.

- **Certificate-Unforgeability (CUnf).** The issuer key is honestly generated and thus satisfies

$$y_I^\star = \mathsf{hon}.$$

Consequently, the gadget branch is disabled. An accepting mode-0 signature additionally requires a canonical opener key. Removing the deterministic wrapper encoding yields an accepting baseline signature on the corresponding derived message.

If the baseline transcript had previously been returned by the baseline signing oracle, the complete mode-0 wrapper transcript would also have been recorded. Hence wrapper freshness implies baseline freshness, and a successful wrapper forgery contradicts certificate-unforgeability of Γ.

- **Key-Unforgeability (KUnf).** The final issuer key is honestly generated and therefore canonical. A successful final forgery cannot use mode 1, and mode 0 additionally requires a canonical opener key.

Signing-oracle queries under noncanonical extended contexts return $\perp$. For canonical contexts, the wrapper signing oracle is a deterministic encoding of the baseline signing oracle on $\widehat{m}_0$. Thus, removing the wrapper encoding from a fresh final forgery yields a fresh accepting baseline forgery, contradicting key-unforgeability of Γ.

- **Opening-Unforgeability (OpU).** The issuer key is honestly generated, so the gadget branch is disabled. Moreover, both Join'_Γ and the ordinary opening-verification branch accept only canonical opener keys. The wrapper join relation is therefore in one-to-one correspondence with the baseline join relation:

$$\big(\mathsf{upk}, (\mathsf{opk}, \mathsf{hon})\big) \longleftrightarrow (\mathsf{upk}, \mathsf{opk}).$$

  An accepted wrapper attribution to a pair outside the wrapper join relation yields an accepted baseline attribution to the corresponding pair outside the baseline join relation, contradicting opening-unforgeability of Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{NF}}$ preserves anonymity, traceability, trace-soundness, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability by black-box reduction to Γ. Nevertheless, colluding issuer and opener contexts can activate the gadget branch and attribute a fresh mode-1 signature to the fixed honest member public key. Hence non-frameability is independent of the other seven notions.

6.1.3 Proof of Theorem 27 (Traceability)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 8–9.

Extended issuer public keys have the form

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}}),$$

where honestly generated issuer keys always satisfy

$$b_{\mathsf{trc}} = 0.$$

An adversary controlling the issuer public key may instead supply an arbitrary well-formed extended key, in particular one with

$$b_{\mathsf{trc}} = 1.$$

To obtain an injective correspondence between wrapper transcripts and baseline transcripts, every wrapper signature explicitly carries the complete wrapper group context. For a wrapper message m, define the derived baseline message

$$\widehat{m} := (m, \mathsf{ipk}', \mathsf{opk}).$$

As specified by the global convention, $\widehat{m}$ is the complete baseline message; no additional context wrapping is applied.

Correctness. Honestly generated issuer keys satisfy

$$b_{\mathsf{trc}} = 0.$$

All wrapper algorithms therefore invoke the corresponding baseline algorithms on the same derived message

$$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}).$$

The group context embedded in an honestly generated wrapper signature also equals the external verification context. Correctness of the wrapper follows directly from correctness of Γ.

All unlisted algorithms are inherited unchanged from Γ. Malformed inputs are rejected according to the global parsing convention. For

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}}), \qquad \widehat{m} := (m, \mathsf{ipk}', \mathsf{opk}),$$

define

$$\mathsf{Ctx} := \Big[\sigma'_m = (\sigma_m, \mathsf{ipk}_{\mathsf{emb}}, \mathsf{opk}_{\mathsf{emb}}) \wedge \mathsf{ipk}_{\mathsf{emb}} = \mathsf{ipk}' \wedge \mathsf{opk}_{\mathsf{emb}} = \mathsf{opk}\Big].$$

$\underline{\mathsf{Kgl}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Run

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{Kgl}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{isk}', \mathsf{ipk}') \leftarrow \Big(\mathsf{isk}, (\mathsf{ipk}, 0)\Big).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}', \mathsf{ipk}', \mathsf{upk}, \mathsf{opk})}$**:**
Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}})$$

and set

$$\mathsf{isk} \leftarrow \mathsf{isk}'.$$

Compute

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk})$$

and return σ_s.

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}', \mathsf{opk}, m)}$**:**
Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}}).$$

Compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}).$$

Return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$\sigma'_m \leftarrow (\sigma_m, \mathsf{ipk}', \mathsf{opk}).$$

$\underline{\mathsf{Vfy}'_\Gamma(\mathsf{ipk}', \mathsf{opk}, m, \sigma'_m)}$**:**
Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}})$$

and

$$\sigma'_m = (\sigma_m, \mathsf{ipk}_{\mathsf{emb}}, \mathsf{opk}_{\mathsf{emb}}).$$

Return 0 unless Ctx. Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m).$$

Construction 8: Wrapper $\Gamma^{\neg\mathsf{TRC}}$ disabling opening under an adversarial issuer context (Part I of II)

Why traceability fails. In the traceability experiment, the adversary receives the honestly generated opener key pair

$$(\mathsf{osk}, \mathsf{opk}).$$

It independently generates

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{Kgl}_\Gamma(\mathsf{pp}_\Gamma)$$

and

$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma).$$

It defines the adversarial issuer public key

$$\mathsf{ipk}' = (\mathsf{ipk}, 1)$$

and computes

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk}).$$

For any message m, it computes the wrapper signature

$$\sigma'_m \leftarrow_\$ \mathsf{Sgn}'_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}', \mathsf{opk}, m).$$

Writing

$$\sigma'_m = (\sigma_m, \mathsf{ipk}', \mathsf{opk}),$$

the underlying signature σ_m is generated on

$$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}).$$

Consequently,

$$\mathsf{Vfy}'_\Gamma(\mathsf{ipk}', \mathsf{opk}, m, \sigma'_m) = 1.$$

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Opn}'_{\Gamma}(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}})$$

and

$$\sigma'_m = (\sigma_m, \mathsf{ipk}_{\mathsf{emb}}, \mathsf{opk}_{\mathsf{emb}}).$$

Return $\perp$ unless

$$\mathsf{Ctx} \wedge (b_{\mathsf{trc}} = 0).$$

Compute

$$(\mathsf{upk}, O) \leftarrow_{\$} \mathsf{Opn}_{\Gamma}(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, \widehat{m}, \sigma_m).$$

Return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$(\mathsf{upk}, O).$$

$\underline{\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}, m, \sigma'_m, \mathsf{upk}^*, O)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, b_{\mathsf{trc}})$$

and

$$\sigma'_m = (\sigma_m, \mathsf{ipk}_{\mathsf{emb}}, \mathsf{opk}_{\mathsf{emb}}).$$

Return 0 unless

$$\mathsf{Ctx} \wedge (b_{\mathsf{trc}} = 0).$$

Otherwise return

$$\mathsf{Jdg}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m, \mathsf{upk}^*, O).$$

Construction 9: Wrapper $\Gamma^{\neg\mathsf{TRC}}$ disabling opening under an adversarial issuer context (Part II of II)

However, since the issuer key satisfies

$$b_{\mathsf{trc}} = 1,$$

the prescribed opening algorithm returns

$$\mathsf{Opn}'_{\Gamma}(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}', m, \sigma'_m) = \perp.$$

The adversary therefore wins the traceability experiment with probability 1. Hence $\Gamma^{\neg\mathsf{TRC}}$ is not traceable.

Why the other seven properties hold.

- **Anonymity (ANON).** The wrapper does not expose any member-dependent information beyond the baseline signature. A wrapper anonymity adversary can choose either value of b_{trc} in its issuer public key.

  For every left-or-right signing query, the reduction forwards the corresponding baseline query using the derived message

  $$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk})$$

  and wraps the returned signature with the public context

  $$(\mathsf{ipk}', \mathsf{opk}).$$

  For opening queries with

  $$b_{\mathsf{trc}} = 1,$$

  the reduction returns $\perp$ directly. For queries with

  $$b_{\mathsf{trc}} = 0,$$

  it forwards the corresponding non-challenge query to the baseline opening oracle.

  Since the embedded group context and the derived message uniquely determine one another, wrapper challenge transcripts correspond injectively to baseline challenge transcripts. The simulation is therefore perfect, and any anonymity adversary against the wrapper yields one against Γ.

- **Non-Frameability (NF).** A successful frame requires

$$\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}'^{*}, \mathsf{opk}^{*}, m^{*}, \sigma'^{*}_{m}, \mathsf{upk}, O^{*}) = 1.$$

  This is impossible when

$$b^{*}_{\mathsf{trc}} = 1.$$

  Hence every successful frame must satisfy

$$b^{*}_{\mathsf{trc}} = 0$$

  and must occur through the baseline branch.

  Removing the deterministic context wrapper from σ'^{*}_{m} yields an accepted baseline attribution to the same honest member on the derived message

$$\widehat{m}^{*} = (m^{*}, \mathsf{ipk}'^{*}, \mathsf{opk}^{*}).$$

  Freshness is preserved because a replay of the baseline transcript under the same derived message and group context would imply a replay of the complete wrapper transcript. Thus, a successful wrapper frame contradicts non-frameability of Γ.

- **Trace-Soundness (TS).** Suppose that a fixed wrapper message-signature pair

$$(m^{*}, \sigma'^{*}_{m})$$

  admits two different accepted attribution tuples.

  Both accepting invocations of Jdg'_{Γ} require

$$b^{*}_{\mathsf{trc},0} = b^{*}_{\mathsf{trc},1} = 0.$$

  Moreover, because the same wrapper signature explicitly contains its issuer and opener public keys, both accepted invocations must use the same wrapper group context:

$$\mathsf{ipk}'^{*}_{0} = \mathsf{ipk}'^{*}_{1} \qquad \text{and} \qquad \mathsf{opk}^{*}_{0} = \mathsf{opk}^{*}_{1}.$$

  They therefore invoke the baseline opening-verification algorithm on the same derived message

$$\widehat{m}^{*} = (m^{*}, \mathsf{ipk}'^{*}_{0}, \mathsf{opk}^{*}_{0})$$

  and the same baseline signature.

  Since the two wrapper attribution tuples differ while their group contexts coincide, their claimed member public keys must differ. This yields two different accepted baseline attributions for the same message-signature pair, contradicting trace-soundness of Γ.

- **Non-Claimability (NC).** The issuer and opener contexts supplied to the signing oracle may be adversarial.

  If a protected transcript was generated under an issuer key with

$$b_{\mathsf{trc}} = 1,$$

  no opening for that transcript can be accepted by Jdg'_{Γ}, so it cannot yield a successful non-claimability attack.

  If instead

$$b_{\mathsf{trc}} = 0,$$

  an accepted reassignment occurs through the baseline branch. The logged wrapper transcript corresponds injectively to a logged baseline transcript on the derived message, and an accepted attribution to a different member public key therefore contradicts non-claimability of Γ.

- **Certificate-Unforgeability (CUnf).** The issuer public key is honestly generated and therefore satisfies
$$b_{\mathsf{trc}} = 0.$$
Every accepting wrapper signature consequently uses the baseline branch.

  Removing the deterministic context wrapper yields an accepting baseline signature under the honest issuer key on the corresponding derived message. If that baseline transcript had previously been returned by the baseline signing oracle, the complete wrapper transcript would also have been recorded. Thus, wrapper freshness implies baseline freshness, and any successful wrapper forgery contradicts certificate-unforgeability of Γ.

- **Key-Unforgeability (KUnf).** The final verification key in the key-unforgeability experiment is the honestly generated issuer key and hence has
$$b_{\mathsf{trc}} = 0.$$
Therefore every successful final forgery is accepted through the baseline branch.

  The signing oracle may be queried under arbitrary extended issuer keys, but each returned wrapper signature explicitly records that issuer context. Consequently, a transcript obtained under an issuer key with $b_{\mathsf{trc}} = 1$, or under any other issuer context, is not converted into a replay under the honest issuer key. Removing the wrapper context from a fresh final forgery yields a fresh accepting baseline forgery, contradicting key-unforgeability of Γ.

- **Opening-Unforgeability (OpU).** The issuer key is honestly generated and therefore satisfies
$$b_{\mathsf{trc}} = 0.$$
The wrapper join oracle and the baseline join oracle induce the same relation on pairs
$$(\mathsf{upk}, \mathsf{opk}).$$
Every accepted wrapper opening consequently corresponds to an accepted baseline opening on the derived message.

  An accepted wrapper attribution to a pair outside the wrapper join relation would therefore yield an accepted baseline attribution to the same pair outside the baseline join relation, contradicting opening-unforgeability of Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{TRC}}$ preserves anonymity, trace-soundness, non-frameability, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability by black-box reduction to Γ, while an adversarial issuer can set $b_{\mathsf{trc}} = 1$ and thereby produce verifying signatures for which the prescribed opening algorithm always fails. Hence traceability is independent of the other seven notions.

6.1.4 Proof of Theorem 28 (Trace-Soundness)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 10–12.

The wrapper uses disjoint encodings for ordinary and gadget signatures:
$$(0, \sigma_m) \qquad \text{and} \qquad (1, \tau),$$

respectively. Likewise, ordinary member public keys and gadget identities are encoded as

$$(0, \mathsf{upk}) \qquad \text{and} \qquad (1, t).$$

Thus, no honestly generated member public key is syntactically equal to a gadget identity.

An extended issuer public key and opener public key have the forms

$$\mathsf{ipk}' = (\mathsf{ipk}, a_I, T_I) \qquad \text{and} \qquad \mathsf{opk}' = (\mathsf{opk}, a_O, T_O),$$

where $a_I, a_O \in \mathsf{Anchor}$ and T_I, T_O are finite sets of bitstrings. We call an extended key *canonical* if its anchor is hon and its tag set is empty.

The ordinary baseline branch is enabled only when both extended public keys are canonical. The gadget branch is enabled only when the issuer and opener anchors agree on the gadget-signature tag.

All unlisted algorithms are inherited unchanged from Γ. Let

$$\widehat{m} := (m, \mathsf{ipk}', \mathsf{opk}'), \qquad \mathsf{Can}_I := \big[(a_I, T_I) = (\mathsf{hon}, \emptyset)\big], \qquad \mathsf{Can}_O := \big[(a_O, T_O) = (\mathsf{hon}, \emptyset)\big].$$

The underlying scheme binds the complete wrapper group context according to the global convention.

$\underline{\mathsf{KgM}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run
$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$$
and return
$$(\mathsf{usk}', \mathsf{upk}') \leftarrow (\mathsf{usk}, (0, \mathsf{upk})).$$

$\underline{\mathsf{KgI}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run
$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$$
and return
$$(\mathsf{isk}', \mathsf{ipk}') \leftarrow (\mathsf{isk}, (\mathsf{ipk}, \mathsf{hon}, \emptyset)).$$

$\underline{\mathsf{KgO}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run
$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$$
and return
$$(\mathsf{osk}', \mathsf{opk}') \leftarrow (\mathsf{osk}, (\mathsf{opk}, \mathsf{hon}, \emptyset)).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}', \mathsf{ipk}', \mathsf{upk}', \mathsf{opk}')}$:
Parse
$$\mathsf{ipk}' = (\mathsf{ipk}, a_I, T_I),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O, T_O),$$
$$\mathsf{upk}' = (b_M, z_M),$$
and set $\mathsf{isk} \leftarrow \mathsf{isk}'$.
Return $\perp$ unless
$$\mathsf{Can}_I \wedge \mathsf{Can}_O \wedge (b_M = 0).$$
Set $\mathsf{upk} \leftarrow z_M$ and compute
$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk}).$$
Return σ_s.

Construction 10: Wrapper $\Gamma^{\neg\mathsf{TS}}$ (Part I of III)

Correctness. For honestly generated keys, all anchors are hon, all tag sets are empty, member public keys have ordinary form $(0, \mathsf{upk})$, and signatures have ordinary form $(0, \sigma_m)$. All wrapper algorithms therefore use the ordinary branch and consistently invoke the underlying algorithms on

$$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}').$$

Correctness follows directly from correctness of Γ.

Why trace-soundness fails. The trace-soundness adversary first generates any well-formed baseline public keys

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma) \qquad \text{and} \qquad (\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma).$$

It chooses a noncanonical gadget tag

$$\tau^\star \neq \mathsf{hon}$$

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Sgn}'_{\Gamma}(\mathsf{usk}', \mathsf{upk}', \sigma_s, \mathsf{ipk}', \mathsf{opk}', m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_I, T_I),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O, T_O),$$
$$\mathsf{upk}' = (b_M, z_M),$$

and set $\mathsf{usk} \leftarrow \mathsf{usk}'$.
Return $\perp$ unless

$$\mathsf{Can}_I \wedge \mathsf{Can}_O \wedge (b_M = 0).$$

Set $\mathsf{upk} \leftarrow z_M$ and compute

$$\sigma_m \leftarrow_{\$} \mathsf{Sgn}_{\Gamma}(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}).$$

Return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$\sigma'_m \leftarrow (0, \sigma_m).$$

$\underline{\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_I, T_I),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O, T_O).$$

If

$$\sigma'_m = (1, \tau) \quad \wedge \quad a_I = a_O = \tau \neq \mathsf{hon},$$

return 1.
Otherwise parse $\sigma'_m = (0, \sigma_m)$. Return 0 unless

$$\mathsf{Can}_I \wedge \mathsf{Can}_O.$$

Return

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m).$$

Construction 11: Wrapper $\Gamma^{\neg\mathsf{TS}}$ (Part II of III)

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Opn}'_{\Gamma}(\mathsf{osk}', \mathsf{opk}', \mathsf{ipk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_I, T_I),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O, T_O),$$

and set $\mathsf{osk} \leftarrow \mathsf{osk}'$.
Return $\perp$ unless

$$\mathsf{Can}_I \wedge \mathsf{Can}_O \wedge \left[\sigma'_m = (0, \sigma_m)\right].$$

Compute

$$(\mathsf{upk}, O) \leftarrow_{\$} \mathsf{Opn}_{\Gamma}(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, \widehat{m}, \sigma_m).$$

Return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$((0, \mathsf{upk}), O).$$

$\underline{\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_I, T_I),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O, T_O).$$

If

$$\sigma'_m = (1, \tau) \quad \wedge \quad a_I = a_O = \tau \neq \mathsf{hon},$$

parse $\mathsf{upk}^* = (b_M, t)$ and return 1 iff

$$b_M = 1 \quad \wedge \quad t \in T_I \cap T_O.$$

Otherwise parse

$$\sigma'_m = (0, \sigma_m), \qquad \mathsf{upk}^* = (b_M, z_M).$$

Return 0 unless

$$\mathsf{Can}_I \wedge \mathsf{Can}_O \wedge (b_M = 0).$$

Set $\mathsf{upk} \leftarrow z_M$ and return

$$\mathsf{Jdg}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m, \mathsf{upk}, O).$$

Construction 12: Wrapper $\Gamma^{\neg\mathsf{TS}}$ (Part III of III)

and two distinct bitstrings

$$t_0 \neq t_1.$$

It then defines

$$T^\star = \{t_0, t_1\}$$

and sets

$$\mathsf{ipk}_0^* = \mathsf{ipk}_1^* = (\mathsf{ipk}, \tau^\star, T^\star)$$

and

$$\mathsf{opk}_0^* = \mathsf{opk}_1^* = (\mathsf{opk}, \tau^\star, T^\star).$$

For any message m^*, let

$$\sigma_m^* = (1, \tau^\star).$$

By construction,

$$\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}_i^*, \mathsf{opk}_i^*, m^*, \sigma_m^*) = 1$$

for both $i \in \{0, 1\}$.

The adversary outputs the two candidate attributions

$$\mathsf{upk}_0^* = (1, t_0), \qquad O_0^* = 0,$$

and

$$\mathsf{upk}_1^* = (1, t_1), \qquad O_1^* = 0.$$

Since

$$t_0, t_1 \in T^\star,$$

both

$$\mathsf{Jdg}'_\Gamma(\mathsf{ipk}_i^*, \mathsf{opk}_i^*, m^*, \sigma_m^*, \mathsf{upk}_i^*, O_i^*) = 1.$$

However,

$$(\mathsf{upk}_0^*, \mathsf{ipk}_0^*, \mathsf{opk}_0^*) \neq (\mathsf{upk}_1^*, \mathsf{ipk}_1^*, \mathsf{opk}_1^*)$$

because

$$\mathsf{upk}_0^* \neq \mathsf{upk}_1^*.$$

Hence $\Gamma^{\neg\mathsf{TS}}$ is not trace-sound.

Why the other seven properties hold. The gadget branch is disabled whenever either the issuer or opener key is honestly generated. Non-frameability and non-claimability require separate arguments because both public-key contexts may be adversarial.

- **Anonymity (ANON).** The opener key is honestly generated and therefore has canonical anchor components

  $$(a_O, T_O) = (\mathsf{hon}, \emptyset).$$

  Consequently, the gadget branch is disabled. Any accepting challenge signature must use the ordinary branch, which additionally forces the adversarial issuer key to be canonical. The experiment then coincides with the baseline anonymity experiment on the derived messages

  $$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}').$$

  Thus, any anonymity adversary against the wrapper yields one against Γ.

- **Traceability (TRC).** The opener key is honestly generated, so the gadget branch is disabled. Noncanonical issuer keys cannot produce accepting ordinary signatures. Every verifying wrapper signature therefore corresponds to a verifying baseline signature under canonical group keys. A failure of the prescribed wrapper opening procedure consequently yields a failure of the baseline opening procedure, contradicting traceability of Γ.

- **Non-Frameability (NF).** An honestly generated member public key has ordinary form

  $$(0, \mathsf{upk}).$$

  The gadget branch accepts only gadget identities of the form

  $$(1, t).$$

  Hence the gadget branch can never attribute a signature to the fixed honest member. Any successful frame must use the ordinary branch and therefore induces a successful frame against Γ. Freshness is preserved by the deterministic ordinary-signature encoding

  $$\sigma'_m = (0, \sigma_m).$$

- **Non-Claimability (NC).** Every non-bottom signing-oracle output has ordinary form
$$(0, \sigma_m).$$
Since the adversary must reuse an exact logged signing transcript, the gadget branch, which requires a signature of the form
$$(1, \tau),$$
cannot be used in a successful non-claimability attack. Any accepted reassignment must therefore occur through the ordinary branch and yields an accepted reassignment of the corresponding baseline transcript, contradicting non-claimability of $\mathsf{\Gamma}$.

- **Certificate-Unforgeability (CUnf).** The issuer key is honestly generated and therefore canonical. The gadget branch is disabled. Any accepting wrapper forgery must also use a canonical opener key and an ordinary signature
$$(0, \sigma_m).$$
Removing the deterministic outer tag yields a fresh accepting baseline forgery. A baseline replay would imply a wrapper replay, so freshness is preserved.

- **Key-Unforgeability (KUnf).** Again, the honest issuer key disables the gadget branch. Every accepting wrapper forgery must use the ordinary branch and maps directly to an accepting baseline forgery under the corresponding derived message. Deterministic wrapping preserves replay freshness.

- **Opening-Unforgeability (OpU).** The honest issuer key disables the gadget branch. Moreover, the ordinary join and opening branches accept only canonical opener keys. Consequently, the wrapper join relation is in one-to-one correspondence with the baseline join relation:
$$\big((0, \mathsf{upk}), (\mathsf{opk}, \mathsf{hon}, \emptyset)\big) \longleftrightarrow (\mathsf{upk}, \mathsf{opk}).$$
An accepted wrapper attribution to a pair outside the wrapper join relation therefore yields an accepted baseline attribution to a pair outside the baseline join relation, contradicting opening-unforgeability of $\mathsf{\Gamma}$.

□

Conclusion. The wrapper $\mathsf{\Gamma}^{\neg\mathsf{TS}}$ preserves anonymity, traceability, non-frameability, non-claimability, certificate-unforgeability, key-unforgeability, and opening-unforgeability by black-box reduction to $\mathsf{\Gamma}$, while the colluding issuer and opener can activate the gadget branch and produce two different accepted attributions for the same message-signature pair. Hence trace-soundness is independent of the other seven notions.

6.1.5 Proof of Theorem 29 (Non-Claimability)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 13–15.

The wrapper uses disjoint encodings for ordinary member public keys and gadget identities:
$$(0, \mathsf{upk}) \qquad \text{and} \qquad (1, t),$$
respectively. Thus, no honestly generated wrapper member public key is syntactically equal to a gadget identity.

Extended issuer and opener public keys have the forms

$$\mathsf{ipk}' = (\mathsf{ipk}, a_G) \qquad \text{and} \qquad \mathsf{opk}' = (\mathsf{opk}, a_O).$$

Honestly generated issuer and opener keys use the canonical anchors

$$a_G = a_O = \mathsf{hon}.$$

An adversary controlling a public-key context may instead supply arbitrary well-formed anchors.

For a wrapper message m, define the message supplied to the underlying scheme as

$$\widehat{m} := (m, \mathsf{ipk}', \mathsf{opk}').$$

Every wrapper signature explicitly records the complete wrapper group context:

$$\sigma'_m = (\sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

The ordinary branch is enabled when

$$a_G = a_O = \mathsf{hon}.$$

The non-claimability gadget is enabled when

$$a_G = a_O = t \neq \mathsf{hon}.$$

All other anchor combinations are rejected.

All algorithms not listed below are inherited unchanged from Γ. Malformed inputs are rejected according to the global parsing convention. For

$$\mathsf{ipk}' = (\mathsf{ipk}, a_G), \qquad \mathsf{opk}' = (\mathsf{opk}, a_O), \qquad \widehat{m} := (m, \mathsf{ipk}', \mathsf{opk}'),$$

define

$$\mathsf{Hon} := [a_G = a_O = \mathsf{hon}],$$
$$\mathsf{Gad} := [a_G = a_O \neq \mathsf{hon}],$$
$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') := [(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') = (\mathsf{ipk}', \mathsf{opk}')].$$

$\underline{\mathsf{KgM}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run
$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$$
and return
$$(\mathsf{usk}', \mathsf{upk}') \leftarrow (\mathsf{usk}, (0, \mathsf{upk})).$$

$\underline{\mathsf{KgI}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run
$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma)$$
and return
$$(\mathsf{isk}', \mathsf{ipk}') \leftarrow (\mathsf{isk}, (\mathsf{ipk}, \mathsf{hon})).$$

$\underline{\mathsf{KgO}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run
$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$$
and return
$$(\mathsf{osk}', \mathsf{opk}') \leftarrow (\mathsf{osk}, (\mathsf{opk}, \mathsf{hon})).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}', \mathsf{ipk}', \mathsf{upk}', \mathsf{opk}')}$:
Parse
$$\mathsf{ipk}' = (\mathsf{ipk}, a_G),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O),$$
$$\mathsf{upk}' = (b_M, z_M),$$
and set
$$\mathsf{isk} \leftarrow \mathsf{isk}'.$$
Return $\perp$ unless
$$\mathsf{Hon} \wedge (b_M = 0).$$
Set
$$\mathsf{upk} \leftarrow z_M$$
and compute
$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk}).$$
Return σ_s.

Construction 13: Wrapper $\Gamma^{\neg\mathsf{NC}}$ (Part I of III)

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}', \mathsf{upk}', \sigma_s, \mathsf{ipk}', \mathsf{opk}', m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_G),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O),$$
$$\mathsf{upk}' = (b_M, z_M),$$

and set

$$\mathsf{usk} \leftarrow \mathsf{usk}'.$$

Return $\perp$ unless

$$b_M = 0 \quad \wedge \quad (\mathsf{Hon} \vee \mathsf{Gad}).$$

Set

$$\mathsf{upk} \leftarrow z_M,$$

and compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}).$$

Return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$\sigma'_m \leftarrow (\sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

$\underline{\mathsf{Vfy}'_\Gamma(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_G),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O),$$
$$\sigma'_m = (\sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Return 0 unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') \quad \wedge \quad (\mathsf{Hon} \vee \mathsf{Gad}).$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m).$$

Construction 14: Wrapper $\Gamma^{\neg\mathsf{NC}}$ (Part II of III)

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Opn}'_\Gamma(\mathsf{osk}', \mathsf{opk}', \mathsf{ipk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_G),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O),$$
$$\sigma'_m = (\sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

and set

$$\mathsf{osk} \leftarrow \mathsf{osk}'.$$

Return $\perp$ unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') \wedge \mathsf{Hon}.$$

Compute

$$(\mathsf{upk}, O) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, \widehat{m}, \sigma_m).$$

Return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$\big((0, \mathsf{upk}), (\mathsf{upk}, O)\big).$$

$\underline{\mathsf{Jdg}'_\Gamma(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O')}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}, a_G),$$
$$\mathsf{opk}' = (\mathsf{opk}, a_O),$$
$$\sigma'_m = (\sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$
$$O' = (\widehat{\mathsf{upk}}, \widetilde{O}).$$

Return 0 unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

If Gad, let

$$t \coloneqq a_G = a_O$$

and parse

$$\mathsf{upk}^* = (b_M, z_M).$$

Return 1 iff

$$b_M = 1, \qquad z_M = t,$$

and

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m, \widehat{\mathsf{upk}}, \widetilde{O}) = 1.$$

Otherwise, return 0 unless

$$\mathsf{Hon}.$$

Parse

$$\mathsf{upk}^* = (b_M, z_M)$$

and return 0 unless

$$b_M = 0 \qquad \text{and} \qquad z_M = \widehat{\mathsf{upk}}.$$

Return

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}, \sigma_m, \widehat{\mathsf{upk}}, \widetilde{O}).$$

Construction 15: Wrapper $\Gamma^{\neg\mathsf{NC}}$ (Part III of III)

Correctness. Honestly generated issuer and opener keys have canonical anchors

$$a_G = a_O = \mathsf{hon},$$

and honestly generated member public keys have ordinary form

$$(0, \mathsf{upk}).$$

Consequently, all honest executions use the ordinary branch and consistently invoke the underlying algorithms on

$$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}').$$

The opening algorithm returns

$$\big((0, \mathsf{upk}), (\mathsf{upk}, O)\big),$$

which is accepted by the ordinary branch of Jdg'_{Γ}. Correctness therefore follows directly from correctness of Γ.

Why non-claimability fails. Let the honest wrapper member key pair generated by the non-claimability experiment be

$$(\mathsf{usk}, \mathsf{upk}') \qquad \text{with} \qquad \mathsf{upk}' = (0, \mathsf{upk}).$$

The adversary independently generates baseline issuer and opener key pairs

$$(\mathsf{isk}, \mathsf{ipk}) \leftarrow_{\$} \mathsf{KgI}_{\Gamma}(\mathsf{pp}_{\Gamma})$$

and

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_{\$} \mathsf{KgO}_{\Gamma}(\mathsf{pp}_{\Gamma}).$$

It chooses any tag

$$t \neq \mathsf{hon}$$

and defines

$$\mathsf{ipk}' = (\mathsf{ipk}, t) \qquad \text{and} \qquad \mathsf{opk}' = (\mathsf{opk}, t).$$

Using the honest member's underlying public key, it computes

$$\sigma_s \leftarrow_{\$} \mathsf{Join}_{\Gamma}(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk}).$$

It then queries the wrapper signing oracle on

$$(\mathsf{ipk}', \mathsf{opk}', \sigma_s, m)$$

and receives a logged wrapper signature

$$\sigma'_m = (\sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

The corresponding baseline signature σ_m is valid on

$$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}').$$

Since the adversary generated the opener key pair itself, it computes

$$(\widehat{\mathsf{upk}}, \widetilde{O}) \leftarrow_{\$} \mathsf{Opn}_{\Gamma}(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, \widehat{m}, \sigma_m).$$

By correctness,

$$\widehat{\mathsf{upk}} = \mathsf{upk}.$$

The adversary outputs the different wrapper member public key

$$\mathsf{upk}^* = (1, t)$$

and opening information

$$O' = (\widehat{\mathsf{upk}}, \widetilde{O}).$$

Since ordinary and gadget member public keys have disjoint encodings,

$$\mathsf{upk}^* = (1, t) \neq (0, \mathsf{upk}) = \mathsf{upk}'.$$

Moreover, the common noncanonical anchor activates the gadget branch, and the embedded baseline opening is valid. Therefore

$$\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O') = 1.$$

The signature σ'_m is exactly a transcript previously returned by the wrapper signing oracle, yet it is accepted as an attribution to a different member public key. Hence $\Gamma^{\neg\mathsf{NC}}$ is not non-claimable.

Why the other seven properties hold.

- **Anonymity (ANON).** The opener key is honestly generated and therefore has

  $$a_O = \mathsf{hon}.$$

  A noncanonical issuer anchor cannot match it, so every accepting challenge signature must use the canonical context

  $$a_G = a_O = \mathsf{hon}.$$

  For noncanonical issuer contexts, both candidate signing executions return $\perp$ independently of the challenge bit. For canonical contexts, the challenge-signing and opening oracles coincide with their baseline counterparts on the derived message

  $$\widehat{m} = (m, \mathsf{ipk}', \mathsf{opk}').$$

  The context components embedded in the wrapper signature are public and independent of the hidden signer bit. Thus, any anonymity adversary against the wrapper yields one against Γ.

- **Traceability (TRC).** The opener key is honestly generated, hence

  $$a_O = \mathsf{hon}.$$

  Every verifying wrapper signature must therefore use the canonical issuer anchor

  $$a_G = \mathsf{hon}.$$

  The prescribed wrapper opening algorithm invokes the prescribed baseline opening algorithm on the same derived message and returns the ordinary wrapper identity. A successful wrapper traceability attack would therefore yield a successful baseline traceability attack.

- **Non-Frameability (NF).** The fixed honest wrapper member public key has ordinary form

  $$(0, \mathsf{upk}).$$

  When the gadget branch is active, it accepts only the unique gadget identity

  $$(1, t),$$

which is syntactically distinct from every honestly generated member public key. Hence the gadget branch cannot frame the fixed honest member.

In the canonical branch, any accepted attribution to the honest wrapper member key yields an accepted baseline attribution to upk. Deterministic context wrapping preserves freshness, so a successful wrapper frame would contradict non-frameability of Γ.

- **Trace-Soundness (TS).** Consider a fixed wrapper message-signature pair admitting two accepted attribution tuples. Since the wrapper signature explicitly contains its complete group context, both accepted interpretations must use the same issuer and opener public keys.

  If the common anchors are noncanonical and equal, the gadget branch accepts only the unique external member public key

  $$(1, t).$$

  Thus, two different accepted attribution tuples cannot arise.

  If the common anchors are both hon, both accepted openings reduce to baseline openings for the same derived message and the same baseline signature. Two different accepted wrapper attributions would therefore contradict trace-soundness of Γ.

- **Certificate-Unforgeability (CUnf).** The issuer key is honestly generated and therefore has

  $$a_G = \mathsf{hon}.$$

  Every accepting final signature must consequently use the canonical opener anchor

  $$a_O = \mathsf{hon}.$$

  Removing the deterministic wrapper context yields an accepting baseline signature under the honest issuer key.

  If the baseline transcript had previously been returned by the baseline signing oracle, the complete wrapper transcript would also have been recorded. Wrapper freshness therefore implies baseline freshness, and a successful wrapper forgery contradicts certificate-unforgeability of Γ.

- **Key-Unforgeability (KUnf).** The final issuer key is honestly generated and therefore canonical. Consequently, every successful final forgery must also use a canonical opener key and reduces to a baseline forgery under the corresponding derived message.

  Signing-oracle outputs obtained under matching noncanonical anchors explicitly record those noncanonical contexts and cannot be reinterpreted under the honest issuer key. Thus, removing the wrapper context from a fresh final forgery yields a fresh accepting baseline forgery, contradicting key-unforgeability of Γ.

- **Opening-Unforgeability (OpU).** The issuer key is honestly generated and has

  $$a_G = \mathsf{hon}.$$

  The join oracle accepts only canonical opener keys, and the gadget branch cannot be activated. The wrapper join relation is therefore in one-to-one correspondence with the baseline join relation:

  $$\big((0, \mathsf{upk}), (\mathsf{opk}, \mathsf{hon})\big) \longleftrightarrow (\mathsf{upk}, \mathsf{opk}).$$

  Any accepted wrapper attribution to a pair outside the wrapper join relation would yield an accepted baseline attribution to the corresponding pair outside the baseline join relation, contradicting opening-unforgeability of Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{NC}}$ preserves anonymity, non-frameability, traceability, trace-soundness, certificate-unforgeability, key-unforgeability, and opening-unforgeability by black-box reduction to Γ. Nevertheless, a colluding issuer–opener pair can activate the gadget branch and relabel an exact signing-oracle transcript from the honest member $(0, \mathsf{upk})$ as an attribution to the distinct gadget identity $(1, t)$. Hence non-claimability is independent of the other seven notions.

6.1.6 Proof of Theorem 30 (Certificate-Unforgeability)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 16–17.

Let

$$\mathcal{H} = \left\{ H_\lambda : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}^{n(\lambda)} \right\}_{\lambda \in \mathbb{N}}$$

be a fixed one-way function family. We write H for H_λ when the security parameter is clear.

The wrapper augments every member secret key with an independently sampled value x, while leaving the corresponding member public key unchanged:

$$\mathsf{usk}' = (\mathsf{usk}, x), \qquad \mathsf{upk}' = \mathsf{upk}.$$

The wrapper issuer key consists of two independently generated baseline issuer key pairs:

$$\mathsf{isk}' = (\mathsf{isk}^0, \mathsf{isk}^1), \qquad \mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1).$$

The two components separate the ordinary branch from the certificate-unforgeability branch.

Extended opener public keys have the form

$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}), \qquad b_{\mathsf{cunf}} \in \{0,1\}.$$

Honestly generated opener keys always use

$$b_{\mathsf{cunf}} = 0.$$

An adversary controlling the opener context may instead supply a key with $b_{\mathsf{cunf}} = 1$.

The wrapper uses three disjoint signature encodings:

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}')$$

for ordinary signatures,

$$(1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}')$$

for authenticated source signatures, and

$$(2, h, \sigma_m, x, \mathsf{ipk}', \mathsf{opk}')$$

for transformed signatures.

For a wrapper message m, define

$$\widehat{m}_0 \coloneqq (m, 0, \mathsf{ipk}', \mathsf{opk}')$$

and

$$\widehat{m}_1(h) \coloneqq (m, 1, h, \mathsf{ipk}', \mathsf{opk}').$$

These are the complete baseline messages under the global convention. The branch selector and complete wrapper keys already determine the selected baseline issuer and opener keys; no additional context wrapping is applied.

All algorithms not listed below are inherited unchanged from Γ. Malformed inputs are rejected according to the global parsing convention. Write

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1), \qquad \mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}),$$

and define

$$\widehat{m}_0 := (m, 0, \mathsf{ipk}', \mathsf{opk}'),$$
$$\widehat{m}_1(h) := (m, 1, h, \mathsf{ipk}', \mathsf{opk}'),$$
$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') := [(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') = (\mathsf{ipk}', \mathsf{opk}')].$$

$\underline{\mathsf{KgM}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run

$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma),$$

and

$$x \leftarrow_\$ \{0,1\}^{\ell(\lambda)},$$

and return

$$(\mathsf{usk}', \mathsf{upk}') \leftarrow ((\mathsf{usk}, x), \mathsf{upk}).$$

$\underline{\mathsf{KgI}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Independently generate

$$(\mathsf{isk}^i, \mathsf{ipk}^i) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma), \qquad i \in \{0,1\}.$$

Return

$$(\mathsf{isk}', \mathsf{ipk}') \leftarrow \big((\mathsf{isk}^0, \mathsf{isk}^1), (\mathsf{ipk}^0, \mathsf{ipk}^1)\big).$$

$\underline{\mathsf{KgO}'_\Gamma(\mathsf{pp}_\Gamma)}$:
Run

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{osk}', \mathsf{opk}') \leftarrow (\mathsf{osk}, (\mathsf{opk}, 0)).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}', \mathsf{ipk}', \mathsf{upk}, \mathsf{opk}')}$:
Parse

$$\mathsf{isk}' = (\mathsf{isk}^0, \mathsf{isk}^1),$$
$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1),$$
$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}).$$

Return $\perp$ unless

$$b_{\mathsf{cunf}} \in \{0,1\}.$$

Otherwise compute

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}^{b_{\mathsf{cunf}}}, \mathsf{ipk}^{b_{\mathsf{cunf}}}, \mathsf{upk}, \mathsf{opk})$$

and return σ_s.

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}', \mathsf{upk}, \sigma_s, \mathsf{ipk}', \mathsf{opk}', m)}$:
Parse

$$\mathsf{usk}' = (\mathsf{usk}, x), \qquad \mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1)$$

and

$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}).$$

If $b_{\mathsf{cunf}} = 0$, compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}^0, \mathsf{opk}, \widehat{m}_0)$$

and return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

If $b_{\mathsf{cunf}} = 1$, set

$$h \leftarrow H(x)$$

and compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}^1, \mathsf{opk}, \widehat{m}_1(h))$$

and return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$(1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

Return $\perp$.

Construction 16: Wrapper $\Gamma^{\neg\mathsf{CUnf}}$ (Part I of II)

Correctness. Honestly generated opener public keys satisfy

$$b_{\mathsf{cunf}} = 0.$$

Consequently, honest joins use the baseline issuer key pair

$$(\mathsf{isk}^0, \mathsf{ipk}^0),$$

and honest signatures have the form

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

Signing, verification, opening, and opening verification consistently invoke the corresponding baseline algorithms on

$$\widehat{m}_0 = (m, 0, \mathsf{ipk}', \mathsf{opk}').$$

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Vfy}'_\Gamma(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1), \qquad \mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}).$$

If

$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

return 0 unless

$$b_{\mathsf{cunf}} = 0 \quad \wedge \quad \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}^0, \mathsf{opk}, \widehat{m}_0, \sigma_m).$$

If

$$\sigma'_m = (1, h, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

return 0 unless

$$b_{\mathsf{cunf}} = 1 \quad \wedge \quad \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}^1, \mathsf{opk}, \widehat{m}_1(h), \sigma_m).$$

If

$$\sigma'_m = (2, h, \sigma_m, x, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

return 0 unless

$$b_{\mathsf{cunf}} = 1, \quad \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'), \quad H(x) = h.$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}^1, \mathsf{opk}, \widehat{m}_1(h), \sigma_m).$$

Return 0.

$\underline{\mathsf{Opn}'_\Gamma(\mathsf{osk}', \mathsf{opk}', \mathsf{ipk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1), \quad \mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}), \quad \sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Set $\mathsf{osk} \leftarrow \mathsf{osk}'$. Return $\perp$ unless

$$b_{\mathsf{cunf}} = 0 \quad \wedge \quad \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Compute

$$(\mathsf{upk}, O) \leftarrow_{\$} \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}^0, \widehat{m}_0, \sigma_m)$$

and return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$(\mathsf{upk}, O).$$

$\underline{\mathsf{Jdg}'_\Gamma(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1), \quad \mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{cunf}}), \quad \sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Return 0 unless

$$b_{\mathsf{cunf}} = 0 \quad \wedge \quad \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}^0, \mathsf{opk}, \widehat{m}_0, \sigma_m, \mathsf{upk}^*, O).$$

Construction 17: Wrapper $\Gamma^{\neg\mathsf{CUnf}}$ (Part II of II)

Correctness therefore follows directly from correctness of Γ.

Why certificate-unforgeability fails. In the certificate-unforgeability experiment, the adversary receives the honest wrapper issuer public key

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1)$$

and chooses an opener key of the form

$$\mathsf{opk}' = (\mathsf{opk}, 1),$$

where

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_{\$} \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma).$$

It queries the member-generation oracle and receives

$$(j, (\mathsf{usk}, x), \mathsf{upk}).$$

It then queries the honest wrapper join oracle on (opk', j) and obtains an opaque certificate handle for a certificate issued under ipk^1.

For any message m, the adversary queries the signing oracle using this handle and receives

$$\sigma'_m = (1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}'),$$

where

$$h = H(x)$$

and

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}^1, \mathsf{opk}, (m, 1, h, \mathsf{ipk}', \mathsf{opk}'), \sigma_m) = 1.$$

The adversary transforms this signing-oracle output into

$$\sigma_m^* = (2, h, \sigma_m, x, \mathsf{ipk}', \mathsf{opk}')$$

and outputs $(\sigma_m^*, m, \mathsf{opk}')$.

Since $H(x) = h$, the transformed signature verifies:

$$\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma_m^*) = 1.$$

Moreover,

$$(2, h, \sigma_m, x, \mathsf{ipk}', \mathsf{opk}') \neq (1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

The exact signature string output by the adversary was therefore never returned by the signing oracle. Hence

$$(\sigma_m^*, m, \mathsf{opk}') \notin \mathcal{Q}_{\sigma}.$$

Thus, the adversary produces a fresh accepting signature without obtaining the certificate hidden behind the join handle. Therefore $\Gamma^{\neg\mathsf{CUnf}}$ is not certificate-unforgeable.

Issuer-key collisions. We use the following observation only in preservation experiments in which the wrapper issuer key is generated honestly. Define

$$\mathsf{Bad}_{\mathsf{coll}} := [\mathsf{ipk}^0 = \mathsf{ipk}^1].$$

Since the two components are then independently generated by Kgl_{Γ}, the issuer-public-key collision argument above implies that $\mathsf{Bad}_{\mathsf{coll}}$ occurs only with negligible probability. We condition on its complement in the key-unforgeability and opening-unforgeability arguments below. No collision abort is introduced in experiments with adversarially chosen issuer public keys; the respective preservation arguments use the explicit mode and full wrapper context to distinguish the branches.

Why the other seven properties hold.

- **Anonymity (ANON).** The opener public key is honestly generated and therefore satisfies

  $$b_{\mathsf{cunf}} = 0.$$

  Challenge signatures consequently have mode-0 and are baseline signatures under ipk^0 on

  $$\widehat{m}_0 = (m, 0, \mathsf{ipk}', \mathsf{opk}').$$

  The auxiliary value x is unused in this branch and does not occur in the challenge transcript. The wrapper context is public and independent of the hidden signer bit. Hence any anonymity adversary against the wrapper yields one against Γ.

- **Non-Frameability (NF).** An accepted wrapper opening is possible only for a mode-0 signature under an opener key satisfying

  $$b_{\mathsf{cunf}} = 0.$$

  Such an opening is exactly a baseline opening under ipk^0 on $\widehat{m}_0$. Mode-1 and mode-2 signatures are never accepted by Jdg'_{Γ}. They therefore cannot be used to frame the fixed honest member. Any successful wrapper frame must occur through the ordinary branch and yields a successful baseline frame. Deterministic context wrapping preserves freshness.

- **Traceability (TRC).** The opener key is honestly generated, hence

$$b_{\mathsf{cunf}} = 0.$$

Only mode-0 signatures can verify in this context. Every verifying wrapper signature therefore corresponds to a verifying baseline signature under ipk^0, and the prescribed wrapper opening algorithm invokes the prescribed baseline opening algorithm on the same derived message. Traceability of the wrapper follows from traceability of Γ.

- **Trace-Soundness (TS).** Every accepted opening concerns a mode-0 wrapper signature under an opener context satisfying

$$b_{\mathsf{cunf}} = 0.$$

Since the complete wrapper issuer and opener context is embedded in the signature, two accepted interpretations of the same wrapper signature must use the same wrapper group context.

Both accepted openings therefore reduce to baseline openings for the same baseline signature and the same derived message $\widehat{m}_0$. Two distinct accepted wrapper attributions would contradict trace-soundness of Γ.

- **Non-Claimability (NC).** The issuer and opener contexts in the non-claimability experiment may be adversarial.

If the signing oracle is queried under an opener key satisfying

$$b_{\mathsf{cunf}} = 1,$$

it returns a mode-1 signature. Neither mode-1 nor mode-2 signatures admit accepted openings, so such a transcript cannot yield a successful non-claimability attack.

If $b_{\mathsf{cunf}} = 0$, the logged wrapper transcript is a deterministic context extension of a logged baseline transcript. Any accepted reassignment to a different member public key therefore yields an accepted reassignment of the corresponding baseline transcript, contradicting non-claimability of Γ.

- **Key-Unforgeability (KUnf).** Consider a fresh accepting wrapper forgery under the honestly generated wrapper issuer key $\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1)$.

For a mode-b forgery with $b \in \{0, 1\}$, a reduction guesses the mode b, embeds the honest baseline issuer public key supplied by the key-unforgeability challenger as ipk^b, and generates the other issuer key pair independently. Removing the wrapper encoding gives a fresh accepting baseline forgery under ipk^b. A wrong guess causes only a constant-factor loss. The reduction obtains each baseline member index and public key from its member-generation oracle and samples the corresponding auxiliary x_j locally. It forwards joins for the embedded issuer component and answers joins for the other component with that component's known issuer secret key. All signing queries, including those using adversarial issuer contexts, are answered with the baseline signing oracle, which accepts arbitrary issuer public keys and certificates. Thus no hidden member secret key is needed to simulate the other branch.

It remains to consider a mode-2 forgery

$$\sigma_m^* = (2, h^*, \sigma_m^*, x^*, \mathsf{ipk}', \mathsf{opk}').$$

Define the corresponding source signature

$$\widetilde{\sigma}_m \coloneqq (1, h^*, \sigma_m^*, \mathsf{ipk}', \mathsf{opk}').$$

If

$$(\widetilde{\sigma}_m, m^*, \mathsf{opk}', \mathsf{ipk}') \notin \mathcal{Q}_\sigma,$$

then the embedded baseline signature σ^*_m is a fresh accepting baseline forgery under ipk^1 on

$$(m^*, 1, h^*, \mathsf{ipk}', \mathsf{opk}').$$

Otherwise, the source signature was produced by the signing oracle using some challenger-generated member secret key (usk_j, x_j). Consequently,

$$h^* = H(x_j).$$

Acceptance of the mode-2 forgery additionally implies

$$H(x^*) = h^*.$$

Thus, x^* is a preimage of the one-way-function value $H(x_j)$.

A one-wayness reduction guesses the relevant member index, embeds its challenge image as the corresponding value h^*, and simulates all other member keys itself. The challenge image suffices to answer mode-1 signing queries for the selected member, since the underlying signing oracle is invoked on $\widehat{m}_1(h^*)$. If the mode-2 forgery refers to the selected member, the extracted value x^* inverts the challenge. The index guess causes only the corresponding polynomial query loss.

Hence a successful key-unforgeability adversary yields either a baseline key-unforgeability adversary or an inverter for H.

- **Opening-Unforgeability (OpU).** Conditioned on $\neg\mathsf{Bad}_{\mathsf{coll}}$, the honest wrapper issuer public key contains two distinct baseline issuer public keys. A reduction embeds the honest baseline issuer public key as ipk^0, generates $(\mathsf{isk}^1, \mathsf{ipk}^1)$ independently, and answers join queries with $b_{\mathsf{cunf}} = 0$ using the baseline join oracle and queries with $b_{\mathsf{cunf}} = 1$ locally using isk^1.

  Accepted openings are possible only for mode-0 signatures, hence only under $b_{\mathsf{cunf}} = 0$ and the issuer component ipk^0. The wrapper join relation restricted to queries with $b_{\mathsf{cunf}} = 0$ is therefore in one-to-one correspondence with the baseline join relation. In particular, if a pair lies outside the complete wrapper join relation, then it also lies outside this restricted relation. Consequently, an accepted wrapper attribution to such a pair yields an accepted baseline attribution under ipk^0 to a pair outside the baseline join relation, contradicting opening-unforgeability of Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{CUnf}}$ preserves anonymity, non-frameability, traceability, trace-soundness, non-claimability, key-unforgeability, and opening-unforgeability under the security of the baseline scheme and the one-wayness of H. Nevertheless, exposure of the auxiliary member secret x in the certificate-unforgeability experiment allows an adversary to transform an authenticated mode-1 signing-oracle output into a distinct, fresh, and accepting mode-2 signature. Hence certificate-unforgeability is independent of the other seven notions.

6.1.7 Proof of Theorem 31 (Key-Unforgeability)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 18–20.

Let

$$\mathcal{H} = \left\{ H_\lambda : \{0,1\}^{\ell(\lambda)} \longrightarrow \{0,1\}^{n(\lambda)} \right\}_{\lambda \in \mathbb{N}}$$

be a fixed one-way function family. We write H for H_λ whenever the security parameter is clear.

The wrapper issuer key consists of two independently generated baseline issuer key pairs:

$$\mathsf{isk}' = (\mathsf{isk}^0, \mathsf{isk}^1), \qquad \mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1).$$

The two issuer components separate the ordinary branch from the key-unforgeability branch.

Extended opener public keys have the form

$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}), \qquad b_{\mathsf{kunf}} \in \{0,1\}.$$

Honestly generated opener keys always use

$$b_{\mathsf{kunf}} = 0.$$

An adversary controlling the opener context may instead supply a key with

$$b_{\mathsf{kunf}} = 1.$$

A wrapper membership certificate has the form

$$\sigma'_s = (\sigma_s, x),$$

where σ_s is a baseline membership certificate and

$$x \leftarrow_{\$} \{0,1\}^{\ell(\lambda)}$$

is an independently sampled auxiliary value.

The wrapper uses three disjoint signature encodings:

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}')$$

for ordinary signatures,

$$(1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}')$$

for authenticated source signatures, and

$$(2, h, \sigma_m, x, \mathsf{ipk}', \mathsf{opk}')$$

for transformed signatures.

For a wrapper message m, define

$$\widehat{m}_0 \coloneqq (m, 0, \mathsf{ipk}', \mathsf{opk}')$$

and

$$\widehat{m}_1(h) \coloneqq (m, 1, h, \mathsf{ipk}', \mathsf{opk}').$$

These are the complete baseline messages under the global convention. The branch selector and complete wrapper keys already determine the selected baseline issuer and opener keys; no additional context wrapping is applied.

The parameter-generation and member-key-generation algorithms are inherited unchanged from Γ. Malformed inputs are rejected according to the global parsing convention. Write

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1),$$
$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}),$$
$$\widehat{m}_0 := (m, 0, \mathsf{ipk}', \mathsf{opk}'),$$
$$\widehat{m}_1(h) := (m, 1, h, \mathsf{ipk}', \mathsf{opk}'),$$
$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') := [(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}') = (\mathsf{ipk}', \mathsf{opk}')].$$

$\underline{\mathsf{KgI}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Independently generate

$$(\mathsf{isk}^i, \mathsf{ipk}^i) \leftarrow_\$ \mathsf{KgI}_\Gamma(\mathsf{pp}_\Gamma), \qquad i \in \{0,1\}.$$

Return

$$(\mathsf{isk}', \mathsf{ipk}') \leftarrow \big((\mathsf{isk}^0, \mathsf{isk}^1), (\mathsf{ipk}^0, \mathsf{ipk}^1)\big).$$

$\underline{\mathsf{KgO}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Run

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{osk}', \mathsf{opk}') \leftarrow (\mathsf{osk}, (\mathsf{opk}, 0)).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}', \mathsf{ipk}', \mathsf{upk}, \mathsf{opk}')}$**:**
Parse

$$\mathsf{isk}' = (\mathsf{isk}^0, \mathsf{isk}^1),$$
$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1),$$
$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}).$$

Return $\perp$ unless

$$b_{\mathsf{kunf}} \in \{0,1\}.$$

Compute

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}^{b_{\mathsf{kunf}}}, \mathsf{ipk}^{b_{\mathsf{kunf}}}, \mathsf{upk}, \mathsf{opk}).$$

Return $\perp$ if $\sigma_s = \perp$. Otherwise sample

$$x \leftarrow_\$ \{0,1\}^{\ell(\lambda)}$$

and return

$$\sigma'_s \leftarrow (\sigma_s, x).$$

Construction 18: Wrapper $\Gamma^{\neg\mathsf{KUnf}}$ (Part I of III)

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma'_s, \mathsf{ipk}', \mathsf{opk}', m)}$**:**
Parse

$$\sigma'_s = (\sigma_s, x),$$
$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1),$$
$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}).$$

If $b_{\mathsf{kunf}} = 0$, compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}^0, \mathsf{opk}, \widehat{m}_0)$$

and return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

If $b_{\mathsf{kunf}} = 1$, set

$$h \leftarrow H(x)$$

and compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}^1, \mathsf{opk}, \widehat{m}_1(h))$$

and return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$(1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

Return $\perp$.

$\underline{\mathsf{Opn}'_\Gamma(\mathsf{osk}', \mathsf{opk}', \mathsf{ipk}', m, \sigma'_m)}$**:**
Parse

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1),$$
$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}),$$
$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

Set $\mathsf{osk} \leftarrow \mathsf{osk}'$. Return $\perp$ unless

$$b_{\mathsf{kunf}} = 0 \quad \wedge \quad \mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Compute

$$(\mathsf{upk}, O) \leftarrow_\$ \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}^0, \widehat{m}_0, \sigma_m)$$

and return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$(\mathsf{upk}, O).$$

Construction 19: Wrapper $\Gamma^{\neg\mathsf{KUnf}}$ (Part II of III)

Correctness. Honestly generated opener public keys satisfy

$$b_{\mathsf{kunf}} = 0.$$

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1), \qquad \mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}).$$

If

$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

return 0 unless

$$b_{\mathsf{kunf}} = 0$$

and

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}^0, \mathsf{opk}, \widehat{m}_0, \sigma_m).$$

If

$$\sigma'_m = (1, h, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

return 0 unless

$$b_{\mathsf{kunf}} = 1$$

and

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}^1, \mathsf{opk}, \widehat{m}_1(h), \sigma_m).$$

If

$$\sigma'_m = (2, h, \sigma_m, x, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

return 0 unless

$$b_{\mathsf{kunf}} = 1,$$
$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}'),$$

and

$$H(x) = h.$$

Otherwise return

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}^1, \mathsf{opk}, \widehat{m}_1(h), \sigma_m).$$

Return 0.

$\underline{\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O)}$**:**

Parse

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1),$$
$$\mathsf{opk}' = (\mathsf{opk}, b_{\mathsf{kunf}}),$$
$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Return 0 unless

$$b_{\mathsf{kunf}} = 0$$

and

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}', \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Jdg}_{\Gamma}(\mathsf{ipk}^0, \mathsf{opk}, \widehat{m}_0, \sigma_m, \mathsf{upk}^*, O).$$

Construction 20: Wrapper $\Gamma^{\neg\mathsf{KUnf}}$ (Part III of III)

Consequently, honest joins use the baseline issuer key pair

$$(\mathsf{isk}^0, \mathsf{ipk}^0),$$

and honest signatures have the form

$$(0, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

Signing, verification, opening, and opening verification consistently invoke the corresponding baseline algorithms on

$$\widehat{m}_0 = (m, 0, \mathsf{ipk}', \mathsf{opk}').$$

Correctness therefore follows directly from correctness of Γ.

Why key-unforgeability fails. In the key-unforgeability experiment, the adversary receives the honest wrapper issuer public key

$$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1)$$

and chooses an opener key of the form

$$\mathsf{opk}' = (\mathsf{opk}, 1),$$

where

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_{\$} \mathsf{KgO}_{\Gamma}(\mathsf{pp}_{\Gamma}).$$

It queries the member-generation oracle and receives

$$(j, \mathsf{upk}).$$

It then queries the honest wrapper join oracle on

$$(\mathsf{opk}', j)$$

and receives the wrapper certificate

$$\sigma'_s = (\sigma_s, x),$$

where σ_s is a baseline membership certificate under ipk^1.

For any message m, the adversary queries the signing oracle using

$$(\mathsf{ipk}', j, \sigma'_s, \mathsf{opk}', m)$$

and receives

$$\sigma'_m = (1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}'),$$

where

$$h = H(x)$$

and

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}^1, \mathsf{opk}, (m, 1, h, \mathsf{ipk}', \mathsf{opk}'), \sigma_m) = 1.$$

Since the join oracle revealed x as part of the wrapper certificate, the adversary transforms this signing-oracle output into

$$\sigma^*_m = (2, h, \sigma_m, x, \mathsf{ipk}', \mathsf{opk}')$$

and outputs

$$(\sigma^*_m, m, \mathsf{opk}').$$

Because

$$H(x) = h,$$

the transformed signature verifies:

$$\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}', \mathsf{opk}', m, \sigma^*_m) = 1.$$

Moreover,

$$(2, h, \sigma_m, x, \mathsf{ipk}', \mathsf{opk}') \neq (1, h, \sigma_m, \mathsf{ipk}', \mathsf{opk}').$$

Hence the exact signature string output by the adversary was never returned by the signing oracle, and therefore

$$(\sigma^*_m, m, \mathsf{opk}', \mathsf{ipk}') \notin \mathcal{Q}_{\sigma}.$$

Thus, the adversary produces a fresh accepting signature without knowing the challenger-generated member secret key. Therefore $\Gamma^{\neg\mathsf{KUnf}}$ is not key-unforgeable.

Issuer-key collisions. We use the following observation only in preservation experiments in which the wrapper issuer key is generated honestly. Define

$$\mathsf{Bad}_{\mathsf{coll}} \coloneqq [\mathsf{ipk}^0 = \mathsf{ipk}^1].$$

Since the two components are then independently generated by Kgl_Γ, the issuer-public-key collision argument above implies that $\mathsf{Bad}_{\mathsf{coll}}$ occurs only with negligible probability. We condition on its complement in the certificate-unforgeability and opening-unforgeability arguments below. No collision abort is introduced in experiments with adversarially chosen issuer public keys; the respective preservation arguments use the explicit mode and full wrapper context to distinguish the branches.

Why the other seven properties hold.

- **Anonymity (ANON).** The opener public key is honestly generated and therefore satisfies
$$b_{\mathsf{kunf}} = 0.$$
Challenge signatures consequently have mode 0 and are baseline signatures under ipk^0 on the derived message
$$\widehat{m}_0 = (m, 0, \mathsf{ipk}', \mathsf{opk}').$$
The auxiliary certificate value x is unused in this branch and does not occur in the challenge transcript. The wrapper context is public and independent of the hidden signer bit. Hence any anonymity adversary against the wrapper yields one against Γ.

- **Non-Frameability (NF).** Accepted wrapper openings are possible only for mode-0 signatures under opener keys satisfying
$$b_{\mathsf{kunf}} = 0.$$
Such openings are exactly baseline openings under ipk^0 on $\widehat{m}_0$.

  Mode-1 and mode-2 signatures are never accepted by Jdg'_Γ. They therefore cannot be used to frame the fixed honest member. Any successful wrapper frame must occur through the ordinary branch and yields a successful baseline frame. Deterministic context wrapping preserves freshness.

- **Traceability (TRC).** The opener key is honestly generated, hence
$$b_{\mathsf{kunf}} = 0.$$
Only mode-0 signatures can verify in this context. Every verifying wrapper signature therefore corresponds to a verifying baseline signature under ipk^0, and the prescribed wrapper opening algorithm invokes the prescribed baseline opening algorithm on the same derived message. Traceability of the wrapper follows from traceability of Γ.

- **Trace-Soundness (TS).** Every accepted opening concerns a mode-0 wrapper signature under an opener context satisfying
$$b_{\mathsf{kunf}} = 0.$$
Since the complete wrapper issuer and opener context is embedded in the signature, two accepted interpretations of the same wrapper signature must use the same wrapper group context.

  Both accepted openings therefore reduce to baseline openings for the same baseline signature and the same derived message $\widehat{m}_0$. Two distinct accepted wrapper attributions would contradict trace-soundness of Γ.

- **Non-Claimability (NC).** The issuer and opener contexts in the non-claimability experiment may be adversarial.

  If the signing oracle is queried under an opener key satisfying

  $$b_{\mathsf{kunf}} = 1,$$

  it returns a mode-1 signature. Neither mode-1 nor mode-2 signatures admit accepted openings, so such a transcript cannot yield a successful non-claimability attack.

  If

  $$b_{\mathsf{kunf}} = 0,$$

  the logged wrapper transcript is a deterministic context extension of a logged baseline transcript. Any accepted reassignment to a different member public key therefore yields an accepted reassignment of the corresponding baseline transcript, contradicting non-claimability of Γ.

- **Certificate-Unforgeability (CUnf).** Consider a fresh accepting wrapper forgery under the honestly generated wrapper issuer public key

  $$\mathsf{ipk}' = (\mathsf{ipk}^0, \mathsf{ipk}^1).$$

  For a mode-b forgery with $b \in \{0, 1\}$, a reduction guesses the mode b, embeds the honest baseline issuer public key supplied by the certificate-unforgeability challenger as ipk^b, and generates the other issuer key pair independently. Join and signing queries for the embedded branch are forwarded to the baseline oracles; queries for the other branch are answered locally using the exposed member secret keys. Removing the wrapper encoding gives a fresh accepting baseline forgery under ipk^b. Indeed, replay of the baseline transcript would imply replay of its deterministic wrapper extension. A wrong guess causes only a constant-factor loss.

  It remains to consider a mode-2 forgery

  $$\sigma_m^* = (2, h^*, \sigma_m^*, x^*, \mathsf{ipk}', \mathsf{opk}').$$

  Define the corresponding source signature

  $$\widetilde{\sigma}_m \coloneqq (1, h^*, \sigma_m^*, \mathsf{ipk}', \mathsf{opk}').$$

  For the following reduction, the honest baseline issuer public key is embedded as ipk^1, while the component $(\mathsf{isk}^0, \mathsf{ipk}^0)$ is generated independently.

  If

  $$(\widetilde{\sigma}_m, m^*, \mathsf{opk}') \notin \mathcal{Q}_\sigma,$$

  then the embedded baseline signature σ_m^* yields a fresh accepting baseline certificate-unforgeability forgery under ipk^1 on

  $$(m^*, 1, h^*, \mathsf{ipk}', \mathsf{opk}').$$

  Otherwise, the source signature was produced by the signing oracle using a wrapper certificate

  $$(\sigma_s, x_j)$$

  generated by the honest join oracle. Consequently,

  $$h^* = H(x_j).$$

Acceptance of the mode-2 forgery additionally implies

$$H(x^*) = h^*.$$

Thus, x^* is a preimage of the one-way-function value $H(x_j)$.

A one-wayness reduction guesses the relevant certificate record and embeds its challenge image as the corresponding value h^*. Since the wrapper certificate is hidden behind an opaque handle, the reduction need not reveal the unknown auxiliary preimage. The challenge image suffices to answer every mode-1 signing query involving the selected handle, since the underlying signing oracle is invoked on $\widehat{m}_1(h^*)$. If the adversary's mode-2 forgery refers to the selected record, the extracted value x^* inverts the challenge. The record guess causes only the corresponding polynomial query loss.

Hence a successful certificate-unforgeability adversary yields either a baseline certificate-unforgeability adversary or an inverter for H.

- **Opening-Unforgeability (OpU).** A reduction embeds the honest baseline issuer public key as ipk^0, generates $(\mathsf{isk}^1, \mathsf{ipk}^1)$ independently, and answers join queries with $b_{\mathsf{kunf}} = 0$ using the baseline join oracle and queries with $b_{\mathsf{kunf}} = 1$ locally using isk^1.

  Accepted openings are possible only for mode-0 signatures, hence only under $b_{\mathsf{kunf}} = 0$ and the issuer component ipk^0. The wrapper join relation restricted to queries with $b_{\mathsf{kunf}} = 0$ is therefore in one-to-one correspondence with the baseline join relation. In particular, if a pair lies outside the complete wrapper join relation, then it also lies outside this restricted relation. Consequently, an accepted wrapper attribution to such a pair yields an accepted baseline attribution under ipk^0 to a pair outside the baseline join relation, contradicting opening-unforgeability of Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{KUnf}}$ preserves anonymity, non-frameability, traceability, trace-soundness, non-claimability, certificate-unforgeability, and opening-unforgeability under the security of the baseline scheme and the one-wayness of H. Nevertheless, the key-unforgeability experiment reveals the complete wrapper certificate

$$(\sigma_s, x),$$

allowing the adversary to transform an authenticated mode-1 signing-oracle output into a distinct, fresh, and accepting mode-2 signature without learning the challenger-generated member secret key. Hence key-unforgeability is independent of the other seven notions.

6.1.8 Proof of Theorem 32 (Opening-Unforgeability)

Proof. We provide a counterexample via the black-box wrapper specified in Constructions 21–23.

The wrapper uses disjoint encodings for ordinary member public keys and gadget identities:

$$(0, \mathsf{upk}) \qquad \text{and} \qquad (1, t),$$

respectively. Thus, no honestly generated wrapper member public key is syntactically equal to a gadget identity.

Extended opener public keys have the form

$$\mathsf{opk}' = (\mathsf{opk}, t),$$

where honestly generated opener keys always use

$$t = \mathsf{hon}.$$

An adversary controlling the opener public key may instead supply an arbitrary well-formed noncanonical tag.

The wrapper uses two disjoint signature encodings:

$$(0, \sigma_m, \mathsf{ipk}, \mathsf{opk}')$$

for ordinary signatures and

$$(1, \sigma_m, \mathsf{ipk}, \mathsf{opk}')$$

for gadget signatures.

For a wrapper message m, define the corresponding baseline messages

$$\widehat{m}_0 \coloneqq (m, 0, \mathsf{ipk}, \mathsf{opk}')$$

and

$$\widehat{m}_1 \coloneqq (m, 1, \mathsf{ipk}, \mathsf{opk}').$$

These are the complete baseline messages under the global convention; no additional context wrapping is applied.

The wrapper signing algorithm always produces ordinary signatures. Gadget signatures can nevertheless be generated locally by an adversary that knows a member secret key and a corresponding baseline membership certificate.

Correctness. Honestly generated member public keys have ordinary form

$$(0, \mathsf{upk}),$$

and honestly generated opener public keys have the form

$$(\mathsf{opk}, \mathsf{hon}).$$

The wrapper signing algorithm always generates an ordinary signature

$$(0, \sigma_m, \mathsf{ipk}, \mathsf{opk}')$$

whose baseline component is computed on

$$\widehat{m}_0 = (m, 0, \mathsf{ipk}, \mathsf{opk}').$$

Verification, opening, and opening verification consistently use the same baseline message and public-key context. Moreover, the prescribed opening algorithm wraps the baseline member public key as

$$(0, \mathsf{upk}).$$

Correctness therefore follows directly from correctness of Γ.

Why opening-unforgeability fails. In the opening-unforgeability experiment, the issuer public key ipk is honestly generated, whereas the adversary may choose the opener public key.

The adversary generates an opener key pair

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_{\$} \mathsf{KgO}_{\Gamma}(\mathsf{pp}_{\Gamma}),$$

The parameter-generation and issuer-key-generation algorithms are inherited unchanged from Γ. Malformed inputs are rejected according to the global parsing convention. Write

$$\mathsf{opk}' = (\mathsf{opk}, t), \qquad \widehat{m}_b \coloneqq (m, b, \mathsf{ipk}, \mathsf{opk}'),$$

and define

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}') \coloneqq \left[(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}') = (\mathsf{ipk}, \mathsf{opk}')\right].$$

$\underline{\mathsf{KgM}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Run

$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{usk}', \mathsf{upk}') \leftarrow (\mathsf{usk}, (0, \mathsf{upk})).$$

$\underline{\mathsf{KgO}'_\Gamma(\mathsf{pp}_\Gamma)}$**:**
Run

$$(\mathsf{osk}, \mathsf{opk}) \leftarrow_\$ \mathsf{KgO}_\Gamma(\mathsf{pp}_\Gamma)$$

and return

$$(\mathsf{osk}', \mathsf{opk}') \leftarrow (\mathsf{osk}, (\mathsf{opk}, \mathsf{hon})).$$

$\underline{\mathsf{Join}'_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}', \mathsf{opk}')}$**:**
Parse

$$\mathsf{upk}' = (b_M, z_M), \qquad \mathsf{opk}' = (\mathsf{opk}, t).$$

Return $\perp$ unless

$$b_M = 0.$$

Set

$$\mathsf{upk} \leftarrow z_M$$

and compute

$$\sigma_s \leftarrow_\$ \mathsf{Join}_\Gamma(\mathsf{isk}, \mathsf{ipk}, \mathsf{upk}, \mathsf{opk})$$

and return $\perp$ if $\sigma_s = \perp$. Otherwise return

$$\sigma_s.$$

$\underline{\mathsf{Sgn}'_\Gamma(\mathsf{usk}', \mathsf{upk}', \sigma_s, \mathsf{ipk}, \mathsf{opk}', m)}$**:**
Parse

$$\mathsf{upk}' = (b_M, z_M), \qquad \mathsf{opk}' = (\mathsf{opk}, t),$$

and set

$$\mathsf{usk} \leftarrow \mathsf{usk}'.$$

Return $\perp$ unless

$$b_M = 0.$$

Set

$$\mathsf{upk} \leftarrow z_M.$$

Compute

$$\sigma_m \leftarrow_\$ \mathsf{Sgn}_\Gamma(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}_0)$$

and return $\perp$ if $\sigma_m = \perp$. Otherwise return

$$(0, \sigma_m, \mathsf{ipk}, \mathsf{opk}').$$

Construction 21: Wrapper $\Gamma^{\neg\mathsf{OpU}}$ (Part I of III)

discards osk, chooses any tag

$$t \neq \mathsf{hon},$$

and defines

$$\mathsf{opk}' = (\mathsf{opk}, t).$$

The opener secret key is not used in the attack.

Next, the adversary generates a baseline member key pair

$$(\mathsf{usk}, \mathsf{upk}) \leftarrow_\$ \mathsf{KgM}_\Gamma(\mathsf{pp}_\Gamma)$$

and defines the corresponding ordinary wrapper member public key

$$\mathsf{upk}' = (0, \mathsf{upk}).$$

It submits

$$(\mathsf{upk}', \mathsf{opk}')$$

to the honest wrapper join oracle and receives a membership certificate

$$\sigma_s \leftarrow_\$ \mathcal{O}^{\mathsf{isk}}_{\mathsf{Join}_\Gamma}(\mathsf{upk}', \mathsf{opk}').$$

For any message m, set

$$\widehat{m}_1 \coloneqq (m, 1, \mathsf{ipk}, \mathsf{opk}').$$

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Opn}'_\Gamma(\mathsf{osk}', \mathsf{opk}', \mathsf{ipk}, m, \sigma'_m)}$**:**

Parse

$$\mathsf{opk}' = (\mathsf{opk}, t),$$

$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}'),$$

and set

$$\mathsf{osk} \leftarrow \mathsf{osk}'.$$

Return $\perp$ unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}').$$

Compute

$$(\mathsf{upk}, O) \leftarrow_{\$} \mathsf{Opn}_\Gamma(\mathsf{osk}, \mathsf{opk}, \mathsf{ipk}, \widehat{m}_0, \sigma_m)$$

and return $\perp$ if $(\mathsf{upk}, O) = \perp$. Otherwise return

$$((0, \mathsf{upk}), O).$$

$\underline{\mathsf{Vfy}'_\Gamma(\mathsf{ipk}, \mathsf{opk}', m, \sigma'_m)}$**:**

Parse

$$\mathsf{opk}' = (\mathsf{opk}, t).$$

If

$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}'),$$

return 0 unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_0, \sigma_m).$$

If

$$\sigma'_m = (1, \sigma_m, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}'),$$

return 0 unless

$$t \neq \mathsf{hon}$$

and

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}').$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_1, \sigma_m).$$

Return 0.

Construction 22: Wrapper $\Gamma^{\neg\mathsf{OpU}}$ (Part II of III)

Continuation of the wrapper from Part I. The notation and conventions are as in Part I.

$\underline{\mathsf{Jdg}'_\Gamma(\mathsf{ipk}, \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, O)}$**:**

Parse

$$\mathsf{opk}' = (\mathsf{opk}, t).$$

If

$$\sigma'_m = (0, \sigma_m, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}'),$$

parse

$$\mathsf{upk}^* = (b_M, z_M).$$

Return 0 unless

$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}')$$

and

$$b_M = 0.$$

Set

$$\mathsf{upk} \leftarrow z_M$$

and return

$$\mathsf{Jdg}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_0, \sigma_m, \mathsf{upk}, O).$$

If

$$\sigma'_m = (1, \sigma_m, \widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}'),$$

parse

$$\mathsf{upk}^* = (b_M, z_M).$$

Return 0 unless

$$t \neq \mathsf{hon},$$
$$\mathsf{Ctx}(\widehat{\mathsf{ipk}}, \widehat{\mathsf{opk}}'),$$
$$b_M = 1, \qquad z_M = t.$$

Otherwise return

$$\mathsf{Vfy}_\Gamma(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_1, \sigma_m).$$

Return 0.

Construction 23: Wrapper $\Gamma^{\neg\mathsf{OpU}}$ (Part III of III)

Using the locally generated member secret key and the certificate returned by the join

oracle, the adversary computes

$$\sigma_m \leftarrow_{\$} \mathsf{Sgn}_{\Gamma}(\mathsf{usk}, \mathsf{upk}, \sigma_s, \mathsf{ipk}, \mathsf{opk}, \widehat{m}_1).$$

It forms the gadget signature

$$\sigma'_m \leftarrow (1, \sigma_m, \mathsf{ipk}, \mathsf{opk}')$$

and claims the gadget identity

$$\mathsf{upk}^* \leftarrow (1, t).$$

By correctness of the baseline scheme,

$$\mathsf{Vfy}_{\Gamma}(\mathsf{ipk}, \mathsf{opk}, \widehat{m}_1, \sigma_m) = 1.$$

Since

$$t \neq \mathsf{hon},$$

the gadget branches of the wrapper verifiers accept:

$$\mathsf{Vfy}'_{\Gamma}(\mathsf{ipk}, \mathsf{opk}', m, \sigma'_m) = 1$$

and

$$\mathsf{Jdg}'_{\Gamma}(\mathsf{ipk}, \mathsf{opk}', m, \sigma'_m, \mathsf{upk}^*, 0) = 1.$$

The only join query made by the adversary added the pair

$$\big((0, \mathsf{upk}), \mathsf{opk}'\big),$$

whereas the accepted attribution concerns

$$\big((1, t), \mathsf{opk}'\big).$$

Because ordinary and gadget member public keys use disjoint encodings,

$$(1, t) \neq (0, \mathsf{upk}).$$

Therefore

$$\big((1, t), \mathsf{opk}'\big) \notin \mathcal{R}_{\mathsf{join}}.$$

Thus, without invoking any opening algorithm, the adversary produces an accepted attribution to a member–opener pair that was never submitted to the honest join oracle. Hence $\Gamma^{\neg\mathsf{OpU}}$ is not opening-unforgeable.

Why the other seven properties hold.

- **Anonymity (ANON).** The opener public key is honestly generated and therefore has

  $$t = \mathsf{hon}.$$

  The gadget branch is consequently disabled. Every challenge signature is an ordinary mode-0 signature obtained from a baseline signature on

  $$\widehat{m}_0 = (m, 0, \mathsf{ipk}, \mathsf{opk}').$$

  The public context embedded in the wrapper signature is independent of the hidden signer bit. Opening-oracle queries are answered by forwarding the corresponding ordinary baseline transcripts. Thus, any anonymity adversary against the wrapper yields one against Γ.

- **Non-Frameability (NF).** The fixed honest wrapper member public key has ordinary form
$$(0, \mathsf{upk}^{\mathsf{hon}}).$$
The gadget branch accepts only the unique gadget identity
$$(1, t).$$
These encodings are syntactically disjoint. Therefore the gadget branch cannot attribute any signature to the fixed honest member.

  Any successful frame must use an ordinary mode-0 signature and opening. Removing the deterministic wrapper encoding yields a successful baseline frame against $\mathsf{upk}^{\mathsf{hon}}$ on the corresponding derived message. Replay freshness is preserved by the deterministic mode and context encoding. This contradicts non-frameability of Γ.

- **Traceability (TRC).** The opener public key is honestly generated and hence satisfies
$$t = \mathsf{hon}.$$
Gadget signatures cannot verify in this context. Every verifying wrapper signature must therefore have mode 0 and corresponds to a verifying baseline signature on
$$\widehat{m}_0.$$
The prescribed wrapper opening algorithm invokes the prescribed baseline opening algorithm on the same baseline transcript and wraps the resulting identity in ordinary form. Any successful wrapper traceability attack would therefore yield a successful traceability attack against Γ.

- **Trace-Soundness (TS).** Consider a fixed wrapper message-signature pair admitting two accepted attribution tuples. The wrapper signature explicitly records its issuer and opener public-key context. Hence both accepted interpretations use the same ipk and opk', and therefore the same opener tag t.

  If the signature has mode 1, the gadget branch accepts only the unique member public key
$$(1, t).$$
Thus, two different accepted attribution tuples cannot arise.

  If the signature has mode 0, both accepted openings reduce to baseline openings for the same derived message, baseline signature, issuer key, and opener key. Two distinct accepted wrapper attributions would therefore contradict trace-soundness of Γ.

- **Non-Claimability (NC).** Every wrapper signing-oracle output has mode 0. A successful non-claimability adversary must reuse an exact logged wrapper transcript and therefore cannot change the mode bit or the embedded public-key context.

  Every accepted attribution of such a transcript consequently uses the ordinary branch and reduces to an accepted attribution of the corresponding logged baseline transcript. An accepted reassignment to a different ordinary wrapper member public key would yield a successful reassignment against Γ. The mode-1 gadget branch cannot be activated using an exact mode-0 signing-oracle transcript.

- **Certificate-Unforgeability (CUnf).** Let
$$\sigma'^{*}_{m}$$

be a fresh accepting wrapper forgery under the honest issuer key.

If σ'^{*}_{m} has mode 0, removing the deterministic mode and context encoding yields an accepting baseline signature on

$$\widehat{m}^{*}_{0}.$$

If the corresponding baseline transcript had previously been returned by the baseline signing oracle, then the complete wrapper transcript would also have been recorded. Hence wrapper freshness implies baseline freshness.

If σ'^{*}_{m} has mode 1, its baseline component is an accepting baseline signature on

$$\widehat{m}^{*}_{1}.$$

The wrapper signing oracle never produces mode-1 signatures and never asks the baseline signing oracle to sign a message of this form. Therefore the embedded baseline signature is fresh.

In either case, a successful wrapper forgery yields a successful baseline certificate-unforgeability forgery.

- **Key-Unforgeability (KUnf).**

  The same case distinction applies to key-unforgeability.

  A fresh mode-0 wrapper forgery yields a fresh baseline forgery on

  $$\widehat{m}^{*}_{0}.$$

  A mode-1 wrapper forgery contains a baseline signature on

  $$\widehat{m}^{*}_{1},$$

  while the wrapper signing oracle only produces baseline signatures on mode-0 messages. Thus, the embedded mode-1 baseline transcript is fresh.

  Consequently, every successful wrapper key-unforgeability adversary yields a successful key-unforgeability adversary against Γ.

□

Conclusion. The wrapper $\Gamma^{\neg\mathsf{OpU}}$ preserves anonymity, non-frameability, traceability, trace-soundness, non-claimability, certificate-unforgeability, and key-unforgeability by black-box reduction to Γ. Nevertheless, an adversarial opener can create a valid mode-1 signature and have it accepted as an attribution to the gadget identity $(1, t)$, although only an ordinary identity of the form $(0, \mathsf{upk})$ was submitted to the honest join oracle. Hence opening-unforgeability is independent of the other seven notions.

Acknowledgements. We are grateful to the anonymous reviewers for their helpful comments and suggestions. This work was in part supported by the European Union's Horizon Europe project LICORICE (grant agreement no. 101168311) and the CHIST-ERA project REMINDER through the Austrian Science Fund (FWF) project number I 6650-N. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the funding agencies. Neither European Union nor the granting authorities can be held responsible for them.

References

[ABCG17] Quentin Alamélou, Olivier Blazy, Stéphane Cauchie, and Philippe Gaborit. A code-based group signature scheme. *Des. Codes Cryptogr.*, 82(1-2):469–493, 2017. URL: https://doi.org/10.1007/s10623-016-0276-6, doi:10.1007/S10623-016-0276-6.

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000. doi:10.1007/3-540-44598-6_16.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceedings*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000. doi:10.1007/3-540-45539-6_18.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve A. Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, volume 9696 of *Lecture Notes in Computer Science*, pages 117–136. Springer, 2016. doi:10.1007/978-3-319-39555-5_7.

[BCC+20] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. *J. Cryptol.*, 33(4):1822–1870, 2020. URL: https://doi.org/10.1007/s00145-020-09357-w, doi:10.1007/S00145-020-09357-W.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 551–572. Springer, 2014. doi:10.1007/978-3-662-45611-8_29.

[BCN+10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, 2010. doi:10.1007/978-3-642-15317-4_24.

[BHS19] Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2181–2198. ACM, 2019. doi:10.1145/3319535.3354257.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003. doi:10.1007/3-540-39200-9_38.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick D. McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, pages 168–177. ACM, 2004. doi:10.1145/1030083.1030106.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011. doi:10.1007/978-3-642-19571-6_16.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005. doi:10.1007/978-3-540-30574-3_11.

[CE26] Shiqi Cao and Keita Emura. Another look at non-frameability in group signatures for anonymous auctions. *IACR Cryptol. ePrint Arch.*, 2026:1213, 2026. URL: https://eprint.iacr.org/2026/1213.

[CLNS17] Jan Camenisch, Anja Lehmann, Gregory Neven, and Kai Samelin. UC-secure non-interactive public-key encryption. In Boris Köpf and Steve Chong, editors, *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 217–233. IEEE Computer Society, 2017. doi:10.1109/CSF.2017.14.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991. doi:10.1007/3-540-46416-6_22.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006. doi:10.1007/11958239_13.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 551–565. ACM, 2018. doi:10.1145/3196494.3196507.

[DSS20] David Derler, Kai Samelin, and Daniel Slamanig. Bringing order to chaos: The case of collision-resistant chameleon-hashes. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 462–492. Springer, 2020. doi:10.1007/978-3-030-45374-9_16.

[Emu26] Keita Emura. On the traceability of group signatures: Uncorrupted user must exist. *Theor. Comput. Sci.*, 1073:115898, 2026. URL: https://doi.org/10.1016/j.tcs.2026.115898, doi:10.1016/J.TCS.2026.115898.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In Colin Boyd and Juan Manuel González Nieto, editors, *Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4-6, 2005, Proceedings*, volume 3574 of *Lecture Notes in Computer Science*, pages 455–467. Springer, 2005. doi:10.1007/11506157_38.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2012. doi:10.1007/978-3-642-34931-7_5.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007. doi:10.1007/978-3-540-76900-2_10.

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 121–145. Springer, 2016. doi:10.1007/978-3-662-53644-5_5.

[KSS19] Stephan Krenn, Kai Samelin, and Christoph Striecks. Practical group-signatures with privacy-friendly openings. In *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019*, pages 10:1–10:10. ACM, 2019. doi:10.1145/3339252.3339256.

[KY05] Aggelos Kiayias and Moti Yung. Efficient secure group signatures with dynamic joins and keeping anonymity against group managers. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*, volume 3715 of *Lecture Notes in Computer Science*, pages 151–170. Springer, 2005. doi:10.1007/11554868_11.

[KY06] Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *Int. J. Secur. Networks*, 1(1/2):24–45, 2006. doi:10.1504/IJSN.2006.010821.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2002. doi:10.1007/3-540-45708-9_8.

[PNH+22] Maharage Nisansala Sevwandi Perera, Toru Nakamura, Masayuki Hashimoto, Hiroyuki Yokoyama, Chen-Mou Cheng, and Kouichi Sakurai. A survey on group signatures and ring signatures: Traceability vs. anonymity. *Cryptogr.*, 6(1):3, 2022. URL: https://doi.org/10.3390/cryptography6010003, doi:10.3390/CRYPTOGRAPHY6010003.

[SSE+12] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, volume 7293 of *Lecture Notes in Computer Science*, pages 715–732. Springer, 2012. doi:10.1007/978-3-642-30057-8_42.

A Use of Generative AI

This section discloses the use of generative-AI tools during the preparation and revision of this manuscript. The authors take full responsibility for the complete manuscript, including all definitions, constructions, experiments, theorem statements, proofs, literature claims, and citations.

A.1 Tool and Scope of Use

We used OpenAI ChatGPT, with models from the GPT-5 family, as an interactive writing, review, and consistency-checking assistant. Manuscript excerpts, mathematical definitions, security experiments, proof drafts, and LaTeX source code were provided through conversational interfaces.

ChatGPT was used for the following categories of tasks.

Language and exposition. The system assisted with revising English prose for clarity, concision, consistency, and academic style. This included the abstract, introduction, motivation, informal explanations of security notions, proof overviews, captions, transitions, and parts of the related-work discussion. In some cases, complete rewritten paragraphs were proposed and subsequently reviewed and edited by the authors before inclusion.

LaTeX preparation and consistency. ChatGPT assisted with LaTeX boilerplate, experiment and game layouts, tables, construction environments, algorithm interfaces, figure formatting, macro consistency, and diagnosis of compilation and layout problems. It was also used to locate unresolved references, inconsistent labels, stale terminology, and mismatches between summary tables and detailed proofs.

The authors independently checked the semantics of all generated experiments, algorithms, constructions, and figures.

Formal definitions and security experiments. The authors supplied the intended adversarial models, trust assumptions, algorithm interfaces, corruption settings, oracle capabilities, and winning conditions. ChatGPT was used to review the resulting definitions and experiments for issues such as inconsistent quantification, omitted parsing or failure conditions, incomplete replay filters, missing public-key context, incorrect oracle interfaces, accidental use of unavailable secret information, and discrepancies between informal descriptions and formal experiments.

Suggested modifications were not adopted automatically. They were compared against the intended security model and either rejected, corrected, or manually incorporated by the authors.

Proof review and presentation. ChatGPT was used to review proof drafts, challenge individual game transitions, and improve the organization of reductions and hybrid arguments. This included checking whether reductions possess the witnesses, trapdoors, secret keys, and oracle access they use; checking statement–proof freshness before invoking simulation-sound extractability; distinguishing decryption-soundness from key-verifiability; reviewing reductions to digital-signature unforgeability and one-wayness; and checking reduction losses and advantage bounds.

The system also generated draft formulations of proof strategies, game hops, reductions, and advantage analyses. Some of this exposition was incorporated after substantial author review and correction.

ChatGPT did not constitute automated proof verification and did not establish the correctness of any theorem independently. The mathematical arguments and the decision to accept each proof step remain the responsibility of the authors.

Constructions and separation results. For the contrived construction, the secure construction, and the separation results, the authors supplied the underlying research ideas, intended security properties, construction principles, and proof goals. ChatGPT assisted with organizing the algorithms, reviewing the interaction between the constructions and the security experiments, checking the preservation arguments for the separation wrappers, and improving the presentation of the corresponding proofs.

In particular, the system was used to compare informal security intuitions with the information actually exposed in the formal experiments, to check freshness preservation between wrapper and baseline transcripts, and to identify inconsistencies between evolving construction variants and their proof descriptions.

The final constructions, security notions, separation claims, and proofs were accepted only after manual inspection by the authors.

A.2 Literature and External Sources

The scientific references and related-work claims were selected and checked by the authors. No citation or bibliographic entry was taken from AI.

Web access used during manuscript preparation served primarily to verify publication details, source material, and applicable disclosure policies; it was not treated as evidence for mathematical claims without independent checking.

A.3 Human Verification and Limitations

Generative-AI systems can produce plausible but incorrect mathematical text. During preparation of this manuscript, ChatGPT occasionally suggested incorrect or incomplete proof transitions, confused different experiments, used information unavailable to a reduction, or referred to outdated versions of constructions.

Accordingly, the authors manually checked:

- every formal definition and security experiment against its intended adversarial setting;
- every oracle interface and the information available to the relevant parties;
- every proof transition against an explicit assumption or distributional argument;
- every extraction argument for statement–proof freshness and availability of the required witnesses, keys, and trapdoors; and
- every final contradiction and reduction loss against the exact target experiment.

AI-generated or AI-rewritten text was revised whenever it altered, overstated, or obscured the intended technical meaning. The complete manuscript was reviewed by the authors after the final AI-assisted consistency pass.

A.4 Authorship and Intellectual Contribution

ChatGPT did not originate the main research question concerning the dependence of dynamic group-signature guarantees on issuer, opener, and registration structure. It did not originate the central distinction between functional opening guarantees and membership semantics, nor the decision to introduce certificate-, key-, and opening-unforgeability.

The authors developed the scientific objectives, selected the formal model, designed the constructions, determined the security notions to be studied, and remain responsible for all proofs and conclusions. ChatGPT contributed to drafting, criticism, restructuring, consistency checking, and presentation, including technically substantive feedback on local proof steps.

ChatGPT is not an author and assumes no responsibility for the work. All scientific and legal responsibility for the manuscript remains with the human authors.