k-Anonymous Group Signatures

Shalini Banerjee[†,‡], Andrey Bozhko[⋆], Andy Rupp[⋆,‡]

[†]Karlsruhe Institute of Technology
[‡]KASTEL Security Research Labs
[⋆]University of Luxembourg
shalini.banerjee@kit.edu, {andrey.bozhko, andy.rupp}@uni.lu

July 9, 2026

Abstract

We review k-anonymity in authentication schemes, group signatures, and ring signatures. Existing constructions either require a signer to maintain state across interactions, or admit tracing algorithms that cost $\mathrm{O}(n^k)$ in the number n of signatures. We introduce k-anonymous group signatures (k-AGS), that achieves stateless signing with a tracing cost $\mathrm{O}(n+k)$, while necessarily sacrificing unlinkability. We present a generic construction of k-AGS together with an efficient instantiation. We additionally construct a proof-of-concept k-unlinkable group signature (k-UGS) scheme that achieve unlinkability at the expense of $\mathrm{O}(n^2k)$ tracing overhead. We pose the question of whether stateless signing, $\mathrm{O}(n+k)$ tracing, and unlinkability are simultaneously achievable, and leave it as an open problem.

Building on our k-AGS framework, we introduce k-Anonymous Set Pre Constrained Group Signatures (k-ASPCGS), a *threshold* variant of Set Pre-Constrained Group Signatures introduced by Bartusek *et al.* (EUROCRYPT '23). As a proof-of-concept, we present two generic constructions of k-ASPCGS.

We show that our notions arise naturally in the context of content moderation in end-to-end encrypted messaging platforms, where users must remain anonymous until a threshold number of distinct illegitimate contents have been reported.

Keywords: k-anonymity, group signatures, set pre-constrained group signatures.

Contents

1 Introduction **3**
- 1.1 Our Contributions . . . 4
- 1.2 Our Approach . . . 5
- 1.3 Discussions . . . 10

2 Preliminaries **11**
- 2.1 Notations . . . 11
- 2.2 Public Key Encryption . . . 12
- 2.3 Signature Schemes . . . 13
- 2.4 Blind Signatures . . . 15
- 2.5 Polynomial Commitments . . . 16
- 2.6 Subversion-Resistant Non-Interactive Zero-Knowledge . . . 17
- 2.7 Set Pre-Constrained Encryption . . . 19
- 2.8 Multi-Dealer Fully Anonymous Secret Sharing . . . 21

3 k-Anonymous Group Signatures **23**
- 3.1 Definition and Security Notions . . . 23
- 3.2 k-AGS Construction . . . 27
- 3.3 Instantiation . . . 31

4 k-Unlinkable Group Signatures **32**
- 4.1 k-UGS Construction . . . 33

5 k-Anonymous Set Pre-constrained Group Signatures **37**
- 5.1 k-ASPCGS Construction . . . 39
- 5.2 Generic Transform . . . 43

A Security Proofs of k-AGS **47**
- A.1 Traceability . . . 47
- A.2 k-Anonymity . . . 50
- A.3 Non-GM Unlinkability . . . 54

B Security Proofs of k-ASPCGS Generic Transform **56**

1 Introduction

Anonymous systems are powerful cryptographic primitives that allow users to interact without revealing their true identities. Quite intuitively, such strong guarantees invite interesting applications, in and of itself, in protecting the privacy of users. Nevertheless, it creates a fundamental accountability problem: there is no way to trace a user who has behaved maliciously. This tension is well-recognized and has led to a spectrum of design choices — from full anonymity which leads to abuse with no recourse, to full traceability where anonymity guarantees collapse entirely. As a result, threshold anonymity (k-anonymity) as a natural middle ground has gained popularity in the literature, where a user can be held accountable only after k instances of malicious behavior, where k is a threshold parameter.

A concrete instantiation of this requirement arises in the context of content moderation in end-to-end encrypted (E2EE) messaging platforms. While E2EE systems enable exclusive content-sharing among end-points without disclosing keys to service providers, content moderation requires service providers to examine user-generated content to identify illegitimate contents and their originators. With such contrasting goals, service providers (such as Facebook, WhatsApp, Signal) take a standard approach : allow a user to report a decrypted content she believes to be illegitimate, and rely on known content detection or classifier based methodologies to flag malicious behavior. The former relies on a database of non-cryptographically hashed *pre-defined illegitimate contents*[1] (perpetual hash functions such as Microsoft PhotoDNA, pHash), and the latter detects *new instances of illegitimate contents* (reminiscent of the "retrospective" approach considered in [GKVL21]).

Nevertheless, deanonymizing a user based on a single reported content might be too demanding and not be desirable in the following sense : an honest user may send an illegitimate content with a non-malicious motive and still get flagged for distributing illegal content [New25]. Additionally, state-of-art moderation systems employ non-cryptographic perpetual hashes with high false positive rates [May19], and classifier-based content moderation systems fail to understand cultural nuances and local context in identifying new illegal contents, scaling up the surface of false positives [fJ21, Let25, May19]. This motivates us to explore k-anonymity *in the sense that anonymity is preserved for a reportee until threshold number k of illegitimate contents has been reported by peers* — the rationale being, a flagged reportee is indeed likely to be malicious given multiple illegitimate contents are reported. We stress that we are not claiming that E2EE is an anonymous system — rather E2EE moderation reduces to a k-anonymity problem over anonymous actions. Henceforth, we focus on the problem in its own right.

We abstract the above, and model a setting that motivates our primitive : a system with a group manager, and registered users communicating over an E2EE platform. Users asynchronously send messages along with signatures. When k signatures are reported by peers, a group manager efficiently traces the identity of the signer — by "efficiently", we mean that the tracing algorithm runs in time $\mathrm{O}(n+k)$, where n is the number of reported signatures. Below this threshold, a signer remains anonymous — even to a group manager. Crucially, we require this guarantee to hold even in a *stateless* setting, since neither the group manager nor the users can maintain a synchronized state across interactions. This naturally leads to existing k-anonymity primitives as potential candidates for realizing this functionality.

The notion of k-anonymity is an established cryptographic tool, studied extensively in the context of authentication schemes, group signatures, ring signatures and traceback systems [TFS04, LRTY21, CLXZ25, TNL+25, BGH+25, ASY06, CHL06]. Interestingly, the property of stateless signing and efficient tracing does not appear together in the literature, to the best of our knowledge. Existing schemes that achieve stateless signing, most notably k-anonymous ring signatures [CLXZ25], do so by relying on combinatorially expensive algorithms. In the setting of our instantiation, the tracing algorithm would then grow as $\mathrm{O}(n^k)$ in the number of user reports. Quite evidently, this solution might not be sufficient to support the online, incremental nature of user reports. Conversely, the schemes with efficient tracing require signers to maintain state using counters [TFS04, CHL06, ASY06], which again does not capture our setting. Consequently, it seems natural to ask:

[1] predefined illegitimate contents are ideally defined by third-parties such as National Center for Missing and Exploited Children (NCMEC), Global Internet Forum to Counter Terrorism (GIFCT), etc.

How to efficiently trace a stateless signer who has produced more than k signatures on distinct messages?

An appropriate candidate for realizing this functionality is group signatures, that provides a clean separation between signature anonymity and authority-controlled tracing, which is precisely what our setting demands. We answer the above question in affirmative by introducing "k-anonymous group signatures" (k-AGS), a primitive that, to the best of our knowledge, has not been formally studied. Nevertheless, while constructing our scheme, we observe that our stateless efficient tracing yields "linkable signatures" with respect to a group manager — signatures produced by the same signer can be linked by the group manager below the threshold, though anonymity is protected. We then show how to achieve unlinkability against a group manager by constructing k-Unlinkable Group Signatures (k-UGS) that use the multi-dealer fully anonymous secret sharing (M-FASS) compiler of Bishop, Green, Ishai, Jain, and Lou [BGI+25] as a black box, albeit at a $\mathrm{O}(n^2k)$ tracing cost and requiring heavyweight obfuscation.

Interestingly, most of the existing k-anonymous constructions achieve unlinkability as a baseline guarantee. Together, these results suggest a tradeoff between unlinkability and efficiency in our k-AGS setting: our primary construction achieves stateless signing and efficient tracing at the cost of linkability, while unlinkability appears to demand either signer state or expensive obfuscation-based techniques. This makes us think whether there is any fundamental hardness in combining all the three properties simultaneously: statelessness, efficient tracing and unlinkability. *We leave the formal characterization of this tradeoff as an open problem.*

We next consider a natural refinement of k-AGS, motivated by our content moderation setting. In particular, we augment the k-AGS framework to incorporate a dataset $\mathcal{D}$ of pre-defined illegitimate contents: a group manager can deanonymize a user only if k signatures on distinct messages that belong to $\mathcal{D}$ has been reported. We refer to the foundational work in this direction which is due to Bartusek, Garg, Jain and Policharla [BGJP22]. The authors introduce set pre-constrained group signatures (SPCGS) — a group manager can trace the originator of a reported content provided it belongs to $\mathcal{D}$, however *lacking the threshold variant*. Notably, the work identifies the threshold variant as a stronger requirement, pointing it as appealing yet non-trivial to realize, and explicitly leaves it as a future work.

In this paper, we augment the SPCGS framework with our k-anonymity requirement to construct "k-anonymous set pre-constrained group signatures" (k-ASPCGS). This combines the content constraint of SPCGS with the threshold anonymity of k-AGS, and inherits the statelessness requirement from our base construction. We note that if the underlying k-AGS construction satisfies unlinkability, then the resulting k-ASPCGS scheme inherits this guarantee.

1.1 Our Contributions

We now summarize our main results.

- We review k-anonymity in authentication schemes [NSN05, TFS04, TS06], group signatures [ASY06, CHL06], ring signatures [CLXZ25, TNL+25], and traceback systems [BGH+25, LRTY21]. Existing constructions either require a signer to maintain state across interactions, or achieve stateless signing at the cost of tracing algorithms that run in $\mathrm{O}(n^k)$ in the number of signatures n. We introduce k-anonymous group signatures — a new primitive in a group signature setting — that achieves stateless signing with tracing cost $\mathrm{O}(n+k)$; we call our variant k-anonymous group signatures.
  - *Definition.* A k-AGS scheme refers to a system with a group manager, and multiple users who asynchronously send messages along with signatures. To incorporate the notion of k-anonymity, we embed the scheme anonymity parameter k directly into the group's public key. When k signatures are reported, a group manager can efficiently trace the identity of a signer.
  - *Constructions.* We generically construct k-AGS, assuming a polynomial commitment scheme, a blind signature scheme, a hash to field function and a subversion-resistant NIZK. Our primary construction does not achieve unlinkability against a group manager. We present an efficient instantiation of our generic k-AGS construction using KZG commitments [KZG10], a

PKE with lifted ElGamal, Schnorr signatures, a two-round blind signature scheme with the round-optimal blind signature of Fuchsbauer, Hanser, and Slamanig (FHS) [FHS15], and two non-interactive zero-knowledge argument systems $\mathsf{NIZK}_{\mathsf{KG}}$ and $\mathsf{NIZK}_{\mathsf{S}}$ with subversion-resistant Groth–Sahai proofs [GS08, ALSZ21].

We next construct k-UGS using an M-FASS compiler [BGI+25] as a black box, and achieve unlinkability against a group manager — nevertheless, admitting $\mathrm{O}(n^2 \cdot k)$ cost, in the number of signatures. We leave as an open problem whether stateless signing, $\mathrm{O}(n + k)$ tracing and full unlinkability are simultaneously achievable.

- Our next contribution is in formulating "k-anonymous set pre-constrained group signatures", which can be viewed as a threshold extension of set pre-constrained group signatures [BGJP22]. Our formalism builds on our k-AGS security definitions, while incorporating an additional property: a stateless sender is traceable only after producing k signatures on messages drawn from a predefined set of illegitimate messages.
  - *Definition.* We encode a database of illegitimate contents $\mathcal{D}$, along with a threshold anonymity parameter k in the group's public key. As in k-AGS, users send messages asynchronously, along with signatures. A group manager can efficiently deanonymize a signer who has generated k reported signatures on messages from $\mathcal{D}$.
  - *Constructions.* As a proof-of-concept, we present two generic constructions of k-ASPCGS—the first one is obtained via a simple modification of a k-AGS, while the second follows from a generic transformation. Both constructions additionally rely on a subversion-resistant NIZK and a set pre-constrained encryption scheme (see Definition 6). Given that our k-ASPCGS inherits the properties of an underlying k-AGS, it is stateless and efficiently traceable, albeit fully linkable. Similarly, instantiating our generic compiler with k-UGS directly yields an unlinkable variant of k-ASPCGS.

1.2 Our Approach

The main focus of this paper is to design a group signature scheme where a group manager can deanonymize a stateless signer, who has obliviously and independently produced k signatures on distinct messages. Intuitively, one might expect to find a solution by employing Shamir's secret sharing [Sha79]. More specifically, a sender (functioning as a "dealer") computes shares of his identity id, and encrypts a fresh share for every message he sends. Unfortunately, this approach alone does not suffice. The reason is that there are "multiple" senders, each encrypting an id-share for every message being communicated, which inarguably indicates that the reconstruction algorithm needs to be able to distinguish k shares corresponding to a specific id from a large pool of reported ciphertexts, with no prior knowledge on shares generated together.

This problem has been studied previously by Eldridge, Beck, Green, Heninger and Jain [EBG+24] as multi-dealer secret sharing (MDSS). They give a solution to MDSS by employing list decoding for Reed-Solomon (RS) codes, which requires constant number of dealers for robust reconstruction. In our setting, however, the number of signers is unbounded — we require support for polynomially many dealers. Another approach [EBG+24] based on interleaved RS codes somewhat relaxes the bounds, however at the cost of achieving only heuristic correctness. Additionally, we observe that the MDSS formalization does not incorporate a centralized authority or group manager, which is essential in our setting. Together, these observations show that the MDSS approach does not address our problem. We note that a recent work by Bishop, Green, Ishai, Jain and Lou [BGI+25] resolves the unbounded dealer limitation by constructing M-FASS — an MDSS scheme supporting polynomially many secrets using compute-and-compare obfuscation. However, M-FASS is a secret sharing primitive and does not directly yield a group signature scheme: it provides no signing or verification algorithm, no group manager, and no threshold tracing mechanism, all of which are essential in our setting. We therefore set M-FASS aside for now, though we will later employ it as a black box to achieve unlinkability against a group manager.

The core challenge underlying k-AGS — allowing anonymous actions till a threshold exceeds — is structurally reminiscent of the seminal work by Teranishi, Furukawa and Sako [TFS04], which introduces the notion of *k-Times Anonymous Authentication* (k-TAA). Conceptually, the notion allows a user to "anonymously" authenticate himself until k-times, beyond which his identity is traceable to a server (relevant in e-cash, e-voting, trial content-browsing applications). The key idea behind their construction is to have the server publish k hints, where users are registered by a separate entity called the group manager. During authenticating, a user computes a tag using a published hint, along with a proof that one of the k hints has been used to compute the tag. As long as a different hint is used per authentication session, the computed tags along with the authentication logs reveal no information on the identity of the user. Therefore, a user cannot send more than k messages anonymously—they are forced to reuse a hint when computing a tag for the $(k+1)$-th message. Obviously, an honest user in such a scheme has to be stateful, keeping track of which of the hints has already been used. Note that a key difference from our setting is that we merge the roles of an authenticating server and group manager (recall our example, where a service provider both registers users and audit originators of illegal contents). Despite the differences, k-TAA could be a natural starting point to realize our functionality.

A Flawed Attempt. Suppose we attempt to transform a stateful k-TAA to a stateless k-TAA as follows: during registration, the server generates k^2 hints $\{\mathsf{hint}_1, \ldots, \mathsf{hint}_{k^2}\}$. To send a message msg, the user computes $c = f(msg)$, where f is a function (possibly a random oracle) mapping the message space to $\{1, \ldots, k^2\}$. The user then uses hint_c to produce a tag, following the procedure of the underlying stateful scheme. Then, the probability that a hint is reused after ℓ messages is approximately $\frac{l^2}{2k^2}$, by the birthday bound. In particular, after $\ell = \Theta(k)$, collisions occur with constant probability, and after $\ell = \Theta(k^2)$, collisions occur with overwhelming probability. At a first glance, this approach seems to work. However, upon closer inspection, we observe two limitations. *Firstly*, if k^2 is chosen to be small, then an honest user sending small number of messages gets deanonymized with noticeable probability due to frequent collisions. *Secondly*, if k^2 is chosen to be large, then an adversary could flood the system with many "bad" messages, since they can track used hints. This transformation is therefore unsatisfactory. Our goal, instead, is to design a scheme in which even honest users who send a small number of messages that are mistakenly marked as "bad" retain anonymity, without relying on a delicate choice of k.

k-Anonymous Group Signatures. We design our k-AGS scheme with a primary focus on efficiency of the tracing procedure. As discussed earlier, the straightforward approach—having a signer include a fresh Shamir secret share of their identity id in each signature—leads to a tracing algorithm that must check all $(k+1)$-subsets of reported messages, i.e. $\binom{n}{k+1}$ candidate groupings, before being able to identify a signer who has crossed the threshold. At the same time, sticking to the secret-sharing approach remains preferable: it yields an efficient signing algorithm and guarantees perfect anonymity of the signer until the threshold is reached. Our goal, therefore, is to retain the secret-sharing of id while designing a scheme that enables efficient tracing.

We now highlight the main benefit of allowing efficient linkability of signatures produced by the same user. Linkability makes tracing efficient while still retaining the advantages of the secret-sharing approach. The tracing algorithm can first link signatures from the same user and then, once the threshold is reached, reconstruct the identity from the shares. The main challenge in designing such a scheme is to "separate" the mechanism used for linking signatures from the user's actual identity, thereby guaranteeing anonymity before the threshold while preserving unforgeability.

In our k-AGS construction, we adopt a direct approach to achieve linkability. During the key generation protocol between the user and the group manager, the user—together with their identity id, which is known to the group manager—obtains a unique *pseudo-identity* pid. Crucially, pid remains hidden from the group manager, and can only be generated by a user who successfully completes the key generation protocol. The pseudo-identity is then used to produce signatures and is explicitly included in each signature.

Before explaining the structure of pid, let us revisit what we mean by "having the signer include a fresh Shamir secret share of their identity id into a signature." Since the signer is stateless, the only[2]

[2]For simplicity, we focus on computing a share in a deterministic manner.

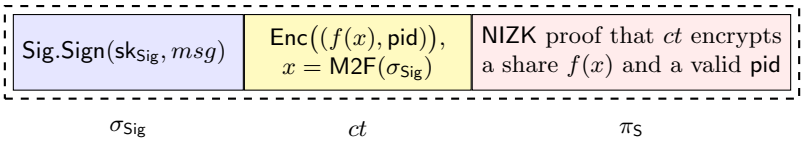

Figure 1: Schematic structure of a signature on msg in our k-AGS scheme.

available source of "freshness" for computing a share is the message itself. Following the thresholding approach of [BBM+21], the signer computes the x-coordinate of a share by signing the message with an underlying conventional signature scheme with a key tied to their pseudo-identity and mapping the resulting signature into a (sufficiently large) finite field, i.e., $x = \mathsf{M2F}(\mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, msg))$, and then sets the share as $\mathsf{sh} = (x, f(x))$, where f is a polynomial of degree k with $f(0) = \mathsf{id}$. The conventional Sig signature is included in the resulting k-AGS signature.

We include the user's Sig public key in the pseudo-identity. The choice to sign the message instead of a simpler approach of hashing the message into the field is motivated by the goal to achieve unframeability (group members shouldn't be able to sign messages on behalf of other members) at the lowest possible cost—we logically separate the unforgeability and tracing components of a k-AGS signature.

At this point, a malicious client[3] could attempt to cheat by including a random y instead of evaluating $f(x)$. Thus, we must enforce that the user (a) correctly evaluates f at x for every signature, and (b) consistently uses the same polynomial f across all signatures. To achieve this, we employ polynomial commitments [KZG10]. During the key generation protocol, the user commits to a chosen polynomial f, proving that it satisfies $f(0) = \mathsf{id}$. Each signature then includes an opening of this commitment at x, along with the commitment com itself.

To prevent the group manager from trivially deanonymizing the user by linking com with id during key generation, the user does not simply hand over com. Instead, the user obtains a blind signature on the commitment along with their Sig public key $\mathsf{pk}_{\mathsf{Sig}}$, while proving in zero-knowledge that the blindly signed message indeed includes a commitment to some polynomial f with $f(0) = \mathsf{id}$, without revealing either the commitment or the public key to the group manager during the protocol. As a result, the user obtains a pair $((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}})$, which we designate as their pseudo-identity pid. Due to blindness, the group manager cannot link pseudo-identities appearing in signatures with any key generation session (and thus cannot connect them to any id). At the same time, unforgeability of the blind signature scheme ensures that valid pseudo-identities can only be produced after successfully running the key generation protocol with the group manager. Finally, the binding property of the polynomial commitment scheme ensures that users cannot cheat by producing incorrect shares.

However, the user does not include the y-coordinate of the share, the opening, or the pseudo-identity pid in the signature in plaintext. Including any of these would break anonymity and unlinkability of k-AGS against parties that don't hold the opening key of the group signature—in particular, a plaintext pid would let such parties link a user's signatures—which conflicts with the usual anonymity property of group signatures. To prevent this, the user encrypts the y-coordinate of the share together with pid using the group manager's public key, and provides a non-interactive zero-knowledge proof that the resulting ciphertext indeed encrypts a y-coordinate of a share of the polynomial committed in com along with a valid pseudo-identity. The group's opening key is the secret key of the PKE scheme.

To summarize, during key generation the user commits to a polynomial f with $f(0) = \mathsf{id}$ by obtaining a blind signature σ_{BS} on the polynomial commitment com and their Sig public key, without revealing these values. The polynomial f together with $\mathsf{sk}_{\mathsf{Sig}}$ form the user's secret key sk. The pair $((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}})$ then serves as the user's pseudo-identity pid. A signature on a message msg consists of a conventional signature σ_{Sig} on msg, a ciphertext ct encrypting the Shamir share $f(x)$ (with $x = \mathsf{M2F}(\sigma_{\mathsf{Sig}})$) together with the pseudo-identity pid, and a proof that ct is well-formed—i.e. that it encrypts a valid pid and a share $f(x)$ of

[3] We note that [BBM+21] does not consider malicious clients against traceability.

the polynomial committed in *com*. The structure of a signature is schematically presented in Figure 1. This design ensures that signatures are linkable through the unforgeable pid by the group manager alone, that the shares are consistent and verifiable through commitment, that the signer's identity remains hidden from the group manager until the threshold k is exceeded, that $f(0)$ can be reconstructed, and that unforgeability follows from the signature Sig on msg.

We present generic construction in Section 3.2, and an efficient instantiation in Section 3.3. Additionally, we briefly discuss some more subtle design choices, such as settling on subversion-resistant NIZKs for computing proofs, at the end of Section 3.2.

k-Unlinkable Group Signatures. As evident from the above discussion, our primary k-AGS construction achieves unlinkability against parties who do not possess the group manager's secret key. A group manager can well decrypt any individual signature and recover pid immediately — and hence link any two signatures with matching pid at any time, even below the threshold k. This naturally motivates a stronger design in which all intermediate computations involving recovering pids must stay hidden until the group manager obtains an authorized set of k ciphertexts. That is, the only operation available to the group manager should be evaluating a function on k ciphertexts, that either returns id or $\perp$.

A natural candidate for realizing this functionality is program obfuscation, which hides program implementation while preserving its input-output behavior. Interestingly, a recent work by Bishop, Green, Ishai, Jain and Lou [BGI+25] already realizes a closely related functionality : a generic compiler that transforms any secret sharing scheme to M-FASS, assuming compute-and-compare obfuscation [WZ17, GKW17] and injective PRGs with polynomial stretch. At a high level, a dealer encrypt shares $(x_i, y_i)_{i\in[n]}$ of a secret s with a secret key sk of a PRF-based SKE scheme. A function $f_{\mathsf{sk},s}$ with hard-coded sk and s is then defined: given a set of ciphertexts, decrypt them, reconstruct using the underlying secret-sharing scheme, and output s if reconstruction succeeds. Under an ideal obfuscation model, all hard-coded values and intermediate computations remain hidden to an adversary. To instantiate this functionality concretely, the authors employ compute and compare obfuscation which obfuscates programs of the form $CC[f, \alpha, s]$ that on input x, output s if $f(x) = \alpha$, and $\perp$ otherwise — with statistical correctness and VBB security for an evasive α[4]. Robust reconstruction requires a certain SKE scheme with the property that decrypting ciphertexts with an incorrect sk results in $\perp$. This essentially gives an MDSS scheme for polynomially many secrets.

Observe that, this is precisely the functionality we require in our setting. Therefore, we use an M-FASS construction as a black-box to obtain a k-UGS scheme. Concretely, we treat each signature in our k-UGS scheme as a share in an M-FASS instantiation: the signer's id plays the role of the secret who outputs encrypted shares ct_i and an obfuscated function $f^i_{\mathsf{sk},\mathsf{id}}$. To trace, a group manager evaluates all obfuscated $f^i_{\mathsf{sk},\mathsf{id}}$ on all sets of ciphertexts. Since each sk is sampled uniformly at random, decrypting ciphertexts with an incorrect sk returns $\perp$, ensuring only ciphertexts generated from the same signer can be successfully decrypted. The final step is to reconstruct the degree k polynomial, and output id.

The overall cost of tracing in our k-UGS is $\mathrm{O}(n^2 \cdot k)$ in the number of reported signatures n, since the group manager must evaluate each of the nk programs on all accumulated n ciphertexts. This stands in sharp contrast to our k-AGS construction, where a group manager decrypts every ciphertext, groups them by matching pid and interpolates — incurring only $\mathrm{O}(n+k)$ cost. In fact, the blowup is the precise price for hiding pid from the group manager below the threshold.

We note that Bishop, Green, Ishai, Jain and Lou [BGI+25] instantiate their compiler using the compute-and-compare obfuscator of Wichs and Zirdelis [WZ17] and Goyal, Koppula, and Waters [GKW17], which achieves only statistical correctness. Goyal, Koppula, Vusirikala and Waters [GKVW20] construct a compute-and-compare obfuscator that achieves perfect correctness while retaining provable security under the LWE assumption. Plugging this obfuscator into the [BGI+25] compiler would improve the correctness of our k-UGS scheme. However, perfect correctness remains out of reach: the underlying SKE scheme requires that decrypting a ciphertext under an incorrect key returns $\perp$ only with overwhelming probability, contributing a negligible correctness error that persists regardless of the obfuscator used.

[4]Evasiveness ensures that for a random α, and any x chosen a-priori, a compute-and-compare program outputs 0 with overwhelming probability.

k-Anonymous Set Pre-Constrained Group Signatures. We now turn to the k-ASPCGS extension. Recall our earlier discussion on set pre-constrained group signatures [BGJP22], which allow tracing the originator of a reported message msg provided it belongs to a predefined dataset $\mathcal{D}$. In fact, the essence of constraining the power of tracing originators of messages is built upon the foundational work of *pre-constrained encryption* (PCE) by Ananth, Jain, Jin and Malavolta [AJ22], which allows encrypting a message that can be decrypted only if it satisfies some predicate, where the predicate is pre-constrained in the system setup phase. At a high level, the [BGJP22] scheme works as follows: a group manager encodes a database $\mathcal{D}$ in the public key of the group. To sign a message msg, a sender encrypts his identity id w.r.t. msg using a PCE scheme, such that id can be recovered from the ciphertext ct iff $msg \in \mathcal{D}$.

Having recalled this machinery, the extension of k-AGS construction to incorporate dataset $\mathcal{D}$ follows smoothly, with no significant new technical challenges. We present two constructions of k-ASPCGS. The first is a direct modification of our k-AGS scheme; the second is a generic transformation that converts any k-AGS into a k-ASPCGS.

Modification of k-AGS. Our first construction is a direct modification of our k-AGS scheme. Our goal is to ensure that signatures on "good" messages do not reveal information that could help the group manager trace the signer's identity. Recall that encrypting the y-coordinate of a share included in a signature is sufficient to prevent outsiders from reconstructing the secret polynomial f, and thus from identifying the signer. We can now achieve our goal by replacing public-key encryption with set pre-constrained encryption (SPCE). This approach is similar to the generic SPCGS framework of [BGJP22]: SPC encrypt identifying information—id in the case of [BGJP22], and a part of a share in ours.

The properties of SPCE guarantee that the group manager cannot extract y-coordinates from signatures—the only information, as previously argued, that would allow identity tracing in our k-AGS scheme. At the same time, the correctness of SPCE ensures that a user remains traceable if they sign more than k "bad" messages. We describe this modification in detail in Section 5.1.

Generic Compiler. The second construction is a generic compiler that transforms any k-AGS scheme into a k-ASPCGS scheme, treating k-AGS as a black box. Rather than modifying the k-AGS scheme directly, the compiler SPC-encrypts the entire k-AGS signature on the message and appends an NIZK proof that the ciphertext indeed contains a valid k-AGS signature. This design preserves the desired security properties in a modular way. The secrecy of the underlying SPCE scheme, combined with the zero-knowledge property of the NIZK, ensures that the hidden k-AGS signature cannot be learned by the server as long as the message is legitimate. Likewise, the k-anonymity of the k-AGS scheme directly carries over to the resulting k-ASPCGS. Traceability is also preserved: the correctness of the SPCE scheme, the knowledge soundness of the NIZK, and the traceability (and crucially, the statelessness) of the k-AGS together imply traceability of our construction.

Finally, the modularity of the approach makes it flexible enough to incorporate stronger properties. For instance, if the underlying scheme is k-UGS, then the k-ASPCGS scheme will inherit the property. Similarly, instantiating SPCE with PSI over authenticated inputs [BGJP25] would allow the group manager's set to be authenticated by a trusted third-party authority. We present our k-ASPCGS scheme in Section 5.2.

Security Notions. We introduce security notions for k-AGS, k-UGS, and k-ASPCGS, which, broadly speaking, are threshold extensions of the standard security properties of group signatures [BMW03] and set pre-constrained group signatures.

In Section 3.1, we define three security properties of k-AGS: traceability, k-anonymity, and non-GM unlinkability. Traceability corresponds to the standard notion but in a threshold setting: an adversary should not be able to produce more than $k \cdot usr$ valid signatures—where usr is the number of users it registered with the group manager—without at least one identity being traced. As in the classical notion, traceability for k-AGS also encompasses unforgeability and unframeability, since the adversary is allowed to register honest users and obtain (more than k) signatures from them.

In contrast, k-anonymity is not a straightforward generalization. For conventional group signatures, anonymity is defined against everyone except the group manager, whereas in our case even the group manager should not be able to identify the signer until that signer has produced more than k signatures. This splits the anonymity guarantee of k-AGS into two complementary notions: k-anonymity against the group manager

itself, bounded by the budget of k signatures per user, and a full unlinkability guarantee against parties that do not hold the opening key, which we call *non-GM unlinkability* and which corresponds to the standard "anonymity against non-openers" guarantee of group signatures.

In the k-anonymity definition, the group manager, after registering two target users, is asked to distinguish whether given signatures were produced by the first or the second user, subject to the budget of at most k signatures per user. In other words, the group manager should not be able to link key-generation sessions to the signatures of those users.

In the non-GM unlinkability definition, the two target users are registered honestly without adversarial interaction, and the adversary is then allowed to query signatures from either of them through a slot-indexed signing oracle. A hidden bit b controls the oracle's behavior: when $b = 0$ both slots are answered using the same user's secret key, so all signatures originate from one user; when $b = 1$ slot i is answered using the i-th target user's key, so the two slots correspond to two distinct users. The adversary's task is to tell the two cases apart.

In Section 4, we define k-unlinkability for k-UGS, which strengthens unlinkability to hold even against a group manager: as long as a user has produced no more than k signatures, the group manager should be unable to link any two of them as coming from the same user.

In Section 5, we define extensions of traceability and k-anonymity to the set pre-constrained setting. We leave out of scope the *set secrecy* property of set pre-constrained primitives—i.e., that no party outside the setup phase can learn anything about the predefined dataset—since it is directly inherited from the underlying SPC primitives and requires simulation-based definitions. Given that the focus of this paper is k-anonymity, we restrict ourselves to game-based adaptations of our k-anonymity and traceability notions.

1.3 Discussions

Other Related Work. k-anonymity in traceback systems (or variants thereof) has been extensively explored by the existing literature [TFS04, LRTY21, CLXZ25, TNL+25, BGH+25, ASY06, CHL06].

Linkable Ring Signatures. Choi, Liu, Xia and Zikas [CLXZ25] design *k-linkable ring signatures* (k-LRS) which allow a *stateless* user to sign anonymously at most k times, beyond which his signatures can be linked. The idea is to have the signer generate a fresh pseudo-identity pid as a linear combination of partial secret keys $x_1, \ldots, x_k$ and freshly sampled random weights $a_1, \ldots, a_k$ in the exponent w.r.t event-label e, and the signature also includes the weights. A link algorithm then solves linear equations over groups for every set of k signatures, with the overall complexity growing combinatorially. In a nutshell, the scheme is stateless and non-traceable while enforcing unlinkability, however only at the cost of an expensive link algorithm.

In [TNL+25], Tran, Nguyen, Liu, Pieprzyk and Susilo introduce a "weaker" variant of k-LRS, which instead of linking a group of $k + 1$ signatures by a user, allows linking at least two signatures from a group of $k + 1$ linked signatures. Their signing algorithm employs a PRF that is evaluated on a signing key and a counter that resets after k.

Bounded Anonymity E-cash. Camenisch, Hohenberger and Lysyanskaya [CHL06] propose a bounded-anonymity business model, where a *stateful* user can anonymously spend k e-coins per merchant, beyond which he is traceable by the merchant. The transactions are also unlinkable (as in [CLXZ25]) up to k spends per merchant.

Linkable Group Signatures. Au, Susilo and Yiu [ASY06] construct event-oriented linkable group signatures, where anonymity is event-specific. A signer remains untraceable and signatures are unlinkable (as in [CLXZ25]) until threshold k is reached. Nevertheless, the signer must maintain a counter specific to each event to track how many times he has signed.

Traceback Systems. In the context of tracing originators of fake news, Liu, Roche, Theriault and Yerukhimovich [LRTY21] design a $(k + \epsilon)$-traceback system over an E2EE messaging platform that traces the originator of a message reported by more than k users, where ϵ is an error bound. Their solution is to let the server maintain a counter whose status is updated obliviously by a user when reporting a message msg. After receiving more than $(1 \pm \epsilon)k$ reports on msg, the server can trace the originator. Note that such a

system restricts the number of reports a user can submit, and *a traceback is triggered when the same message has been reported by more than k users.*

Along this line of work, Branco, Green, Hedge, Jain and Kaptchuk [BGH+25] propose a traceback system for misinformation that identifies the originator of a message which has traversed at least k hops (forwarded k-times), before getting reported.

These are worthy tasks, and we do not disagree with it. However, our k-AGS formalism applies to a different context, which we think is also relevant. In particular, we allow a group manager to trigger a traceback on an originator only when k distinct messages have been reported — since a share coordinate x is derived from $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, c \,\|\, msg)$, repeated reports of the same message yield identical share coordinates, and hence do not contribute to additional independent shares for reconstruction.

Avenues for Further Research. We now highlight several prospective avenues for future research. To begin, our primary k-AGS construction achieves stateless signing and efficient tracing, at the cost of full linkability against group manager, whereas our unlinkability variant relies on heavyweight obfuscation-based techniques. This raises a fundamental open question: *can one construct a k-AGS scheme that simultaneously achieves stateless signing, efficient (linear) tracing, and k-unlinkability (against a group manager)?*

Our unlinkable scheme relies on M-FASS instantiated from compute-and-compare obfuscation and injective PRGs, and as such remains a *proof-of-concept* than a concrete proposal. A natural question is *whether unlinkability can be achieved from simpler assumptions*, such as pairings, lattices, or standard public-key primitives. This is interesting even if the combinatorial overhead persists, as it would broaden the cryptographic foundations on which k-UGS schemes can be built.

Our linkable k-AGS notion is conceptually distinct from notions such as "user-controlled linkability" in group signatures introduced by Diaz and Lehmann [DL21], where signers can selectively link their signatures. In our setting, linkability is, instead, controlled by a group manager. We leave the study of "group manager controlled linkability", including its relationship to existing linkability notions, its trust assumptions, and its privacy-accountability tradeoffs—as an interesting direction for future work.

We did not consider the scenario where the same message from a sender is reported by k peers. If such a functionality were to be implemented, it may be handled at the application level, for instance, by including receiver's identity, timestamp, etc. in the payload. Nevertheless, we do without it, and aim furthering this direction in the future.

A complementary direction is to investigate anonymous reputation systems. Reputation systems track past user behavior to govern future interactions. However, designing them for anonymous platforms is challenging, as actions cannot be linked to identities. In [SRKM25], Shih, Rosenberg, Kailad and Miers address this via stateful anonymous reputation systems, enforcing correctness through expensive zero-knowledge proofs. This raises an intriguing question: *can k-AGS serve as a stateless building-block for anonymous reputation systems, where k accumulated negative reports progressively restrict user privileges and eventually trigger deanonymization?* We leave the formal exploration of this connection as an interesting direction for future work.

Finally, following [BGJP22, BGJP25], we let $\mathcal{D}$ denote a database of illegitimate contents, rather than relying on perpetually hashed images as in real world deployments. Since, perpetual hashes do not inherit the full collision resistance guarantees of a standard cryptographic hash function, they introduce additional attack vectors, which are beyond the scope of this work. We leave investigating this direction to future work.

2 Preliminaries

2.1 Notations

We denote by $|S|$, the size of a finite set S. For any $n \in \mathbb{N}$, we consider $[n] \coloneqq \{1, \ldots, n\}$. We use $x \leftarrow y$ to assign x the value of y, and $x \leftarrow S$ to denote uniform sampling of x from the set S. For a probabilistic algorithm $\mathcal{A}$, we let $y \leftarrow \mathcal{A}(x)$ denote the process of running $\mathcal{A}(x)$ with fresh randomness and assigning the result to y. We write $(y; R) \leftarrow \mathcal{A}(x)$ when we want to store the random tape used by $\mathcal{A}$ in R, and

$y \leftarrow \mathcal{A}(x; R)$ to denote running $\mathcal{A}$ with random tape R. We use $(x_1, \ldots, x_\ell) \Leftarrow y$ to parse a tuple y and assign its components to $x_1, \ldots, x_\ell$.

In our pseudocodes we use lists, denoted by L with a suitable subscript. To initialize a list we write $L \leftarrow \emptyset$. A list contains tuples $(x_1, \ldots, x_\ell)$, and we index a list by the first element of a tuple: $L[x_1]$ returns $(x_2, \ldots, x_\ell)$ if such a tuple exists in L, and $\emptyset$ otherwise. The expression $(x_1, \ldots, x_\ell) \in L$ (resp. $\notin L$) evaluates to 1 (resp. 0) if the tuple is present in the list, and 0 (resp. 1) otherwise. The construct **for** $x \in L$ iterates over all tuples in L. To add a tuple to a list we write $L \leftarrow L \cup \{(x_1, \ldots, x_\ell)\}$, and to update the value indexed by x with y we write $L[x] \leftarrow y$.

We let λ denote the security parameter. A function $f : \mathbb{N} \to \mathbb{N}$ is said to be polynomial if there exists a constant c such that $f(n) \leq n^c$, for all $n \in \mathbb{N}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is called negligible in n, if it grows slower than n^{-c}, for every constant c. We measure negligibility with respect to the security parameter λ. We write PPT to denote probabilistic polynomial-time algorithms. We model adversaries and simulators as non-uniform PPT algorithms.

2.2 Public Key Encryption

We recall the standard syntax of PKE which we require to have indistinguishable encryptions under the chosen plaintext attack (INDCPA).

Definition 1 (Public Key Encryption) *We say that a triplet $\Pi_{\mathsf{PKE}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of PPT algorithms with a message space $\mathcal{M}$ is an* INDCPA *secure public key encryption (PKE) scheme if the following holds:*

$\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) :}$ *The key generation algorithm takes as input the security parameter 1^λ, and outputs a public key pk and a secret key sk.*

$\underline{ct \leftarrow \mathsf{Enc}(\mathsf{pk}, msg) :}$ *The encryption algorithm takes as input the public key pk, and a message $msg \in \mathcal{M}$, and outputs a ciphertext ct.*

$\underline{msg \leftarrow \mathsf{Dec}(\mathsf{sk}, ct) :}$ *The decryption algorithm takes as input the secret key sk and a ciphertext ct, and outputs a message msg.*

Furthermore, the algorithms must satisfy the following correctness and security properties.

Correctness: *For every message $msg \in \mathcal{M}$, the following holds:*

$$\Pr\left[msg = \mathsf{Dec}(\mathsf{sk}, ct) : \begin{matrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ ct \leftarrow \mathsf{Enc}(\mathsf{pk}, msg)\end{matrix}\right] = 1.$$

INDCPA **security**: *For a PPT adversary $\mathcal{A}$, its advantage against the* INDCPA *security of a public key encryption scheme PKE is defined as follows:*

$$\mathsf{Adv}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(\mathcal{A}, n) = \Pr\left[\mathbf{Exp}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(1^n, \mathcal{A}) \to 1 \mid b = 0\right] - \Pr\left[\mathbf{Exp}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(1^n, \mathcal{A}) \to 1 \mid b = 1\right],$$

where experiment $\mathbf{Exp}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(1^n, \mathcal{A})$ is defined on Figure 2.
A PKE scheme PKE is INDCPA *secure if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every PPT adversary $\mathcal{A}$ and for every $\lambda \in \mathbb{N}$ the following holds:*

$$\mathsf{Adv}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(\mathcal{A}, n) \leqslant \mathsf{negl}(\lambda).$$

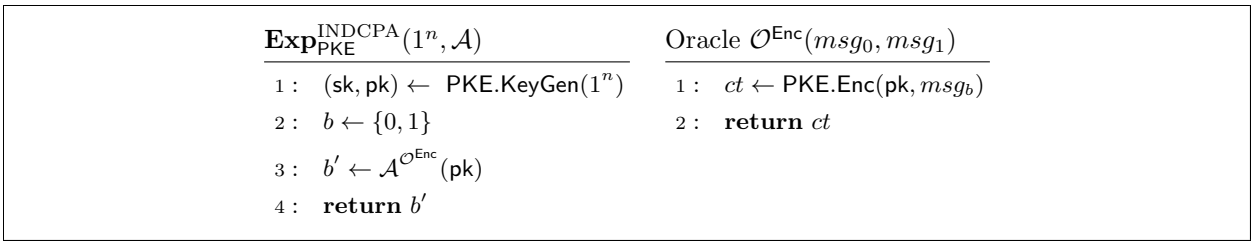

Figure 2: Experiment $\mathbf{Exp}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(1^n, \mathcal{A})$

2.3 Signature Schemes

We recall the standard syntax of a digital signature scheme, augmented with a deterministic mapping from the signature space into a prime field $\mathbb{Z}_p$. We require three security properties. First, strong unforgeability under chosen-message attack (sUFCMA). Second, that the composition $\mathsf{M2F} \circ \mathsf{Sig.Sign}$ is collision-resistant in the sense that no efficient algorithm—even one given the secret key—can produce two valid message–signature pairs (msg_0, σ_0) and (msg_1, σ_1) with $msg_0 \neq msg_1$ such that $\mathsf{M2F}(\sigma_0) = \mathsf{M2F}(\sigma_1)$. Third, *key anonymity* (kAnonSig): a signature carries no information about the signing key to a party that does not hold the corresponding public key. The first two properties are implied by the collision resistance of the underlying hash function in essentially every natural "hash-then-sign" instantiation (BLS, Schnorr, ECDSA, RSA-FDH, Ed25519) when $\mathsf{M2F}$ is itself a collision-resistant projection (e.g., the projection of a Schnorr signature (R, s) onto its scalar component s); the third holds, e.g., for Schnorr, where a signature (R, s) is uniformly distributed over $\mathbb{G} \times \mathbb{Z}_p$ and is therefore independent of the signing key absent the public key.

Definition 2 (Signature Scheme with Mapping to a Field) *A signature scheme* Sig *with a message space* $\mathcal{M}$ *and a signature space* Σ *is a tuple*

$$\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{M2F})$$

of PPT algorithms defined as follows:

$\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda):}$ *The key generation algorithm takes as input a security parameter* 1^λ*, and outputs a public key* pk *and a secret key* sk*.*

$\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg):}$ *The signing algorithm takes as input a secret key* sk *and a message* $msg \in \mathcal{M}$*, and outputs a signature* $\sigma \in \Sigma$*.*

$\underline{b \leftarrow \mathsf{Verify}(\mathsf{pk}, msg, \sigma):}$ *The verification algorithm takes as input a public key* pk*, a message* $msg \in \mathcal{M}$*, and a candidate signature* σ*, and outputs* $b = 1$ *if* σ *is a valid signature on* msg*, and* $b = 0$ *otherwise.*

$\underline{x \leftarrow \mathsf{M2F}(\sigma):}$ *The mapping algorithm is a deterministic function* $\mathsf{M2F}\colon \Sigma \to \mathbb{Z}_p$ *that maps signatures into the prime field* $\mathbb{Z}_p$*.*

For convenience, we let $\mathcal{R}_{\mathsf{Sig}}$ *denote the key-pair relation associated with* Sig*, defined as*

$$\mathcal{R}_{\mathsf{Sig}}(\mathsf{pk}, \mathsf{sk}) = 1 \iff \exists\, R\colon (\mathsf{pk}, \mathsf{sk}) = \mathsf{Sig.KeyGen}(1^\lambda; R).$$

Furthermore, the algorithms must satisfy the following correctness and security properties.

Correctness: *For every* $msg \in \mathcal{M}$*, the following holds:*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, msg, \sigma) = 1 : \begin{matrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, msg)\end{matrix}\right] = 1.$$

Strong Unforgeability under Chosen-Message Attack (sUFCMA): *For a PPT adversary $\mathcal{A}$, its advantage against the* sUFCMA *security of a signature scheme* Sig *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(\mathcal{A},\lambda) = \Pr[\mathbf{Exp}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(1^\lambda,\mathcal{A}) \to 1],$$

where the strong unforgeability experiment $\mathbf{Exp}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}$ *is defined in Figure 3.*
A signature scheme Sig *is* sUFCMA*-secure if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and for every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\mathsf{Adv}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(\mathcal{A},\lambda) \leqslant \mathsf{negl}(\lambda).$$

$\mathbf{Exp}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(1^\lambda,\mathcal{A})$

$Q \leftarrow \emptyset$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
$(msg^*,\sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(1^\lambda,\mathsf{pk})$
if $(msg^*,\sigma^*) \in Q$:
  return 0
if $\mathsf{Sig.Verify}(\mathsf{pk},msg^*,\sigma^*) = 0$:
  return 0
return 1

Oracle $\mathcal{O}^{\mathsf{Sign}}(msg)$

$\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk},msg)$
$Q \leftarrow Q \cup \{(msg,\sigma)\}$
return σ

Figure 3: Strong unforgeability experiment $\mathbf{Exp}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}$

Collision Resistance of $\mathsf{M2F} \circ \mathsf{Sig.Sign}$ (MFcr): *For a PPT adversary $\mathcal{A}$, its advantage against the collision resistance of* $\mathsf{M2F} \circ \mathsf{Sig.Sign}$ *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{MFcr}}_{\mathsf{Sig}}(\mathcal{A},\lambda) = \Pr[\mathbf{Exp}^{\mathrm{MFcr}}_{\mathsf{Sig}}(1^\lambda,\mathcal{A}) \to 1],$$

where the experiment $\mathbf{Exp}^{\mathrm{MFcr}}_{\mathsf{Sig}}$ *is defined in Figure 4.*
A signature scheme Sig *is* MFcr*-secure if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\mathsf{Adv}^{\mathrm{MFcr}}_{\mathsf{Sig}}(\mathcal{A},\lambda) \leqslant \mathsf{negl}(\lambda).$$

The experiment hands the adversary the secret key sk alongside pk, capturing the strongest natural setting: even a malicious signer with full knowledge of sk cannot produce two valid message–signature pairs on distinct messages whose signatures collide under M2F.

Key Anonymity (kAnonSig): *For a PPT adversary $\mathcal{A}$, its advantage against the key anonymity of a signature scheme* Sig *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(\mathcal{A},\lambda) = \Pr[\mathbf{Exp}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(1^\lambda,\mathcal{A}) \to 1 \mid \beta = 0] - \Pr[\mathbf{Exp}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(1^\lambda,\mathcal{A}) \to 1 \mid \beta = 1],$$

where the key anonymity experiment $\mathbf{Exp}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}$ *is defined in Figure 5.*
A signature scheme Sig *is key-anonymous if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\mathsf{Adv}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(\mathcal{A},\lambda) \leqslant \mathsf{negl}(\lambda).$$

$\underline{\mathbf{Exp}^{\mathrm{MFcr}}_{\mathsf{Sig}}(1^\lambda, \mathcal{A})}$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
2 : $(msg_0, \sigma_0, msg_1, \sigma_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}, \mathsf{sk})$
3 : **if** $msg_0 = msg_1$:
4 : **return** 0
5 : **if** $\mathsf{Sig.Verify}(\mathsf{pk}, msg_0, \sigma_0) = 0$:
6 : **return** 0
7 : **if** $\mathsf{Sig.Verify}(\mathsf{pk}, msg_1, \sigma_1) = 0$:
8 : **return** 0
9 : **if** $\mathsf{M2F}(\sigma_0) \neq \mathsf{M2F}(\sigma_1)$:
10 : **return** 0
11 : **return** 1

Figure 4: Collision resistance experiment $\mathbf{Exp}^{\mathrm{MFcr}}_{\mathsf{Sig}}$ for $\mathsf{M2F} \circ \mathsf{Sig.Sign}$

The adversary is given access to two signing oracles: $\mathcal{O}_0^{\mathsf{Sign}}$, which always signs under sk_0, and $\mathcal{O}_1^{\mathsf{Sign}}$, which signs under sk_β for the hidden bit β. Crucially, the adversary is given *neither* public key, and—to rule out the trivial attack against a deterministic scheme—it is forbidden from querying the same message to both oracles. Key anonymity then states that signatures produced under one key are indistinguishable from signatures produced under an independently generated key.

$\underline{\mathbf{Exp}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(1^\lambda, \mathcal{A})}$

1 : $Q_0, Q_1 \leftarrow \emptyset$
2 : $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
3 : $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
4 : $b \leftarrow \{0, 1\}$
5 : $b' \leftarrow \mathcal{A}^{\mathcal{O}_0^{\mathsf{Sign}}, \mathcal{O}_1^{\mathsf{Sign}}}(1^\lambda)$
6 : **return** b'

Oracle $\mathcal{O}_0^{\mathsf{Sign}}(msg)$

1 : **if** $msg \in Q_1$: **return** $\perp$
2 : $Q_0 \leftarrow Q_0 \cup \{msg\}$
3 : **return** $\mathsf{Sig.Sign}(\mathsf{sk}_0, msg)$

Oracle $\mathcal{O}_1^{\mathsf{Sign}}(msg)$

1 : **if** $msg \in Q_0$: **return** $\perp$
2 : $Q_1 \leftarrow Q_1 \cup \{msg\}$
3 : **return** $\mathsf{Sig.Sign}(\mathsf{sk}_b, msg)$

Figure 5: Key anonymity experiment $\mathbf{Exp}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}$

2.4 Blind Signatures

Definition 3 *A two-move blind signature scheme* BS *between a user* U *and a server* S *is a tuple*

$$\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{U}_1, \mathsf{S}_1, \mathsf{U}_2, \mathsf{Verify})$$

of PPT algorithms defined as follows:

$\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n) :}$ *The key generation algorithm takes as input a security parameter* 1^n *and outputs a public key* pk *and a secret key* sk*.*

$(m_1, \mathsf{st}_\mathsf{U}) \leftarrow \mathsf{U}_1(\mathsf{pk}, msg)$: *The first user algorithm takes as input a public key* pk *and a message* msg *to be signed, and outputs a message* m_1 *to be sent to the server and a state* st_U *to be kept locally.*

$m_2 \leftarrow \mathsf{S}_1(\mathsf{sk}, m_1)$: *The server algorithm takes as input a secret key* sk *and a user's first message* m_1*, and outputs a response message* m_2*.*

$\sigma \leftarrow \mathsf{U}_2(\mathsf{st}_\mathsf{U}, m_2)$: *The second user algorithm takes as input a user's state* st_U *and a server's response* m_2*, and outputs a signature* σ *on the message* msg*.*

$b \leftarrow \mathsf{Verify}(\mathsf{pk}, msg, \sigma)$: *The verification algorithm takes as input a public key* pk*, a message* msg*, and a candidate signature* σ*, and outputs* $b = 1$ *if* σ *is a valid signature on* msg*, and* $b = 0$ *otherwise.*

We recall the following security properties of blind signatures.

Blindness: *For a PPT adversary* $\mathcal{A}$*, its advantage against the blindness of a blind signature scheme* BS *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{Blind}}_{\mathsf{BS}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\mathrm{Blind}}_{\mathsf{BS}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathrm{Blind}}_{\mathsf{BS}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1],$$

where the blindness experiment $\mathbf{Exp}^{\mathrm{Blind}}_{\mathsf{BS}}$ *is defined in Figure* **??**.
A BS scheme BS *is blind if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and for every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\mathsf{Adv}^{\mathrm{Blind}}_{\mathsf{BS}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

Remark 1 For convenience, in addition to the presented definition, we will also make use of an n-message definition of the experiment $\mathbf{Exp}^{\mathrm{Blind}}_{\mathsf{BS}}$, in which the adversary, upon initialization, outputs a permutation π on $\{0, \ldots, n-1\}$ along with other values, and is allowed to output n messages. If bit $b = 1$, when $\mathcal{A}$ queries to sign i-th message, message $msg_{\pi(i)}$ is signed instead. It is easy to see that, since every permutation can be represented as a product of transpositions, the n-message security follows from the 2-message case via a hybrid argument.

One-More Unforgeability: *For a PPT adversary* $\mathcal{A}$*, its advantage against the one-more unforgeability of a blind signature scheme* BS *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{omUnforg}}_{\mathsf{BS}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\mathrm{omUnforg}}_{\mathsf{BS}}(1^\lambda, \mathcal{A}) \to 1],$$

where the one-more unforgeability experiment $\mathbf{Exp}^{\mathrm{omUnforg}}_{\mathsf{BS}}$ *is defined in Figure 7.*
A BS scheme BS *is one-more unforgeable if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and for every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\mathsf{Adv}^{\mathrm{omUnforg}}_{\mathsf{BS}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

2.5 Polynomial Commitments

Definition 4 *A polynomial commitment scheme* PC *is a tuple* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *of PPT algorithms defined as follows:*

$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d)$: *The setup algorithm takes as input a security parameter* 1^λ *and a maximum degree bound* d*, and outputs a commitment key* ck*.*

$com \leftarrow \mathsf{Commit}(\mathsf{ck}, f)$: *The* deterministic *commitment algorithm takes as input a commitment key* ck *and a polynomial* $f \in \mathbb{Z}_p[x]$ *of degree at most* d*, and outputs a commitment* com*.*

$\mathbf{Exp}^{\text{Blind}}_{\text{omUnforg}}(1^\lambda, \mathcal{A})$

1 : $count \leftarrow 0$
2 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$
3 : $(s, \{(msg_i, \sigma_i)\}_{i=1,\ldots,s\}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{S}_1}}(1^n, \mathsf{pk})$
4 : **if** $s \leq count$:
5 : **return** 0
6 : **if** $\exists (i,j)$: $(msg_i, \sigma_i) = (msg_j, \sigma_j)$:
7 : **return** 0
8 : **for** $i = 1, \ldots, s$:
9 : **if** $\mathsf{Verify}(\mathsf{pk}, msg_i, \sigma_i) = 0$:
10 : **return** 0
11 : **return** 1

Oracle $\mathcal{O}^{\mathsf{S}_1}(m)$

1 : $count \leftarrow count + 1$
2 : $m_{out} \leftarrow \mathsf{S}_1(\mathsf{sk}, m)$
3 : **return** m_{out}

Figure 7: One-More Unforgeability Experiment $\mathbf{Exp}^{\text{omUnforg}}_{\mathsf{BS}}$

$\omega \leftarrow \mathsf{Open}(\mathsf{ck}, f, x, y)$: *The opening algorithm takes as input a commitment key* ck*, a committed polynomial* f*, and points* $x, y \in \mathbb{Z}_p$*, and outputs an opening* ω*.*

$b \leftarrow \mathsf{Verify}(\mathsf{ck}, com, x, y, \omega)$: *The verification algorithm takes as input a commitment key* ck*, a commitment* com*, points* x, y*, and a proof* π*, and outputs a bit* b*.*

We recall the following security properties of polynomial commitments.

Binding: *A PC scheme* PC *is binding if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and for every* $d, \lambda \in \mathbb{N}$ *the following holds:*

$$\Pr\left[\begin{matrix} y_1 \neq y_2 \\ \wedge \mathsf{Verify}(\mathsf{ck}, com, x, y_1, \omega_1) : \\ \wedge \mathsf{Verify}(\mathsf{ck}, com, x, y_2, \omega_2) \end{matrix} \begin{matrix} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (com, x, y_1, y_2, \omega_1, \omega_2) \leftarrow \mathcal{A}(\mathsf{pk}) \end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Hiding: *For a PPT adversary* $\mathcal{A}$*, its advantage against the hiding property of a polynomial commitment scheme* PC *is defined as follows:*

$$\mathsf{Adv}^{\text{Hiding}}_{\mathsf{PC}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1],$$

where the hiding experiment $\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}$ *is defined in Figure 8.*
A PC scheme PC *is hiding if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every PPT adversary* $\mathcal{A}$ *and for every* $\lambda \in \mathbb{N}$ *the following holds:*

$$\mathsf{Adv}^{\text{Hiding}}_{\mathsf{PC}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

2.6 Subversion-Resistant Non-Interactive Zero-Knowledge

In this section, we recall definition of subversion-resistant non-interactive zero-knowledge (NIZK) schemes [BFS16, ALSZ21]. These are NIZK schemes that maintain the zero-knowledge property even when the common reference string (CRS) is generated by an adversary, rather than by a trusted setup. In other words,

$\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{A})$

1 : $(\mathsf{ck}, d, (x_0, y_0), \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$
2 : $b \leftarrow \{0,1\}$
3 : $count \leftarrow 0$
4 : **if** $b = 0$:
5 : $\quad f \leftarrow \{f \in \mathbb{Z}_p[x] : deg(f) = d \wedge f(x_0) = y_0\}$
6 : **else** :
7 : $\quad f \leftarrow \{f \in \mathbb{Z}_p[x] : deg(f) = d\}$
8 : $com \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, f)$
9 : $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Open}}}(\mathsf{st}, com)$
10 : **return** b'

Oracle $\mathcal{O}^{\mathsf{Open}}(x)$

1 : **if** $count = d \vee x = x_0$
2 : $\quad$ **return** $\perp$
3 : $y \leftarrow f(x)$
4 : $\omega \leftarrow \mathsf{PC.Open}(\mathsf{ck}, f, x, y)$
5 : $count \leftarrow count + 1$
6 : **return** $(\omega, (x, y))$

Figure 8: Hiding experiment $\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}$

even if the CRS is maliciously crafted, the adversary should not learn any additional information from the proof beyond its validity. For example, slight modifications of Groth16 were proven to be subversion zero-knowledge [ALSZ21, Fuc18]. Along with conventional knowledge-soundness, we will use no less conventional, but stronger definition of simulation-extractability.

Let $\mathsf{RelGen}(1^n)$ be a relation generator that outputs an NP-relation $\mathcal{R}(x, w) \to \{0, 1\}$, where x is a statement and w is a witness. We denote the corresponding NP language by $\mathcal{L}_{\mathcal{R}}$. For convenience, we also write $(x, w) \in \mathcal{R}$ for pairs (x, w) such that $\mathcal{R}(x, w) = 1$.

Definition 5 *A subversion-resistant non-interactive zero-knowledge (NIZK) argument system* NIZK *for a relation generator* RelGen *is a tuple* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{CVerify}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ *of PPT algorithms defined as follows:*

$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R})$: *The setup algorithm takes as input a security parameter* 1^λ *and a relation* $\mathcal{R}$ *and outputs a common reference string* crs *and a simulation trapdoor* td.

$b \leftarrow \mathsf{CVerify}(\mathcal{R}, \mathsf{crs})$: *The CRS verification algorithm takes as input a relation* $\mathcal{R}$ *and a CRS* crs*, and outputs* $b = 1$ *if* crs *is well-formed, and* $b = 0$ *otherwise.*

$\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$: *The prove algorithm takes as input a CRS* crs*, a statement–witness pair* $(x, w) \in \mathcal{R}$*, and outputs a proof* π.

$b \leftarrow \mathsf{Verify}(\mathsf{crs}, x, \pi)$: *The verify algorithm takes as input a CRS* crs*, a statement* x*, and a proof* π*, and outputs and outputs* $b = 1$ *if the proof is valid, and* $b = 0$ *otherwise.*

$\pi \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, x)$: *The simulator algorithm takes as input a CRS* crs*, a simulation trapdoor* td*, and a statement* x*, and outputs a simulated proof* π.

Subversion Zero-Knowledge: *Let* $\mathcal{S}$ *be a PPT subverter and* $\mathcal{E}_{\mathcal{S}}$ *be a PPT simulation trapdoor extractor. Let* NIZK *be a NIZK scheme for* $\mathsf{RelGen}(1^n)$ *and* $\mathcal{R}$ *some relation from* $\mathsf{RelGen}(1^n)$*. For a pair of the subverter and a PPT adversary* $(\mathcal{S}, \mathcal{A})$*, an advantage against the subversion zero-knowledge of* NIZK*, the extractor* $\mathcal{E}_{\mathcal{S}}$ *and relation* $\mathcal{R}$ *is defined as follows:*

$$\mathsf{Adv}^{\text{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}},\mathcal{R}}((\mathcal{S}, \mathcal{A}), \lambda) = \Pr[\mathbf{Exp}^{\text{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}},\mathcal{R}}(1^\lambda, (\mathcal{S}, \mathcal{A})) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}},R}(1^\lambda, (\mathcal{S}, \mathcal{A})) \to 1 \mid b = 1],$$

where the subversion zero-knowledge experiment $\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}}}$ *is defined in Figure 9.*
A NIZK scheme NIZK *is subversion zero-knowledge (Sub-ZK) if there exists a negligible function* $\mathsf{negl}(\lambda)$ *and for every subverter* $\mathcal{S}$ *there exists a PPT simulation trapdoor extractor* $\mathcal{E}_{\mathcal{S}}$ *such that for every* $\lambda \in \mathbb{N}$*, every* $\mathcal{R} \in \mathsf{RelGen}(1^\lambda)$ *and every PPT adversary* $\mathcal{A}$*, the following holds:*

$$\mathsf{Adv}^{\mathrm{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}},\mathcal{R}}((\mathcal{S},\mathcal{A}),\lambda) \leqslant \mathsf{negl}(\lambda).$$

Knowledge Soundness: *Let* NIZK *be a NIZK scheme for* $\mathsf{RelGen}(1^n)$ *and* $\mathcal{R}$ *some relation from* $\mathsf{RelGen}(1^n)$*. For an adversary* $\mathcal{A}$*, an advantage against the knowledge soundness of a NIZK scheme* NIZK*, a PPT extractor* $\mathcal{E}_{\mathcal{A}}$ *and a relation* $\mathcal{R}$ *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{knSound}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}(\mathcal{A},\lambda) = \Pr[\mathbf{Exp}^{\mathrm{knSound}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}(1^\lambda,\mathcal{A}) \to 1],$$

where the knowledge soundness experiment $\mathbf{Exp}^{\mathrm{knSound}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}$ *is defined in Figure 9.*
A NIZK scheme NIZK *is knowledge-sound if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for PPT every adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathcal{E}_{\mathcal{A}}$ *and for every* $\lambda \in \mathbb{N}$*, every* $\mathcal{R} \in \mathsf{RelGen}(1^\lambda)$*, the following holds:*

$$\mathsf{Adv}^{\mathrm{knSound}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}(\mathcal{A},\lambda) \leqslant \mathsf{negl}(\lambda).$$

Simulation-Extractability: *We call a* NIZK *scheme simulation-extractable if the above definition of knowledge-soundness continues to hold even when* $\mathcal{A}$ *is additionally given oracle access to* $\mathsf{Sim}(\mathsf{crs},\mathsf{td},x)$ *for statements* x *of its choice, provided that* $\mathcal{A}$ *is restricted from outputting proofs for statements it queried to* Sim*. We denote corresponding experiment by* $\mathbf{Exp}^{\mathrm{SE}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}$*.*

$\mathbf{Exp}^{\mathrm{knSound}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}(1^\lambda,\mathcal{A})$

$(\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda,\mathcal{R})$
$(x,\pi;R) \leftarrow \mathcal{A}(1^\lambda,\mathsf{crs})$
$w \leftarrow \mathcal{E}_{\mathcal{A}}(1^\lambda,\mathsf{crs},\mathsf{td},R)$
if $(x,w) \notin \mathcal{R} \wedge \mathsf{NIZK.Verify}(\mathsf{crs},x,\pi) = 1$
  return 1
return 0

$\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}},\mathcal{R}}(1^\lambda,(\mathcal{S},\mathcal{A}))$

$b \leftarrow \{0,1\}$
$(\mathsf{crs},\mathsf{st};R) \leftarrow \mathcal{S}(1^\lambda,\mathcal{R})$
if $\mathsf{CVerify}(\mathcal{R},\mathsf{crs}) = 0$:
  return 0
$\mathsf{td} \leftarrow \mathcal{E}_{\mathcal{S}}(1^\lambda,\mathcal{R},R)$
$(x,w,\mathsf{st}) \leftarrow \mathcal{A}(1^\lambda,\mathsf{td},\mathsf{st})$
if $(x,w) \notin \mathcal{R}$: **return** 0
if $b = 0$: $\pi \leftarrow \mathsf{Prove}(\mathsf{crs},x,w)$
if $b = 1$: $\pi \leftarrow \mathsf{Sim}(\mathsf{crs},\mathsf{td},x)$
$b' \leftarrow \mathcal{A}(\mathsf{st},\pi)$
return b'

Figure 9: Experiments $\mathbf{Exp}^{\mathrm{knSound}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{A}},\mathcal{R}}$ and $\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK},\mathcal{E}_{\mathcal{S}},\mathcal{R}}$

2.7 Set Pre-Constrained Encryption

We recall the syntax of set pre-constrained encryption from [BGJP22, BGJP25], state its correctness condition, and then formulate security properties we rely on.

Definition 6 (Set Pre-Constrained Encryption) *A set pre-constrained encryption scheme (SPCE)* Π_{SPCE} *over a universe* U *of items and a message space* $\mathcal{M}$ *is a tuple of PPT algorithms*

$$\Pi_{\mathsf{SPCE}} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec}),$$

parameterized by a maximum set size n. It is defined as follows:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{D})$: *The key-generation algorithm takes as input the security parameter 1^λ and a set $\mathcal{D} \subseteq \mathsf{U}$ of size at most n, and outputs a public key pk and a secret key sk.*

$ct \leftarrow \mathsf{Enc}(\mathsf{pk}, x, msg)$: *The encryption algorithm takes as input a public key pk, an item $x \in \mathsf{U}$, and a message $msg \in \mathcal{M}$, and outputs a ciphertext ct.*

$msg' \leftarrow \mathsf{Dec}(\mathsf{sk}, ct)$: *The decryption algorithm takes as input a secret key sk and a ciphertext ct, and outputs either a message $msg' \in \mathcal{M}$ or the failure symbol $\perp$.*

Perfect Correctness: *An SPCE scheme is said to be perfectly correct if for every $\lambda \in \mathbb{N}$ and $\mathcal{D} \subseteq \mathsf{U}, |\mathcal{D}| \leq n$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that for all $msg \in \mathcal{M}, x \in \mathcal{D}$ and any random tape R it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, ct) \neq msg : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{D}) \\ ct \leftarrow \mathsf{Enc}(\mathsf{pk}, x, msg; R)\end{array}\right] \leq \mathsf{negl}(\lambda),$$

and for all $msg \in \mathcal{M}, x \notin \mathcal{D}$ and any random tape R it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, ct) \neq \{msg, \perp\} : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{D}) \\ ct \leftarrow \mathsf{Enc}(\mathsf{pk}, x, msg; R)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Scope of the security definitions: In the literature, SPCE is conventionally formalized through a simulation-based ideal functionality $\mathcal{F}_{\mathsf{SPCE}}$ [BGJP22, BGJP25] that bundles two guarantees: *(i)* a ciphertext under a tag x hides the underlying plaintext as long as $x \notin \mathcal{D}$, whether or not the distinguisher holds the decryption key, and *(ii)* a malicious client cannot misuse the protocol to learn information about the server's set $\mathcal{D}$. The present work focuses on k-anonymity, which depends only on guarantee *(i)*; the client-side set-security property *(ii)* is orthogonal to anonymity and is out of scope for this paper. We therefore formulate two game-based notions that together cover guarantee *(i)* in the two adversarial regimes we encounter: *outsider IND-CPA* (oINDCPA), for adversaries that do not hold the secret key, and *receiver IND-CPA* (rINDCPA), for adversaries that play the role of a malicious server and choose $(\mathsf{pk}, \mathsf{sk})$ themselves. Both notions are implied by the UC formalization of [BGJP22, BGJP25].

Outsider IND-CPA (oINDCPA).: For a fixed set $\mathcal{D} \subseteq \mathsf{U}$ with $|\mathcal{D}| \leq n$, oINDCPA captures the requirement that a ciphertext under a tag $x \notin \mathcal{D}$ hides the underlying plaintext from anyone who does not hold the decryption key. In the experiment $\mathbf{Exp}^{\text{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}$ (Figure 10), the challenger runs $\mathsf{SPCE.Gen}(1^\lambda, \mathcal{D})$, hands $(\mathsf{pk}, \mathcal{D})$ to $\mathcal{A}$, and answers encryption queries (x, msg) with $x \notin \mathcal{D}$ by returning either $\mathsf{Enc}(\mathsf{pk}, x, msg)$ (when $b = 0$) or $\mathsf{Enc}(\mathsf{pk}, x, msg')$ for a uniformly random $msg' \xleftarrow{\$} \mathcal{M}$ (when $b = 1$). Queries with $x \in \mathcal{D}$ are answered with $\perp$.

For a PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\text{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(\mathcal{A}, \lambda) = \left|\Pr[\mathbf{Exp}^{\text{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1]\right|.$$

SPCE *is* oINDCPA*-secure if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every PPT $\mathcal{A}$, every $\mathcal{D} \subseteq \mathsf{U}$ with $|\mathcal{D}| \leq n$, and every $\lambda \in \mathbb{N}$, $\mathsf{Adv}^{\text{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda)$.*

Receiver IND-CPA (rINDCPA): Receiver IND-CPA targets a different adversarial regime: the adversary itself plays the role of the receiver, choosing the key pair $(\mathsf{pk}, \mathsf{sk})$. Without a constraint on pk this notion would be trivially unsatisfiable—a corrupt receiver could output a "broken" pk whose effective set is empty, after which the property fails for every x. We therefore parameterize the definition by an efficiently computable *validator* $\mathsf{Auth}(\mathsf{pk}, \mathcal{D}) \in \{0, 1\}$ that checks consistency of pk with the declared set $\mathcal{D}$ and gates the game on this check. As in subversion zero-knowledge or knowledge soundness, we let Auth depend on the adversary: SPCE

$\mathbf{Exp}^{\mathrm{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(1^\lambda, \mathcal{A})$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SPCE.Gen}(1^\lambda, \mathcal{D})$
2 : $b \xleftarrow{\$} \{0,1\}$
3 : $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Enc}}}(\mathsf{pk}, \mathcal{D})$
4 : **return** b'

Oracle $\mathcal{O}^{\mathsf{Enc}}(x, msg)$

1 : **if** $x \in \mathcal{D}$: **return** $\perp$
2 : **if** $b = 0$:
3 : $\quad ct \leftarrow \mathsf{SPCE.Enc}(\mathsf{pk}, x, msg)$
4 : **else** :
5 : $\quad msg' \xleftarrow{\$} \mathcal{M}$
6 : $\quad ct \leftarrow \mathsf{SPCE.Enc}(\mathsf{pk}, x, msg')$
7 : **return** ct

Figure 10: Outsider IND-CPA experiment $\mathbf{Exp}^{\mathrm{oINDCPA}}_{\mathsf{SPCE},\mathcal{D}}$.

is rINDCPA-secure if for every PPT $\mathcal{A}$ there exists a PPT Auth for which no adversary whose $(\mathsf{pk}, \mathcal{D})$ output is accepted by Auth can win the game below. This adversary-dependent flavor accommodates concrete SPCE constructions whose validators have to observe the adversary's queries to an idealized primitive (random oracle, generic group, etc.) to extract a witness of pk-consistency.

In the experiment $\mathbf{Exp}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D}}$ (Figure 11), the adversary outputs $(\mathsf{pk}, \mathsf{sk}, state)$ on input $(1^\lambda, \mathcal{D})$; if $\mathsf{Auth}(\mathsf{pk}, \mathcal{D}) = 0$ the experiment aborts with output 0, otherwise it samples b and answers encryption queries as in the outsider game (with the adversary now holding sk).

For PPT $\mathcal{A}$ and Auth*,*

$$\mathsf{Adv}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(\mathcal{A}, \mathsf{Auth}, \lambda) = \left| \Pr[\mathbf{Exp}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D},\mathsf{Auth}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D},\mathsf{Auth}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1] \right|.$$

SPCE *is* rINDCPA*-secure if for every PPT $\mathcal{A}$ there exist a PPT validator* Auth *and a negligible function* $\mathsf{negl}(\lambda)$ *such that for every $\mathcal{D} \subseteq \mathsf{U}$ with $|\mathcal{D}| \le n$ and every $\lambda \in \mathbb{N}$,* $\mathsf{Adv}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D}}(\mathcal{A}, \mathsf{Auth}, \lambda) \leqslant \mathsf{negl}(\lambda)$.

$\mathbf{Exp}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D},\mathsf{Auth}}(1^\lambda, \mathcal{A})$

1 : $(\mathsf{pk}, \mathsf{sk}, state) \leftarrow \mathcal{A}(1^\lambda, \mathcal{D})$
2 : **if** $\mathsf{Auth}(\mathsf{pk}, \mathcal{D}) = 0$: **return** 0
3 : $b \xleftarrow{\$} \{0,1\}$
4 : $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Enc}}}(state)$
5 : **return** b'

Oracle $\mathcal{O}^{\mathsf{Enc}}(x, msg)$

1 : **if** $x \in \mathcal{D}$: **return** $\perp$
2 : **if** $b = 0$:
3 : $\quad ct \leftarrow \mathsf{SPCE.Enc}(\mathsf{pk}, x, msg)$
4 : **else** :
5 : $\quad msg' \xleftarrow{\$} \mathcal{M}$
6 : $\quad ct \leftarrow \mathsf{SPCE.Enc}(\mathsf{pk}, x, msg')$
7 : **return** ct

Figure 11: Receiver IND-CPA experiment $\mathbf{Exp}^{\mathrm{rINDCPA}}_{\mathsf{SPCE},\mathcal{D},\mathsf{Auth}}$.

2.8 Multi-Dealer Fully Anonymous Secret Sharing

We recall the notion of M-FASS introduced by Bishop, Green, Ishai, Jain, and Lou [BGI+25], in the threshold setting and with robust reconstruction. Informally, M-FASS strengthens standard threshold secret sharing in two ways: (i) any unauthorized collection of shares is indistinguishable from a uniformly random multiset of the same size, even when it contains a mixture of shares produced by independent dealers sharing different secrets; (ii) the reconstruction algorithm is anonymous—it does not require party identities and is robust against arbitrary mixtures of shares of multiple secrets.

$\mathbf{Exp}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}(1^\lambda, \mathcal{A})$

$(k, s_0, s_1, m, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$
if $m > k$: **return** 0
$\mathsf{pp} \leftarrow \mathsf{M\text{-}FASS.Setup}(1^\lambda, k)$
$\mathsf{sks}_0 \leftarrow \mathsf{M\text{-}FASS.KeyGen}(\mathsf{pp}, s_0)$
$\mathsf{sks}_1 \leftarrow \mathsf{M\text{-}FASS.KeyGen}(\mathsf{pp}, s_1)$
$b \leftarrow \{0, 1\}$
if $b = 0$:
  for $j = 1, \ldots, m$:
    $\mathsf{sh}_j \leftarrow \mathsf{M\text{-}FASS.Share}(\mathsf{pp}, \mathsf{sks}_0)$
else :
  for $j = 1, \ldots, m-1$:
    $\mathsf{sh}_j \leftarrow \mathsf{M\text{-}FASS.Share}(\mathsf{pp}, \mathsf{sks}_0)$
  $\mathsf{sh}_m \leftarrow \mathsf{M\text{-}FASS.Share}(\mathsf{pp}, \mathsf{sks}_1)$
$b' \leftarrow \mathcal{A}(\mathsf{st}_{\mathcal{A}}, \mathsf{pp}, \{\mathsf{sh}_j\}_{j\in[m]})$
return b'

Figure 12: Multi-dealer anonymity experiment $\mathbf{Exp}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}$

For our purposes, we adopt two adaptations to the formalization of [BGI+25]. First, we restrict attention to threshold access structures: a secret s is recoverable from any $k+1$ shares produced under a single sharing of s. Second, we adopt a *stateless sharing* syntax in which each invocation of Share produces a single fresh share rather than all shares at once. The latter captures applications where shares of the same secret are emitted asynchronously over time and tied to a long-term per-secret sharing key, rather than batched up-front by a single dealer. We refer to [BGI+25] for instantiations from compute-and-compare obfuscation [**?**, **?**] and injective PRGs.

Definition 7 (Stateless Robust M-FASS) *A stateless robust multi-dealer fully anonymous secret sharing scheme* M-FASS *for an anonymity threshold k and a secret space $\mathcal{S}$ is a tuple*

$$\mathsf{M\text{-}FASS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Share}, \mathsf{Recon})$$

of PPT algorithms defined as follows:

$\underline{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, k) :}$ *The setup algorithm takes as input a security parameter 1^λ and a threshold parameter k, and outputs public parameters* pp.

$\underline{\mathsf{sks} \leftarrow \mathsf{KeyGen}(\mathsf{pp}, s) :}$ *The dealer key generation algorithm takes as input the public parameters* pp *and a secret $s \in \mathcal{S}$, and outputs a sharing key* sks *that will be used to produce all shares of s.*

$\underline{\mathsf{sh} \leftarrow \mathsf{Share}(\mathsf{pp}, \mathsf{sks}) :}$ *The (stateless) sharing algorithm takes as input the public parameters* pp *and a sharing key* sks*, and outputs a single fresh share* sh*. Independent invocations on the same* sks *produce independent shares of the same underlying secret.*

$\underline{L \leftarrow \mathsf{Recon}(\mathsf{pp}, \mathsf{S}) :}$ *The (anonymous, robust) reconstruction algorithm takes as input the public parameters* pp *and an unordered multiset of shares* S*, and outputs a list $L \subseteq \mathcal{S}$ of recovered secrets.*

We require the following correctness, anonymity, and robustness properties.

Robust Correctness: *For every $\lambda \in \mathbb{N}$, every polynomial $\ell = \ell(\lambda)$, and every choice of secrets $s_1, \ldots, s_\ell \in \mathcal{S}$ and counts $m_1, \ldots, m_\ell \in \mathbb{N}$, the following holds:*

$$\Pr\left[\begin{array}{c} \mathsf{Recon}(\mathsf{pp}, \mathsf{S}) = \\ \{s_i : m_i \geq k+1\} \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, k) \\ \forall i \in [\ell] \colon \mathsf{sks}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}, s_i) \\ \forall i \in [\ell],\, j \in [m_i] \colon \mathsf{sh}_{i,j} \leftarrow \mathsf{Share}(\mathsf{pp}, \mathsf{sks}_i) \\ \mathsf{S} \leftarrow \bigcup_{i \in [\ell]} \{\mathsf{sh}_{i,j}\}_{j \in [m_i]} \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Multi-Dealer Anonymity (MAnon): *For a PPT adversary $\mathcal{A}$, its advantage against the multi-dealer anonymity of* M-FASS *is defined as follows:*

$$\mathsf{Adv}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1],$$

where the multi-dealer anonymity experiment $\mathbf{Exp}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}$ is defined in Figure 12.
An M-FASS *scheme is multi-dealer anonymous if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every PPT adversary $\mathcal{A}$ and every $\lambda \in \mathbb{N}$ the following holds:*

$$\mathsf{Adv}^{\mathrm{MAnon}}_{\mathsf{M\text{-}FASS}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

The experiment captures the strongest natural notion of multi-dealer anonymity in the threshold setting. The adversary first chooses two candidate secrets s_0, s_1 and an unauthorized total count $m \leq k$. The challenger samples m shares either entirely from a fresh sharing of s_0, or from a mixture of $m-1$ shares of s_0 and one share from a fresh independent sharing of s_1. The adversary's task is to distinguish the two cases. Indistinguishability of these two distributions implies that no unauthorized collection of shares reveals whether it was generated by a single dealer or by multiple dealers sharing different secrets.
As shown in [BGI+25] (Theorem 8.5), an M-FASS scheme satisfying Definition 7 can be obtained generically from any (information-theoretic or computational) threshold secret sharing scheme by composition with multi-bit compute-and-compare obfuscation [WZ17,GKVW20] and an injective PRG with polynomial stretch in NC^1, both instantiable under the LWE assumption.

3 k-Anonymous Group Signatures

3.1 Definition and Security Notions

We now define k-AGS along with its correctness and security notions.

Definition 8 *A k-Anonymous Group Signature (k-AGS) scheme $\Pi[k, \mathcal{M}, \mathcal{ID}]$ with a message space $\mathcal{M}$, an identity space $\mathcal{ID}$ is a tuple* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace})$ *of PPT algorithms, along with an interactive protocol* KeyGen *defined as follows:*

$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, k)$: *The master key generation algorithm takes as input a security parameter 1^λ, and the scheme anonymity parameter k, and outputs a master public key* mpk*, a master secret key* msk*, and a master opening key* mok*.*

$\langle \mathsf{id}, (\mathsf{id}, \mathsf{sk}) \rangle \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C} \rangle$: *The interactive key generation protocol is run by a group manager* GM *with an input* msk*, and by a client* C*. It delivers an identity $\mathsf{id} \in \mathcal{ID}$ to both* GM *and* C*, and an identity signing key* sk *to* C*. The master public key* mpk *is an implicit input for both parties.*

$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)$: *The signing algorithm takes as input a master public key* mpk*, an identity signing key* sk*, and a message $msg \in \mathcal{M}$, and outputs a signature σ.*

$\{0, 1\} \leftarrow \mathsf{Verify}(\mathsf{mpk}, msg, \sigma)$: *The signature verification algorithm takes as input a master public key* mpk*, a message $msg \in \mathcal{M}$, and a signature σ, and outputs either 1 or 0, indicating accept or reject.*

$\{\mathsf{id}_1, \ldots, \mathsf{id}_m\} \leftarrow \mathsf{Trace}(\mathsf{mok}, k, \{\sigma_i\}_{i \in \mathcal{I}})$: *The identity reconstruction algorithm takes as input a master opening key* mok*, the scheme anonymity parameter k, a set of signatures $\{\sigma_i\}_{i \in \mathcal{I}}$, and outputs a potentially empty set of corresponding identities $\{\mathsf{id}_1, \ldots, \mathsf{id}_m\}, \mathsf{id}_i \in \mathcal{ID}$.*

$\mathbf{Exp}^{\text{kAnon}}_{\Pi}(1^\lambda, \mathcal{A})$

$\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{id}_0, \mathsf{id}_1 \leftarrow \perp$
$L_{\mathsf{st}} \leftarrow \emptyset$
$CSigns_0, CSigns_1 \leftarrow 0$
$(\mathsf{mpk}, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$
$\mathsf{st} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KeyGen}}}(\mathsf{st})$
if $\mathsf{id}_0 = \perp \vee \mathsf{id}_1 = \perp$:
  return 0
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{st})$
return b'

Oracle $\mathcal{O}^{\mathsf{KeyGen}}(i, msg_{in})$

if $i \notin \{0,1\}$: **return** $\perp$
if $L_{\mathsf{st}}[i] = \emptyset$: $(step, \mathsf{st}) \leftarrow (1, \mathsf{mpk})$
else : $(step, \mathsf{st}) \leftarrow L_{\mathsf{st}}[i]$
$(\mathsf{st}, msg_{out}) \leftarrow \mathsf{KeyGen}.\mathsf{C}_{step}(\mathsf{st}, msg_{in})$
$L_{\mathsf{st}}[i] \leftarrow (step + 1, \mathsf{st})$
if $\mathsf{accept} \in \mathsf{st}$:
  $\mathsf{id}_i \leftarrow \mathsf{st.id}, \mathsf{sk}_i \leftarrow \mathsf{st.sk}$
return msg_{out}

Oracle $\mathcal{O}^{\mathsf{Sign}}(i, m)$

if $i \notin \{0,1\}$: **return** $\perp$
if $CSigns_{i \oplus b} \geq k$: **return** $\perp$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{i \oplus b}, m)$
$CSigns_{i \oplus b} \leftarrow CSigns_{i \oplus b} + 1$
return σ

Figure 13: k-anonymity Experiment $\mathbf{Exp}^{\text{kAnon}}_{\Pi}$

Key Generation Protocol: The key generation protocol is defined by a tuple of PPT algorithms $(\mathsf{GM}_1, \mathsf{C}_1, \mathsf{GM}_2, \mathsf{C}_2, \ldots, \mathsf{GM}_i, \mathsf{C}_i)$. We consider the first message in the protocol to be sent by the group manager. GM_1 takes as input the group manager's protocol input and outputs a message to be sent to the client along with the group manager's state. C_1 takes the client's protocol input and the first message, and outputs a response message together with the client's state. Each subsequent algorithm takes as input a party's state and the received message, and outputs an updated state and a new message. The final algorithms output, within the state, the values designated for each party at the end of the protocol, and additionally include the keyword accept. If any algorithm fails, it outputs the symbol $\perp$ for each of its outputs.

Correctness: *For every $\lambda \in \mathbb{N}$, every $k = k(\lambda)$, and every $msg \in \mathcal{M}$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that*

$$\Pr\left[\mathsf{Verify}(\mathsf{mpk}, msg, \sigma) \neq 1 : \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, k) \\ \langle(\mathsf{id}, \mathsf{sk}), \mathsf{id}\rangle \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where probability is taken over the random coin tosses of Gen, KeyGen *and* Sign.

Security. We now regard the security notions of a k-AGS scheme. We first define the notion of *k-anonymity* which says that no adversary can deanonymize a user who has signed less than $k+1$ messages.

k-Anonymity: *For a PPT adversary $\mathcal{A}$, let its advantage against k-anonymity of a k-AGS scheme Π be defined as follows:*

$$\mathsf{Adv}^{\text{kAnon}}_{\Pi}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\text{kAnon}}_{\Pi}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{kAnon}}_{\Pi}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1],$$

where the k-anonymity experiment $\mathbf{Exp}^{\text{kAnon}}_{\Pi}$ is defined in Figure 13.

The k-AGS scheme Π *is k-anonymous if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, the following holds:*

$$\mathsf{Adv}_{\Pi}^{\mathrm{kAnon}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

The idea behind the k-anonymity experiment is to consider two users. An adversary $\mathcal{A}$ can collude with the GM and all the users except the target users. $\mathcal{A}$ determines the threshold k, along with master public key mpk and state st. When $\mathcal{A}$ queries the Sign oracle with a message msg, the oracle signs msg on behalf of one of the target users, but does not disclose its choice. For a target user, $\mathcal{A}$ is only allowed to query the oracle within k times. Finally, $\mathcal{A}$ wins if she could make the correct guess.

Next, We first define *traceability* which ensures that any user who signs more than k messages that are accepted by the Verify algorithm, can be traced by the master opening key holder.

Traceability: *For a PPT adversary* $\mathcal{A}$*, let its advantage against traceability of a k-AGS scheme* Π *be defined as follows:*

$$\mathsf{Adv}_{\Pi}^{\mathrm{Trace}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}_{\Pi}^{\mathrm{Trace}}(1^{\lambda}, \mathcal{A}) \to 1],$$

where the traceability experiment $\mathbf{Exp}_{\mathrm{Trace}}^{\Pi}$ *is defined in Figure 14.*

A k-AGS scheme Π *is traceable if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, the following holds:*

$$\mathsf{Adv}_{\Pi}^{\mathrm{Trace}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

We additionally consider a notion capturing *unlinkability against parties other than the group manager*, which we call *non-GM unlinkability*. It models the natural setting in which an outsider — or a coalition of malicious users within the system — attempts to decide whether two signatures were produced by the same user, while the group manager remains honest.

Non-GM Unlinkability: *For a PPT adversary* $\mathcal{A}$*, let its advantage against non-GM unlinkability of a k-AGS scheme* Π *be defined as follows:*

$$\mathsf{Adv}_{\Pi}^{\mathrm{nGMUnlink}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}_{\Pi}^{\mathrm{nGMUnlink}}(1^{\lambda}, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}_{\Pi}^{\mathrm{nGMUnlink}}(1^{\lambda}, \mathcal{A}) \to 1 \mid b = 1],$$

where the non-GM unlinkability experiment $\mathbf{Exp}_{\Pi}^{\mathrm{nGMUnlink}}$ *is defined in Figure 15.*

The k-AGS scheme Π *is non-GM unlinkable if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, the following holds:*

$$\mathsf{Adv}_{\Pi}^{\mathrm{nGMUnlink}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

In the non-GM unlinkability experiment, the master keys $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok})$ are generated honestly and only mpk is given to the adversary; in particular, $\mathcal{A}$ obtains neither the master secret key msk nor the master opening key mok. Two target users $\mathsf{id}_0, \mathsf{id}_1$ are registered honestly by the experiment, and their identities are revealed to $\mathcal{A}$. The adversary may interact with the honest group manager through the oracle $\mathcal{O}^{\mathsf{KeyGen}}$ to register arbitrary corrupted users as a client, and may additionally request the registration of further honest users via $\mathcal{O}^{\mathsf{HonestKG}}$. The signing oracle $\mathcal{O}^{\mathsf{Sign}}$ takes a slot index $i \in \{0, 1\}$ and a message m, and answers using $\mathsf{sk}_{i \cdot b}$: when $b = 0$, both slots are answered with sk_0, so all signatures originate from the *same* user; when $b = 1$, slot i is answered with sk_i, so the two slots originate from *two different* users. The adversary wins if it correctly guesses b, i.e., if it can tell whether the two slots are linked. Unlike in k-anonymity, no upper bound is placed on the number of signing queries; to rule out a trivial attack against a deterministic signature scheme, the experiment requires that no message be queried to $\mathcal{O}^{\mathsf{Sign}}$ more than once.

$\mathbf{Exp}^{\text{Trace}}_{\Pi[k,\mathcal{M},\mathcal{ID}]}(1^\lambda, \mathcal{A})$

$L_{\mathsf{id}}, L_{\mathsf{st}}, L_{\mathsf{sk}}, L_{hon} \leftarrow \emptyset$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, k)$
$(s, \{(msg_i, \sigma_i)\}_{i=1,\ldots,s\}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KeyGen},\ldots}}(1^\lambda, \mathsf{mpk}, \mathsf{mok})$
if $\exists(i \neq j)$: $msg_i = msg_j \vee msg_i \in L_{hon}$:
  return 0
for $i = 1, \ldots, s$:
  if $\mathsf{Verify}(\mathsf{mpk}, msg_i, \sigma_i) = 0$:
    return 0
// Check if the adversary abused the system
if $s \leqslant k \cdot |L_{\mathsf{id}}|$: **return** 0
// Trace identities
$\{\mathsf{id}_1, \ldots, \mathsf{id}_m\} \leftarrow \mathsf{Trace}(\mathsf{mok}, k, \{\sigma_1, \ldots, \sigma_s\})$
// If there is at least one id that belongs
// to the adversary, they lose
if $\{\mathsf{id}_1, \ldots, \mathsf{id}_m\} \cap L_{\mathsf{id}} \neq \emptyset$:
  return 0
// Else they succeeded in hiding their identity
return 1

Oracle $\mathcal{O}^{\mathsf{HonestKG}}()$

// Oracle for $\mathcal{A}$ to register honest users
$(\mathsf{id}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$
$\mathcal{L}_{\mathsf{sk}} \leftarrow (\mathsf{id}, \mathsf{sk})$
return id

Oracle $\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{sid}, msg_{in})$

// An interface for $\mathcal{A}$ to run
// the KeyGen protocol with GM
if $L_{\mathsf{st}}[\mathsf{sid}] = \emptyset$:
  $(step, \mathsf{st}) \leftarrow (1, (\mathsf{msk}, \mathsf{mpk}))$
else :
  $(step, \mathsf{st}) \leftarrow L_{\mathsf{st}}[\mathsf{sid}]$
$(\mathsf{st}, msg_{out}) \leftarrow \mathsf{KeyGen.GM}_{step}(\mathsf{st}, msg_{in})$
$L_{\mathsf{st}}[\mathsf{sid}] \leftarrow (step + 1, \mathsf{st})$
if accept $\in \mathsf{st}$:
  $L_{\mathsf{id}} \leftarrow L_{\mathsf{id}} \cup \mathsf{st.id}$
return msg_{out}

Oracle $\mathcal{O}^{\mathsf{Sign}}(\mathsf{id}, msg)$

// An oracle for $\mathcal{A}$ to get
// signatures from honest users
if $L_{\mathsf{sk}}[\mathsf{id}] = \emptyset$:
  return 0
$\mathsf{sk} \leftarrow L_{\mathsf{sk}}[\mathsf{id}]$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)$
$L_{hon} \leftarrow L_{hon} \cup msg$
return σ

Figure 14: Traceability Experiment $\mathbf{Exp}^{\text{Trace}}_{\Pi}$

$\mathbf{Exp}^{\text{nGMUnlink}}_{\Pi}(1^\lambda, \mathcal{A})$

$L_{\mathsf{id}}, L_{\mathsf{st}}, L_{\mathsf{sk}}, L_{msg} \leftarrow \emptyset$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, k)$
$(\mathsf{id}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$
$(\mathsf{id}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KeyGen}}, \mathcal{O}^{\mathsf{HonestKG}}, \mathcal{O}^{\mathsf{Sign}}}(1^\lambda, \mathsf{mpk}, \mathsf{id}_0, \mathsf{id}_1)$
return b'

Oracle $\mathcal{O}^{\mathsf{Sign}}(i, m)$

if $i \notin \{0,1\} \vee m \in L_{msg}$:
  return $\perp$
$L_{msg} \leftarrow L_{msg} \cup \{m\}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{i \cdot b}, m)$
return σ

Oracle $\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{sid}, msg_{in})$

if $L_{\mathsf{st}}[\mathsf{sid}] = \emptyset$:
  $(step, \mathsf{st}) \leftarrow (1, (\mathsf{msk}, \mathsf{mpk}))$
else :
  $(step, \mathsf{st}) \leftarrow L_{\mathsf{st}}[\mathsf{sid}]$
$(\mathsf{st}, msg_{out}) \leftarrow \mathsf{KeyGen.GM}_{step}(\mathsf{st}, msg_{in})$
$L_{\mathsf{st}}[\mathsf{sid}] \leftarrow (step + 1, \mathsf{st})$
if accept $\in \mathsf{st}$:
  $L_{\mathsf{id}} \leftarrow L_{\mathsf{id}} \cup \mathsf{st.id}$
return msg_{out}

Oracle $\mathcal{O}^{\mathsf{HonestKG}}()$

$(\mathsf{id}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$
$L_{\mathsf{sk}} \leftarrow L_{\mathsf{sk}} \cup (\mathsf{id}, \mathsf{sk})$
return id

Figure 15: Non-GM unlinkability Experiment $\mathbf{Exp}^{\text{nGMUnlink}}_{\Pi}$

Remark 2 Non-GM unlinkability is incomparable with k-anonymity. On the one hand, the adversary is strictly weaker, as it does not corrupt GM and therefore knows neither msk nor mok. On the other hand, the adversary is granted strictly more power on the signing side, as it may obtain an arbitrary number of signatures from each target user. We introduce this notion to capture the natural guarantee that, against any party other than the group manager, the signatures of a k-AGS user remain unlinkable even beyond the k-th signature.

3.2 k-AGS Construction

We now present our generic construction $\Pi[k, \mathcal{M}, \mathcal{ID}]$ of a k-AGS for an arbitrary message space $\mathcal{M}$ and an identity space $\mathcal{ID} = \mathbb{Z}_p$. We make use of the following primitives in a black-box manner:

- a polynomial commitment scheme PC for polynomials over $\mathbb{Z}_p$;
- a two-round blind signature scheme BS;
- a public-key encryption scheme PKE whose message space contains pairs (y, pid), where $y \in \mathbb{Z}_p$ and pid is a pseudo-identity;
- a signature scheme Sig with a field mapping M2F from the signature space to $\mathbb{Z}_p$;
- a subversion-resistant non-interactive zero-knowledge argument system $\mathsf{NIZK}_{\mathsf{KG}}$ for a relation generator $\mathsf{RelGen}(1^\lambda)$, where the generator outputs relations $\mathcal{R}^{\mathsf{KG}}_{k,\mathsf{ck},\mathsf{pk}_{\mathsf{BS}}}$ corresponding to languages of the form[5]:

$$\mathcal{L} = \left\{ (msg, \mathsf{id}) \;\middle|\; \exists\, f \in \mathbb{Z}_p[x],\, \mathsf{sk}_{\mathsf{Sig}},\, \mathsf{pk}_{\mathsf{Sig}},\, R : \begin{array}{c} \mathcal{R}_{\mathsf{Sig}}(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) = 1, \\ f(0) = \mathsf{id},\;\; com = \mathsf{PC.Commit}(\mathsf{ck}, f), \\ msg = \mathsf{BS.U}_1(\mathsf{pk}_{\mathsf{BS}}, (com, \mathsf{pk}_{\mathsf{Sig}}); R) \end{array} \right\}.$$

[5] $\mathsf{RelGen}(1^\lambda)$ runs $\mathsf{PC.Setup}(1^\lambda, k)$ and $\mathsf{BS.KeyGen}(1^\lambda)$ to generate such a relation.

- a subversion-resistant non-interactive zero-knowledge argument system $\mathsf{NIZK}_{\mathsf{S}}$ for a relation generator $\mathsf{RelGen}_{\mathsf{S}}(1^\lambda)$, where the generator outputs relations $\mathcal{R}^{\mathsf{S}}_{k,\mathsf{ck},\mathsf{pk}_{\mathsf{PKE}},\mathsf{pk}_{\mathsf{BS}}}$ corresponding to languages of the form[6]:

$$\mathcal{L} = \left\{ (msg, \sigma_{\mathsf{Sig}}, ct) \;\middle|\; \exists\, c,\, y \in \mathbb{Z}_p,\, \mathsf{pid},\, \omega_{\mathsf{PC}},\, R : \begin{array}{r} \mathsf{pid} = ((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}}), \\ ct = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, (y, \mathsf{pid}); R), \\ \mathsf{PC.Verify}(\mathsf{ck}, com, \omega_{\mathsf{PC}}, \mathsf{M2F}(\sigma_{\mathsf{Sig}}), y) = 1, \\ \mathsf{BS.Verify}(\mathsf{pk}_{\mathsf{BS}}, \sigma_{\mathsf{BS}}, (com, \mathsf{pk}_{\mathsf{Sig}})) = 1, \\ \mathsf{Sig.Verify}(\mathsf{pk}_{\mathsf{Sig}}, \sigma_{\mathsf{Sig}}, c \| msg) = 1 \end{array} \right\}.$$

The algorithms and the key generation protocol of Π are defined in Figure 16 and Figure 17 correspondingly. In the Trace algorithm, $\mathsf{Interpolate}$ denotes any efficient polynomial interpolation procedure that, given $d+1$ points $(x_i, y_i) \in \mathbb{Z}_p^2$, outputs a polynomial $f \in \mathbb{Z}_p[x]$ of degree at most d.

In contrast to a textbook group signature, the pseudo-identity pid is *not* transmitted in the clear. Instead, it is encrypted—together with the share's y-coordinate—towards the group manager under $\mathsf{pk}_{\mathsf{PKE}}$, and it appears only as a witness of the $\mathsf{NIZK}_{\mathsf{S}}$ proof. The proof attests that the ciphertext is well-formed and that the encrypted pseudo-identity carries a valid blind signature on $(com, \mathsf{pk}_{\mathsf{Sig}})$ and is consistent with the conventional signature σ_{Sig} and the committed share. As a result, a signature reveals nothing about the signer to any party that does not hold the master opening key mok, while the holder of mok can still recover pid and trace as before.

The key generation protocol KeyGen is specified by four algorithms: $\{\mathsf{GM}_1, \mathsf{C}_1, \mathsf{GM}_2, \mathsf{C}_2\}$. For the sake of compactness, we omit in the pseudo code the details of how the output states of GM_1 and C_1 are constructed and later parsed by GM_2 and C_2. We assume that the states contain intermediate values required for subsequent computations. The output messages of GM_1, C_1, and GM_2 are shown above the arrows.

The correctness of Π follows straightforwardly from the correctness of the underlying primitives.

Security. The theorems below state the security properties of our k-AGS scheme, namely traceability, k-anonymity, and non-GM unlinkability. We provide brief proof sketches in the main body—more precisely, intuitions—for each theorem, while the full proofs are deferred to Appendix A.

Theorem 1 *Let $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a k-AGS scheme as defined before. Assume the following building blocks: $\mathsf{NIZK}_{\mathsf{KG}}$ is a knowledge-sound NIZK scheme, $\mathsf{NIZK}_{\mathsf{S}}$ is simulation-extractable, BS is a one-more unforgeable blind signature scheme, PKE is a correct public-key encryption scheme, Sig is a* sUFCMA *secure signature scheme with mapping to a field, resistant to collisions, and PC is a binding and hiding polynomial commitment scheme. Then $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is a traceable k-AGS scheme.*

Proof. (Sketch) As usual, we prove traceability via a sequence-of-games argument. In the initial games, we use the knowledge-soundness of $\mathsf{NIZK}_{\mathsf{KG}}$ and the one-more unforgeability of BS to show that all pseudo-identities in the adversary's output signatures were correctly formed with the participation of the group manager. That is, each pid is a commitment to a polynomial f of degree k, with $f(0) = \mathsf{id}$ and a Sig public key, blindly signed by GM.

Next, by relying on the simulation-extractability of $\mathsf{NIZK}_{\mathsf{S}}$—that is, knowledge soundness in the presence of simulated proofs (in the full proof, we must replace real NIZK proofs with simulated ones in signatures produced by honest users from the outset)—along with the correctness of PKE and the binding property of PC, we show that the PKE ciphertext in each signature decrypts to the y-coordinate of a share corresponding to the polynomial f committed in the associated $\mathsf{pid}.com$. From this, it follows that any $k+1$ signatures with the same pid will reveal the polynomial f, such that $f(0)$ equals some id previously registered with GM.

It remains to show that the adversary can only submit signatures from users it controls. In other words, $\mathcal{A}$ cannot forge signatures of honest users. This follows directly from the sUFCMA security of Sig.

Therefore, the adversary can only produce correctly formed signatures for identities it has registered with GM. Consequently, at least one identity must be used more than k times, which allows GM to trace it. □

[6] $\mathsf{RelGen}_{\mathsf{S}}(1^\lambda)$ runs $\mathsf{PC.Setup}(1^\lambda, k)$, $\mathsf{PKE.KeyGen}(1^\lambda)$ and $\mathsf{BS.KeyGen}(1^\lambda)$ to generate such a relation.

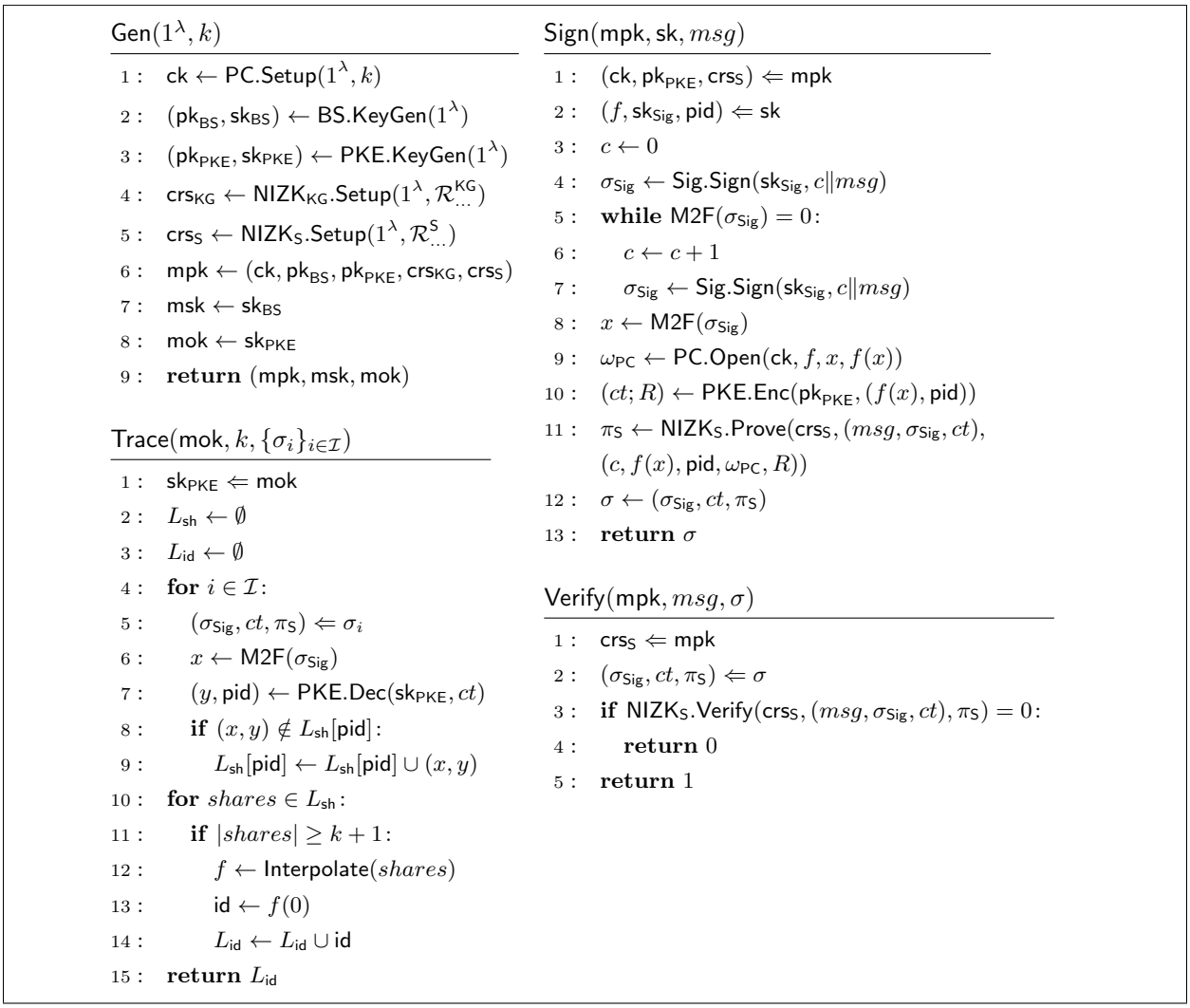

Figure 16: k-AGS construction $\Pi[k, \mathcal{M}, \mathcal{ID}]$

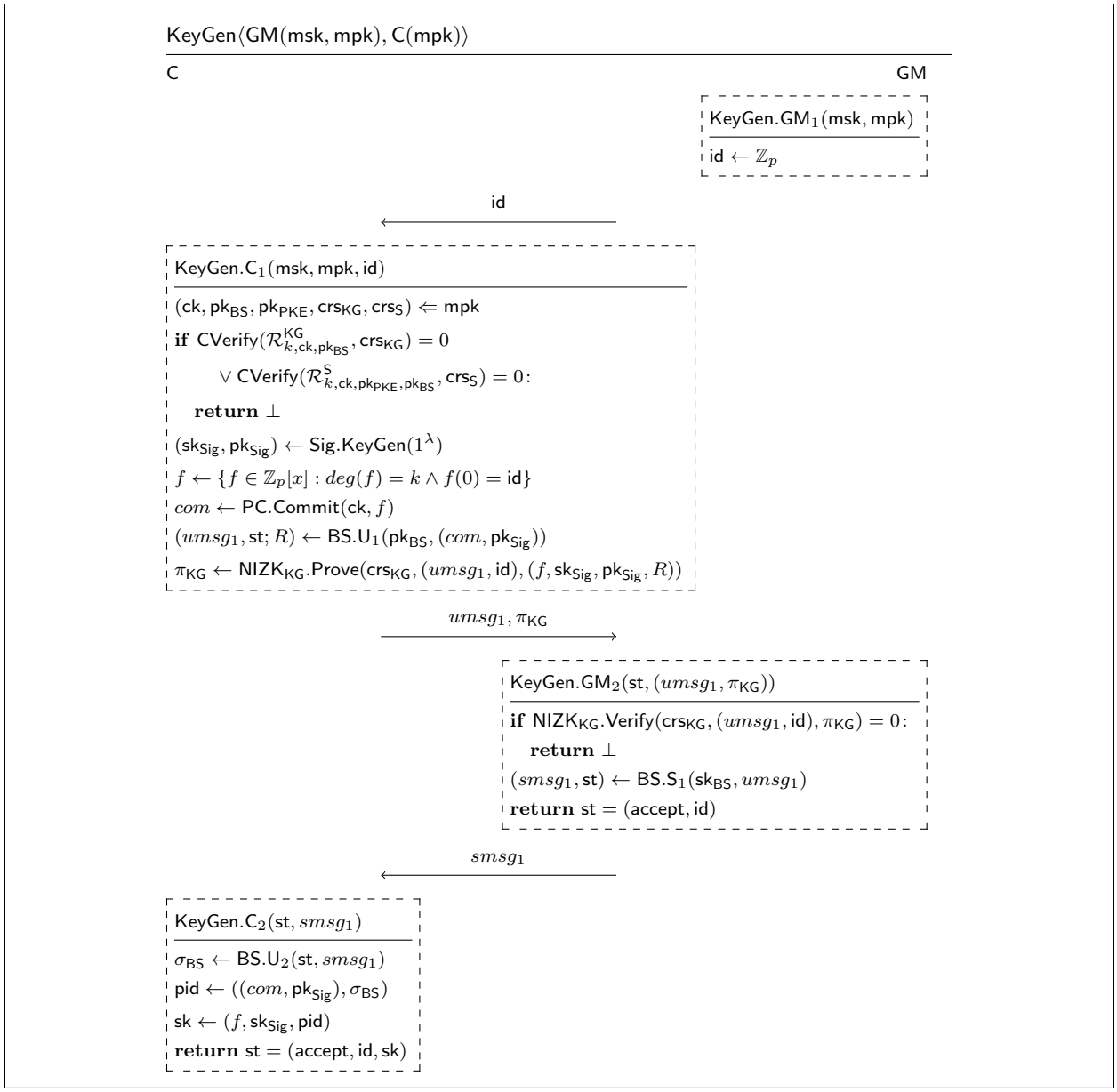

Figure 17: KeyGen protocol of k-AGS $\Pi[k, \mathcal{M}, \mathcal{ID}]$

Theorem 2 *Let $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a k-AGS scheme as defined before. Assume the following building blocks:* $\mathsf{NIZK}_{\mathsf{KG}}$ *is a subversion-zero-knowledge NIZK scheme,* PC *is a hiding polynomial commitment scheme, and* BS *is a secure blind signature scheme. Then $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is a k-anonymous k-AGS scheme.*

Proof. (Sketch) Since the k-Anonymity experiment is defined with the group manager as the adversary, we immediately consider a modification of Π: instead of including an encryption of the y-coordinate of a share in a signature, the signature directly contains the y-coordinate, as well as the polynomial commitment ω_{PC} in place of π_{S}. In other words, for k-Anonymity we do not rely on any properties of PKE or $\mathsf{NIZK}_{\mathsf{S}}$.

The proof is straightforward. First, we use the subversion-zero-knowledge of $\mathsf{NIZK}_{\mathsf{KG}}$ and the hiding property of PC to replace identities sent by the adversary in the KeyGen protocol with fresh, independently chosen ones. This makes the shares and commitments contained in signatures independent of the identities chosen by $\mathcal{A}$.

The only remaining way for the adversary to link signatures and registration sessions is through the blind signature σ_{BS} over commitments contained in pseudo-identities. However, $\mathcal{A}$ cannot achieve this due to the blindness property of BS. □

Theorem 3 *Let $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a k-AGS scheme as defined before. Assume the following building blocks:* $\mathsf{NIZK}_{\mathsf{S}}$ *is a zero-knowledge NIZK scheme,* PKE *is an* INDCPA*-secure public-key encryption scheme, and* Sig *is a key-anonymous (*kAnonSig*) signature scheme. Then $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is non-GM unlinkable.*

Proof. (Sketch) The non-GM unlinkability experiment denies the adversary both msk and mok, and the two target users are registered honestly, so the only signer-bearing information in $\mathcal{A}$'s view lives in the three components of each $\mathcal{O}^{\mathsf{Sign}}$ response: the conventional signature σ_{Sig}, the PKE ciphertext ct, and the $\mathsf{NIZK}_{\mathsf{S}}$ proof π_{S}. We bridge from $b = 0$ to $b = 1$ in three steps.

First, by the zero-knowledge property of $\mathsf{NIZK}_{\mathsf{S}}$, every $\mathsf{NIZK}_{\mathsf{S}}$ proof produced by $\mathcal{O}^{\mathsf{Sign}}$ is replaced by a simulated one; the proof no longer depends on its witness $(c, f(x), \mathsf{pid}, \omega_{\mathsf{PC}}, R)$. Second, by the INDCPA security of PKE, every PKE ciphertext is replaced by an encryption of a fixed dummy plaintext; the encrypted credentials (y, pid) no longer appear in $\mathcal{A}$'s view at all.

With π_{S} and ct neutralised, the only remaining user-dependent component is σ_{Sig}. Switching the signing key embedded in σ_{Sig} for slot-1 queries from $\mathsf{sk}_{\mathsf{Sig},0}$ to $\mathsf{sk}_{\mathsf{Sig},1}$ is then a single hop reducing to the key anonymity (kAnonSig) of Sig: since the non-GM unlinkability experiment forbids repeated messages to $\mathcal{O}^{\mathsf{Sign}}$, the reduction never queries the same Sig message to both of its kAnonSig oracles and the restriction of the kAnonSig experiment is respected.

Reverting the two preparatory hops in the opposite direction yields the $b = 1$ experiment. □

Discussion. We choose subversion-resistant NIZKs [BFS16, ALSZ21] as building blocks because they allow us to delegate the entire burden of generating the common reference string to the group manager. This contrasts with, for example, using the common random string model and making the user verifiably generate a pseudo-random crs for each signature. In our setting, the group manager performs this step only once during master key generation.

This choice benefits us in two ways: it simplifies both the presentation of the construction and the security proofs, and it allows us to hard-code crs after setting up the group manager. We note that if the crs setup is performed in a trusted manner, the subversion-resistance requirement can be relaxed.

From the KeyGen protocol, we also identify another interesting—albeit slightly tangential—open question: designing an efficient NIZK-blind signature pair that allows the prover to demonstrate to the server that the message being signed satisfies a fixed NP relation. For instantiating our construction, we consider pairing-based primitives along with NIZKs tailored for proving relations over pairings to be the most straightforward approach.

3.3 Instantiation

We sketch an efficient pairing-based instantiation of our generic k-AGS construction $\Pi[k, \mathcal{M}, \mathcal{ID}]$ (Section 3.2). All primitives operate over a single asymmetric (Type-3) bilinear group $\mathcal{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$,

and both relations $\mathcal{R}^{\mathsf{KG}}$ and $\mathcal{R}^{\mathsf{S}}$ admit a structure-preserving formulation as systems of pairing-product (PPE) and multi-scalar-multiplication (MSM) equations, directly suitable for Groth–Sahai. We rely on SXDH, the k-SDH assumption (KZG binding), and EQS-EUF-CMA.

Building blocks: We instantiate the polynomial commitment PC with the hiding KZG scheme [KZG10]: $com = g_1^{f(\tau)}$, with the opening at x verified by the pairing-product equation $e(com \cdot Y^{-1}, g_2) = e(\omega, g_2^{\tau}/g_2^{x})$ in *group form*, i.e. with the lifted share $Y = g_1^{y}$ in place of the scalar y; hiding is for free since f is sampled uniformly subject to $f(0) = \mathsf{id}$, and binding follows from k-SDH. The encryption scheme PKE is lifted ElGamal in $\mathbb{G}_1$, applied component-wise to encrypt tuples of group elements: a vector $(W_1, \ldots, W_t) \in \mathbb{G}_1^t$ is encrypted as $(g_1^{R_j}, W_j \cdot \mathsf{pk}_{\mathsf{PKE}}^{R_j})_{j=1}^{t}$ with independent per-component randomness $R_1, \ldots, R_t$, which is IND-CPA secure over $\mathbb{G}_1^t$ under SXDH by a standard hybrid argument. The signature scheme Sig is Schnorr over $\mathbb{G}_1$ with field map $\mathsf{M2F}((R_{\mathsf{Sig}}, s)) = s$; its sUFCMA security reduces to discrete log in $\mathbb{G}_1$ in the random-oracle model, MFcr follows from collision resistance of the random oracle $\mathsf{H2F}$, and it is key-anonymous (kAnonSig) since a Schnorr signature (R_{Sig}, s) is distributed uniformly over $\mathbb{G}_1 \times \mathbb{Z}_p$ independently of $\mathsf{pk}_{\mathsf{Sig}}$. The blind signature BS is the round-optimal FHS scheme [FHS15] on the pair $(com, \mathsf{pk}_{\mathsf{Sig}}) \in \mathbb{G}_1^2$, with blindness from perfect adaptation of structure-preserving signatures on equivalence classes (SPS-EQ) [HS14] and one-more unforgeability from EQS-EUF-CMA. Finally, $\mathsf{NIZK}_{\mathsf{KG}}$ and $\mathsf{NIZK}_{\mathsf{S}}$ are instantiated with subversion-resistant Groth–Sahai proofs [GS08, ALSZ21], with $\mathsf{NIZK}_{\mathsf{S}}$ additionally made simulation-extractable by a standard transform; the user-side $\mathsf{CVerify}$ checks let the group manager generate $\mathsf{crs}_{\mathsf{KG}}, \mathsf{crs}_{\mathsf{S}}$ unilaterally.

Lifted shares: For efficiency we work in *lifted* form: the share $y = f(x)$ is carried as $Y = g_1^{y}$, the KZG opening is verified in group form as above, and Trace recovers $g_1^{f(0)}$ by Lagrange interpolation *in the exponent*, matching it against a public table $\{\mathsf{id} \mapsto g_1^{\mathsf{id}}\}$ that the group manager populates during KeyGen. This is equivalent to the abstract scheme up to a change of PKE message space from (y, pid) with $y \in \mathbb{Z}_p$ to a vector $(Y, \mathsf{pid}) \in \mathbb{G}_1^t$ of $t = 1 + |\mathsf{pid}|$ group elements.

Hiding the pseudo-identity: Following the abstract construction, the pseudo-identity $\mathsf{pid} = ((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}})$ is not transmitted in the clear: a signature is $\sigma = (\sigma_{\mathsf{Sig}}, c, ct, \pi_{\mathsf{S}})$, where ct lifted-ElGamal-encrypts the tuple (Y, pid) component-wise (all components being group elements) and π_{S} is the $\mathsf{NIZK}_{\mathsf{S}}$ proof. On statement $(msg, c, \sigma_{\mathsf{Sig}}, ct)$ and witnesses $(Y, \mathsf{pid}, \omega, R)$, the relation $\mathcal{R}^{\mathsf{S}}$ encodes: (i) ElGamal correctness of ct with respect to the committed group elements $Y, com, \mathsf{pk}_{\mathsf{Sig}}, \sigma_{\mathsf{BS}}$ (MSM equations); (ii) the KZG opening of com at $(\mathsf{M2F}(\sigma_{\mathsf{Sig}}), Y)$ (one PPE, group form); (iii) FHS/SPS-EQ verification of σ_{BS} on $(com, \mathsf{pk}_{\mathsf{Sig}})$ (PPEs); and (iv) Schnorr verification of $\sigma_{\mathsf{Sig}} = (R_{\mathsf{Sig}}, s)$ under the committed $\mathsf{pk}_{\mathsf{Sig}}$. For (iv) we keep the counter c in the clear so that $\mathsf{H2F}(R_{\mathsf{Sig}} \| c \| msg)$ is a public coefficient and the Schnorr equation $g_1^{s} = R_{\mathsf{Sig}} \cdot \mathsf{pk}_{\mathsf{Sig}}^{\mathsf{H2F}(R_{\mathsf{Sig}} \| c \| msg)}$ is a linear MSM equation whose only witness is $\mathsf{pk}_{\mathsf{Sig}}$. All equations carry 1 on the right-hand side, so the GS proof is zero-knowledge in one shot and of size $O(1)$.

Sizes: The pseudo-identity is $O(1)$ group elements in $\mathbb{G}_1$, and a k-AGS signature $\sigma = (\sigma_{\mathsf{Sig}}, c, ct, \pi_{\mathsf{S}})$ is $O(1)$ group elements and scalars, both independent of k. The master public key mpk and the proof π_{KG} are $O(k)$, dominated by the KZG SRS and the k committed coefficients of f. Verification costs a constant number of pairings. The choice of Sig is largely orthogonal: any key-anonymous signature scheme equipped with an efficiently computable, collision-resistant field map $\mathsf{M2F}$, e.g. BLS with $\mathsf{M2F}$ a hash from $\mathbb{G}_2$ to $\mathbb{Z}_p$, is a drop-in replacement.

4 k-Unlinkable Group Signatures

k-unlinkable group signatures strengthen the anonymity guarantee of k-AGS to hold even against the group manager itself: as long as a user has produced no more than k signatures, the group manager should be unable to link any two of them as originating from the same user. While the k-anonymity experiment of

$\mathbf{Exp}^{\mathrm{kUnlink}}_{\Pi}(1^\lambda, \mathcal{A})$

$\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{id}_0, \mathsf{id}_1 \leftarrow \bot$
$L_{\mathsf{st}} \leftarrow \emptyset$
$(\mathsf{mpk}, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{st}, l, \{m_1, \ldots, m_l\}, \mathcal{I}_0, \mathcal{I}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KeyGen}}}(\mathsf{st})$
if $|\mathcal{I}_0| > k \vee |\mathcal{I}_1| > k \vee$
  $|\mathcal{I}_0| + |\mathcal{I}_1| \neq l$:
  return 0
$b \xleftarrow{\$} \{0,1\}$
for $i = 1, \ldots, l$:
  if $i \in \mathcal{I}_b$:
    $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_0, m_i)$
  else :
    $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_1, m_i)$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \{\sigma_1, \ldots, \sigma_l\})$
return $b = b'$

Oracle $\mathcal{O}^{\mathsf{KeyGen}}(i, msg_{in})$

// An interface for $\mathcal{A}$ acting as GM
// to run the KeyGen protocol
if $i \notin \{0,1\}$: **return** $\bot$
if $L_{\mathsf{st}}[i] = \emptyset$:
  $(step, \mathsf{st}) \leftarrow (1, \mathsf{mpk})$
else :
  $(step, \mathsf{st}) \leftarrow L_{\mathsf{st}}[i]$
$(\mathsf{st}, msg_{out}) \leftarrow \mathsf{KeyGen}.\mathsf{C}_{step}(\mathsf{st}, msg_{in})$
$L_{\mathsf{st}}[i] \leftarrow (step + 1, \mathsf{st})$
if accept $\in \mathsf{st}$:
  $\mathsf{id}_i \leftarrow \mathsf{st}.\mathsf{id}$
  $\mathsf{sk}_i \leftarrow \mathsf{st}.\mathsf{sk}$
return msg_{out}

Figure 18: k-unlinkability experiment $\mathbf{Exp}^{\mathrm{kUnlink}}_{\Pi}$.

k-AGS challenges the group manager to distinguish between two slot-indexed signing queries, k-unlinkability presents the group manager with a batch of l signatures produced jointly by two target users and asks it to identify which subset originated from which user, subject to a budget of at most k signatures per user. This rules out, in particular, the use of any persistent pseudo-identity pid that the group manager could read off the signature; in k-AGS the pseudonym is recoverable by the group manager via mok, which is precisely what k-unlinkability denies.

k-Unlinkability: *For a PPT adversary $\mathcal{A}$, let its advantage against k-unlinkability of a k-AGS scheme Π be defined as follows:*

$$\mathsf{Adv}^{\mathrm{kUnlink}}_{\Pi}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\mathrm{kUnlink}}_{\Pi}(1^\lambda, \mathcal{A}) \to 1] - \tfrac{1}{2},$$

where the k-unlinkability experiment $\mathbf{Exp}^{\mathrm{kUnlink}}_{\Pi}$ is defined in Figure 18. The k-AGS scheme Π is k-unlinkable if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every $\lambda \in \mathbb{N}$ and every PPT adversary $\mathcal{A}$, the following holds:

$$\mathsf{Adv}^{\mathrm{kUnlink}}_{\Pi}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

4.1 k-UGS Construction

In this section, we are presenting a (theoretical) generic construction of a k-UGS scheme for an arbitrary message space $\mathcal{M}$ and an identity space $\mathcal{ID} = \mathbb{Z}_p$. We base our construction on a recent fully-anonymous secret share scheme of [BGI+25], which here we are using in a block-box way. All in all, we make use of the following primitives in a black-box manner:

- a two-round blind signature scheme BS;
- a public-key encryption scheme PKE for the message space $\mathbb{Z}_p$;
- a multi-dealer fully anonymous secret sharing scheme M-FASS;
- a signature scheme Sig with an injective mapping M2F from the signature space to $\mathbb{Z}_p$;

- a subversion-resistant non-interactive zero-knowledge argument system $\mathsf{NIZK}_{\mathsf{KG}}$ for a relation generator $\mathsf{RelGen}(1^\lambda)$, where the generator outputs relations $\mathcal{R}^{\mathsf{KG}}_{k,\mathsf{pp}_{\mathsf{sh}},\mathsf{pk}_{\mathsf{BS}}}$ corresponding to languages of the form:

$$\mathcal{L} = \left\{ (msg, \mathsf{id}) \;\middle|\; \exists \mathsf{sk}_{\mathsf{sh}}, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, R_1, R_2 : \begin{array}{c} \mathcal{R}_{\mathsf{Sig}}(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) = 1, \\ \mathsf{sk}_{\mathsf{sh}} = \mathsf{M\text{-}FASS.KeyGen}(\mathsf{pp}_{\mathsf{sh}}, \mathsf{id}; R_1) \\ msg = \mathsf{BS.U}_1(\mathsf{pk}_{\mathsf{BS}}, (\mathsf{sk}_{\mathsf{sh}}, \mathsf{pk}_{\mathsf{Sig}}); R_2) \end{array} \right\}.$$

- a subversion-resistant non-interactive zero-knowledge argument system $\mathsf{NIZK}_{\mathsf{S}}$ for a relation generator $\mathsf{RelGen}_{\mathsf{S}}(1^\lambda)$, where the generator outputs relations $\mathcal{R}^{\mathsf{S}}_{k,\mathsf{pp}_{\mathsf{sh}},\mathsf{pk}_{\mathsf{PKE}},\mathsf{pk}_{\mathsf{BS}}}$ corresponding to languages of the form:

$$\mathcal{L} = \left\{ (msg, \sigma_{\mathsf{Sig}}, ct) \;\middle|\; \exists \mathsf{sk}_{\mathsf{sh}}, \sigma_{\mathsf{BS}}, \mathsf{pk}_{\mathsf{Sig}}, R_{\mathsf{Enc}} : \begin{array}{c} ct = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{sh}; R_{\mathsf{Enc}}) \\ \mathsf{sh} = \mathsf{M\text{-}FASS.Share}(\mathsf{pp}_{\mathsf{sh}}, \mathsf{sk}_{\mathsf{sh}}; \mathsf{M2F}(\sigma_{\mathsf{Sig}})) \\ \mathsf{Sig.Verify}(\mathsf{pk}_{\mathsf{Sig}}, \sigma_{\mathsf{Sig}}, msg) = 1 \\ \mathsf{BS.Verify}(\mathsf{pk}_{\mathsf{BS}}, \sigma_{\mathsf{BS}}, (\mathsf{sk}_{\mathsf{sh}}, \mathsf{pk}_{\mathsf{Sig}})) = 1 \end{array} \right\}.$$

The algorithms and the key generation protocol of Π are defined in Figure 19 and Figure 20 correspondingly.

The key generation protocol KeyGen is specified by four algorithms: $\{\mathsf{GM}_1, \mathsf{C}_1, \mathsf{GM}_2, \mathsf{C}_2\}$. For the sake of compactness, we omit in the pseudo code the details of how the output states of GM_1 and C_1 are constructed and later parsed by GM_2 and C_2. We assume that the states contain intermediate values required for subsequent computations. The output messages of GM_1, C_1, and GM_2 are shown above the arrows.

The correctness of Π follows straightforwardly from the correctness of the underlying primitives.

Comparing Design with k-AGS. We design k-UGS to closely follow the overall structure of k-AGS. As in k-AGS, a conventional signature on the message msg serves as a stateless random seed for the underlying fully anonymous secret-sharing scheme—where in k-AGS this seed becomes the x-coordinate of the Shamir share, here it plays the analogous role inside the $\mathsf{M\text{-}FASS}$ instance. Likewise, the role of the secret-sharing polynomial f in k-AGS is played by a secret-sharing key $\mathsf{sk}_{\mathsf{sh}}$ in k-UGS. The pair $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{sh}})$ constitutes the user's credentials, blindly signed by GM during KeyGen, analogously to pid in k-AGS. The crucial difference is that, unlike in k-AGS, we do not encrypt the credentials toward GM: they are fully hidden inside the $\mathsf{NIZK}_{\mathsf{S}}$ witness. This is made possible by the robust reconstruction of $\mathsf{M\text{-}FASS}$, which enables GM to trace without relying on any per-signature linking information; in k-AGS, by contrast, the credentials revealed to GM through the ciphertext are essential for efficient linking and polynomial reconstruction.

Security. Traceability of k-UGS follows the same blueprint as the traceability proof of k-AGS (Theorem 1): knowledge soundness of $\mathsf{NIZK}_{\mathsf{KG}}$ and $\mathsf{NIZK}_{\mathsf{S}}$, together with one-more unforgeability of BS and sUFCMA of Sig, force every output signature to come from a credential that was honestly blind-signed during KeyGen, after which any user accumulating more than k such signatures is traced. The only structural differences are that the witness extracted from $\mathsf{NIZK}_{\mathsf{S}}$ here consists of the secret-sharing credentials $(\mathsf{sk}_{\mathsf{sh}}, \mathsf{pk}_{\mathsf{Sig}}, \sigma_{\mathsf{BS}})$ rather than a polynomial-commitment opening, and that the role of polynomial interpolation in Trace is played by the robust reconstruction of $\mathsf{M\text{-}FASS}$, which recovers id once the shares of a single dealer exceed the reconstruction threshold. We therefore omit the formal traceability statement and focus from here on k-unlinkability, whose proof closely resembles the k-anonymity proof of k-AGS.

Theorem 4 *Let $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be the k-UGS scheme defined above. Assume the following building blocks:*

- $\mathsf{NIZK}_{\mathsf{KG}}$ *and* $\mathsf{NIZK}_{\mathsf{S}}$ *are subversion-zero-knowledge NIZK schemes,*
- $\mathsf{M\text{-}FASS}$ *is a fully anonymous multi-dealer secret-sharing scheme,*
- BS *is a secure blind signature scheme.*

Then $\Pi[k, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is k-unlinkable.

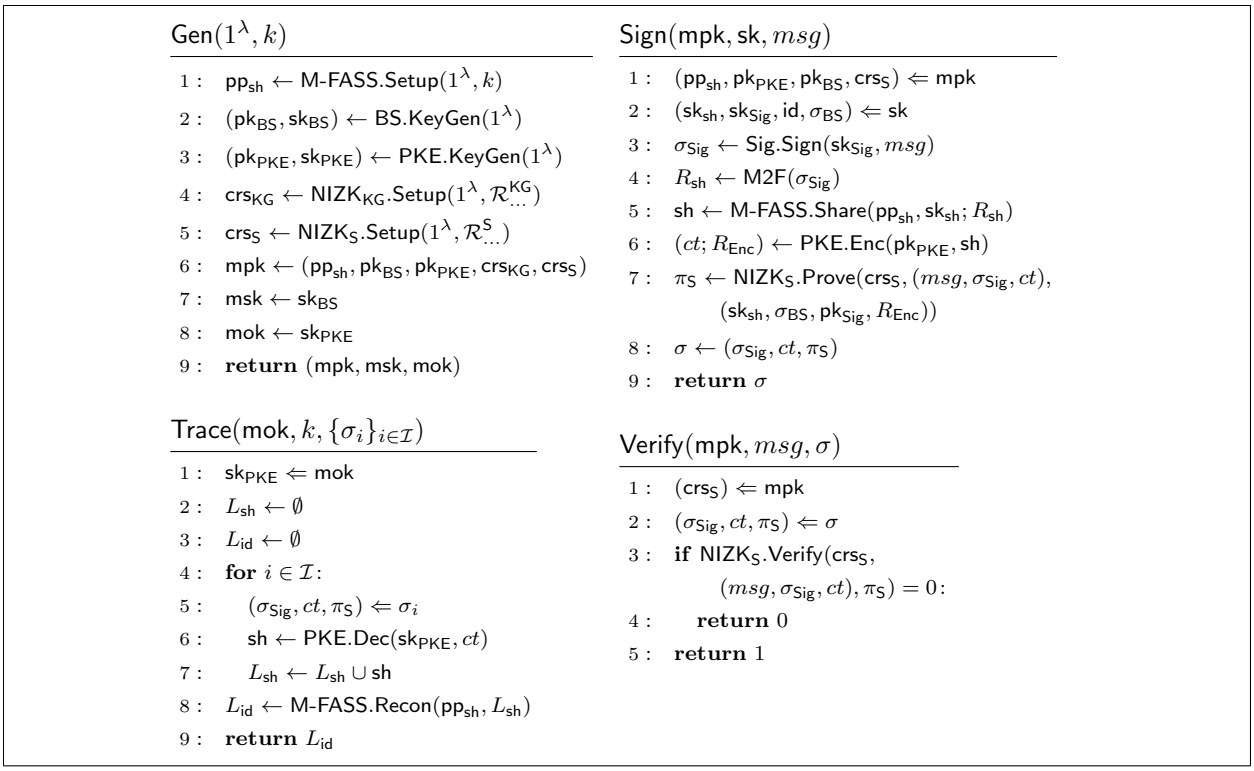

Figure 19: k-UGS construction $\Pi[k, \mathcal{M}, \mathcal{ID}]$

Proof. (Sketch) The proof mirrors the structure of k-anonymity for k-AGS (Theorem 2), with one essential addition. Because the signing credentials $\mathsf{sk_{sh}}$, $\mathsf{pk_{Sig}}$, and the blind signature σ_{BS} over them live entirely inside the $\mathsf{NIZK_S}$ witness rather than being recoverable by the group manager through mok, the proof additionally simulates the $\mathsf{NIZK_S}$ proofs produced by $\mathcal{O}^{\mathsf{Sign}}$ via the subversion-zero-knowledge of $\mathsf{NIZK_S}$, removing the credentials from the adversary's view of every signature.

With this preparatory hop in place, the proof proceeds as in the k-AGS case: the subversion-zero-knowledge of $\mathsf{NIZK_{KG}}$ is used to simulate the KeyGen proofs, after which the blindness of BS allows swapping the assignment of $(\mathsf{sk_{sh}}, \mathsf{pk_{Sig}})$ between the two target users, making their credentials i.i.d. and symmetric across the challenge bit b. The role played by the hiding property of PC in the k-AGS proof—hiding the secret-sharing polynomial f—is played here by the anonymity of M-FASS: with at most k shares per dealer in the adversary's view, the two streams of MFASS shares corresponding to the two target users are below the reconstruction threshold and hence computationally indistinguishable. Combined with the credential swap absorbed by the BS-blindness step, this completes the reduction. □

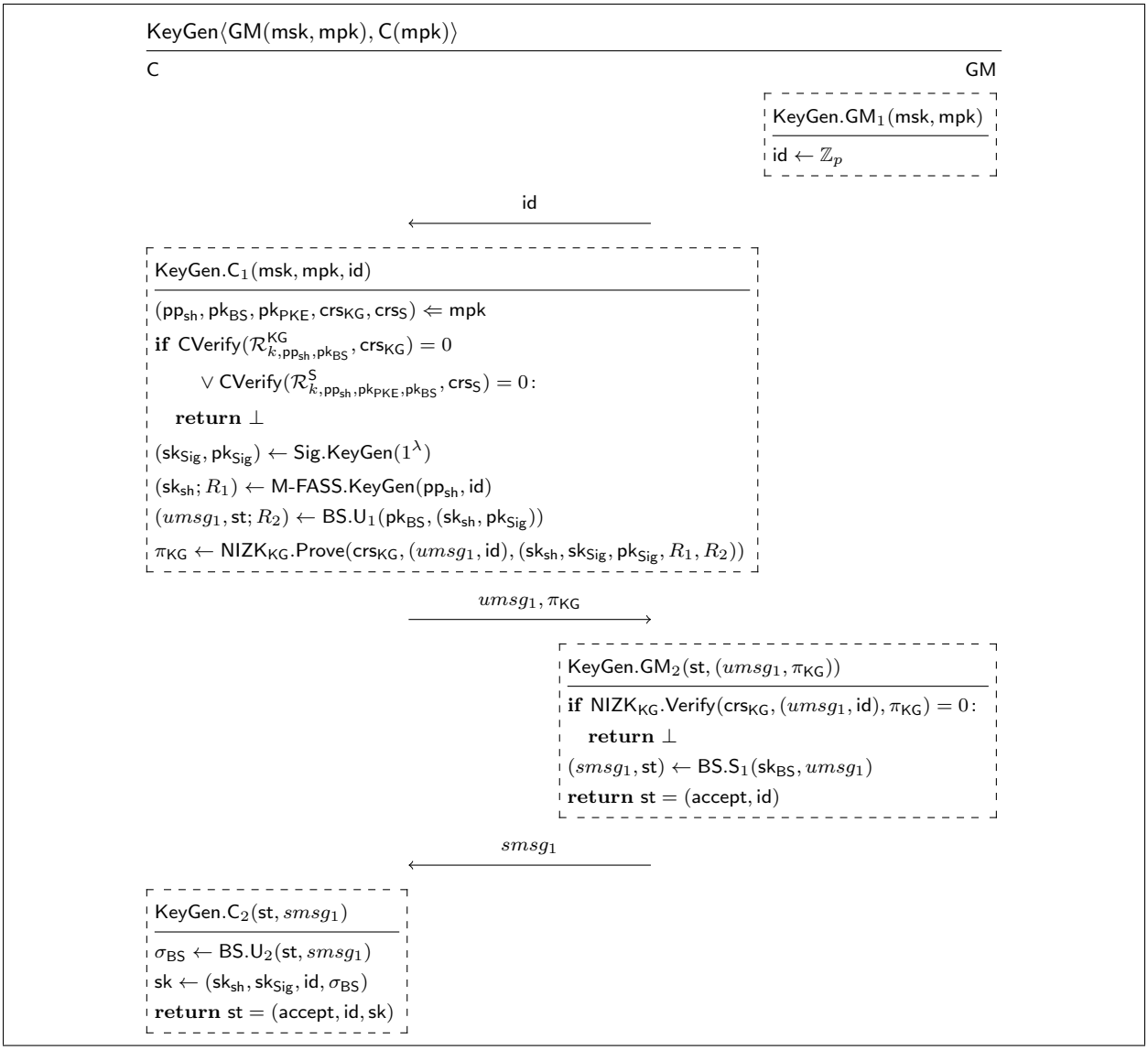

Figure 20: KeyGen protocol of k-UGS $\Pi[k, \mathcal{M}, \mathcal{ID}]$

5 k-Anonymous Set Pre-constrained Group Signatures

We now define k-ASPCGS along with its correctness and security notions.

Definition 9 *A k-Anonymous Set Pre-constrained group signature (k-ASPCGS) scheme $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$ with a message space $\mathcal{M}$, an identity space $\mathcal{ID}$, and a set size n, is a tuple* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{OpenShare}, \mathsf{Trace})$ *of PPT algorithms along with an interactive protocol* KeyGen *defined as follows:*

$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{D}, k)$: *The master key generation algorithm takes as input a security parameter 1^λ, a set $\mathcal{D} \subseteq \mathcal{M}$ of size at most n, and a threshold parameter k, and outputs a master public key* mpk*, a master secret key* msk*, and a master opening key* mok*.*

$\langle \mathsf{id}, (\mathsf{id}, \mathsf{sk}) \rangle \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C} \rangle$: *The interactive key generation protocol is run by a group manager* GM *with an input* msk*, and by a client* C*. It delivers an identity $\mathsf{id} \in \mathcal{ID}$ to both* GM *and* C*, and an identity signing key* sk *to* C*.*

$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)$: *The signing algorithm takes as input a master public key* mpk*, an identity signing key* sk*, and a message $msg \in \mathcal{M}$, and outputs a signature σ.*

$\{1, 0\} \leftarrow \mathsf{Verify}(\mathsf{mpk}, msg, \sigma)$: *The signature verification algorithm takes as input a master public key* mpk*, a message $msg \in \mathcal{M}$, and a signature σ, and outputs either 1 or 0, indicating accept or reject.*

$\{\mathsf{id}_1, \ldots, \mathsf{id}_m\} \leftarrow \mathsf{Trace}(\mathsf{mok}, k, \{\sigma_i\}_{i \in \mathcal{I}})$: *The identity reconstruction algorithm takes as input a master opening key* mok*, the scheme anonymity parameter k, a set of signatures $\{\sigma_i\}_{i \in \mathcal{I}}$, and outputs a potentially empty set of corresponding identities $\{\mathsf{id}_1, \ldots, \mathsf{id}_m\}, \mathsf{id}_i \in \mathcal{ID}$.*

Correctness: *There exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for every $\lambda \in \mathbb{N}$, for every $\mathcal{D} \subset \mathcal{M}$, and for every $m \in \mathcal{M}$, the following holds:*

$$\Pr\left[\mathsf{Verify}(\mathsf{mpk}, msg, \sigma) = 1 \;:\; \begin{array}{c}(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{D}, k) \\ \langle \mathsf{id}, (\mathsf{id}, \mathsf{sk}) \rangle \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}, k), \mathsf{C} \rangle \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)\end{array}\right] = 1$$

Security. We now regard the security notions of a k-ASPCGS scheme. While Non-GM Unlinkability remains without changes, traceability and k-anonymity are adapted for the set pre-constrained setting.

Traceability: *For a PPT adversary $\mathcal{A}$, let its advantage against traceability of a k-ASPCGS scheme Π be defined as follows:*

$$\mathsf{Adv}^{\mathrm{Trace}}_{\Pi,\mathcal{D}}(\mathcal{A}, \lambda) = \Pr[\mathbf{Exp}^{\mathrm{Trace}}_{\Pi,\mathcal{D}}(1^\lambda, \mathcal{A}) \to 1],$$

where the traceability experiment $\mathbf{Exp}^{\Pi}_{\mathrm{Trace}}$ for k-ASPCGS is defined in Figure 14 .

A k-ASPCGS scheme Π is traceable if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every $\lambda \in \mathbb{N}$, $\mathcal{D} \subseteq \mathcal{M}, |\mathcal{D}| \leq n$, and for every PPT adversary $\mathcal{A}$, the following holds:

$$\mathsf{Adv}^{\mathrm{Trace}}_{\Pi,\mathcal{D}}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}(\lambda).$$

The k-anonymity notion for k-ASPCGS is the natural variant of the game-based k-anonymity for k-AGS, adapted to the set-pre-constrained setting. As before, an adversary $\mathcal{A}$ corrupts the group manager, registers two target users $\mathsf{id}_0, \mathsf{id}_1$ through $\mathcal{O}^{\mathsf{KeyGen}}$, and then queries $\mathcal{O}^{\mathsf{Sign}}$ on slot indices $i \in \{0, 1\}$ and messages of its choice; a hidden bit b determines whether the two slots are swapped, so that slot i is answered using $\mathsf{sk}_{i \oplus b}$. Two differences from the k-AGS experiment reflect the set-pre-constrained nature of the scheme. First, the per-user budget of k signatures applies *only to messages in $\mathcal{D}$*: $\mathcal{A}$ may obtain unlimited signatures on messages outside $\mathcal{D}$ from either slot, but at most k signatures on $\mathcal{D}$-messages from each. This captures the design intent

$\mathbf{Exp}^{\text{Trace}}_{\Pi,\mathcal{D}}(1^\lambda, \mathcal{A})$

$L_{\mathsf{id}}, L_{\mathsf{st}}, L_{\mathsf{sk}}, L_{msg} \leftarrow \emptyset$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{D}, k)$
$Dmsg \leftarrow 0$
$(s, \{(msg_i, \sigma_i)\}_{i=1,\ldots,s}\}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KeyGen}}, \ldots}(1^\lambda, \mathsf{mpk}, \mathsf{mok})$
if $\exists (i,j)$: $msg_i = msg_j \vee msg_i \in L_{msg}$:
  return 0
for $i = 1, \ldots, s$:
  if $\mathsf{Verify}(\mathsf{mpk}, msg_i, \sigma_i) = 0$:
    return 0
  if $msg_i \in \mathcal{D}$: $Dmsg \leftarrow Dmsg + 1$
// Check if the adversary abused the system
if $Dmsg \leqslant k \cdot |L_{\mathsf{id}}|$:
  return 0
// Trace identities
$\{\mathsf{id}_1, \ldots, \mathsf{id}_m\} \leftarrow \mathsf{Trace}(\mathsf{mok}, k, \{\sigma_1, \ldots, \sigma_s\})$
// If there is at least one id that belongs
// to the adversary, they lose
if $\{\mathsf{id}_1, \ldots, \mathsf{id}_m\} \cap L_{\mathsf{id}} \neq \emptyset$:
  return 0
// Else they succeeded in hiding identity
return 1

Oracle $\mathcal{O}^{\mathsf{HonestKG}}()$

// Oracle for $\mathcal{A}$ to register honest users
$(\mathsf{id}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C} \rangle$
$\mathcal{L}_{\mathsf{sk}} \leftarrow (\mathsf{id}, \mathsf{sk})$
return id

Oracle $\mathcal{O}^{\mathsf{KeyGen}}(\mathsf{sid}, msg_{in})$

// An interface for $\mathcal{A}$ to run
// the KeyGen protocol with GM
if $L_{\mathsf{st}}[\mathsf{sid}] = \emptyset$:
  $(step, \mathsf{st}) \leftarrow (1, (\mathsf{msk}, \mathsf{mpk}))$
else :
  $(step, \mathsf{st}) \leftarrow L_{\mathsf{st}}[\mathsf{sid}]$
$(\mathsf{st}, msg_{out}) \leftarrow \mathsf{KeyGen.GM}_{step}(\mathsf{st}, msg_{in})$
$L_{\mathsf{st}}[\mathsf{sid}] \leftarrow (step + 1, \mathsf{st})$
if accept $\in \mathsf{st}$:
  $L_{\mathsf{id}} \leftarrow L_{\mathsf{id}} \cup \mathsf{st.id}$
return msg_{out}

Oracle $\mathcal{O}^{\mathsf{Sign}}(\mathsf{id}, msg)$

// An oracle for $\mathcal{A}$ to get
// signatures from honest users
$L_{msg} \leftarrow L_{msg} \cup msg$
if $\notin L_{\mathsf{sk}}[\mathsf{id}] = \emptyset$:
  return 0
$\mathsf{sk} \leftarrow L_{\mathsf{sk}}[\mathsf{id}]$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)$
return σ

Figure 21: Traceability experiment $\mathbf{Exp}^{\text{Trace}}_{\Pi}$ for k-ASPCGS.

of k-ASPCGS, in which an honest user is identifiable from $\mathcal{D}$-signatures after the threshold while signatures on unconstrained messages are not informative at all. Second, since the master public key produced by the corrupted group manager contains a public key for the underlying SPCE—and an inconsistent SPCE public key would trivially defeat anonymity, exactly as in the receiver IND-CPA notion of SPCE (Figure 11)—the experiment is parameterized by an efficiently computable *validator* $\mathsf{Auth}(\mathsf{mpk}, \mathcal{D}) \in \{0,1\}$ that gates the game on consistency of mpk with $\mathcal{D}$. As in subversion zero-knowledge or knowledge soundness, we let Auth depend on the adversary: the scheme is k-anonymous if for every PPT adversary there exists a PPT Auth for which the advantage is negligible.

k-Anonymity: *For a PPT adversary $\mathcal{A}$ and a PPT validator Auth, let the advantage of $\mathcal{A}$ against k-anonymity of a k-ASPCGS scheme Π be defined as follows:*

$$\mathsf{Adv}^{\text{kAnon}}_{\Pi,\mathcal{D}}(\mathcal{A}, \mathsf{Auth}, \lambda) = \Pr[\mathbf{Exp}^{\text{kAnon}}_{\Pi,\mathcal{D},\mathsf{Auth}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{kAnon}}_{\Pi,\mathcal{D},\mathsf{Auth}}(1^\lambda, \mathcal{A}) \to 1 \mid b = 1],$$

where the k-anonymity experiment $\mathbf{Exp}^{\text{kAnon}}_{\Pi,\mathcal{D},\mathsf{Auth}}$ is defined in Figure 22 .
A k-ASPCGS scheme Π is k-anonymous if for every PPT adversary $\mathcal{A}$ there exist a PPT validator Auth and a negligible function $\mathsf{negl}(\lambda)$ such that for every $\lambda \in \mathbb{N}$ and every $\mathcal{D} \subseteq \mathcal{M}$ with $|\mathcal{D}| \leq n$, the following

$\mathbf{Exp}^{\mathrm{kAnon}}_{\Pi,\mathcal{D},\mathsf{Auth}}(1^\lambda,\mathcal{A})$

1 : $\mathsf{sk}_0,\mathsf{sk}_1,\mathsf{id}_0,\mathsf{id}_1 \leftarrow \bot$
2 : $L_{\mathsf{st}} \leftarrow \emptyset$
3 : $CSigns_0, CSigns_1 \leftarrow 0$
4 : $(\mathsf{mpk},\mathsf{st}) \leftarrow \mathcal{A}(1^\lambda,k,\mathcal{D})$
5 : **if** $\mathsf{Auth}(\mathsf{mpk},\mathcal{D}) = 0$:
6 : **return** 0
7 : $\mathsf{st} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KeyGen}}}(\mathsf{st})$
8 : **if** $\mathsf{id}_0 = \bot \vee \mathsf{id}_1 = \bot$:
9 : **return** 0
10 : $b \xleftarrow{\$} \{0,1\}$
11 : $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{st})$
12 : **return** b'

Oracle $\mathcal{O}^{\mathsf{Sign}}(i,m)$

1 : **if** $i \notin \{0,1\}$: **return** $\bot$
2 : **if** $m \in \mathcal{D} \wedge CSigns_{i\oplus b} \geq k$:
3 : **return** $\bot$
4 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk},\mathsf{sk}_{i\oplus b},m)$
5 : **if** $m \in \mathcal{D}$:
6 : $CSigns_{i\oplus b} \leftarrow CSigns_{i\oplus b} + 1$
7 : **return** σ

Oracle $\mathcal{O}^{\mathsf{KeyGen}}(i,msg_{in})$

1 : // An interface for $\mathcal{A}$ acting as GM
2 : // to run the KeyGen protocol
3 : **if** $i \notin \{0,1\}$: **return** $\bot$
4 : **if** $L_{\mathsf{st}}[i] = \emptyset$:
5 : $(step,\mathsf{st}) \leftarrow (1,\mathsf{mpk})$
6 : **else** :
7 : $(step,\mathsf{st}) \leftarrow L_{\mathsf{st}}[i]$
8 : $(\mathsf{st},msg_{out}) \leftarrow \mathsf{KeyGen}.\mathsf{C}_{step}(\mathsf{st},msg_{in})$
9 : $L_{\mathsf{st}}[i] \leftarrow (step+1,\mathsf{st})$
10 : **if** $\mathsf{accept} \in \mathsf{st}$:
11 : $\mathsf{id}_i \leftarrow \mathsf{st.id}$
12 : $\mathsf{sk}_i \leftarrow \mathsf{st.sk}$
13 : **return** msg_{out}

Figure 22: k-anonymity experiment $\mathbf{Exp}^{\mathrm{kAnon}}_{\Pi,\mathcal{D},\mathsf{Auth}}$ for k-ASPCGS.

holds:

$$\mathsf{Adv}^{\mathrm{kAnon}}_{\Pi,\mathcal{D}}(\mathcal{A},\mathsf{Auth},\lambda) \leqslant \mathsf{negl}(\lambda).$$

5.1 k-ASPCGS Construction

In this section we briefly discuss a simple modification of our k-AGS construction, that achieves Set Pre-Constrained definitions. Our k-ASPCGS construction $\Pi[k,n,\mathcal{M},\mathcal{ID}]$ uses the same building blocks as our k-AGS construction (see Section 3.2), except for:

- a set pre-constrained encryption scheme SPCE over a universe $\mathcal{M}$, whose message space contains pairs (y,pid) with $y \in \mathbb{Z}_p$ and pid a pseudo-identity, is used instead of a public-key encryption scheme PKE;
- a subversion-resistant non-interactive zero-knowledge argument system $\mathsf{NIZK}_{\mathsf{S}}$ is defined for a different relation generator $\mathsf{RelGen}_{\mathsf{S}}(1^\lambda)$. The generator outputs relations $\mathcal{R}^{\mathsf{S}}_{k,\mathcal{D},\mathsf{ck},\mathsf{pk}_{\mathsf{SPCE}},\mathsf{pk}_{\mathsf{BS}}}$ corresponding to languages of the form[7]:

$$\mathcal{L} = \left\{ (msg,\sigma_{\mathsf{Sig}},ct) \;\middle|\; \exists\, c,\, y \in \mathbb{Z}_p,\, \mathsf{pid},\, \omega_{\mathsf{PC}},\, R\colon \begin{array}{c} \mathsf{pid} = ((com,\mathsf{pk}_{\mathsf{Sig}}),\sigma_{\mathsf{BS}}), \\ ct = \mathsf{SPCE.Enc}(\mathsf{pk}_{\mathsf{SPCE}},msg,(y,\mathsf{pid});R), \\ \mathsf{PC.Verify}(\mathsf{ck},com,\omega_{\mathsf{PC}},\mathsf{M2F}(\sigma_{\mathsf{Sig}}),y) = 1, \\ \mathsf{BS.Verify}(\mathsf{pk}_{\mathsf{BS}},\sigma_{\mathsf{BS}},(com,\mathsf{pk}_{\mathsf{Sig}})) = 1, \\ \mathsf{Sig.Verify}(\mathsf{pk}_{\mathsf{Sig}},\sigma_{\mathsf{Sig}},c\|msg) = 1 \end{array} \right\}.$$

[7]Formally, $\mathsf{RelGen}_{\mathsf{S}}(1^\lambda)$ samples some set $\mathcal{D}, |\mathcal{D}| = n$, runs $\mathsf{PC.Setup}(1^\lambda,k)$, $\mathsf{SPCE.KeyGen}(1^\lambda,\mathcal{D})$ and $\mathsf{BS.KeyGen}(1^\lambda)$ to generate a relation.

The modification is, thereof, straightforward:

- in the master key generation algorithm Gen, instead of $\mathsf{PKE.KeyGen}(1^\lambda)$ the SPCE key generation algorithm $\mathsf{SPCE.KeyGen}(1^\lambda, \mathcal{D})$ with the set $\mathcal{D}$ as input is used;
- in the signing algorithm Sign, the pair consisting of the share's y-coordinate $f(x)$ and the pseudo-identity pid is SPC encrypted with respect to the message msg being signed, and—exactly as in the k-AGS construction—the pseudo-identity is no longer transmitted in the clear but appears only as a witness of the $\mathsf{NIZK_S}$ proof, which is computed and verified on the statement $(msg, \sigma_{\mathsf{Sig}}, ct)$;
- in the tracing algorithm Trace, SPCE decryption is used instead of PKE decryption to recover the pair (y, pid), with the master opening key mok containing $\mathsf{sk_{SPCE}}$ instead of $\mathsf{sk_{PKE}}$.

The algorithms and the key generation protocol of Π are defined in Figure 23 and Figure 24 correspondingly. Because pid is hidden inside the SPC ciphertext, the group manager recovers it—and can therefore link and trace—only for the constrained messages $msg \in \mathcal{D}$; a signature on a message outside $\mathcal{D}$ leaks nothing about the signer, not even a pseudonym linking it to that signer's other signatures. Note that Gen may become a protocol between the group manager and an authority that authenticates (or owns) the set $\mathcal{D}$, if the underlying SPCE provides such functionality. We provide full description of the scheme . The correctness of Π follows straightforwardly from the correctness of the underlying primitives.

Security. Next, we discuss traceability and k-anonymity of our k-ASPCGS construction $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$.

Theorem 5 *Let $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a k-ASPCGS scheme as defined above. Assume the following building blocks: $\mathsf{NIZK_{KG}}$ and $\mathsf{NIZK_S}$ are knowledge-sound NIZK schemes, BS is a one-more unforgeable blind signature scheme, SPCE is a perfectly correct and* oINDCPA*-secure set pre-constrained encryption scheme, Sig is an* sUFCMA- *and* MFcr*-secure signature scheme, and PC is a binding polynomial commitment scheme. Then $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is a traceable k-ASPCGS scheme.*

Proof. (Sketch) The proof follows the same structure as the proof of Theorem 1, with the following exceptions:

- As in the k-AGS proof, the pseudo-identity pid is no longer transmitted in the clear, so it is recovered by the knowledge-soundness extractor of $\mathsf{NIZK_S}$: from each valid output signature the challenger extracts a witness $(c, y, \mathsf{pid}, \omega_{\mathsf{PC}}, R)$, and it is this extracted pid that the subsequent blind-signature one-more-unforgeability step uses to argue that every pseudo-identity originates from an honest KeyGen session.
- The perfect correctness of SPCE is used in place of the perfect correctness of PKE. This ensures that for each signature of a "bad" message $msg \in \mathcal{D}$, decrypting the ct part of the signature yields the same pair (y, pid) that was extracted from the $\mathsf{NIZK_S}$ proof, so that Trace groups the shares under the correct pseudo-identity.
- The $\mathsf{NIZK_S}$ relation binds the message msg as part of the statement; knowledge soundness of $\mathsf{NIZK_S}$ therefore extracts a witness for the SPCE encryption with the correct msg, ensuring that SPCE decryption in Trace recovers the share's y-coordinate and pseudo-identity whenever $msg \in \mathcal{D}$. □

Theorem 6 *Let $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a k-ASPCGS scheme as defined above. Assume the following building blocks: $\mathsf{NIZK_{KG}}$ and $\mathsf{NIZK_S}$ are subversion-zero-knowledge NIZK schemes, SPCE is an* rINDCPA*-secure set pre-constrained encryption scheme, PC is a hiding polynomial commitment scheme, Sig is a secure signature scheme, and BS is a secure blind signature scheme. Then $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is k-anonymous.*

Proof. (Sketch) Two preparatory game hops reduce the experiment to the setting of the proof of Theorem 2:

- By the subversion zero-knowledge of $\mathsf{NIZK_S}$, every $\mathsf{NIZK_S}$ proof produced by $\mathcal{O}^{\mathsf{Sign}}$ is replaced by a simulated one.
- By the rINDCPA security of SPCE, the SPCE ciphertext in each $\mathcal{O}^{\mathsf{Sign}}$ response with $msg \notin \mathcal{D}$ is replaced by an encryption of a uniformly random plaintext.

$\mathsf{Gen}(1^\lambda, k, \mathcal{D})$

$\mathsf{ck} \leftarrow \mathsf{PC.Setup}(1^\lambda, k)$
$(\mathsf{pk_{BS}}, \mathsf{sk_{BS}}) \leftarrow \mathsf{BS.KeyGen}(1^\lambda)$
$(\mathsf{pk_{SPCE}}, \mathsf{sk_{SPCE}}) \leftarrow \mathsf{SPCE.Gen}(1^\lambda, \mathcal{D})$
$\mathsf{crs_{KG}} \leftarrow \mathsf{NIZK_{KG}.Setup}(1^\lambda, \mathcal{R}^{\mathsf{KG}}_{\dots})$
$\mathsf{crs_S} \leftarrow \mathsf{NIZK_S.Setup}(1^\lambda, \mathcal{R}^{\mathsf{S}}_{\dots})$
$\mathsf{mpk} \leftarrow (\mathsf{ck}, \mathsf{pk_{BS}}, \mathsf{pk_{SPCE}}, \mathsf{crs_{KG}}, \mathsf{crs_S})$
$\mathsf{msk} \leftarrow \mathsf{sk_{BS}}$
$\mathsf{mok} \leftarrow \mathsf{sk_{SPCE}}$
return $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok})$

$\mathsf{Trace}(\mathsf{mok}, k, \{\sigma_i\}_{i\in\mathcal{I}})$

$\mathsf{sk_{SPCE}} \Leftarrow \mathsf{mok}$
$L_{\mathsf{sh}} \leftarrow \emptyset$
$L_{\mathsf{id}} \leftarrow \emptyset$
for $i \in \mathcal{I}$:
  $(\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}}) \Leftarrow \sigma_i$
  $x \leftarrow \mathsf{M2F}(\sigma_{\mathsf{Sig}})$
  $(y, \mathsf{pid}) \leftarrow \mathsf{SPCE.Dec}(\mathsf{sk_{SPCE}}, ct)$
  if $(x, y) \notin L_{\mathsf{sh}}[\mathsf{pid}]$:
    $L_{\mathsf{sh}}[\mathsf{pid}] \leftarrow L_{\mathsf{sh}}[\mathsf{pid}] \cup (x, y)$
for $shares \in L_{\mathsf{sh}}$:
  if $|shares| \geq k + 1$:
    $f \leftarrow \mathsf{Interpolate}(shares)$
    $\mathsf{id} \leftarrow f(0)$
    $L_{\mathsf{id}} \leftarrow L_{\mathsf{id}} \cup \mathsf{id}$
return L_{id}

$\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, msg)$

$(\mathsf{ck}, \mathsf{pk_{SPCE}}, \mathsf{crs_S}) \Leftarrow \mathsf{mpk}$
$(f, \mathsf{sk_{Sig}}, \mathsf{pid}) \Leftarrow \mathsf{sk}$
$c \leftarrow 0$
$\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk_{Sig}}, c \| msg)$
while $\mathsf{M2F}(\sigma_{\mathsf{Sig}}) = 0$:
  $c \leftarrow c + 1$
  $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk_{Sig}}, c \| msg)$
$x \leftarrow \mathsf{M2F}(\sigma_{\mathsf{Sig}})$
$\omega_{\mathsf{PC}} \leftarrow \mathsf{PC.Open}(\mathsf{ck}, f, x, f(x))$
$(ct; R) \leftarrow \mathsf{SPCE.Enc}(\mathsf{pk_{SPCE}}, msg, (f(x), \mathsf{pid}))$
$\pi_{\mathsf{S}} \leftarrow \mathsf{NIZK_S.Prove}(\mathsf{crs_S}, (msg, \sigma_{\mathsf{Sig}}, ct), (c, f(x), \mathsf{pid}, \omega_{\mathsf{PC}}, R))$
$\sigma \leftarrow (\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}})$
return σ

$\mathsf{Verify}(\mathsf{mpk}, msg, \sigma)$

$\mathsf{crs_S} \Leftarrow \mathsf{mpk}$
$(\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}}) \Leftarrow \sigma$
if $\mathsf{NIZK_S.Verify}(\mathsf{crs_S}, (msg, \sigma_{\mathsf{Sig}}, ct), \pi_{\mathsf{S}}) = 0$:
  return 0
return 1

Figure 23: k-ASPCGS construction $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$

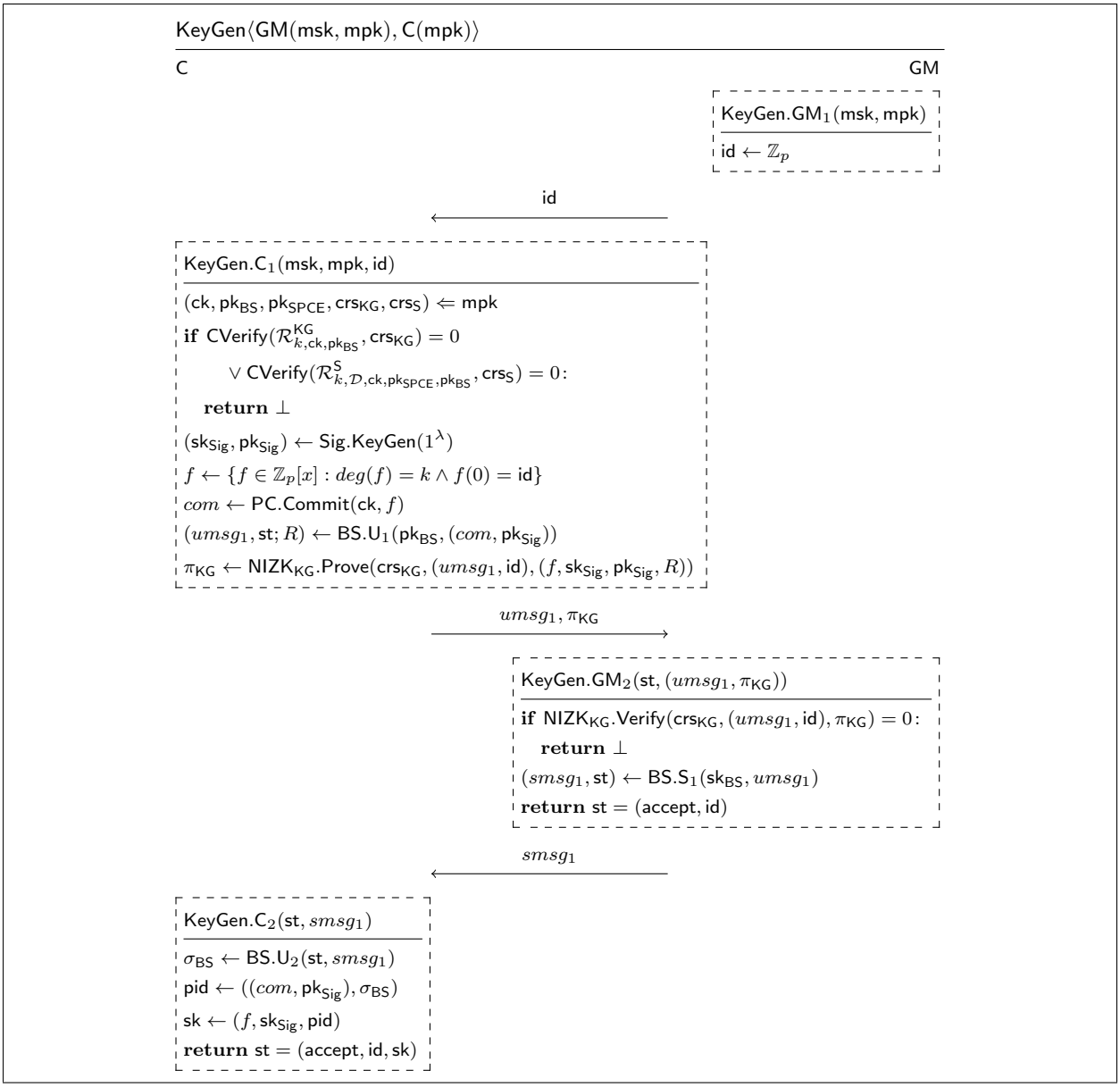

Figure 24: KeyGen protocol of k-ASPCGS $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$

After these hops, for $msg \in \mathcal{D}$ the group manager (who holds $\mathsf{mok} = \mathsf{sk}_{\mathsf{SPCE}}$) transparently decrypts to recover $(f(x), \mathsf{pid})$, so the $msg \in \mathcal{D}$ portion of $\mathcal{A}$'s view coincides with the modified signature used in the proof of Theorem 2; for $msg \notin \mathcal{D}$, the response no longer depends on the target users beyond σ_{Sig}, which is absorbed into the same bit-flip step over i.i.d. user keys as in the k-AGS proof. The remainder of the proof of Theorem 2 then carries over, using the $\leq k$ budget on $\mathcal{D}$-messages exactly as the $\leq k$ budget in the k-AGS experiment. □

5.2 Generic Transform

In this section, we introduce our generic construction $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$ of a k-ASPCGS for an arbitrary message space $\mathcal{M}$ and an identity space $\mathcal{ID} = \mathbb{Z}_p$. We make use of the following primitives in a black-box manner:

- a kAGS for a threshold k, message space $\mathcal{M}$, and identity space $\mathcal{ID}$;
- a set pre-constrained encryption scheme SPCE over a universe $\mathcal{M}$, with the message space being the signature space of kAGS;
- a subversion-resistant non-interactive zero-knowledge argument system NIZK for a relation generator $\mathsf{RelGen}(1^\lambda)$, where the generator outputs relations $\mathcal{R}^{kASPCGS}_{k,\mathsf{pk}_{\mathsf{SPCE}},\mathsf{mpk}_{\mathsf{kAGS}}}$ corresponding to languages of the form[8]:

$$\mathcal{L} = \left\{ (msg, ct) \;\middle|\; \exists \sigma,\, R\colon \begin{array}{c} 1 \leftarrow \mathsf{kAGS.Verify}(\mathsf{mpk}_{\mathsf{kAGS}}, (msg, \sigma)), \\ ct \leftarrow \mathsf{SPCE.Enc}(\mathsf{pk}_{\mathsf{SPCE}}, msg, \sigma; R) \end{array} \right\}.$$

The algorithms and the key generation protocol of Π are defined in Figure 25. Note that the key generation protocol KeyGen is identical to the key generation protocol of kAGS. Furthermore, Gen may also be a protocol between the group manager and an authority that authenticates (or owns) the set $\mathcal{D}$, if the underlying SPCE provides such functionality.

The correctness of Π follows straightforwardly from the correctness of the underlying primitives.

Next, we discuss the security of the generic k-ASPCGS transform $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$. We provide a brief proof sketch for the traceability theorem, with the full proof deferred to Appendix B. The k-anonymity theorem is straightforward.

Theorem 7 *Let $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a generic k-ASPCGS transform as defined above. Assume NIZK is a knowledge-sound NIZK scheme, SPCE is a perfectly correct set pre-constrained encryption scheme, and kAGS is a traceable k-anonymous group signature scheme. Then $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is a traceable k-ASPCGS scheme.*

Proof. (Sketch) The proof is straightforward. First, we use the knowledge soundness of NIZK and the correctness of SPCE to show that the SPCE ciphertext in each signature decrypts to a valid k-AGS signature whenever the corresponding message is "bad." Then, we construct a direct reduction to the traceability of the underlying k-AGS. □

Theorem 8 *Let $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ be a k-ASPCGS scheme as defined above. Assume NIZK is a subversion-zero-knowledge NIZK scheme, SPCE is an* rINDCPA*-secure set pre-constrained encryption scheme, and kAGS is a k-anonymous group signature scheme. Then $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$ is k-anonymous.*

Proof. The statement follows immediately from the subversion-zero-knowledge property of NIZK, the rINDCPA security of SPCE, and the k-anonymity of the underlying kAGS. □

[8] Formally, $\mathsf{RelGen}(1^\lambda)$ samples some set $\mathcal{D}, |\mathcal{D}| = n$, runs $\mathsf{kAGS.Gen}(1^\lambda, k)$ and $\mathsf{SPCE.Gen}(1^\lambda, \mathcal{D})$ to generate such a relation.

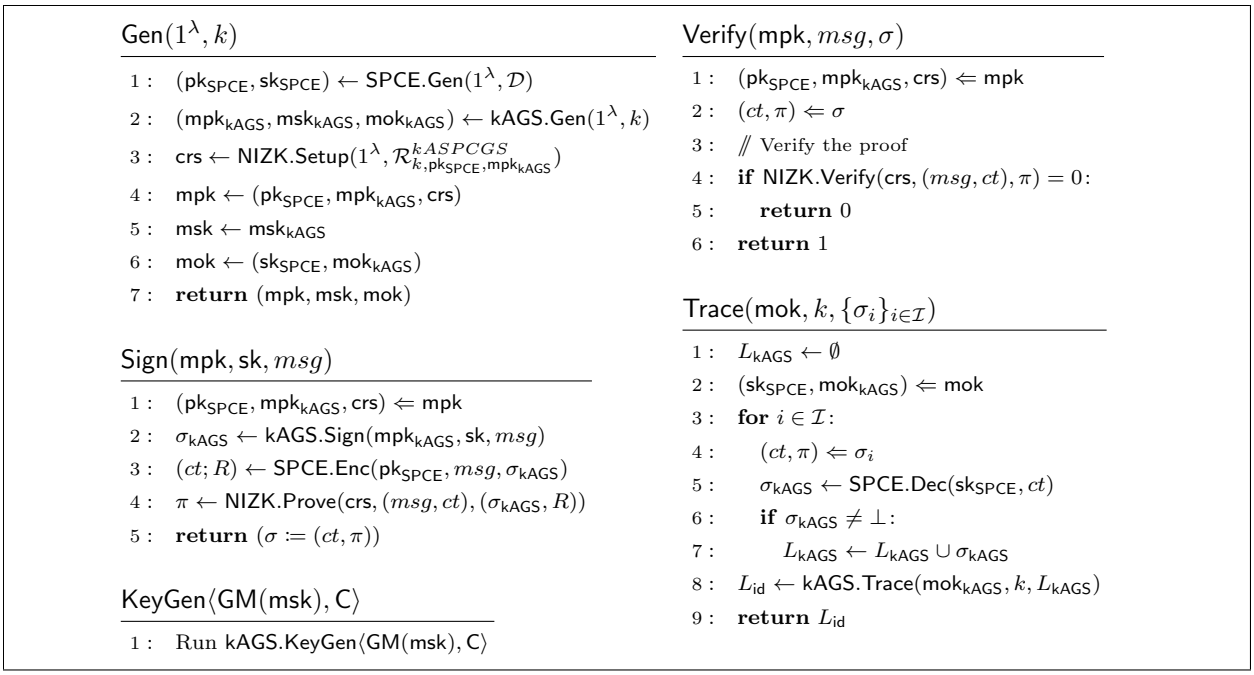

Figure 25: k-ASPCGS construction $\Pi[k, n, \mathcal{M}, \mathcal{ID}]$

References

[AJ22] Prabhanjan Ananth and Abhishek Jain. Pre-constrained encryption. In *13th Innovations in Theoretical Computer Science Conference*, 2022.

[ALSZ21] Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michał Zajac. On subversion-resistant snarks. *Journal of Cryptology*, 34(3):17, 2021.

[ASY06] Man Ho Au, Willy Susilo, and Siu-Ming Yiu. Event-oriented k-times revocable-iff-linked group signatures. In *Australasian Conference on Information Security and Privacy*, pages 223–234. Springer, 2006.

[BBM+21] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI system. 2021.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted crs: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, page 777–804, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BGH+25] Pedro Branco, Matthew Green, Aditya Hegde, Abhishek Jain, and Gabriel Kaptchuk. How to trace viral content in end-to-end encrypted messaging. *Cryptology ePrint Archive*, 2025.

[BGI+25] Allison Bishop, Matthew Green, Yuval Ishai, Abhishek Jain, and Paul Lou. Fully anonymous secret sharing. In *Annual International Cryptology Conference*, pages 356–389. Springer, 2025.

[BGJP22] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. 2022.

[BGJP25] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. Laconic PSI on authenticated inputs and applications. 2025.

[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*, volume 2656 of *Lecture Notes in Computer Science*, page 614–629, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[CHL06] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash. In *International conference on security and cryptography for networks*, pages 141–155. Springer, 2006.

[CLXZ25] Wonseok Choi, Xiangyu Liu, Lirong Xia, and Vassilis Zikas. K-linkable ring signatures and applications in generalized voting. *Cryptology ePrint Archive*, 2025.

[DL21] Jesus Diaz and Anja Lehmann. Group signatures with user-controlled and sequential linkability. In *IACR International Conference on Public-Key Cryptography*, pages 360–388. Springer, 2021.

[EBG+24] Harry Eldridge, Gabrielle Beck, Matthew Green, Nadia Heninger, and Abhishek Jain. Abuse-resistant location tracking: Balancing privacy and safety in the offline finding ecosystem. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5431–5448, 2024.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 233–253. Springer, Berlin, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_12.

[fJ21] Brennan Center for Justice. Facebook's content moderation rules are a mess. https://www.brennancenter.org/our-work/analysis-opinion/facebooks-content-moderation-rules-are-mess, 2021.

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge snarks. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, volume 10769 of *Lecture Notes in Computer Science*, page 315–347, Cham, 2018. Springer International Publishing.

[GKVL21] Matthew Green, Gabriel Kaptchuk, and Gijs Van Laer. Abuse resistant law enforcement access systems. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 553–583. Springer, 2021.

[GKVW20] Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In *Theory of Cryptography Conference*, pages 229–259. Springer, 2020.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–621. IEEE, 2017.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, Berlin, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_24.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511. Springer, Berlin, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_26.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Let25] Open Letter. Open letter on the position of scientists and researchers on the eu's proposed child sexual abuse regulation. https://csa-scientist-open-letter.org/Sep2025, 2025.

[LRTY21] Linsheng Liu, Daniel S Roche, Austin Theriault, and Arkady Yerukhimovich. Fighting fake news in encrypted messaging with the fuzzy anonymous complaint tally system (facts). *arXiv preprint arXiv:2109.04559*, 2021.

[May19] Jonathan Mayer. Content moderation for end-to-end encrypted messaging. *Princeton University*, 2019.

[New25] The News. Google flags child's groin images taken for doctor as sexual abuse material. https://www.thenews.com.pk/latest/985350-google-flags-childs-groin-images-taken-for-doctor-as-sexual-abuse-material, 2025.

[NSN05] Lan Nguyen and Rei Safavi-Naini. Dynamic k-times anonymous authentication. In *International Conference on Applied Cryptography and Network Security*, pages 318–333. Springer, 2005.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SRKM25] Maurice Shih, Michael Rosenberg, Hari Kailad, and Ian Miers. Zk-promises: Anonymous moderation, reputation, and blocking from anonymous credentials with callbacks. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 4995–5014, 2025.

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. K-times anonymous authentication. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 308–322. Springer, 2004.

[TNL+25] Nam Tran, Khoa Nguyen, Dongxi Liu, Josef Pieprzyk, and Willy Susilo. Many-time linkable ring signatures. *Cryptology ePrint Archive*, 2025.

[TS06] Isamu Teranishi and Kazue Sako. K-times anonymous authentication with a constant proving cost. In *International Workshop on Public Key Cryptography*, pages 525–542. Springer, 2006.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611. IEEE, 2017.

SUPPLEMENTAL MATERIALS

A Security Proofs of k-AGS

A.1 Traceability

Theorem 9 *Let $\mathcal{A}$ be an adversary against the traceability of the k-AGS scheme $\Pi[\mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$. Let $\mathsf{NIZK}_{\mathsf{KG}}$ and $\mathsf{NIZK}_{\mathsf{S}}$ be knowledge-sound NIZK schemes, let BS be a one-more-unforgeable blind signature, let PKE be a correct* INDCPA *secure public-key encryption scheme, let Sig be an* sUFCMA- *and* MFcr*-secure signature scheme, and let PC be a binding polynomial commitment scheme. Then there exist negligible functions $\mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda)$, $\mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}_{\mathsf{S}}}(\lambda)$, $\mathsf{negl}^{\mathrm{omUnforg}}_{\mathsf{BS}}(\lambda)$, $\mathsf{negl}^{\mathrm{corr}}_{\mathsf{PKE}}(\lambda)$, $\mathsf{negl}^{\mathrm{MFcr}}_{\mathsf{Sig}}(\lambda)$, $\mathsf{negl}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(\lambda)$, and $\mathsf{negl}^{\mathrm{Bind}}_{\mathsf{PC}}(\lambda)$, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{Trace}}_{\Pi}(\mathcal{A},\lambda) \leqslant\ & \mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda) + \mathsf{negl}^{\mathrm{omUnforg}}_{\mathsf{BS}}(\lambda) + \mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}_{\mathsf{S}}}(\lambda) \\ & + \mathsf{negl}^{\mathrm{corr}}_{\mathsf{PKE}}(\lambda) + \mathsf{negl}^{\mathrm{MFcr}}_{\mathsf{Sig}}(\lambda) + \mathsf{negl}^{\mathrm{Bind}}_{\mathsf{PC}}(\lambda) + q_{hon} \cdot \mathsf{negl}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(\lambda),\end{aligned} \tag{1}$$

where q_{hon} is a maximum number of honest users $\mathcal{A}$ registers.

Proof. We construct a sequence of indistinguishable games $G_0, \ldots, G_7$, where G_0 corresponds exactly to the experiment $\mathbf{Exp}^{\mathrm{Trace}}_{\Pi}(1^\lambda, \mathcal{A})$, and G_7 is a game in which the adversary cannot win.

G_1: By the knowledge-soundness property of $\mathsf{NIZK}_{\mathsf{KG}}$, for the adversary $\mathcal{A}$ there exists a knowledge extractor $\mathcal{E}_{\mathcal{A}}$. In G_1, using the extractor $\mathcal{E}_{\mathcal{A}}$, we modify the KeyGen oracle as follows.

If an adversarial query to $\mathcal{O}^{\mathsf{KeyGen}}$ results in the state accept, this means that in that query the adversary provided a message $umsg_1$ and a proof π_{KG} such that $\mathsf{NIZK}_{\mathsf{KG}}.\mathsf{Verify}(\mathsf{crs}, (umsg_1, \mathsf{id}), \pi_{\mathsf{KG}}) = 1$, where id is an identity generated in a previous adversarial query to KeyGen corresponding to that key-generation session. The challenger then uses the extractor $\mathcal{E}_{\mathcal{A}}$ to obtain a corresponding witness w and checks whether $\mathcal{R}_{k,\mathsf{ck},\mathsf{pk}_{BS}}((umsg_1, \mathsf{id}), w) = 1$. If this check fails, the challenger aborts the game (outputting 0). We call this a failure event and denote it by fail_{01}. Note that if the game does not abort, it holds that $f(0) = \mathsf{id} \in L_{\mathsf{id}}$.

We now construct an adversary $\mathcal{B}$ against the knowledge-soundness of $\mathsf{NIZK}_{\mathsf{KG}}$ as follows. $\mathcal{B}$ perfectly simulates G_1 for $\mathcal{A}$, using the $\mathsf{NIZK}_{\mathsf{KG}}$ parameters received from its own challenger. If the failure event occurs, $\mathcal{B}$ outputs $((umsg_1, \mathsf{id}), \pi_{\mathsf{KG}})$ as its solution in the knowledge-soundness game and wins. If the failure event does not occur, $\mathcal{B}$ outputs random data. Hence, the probability of $\mathcal{B}$ winning the knowledge-soundness game is at least the probability of the failure event in G_1.

Since G_0 and G_1 behave identically unless the failure event occurs, we have:

$$\begin{aligned}\Pr[G_0(\mathcal{A}, 1^\lambda) \to 1] &\leqslant \Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{01} \textit{ in } G_1(\mathcal{A}, 1^\lambda)] \\ &\leqslant \Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] + \mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda).\end{aligned} \tag{2}$$

G_2: By the knowledge-soundness of $\mathsf{NIZK}_{\mathsf{S}}$, for the adversary $\mathcal{A}$ there exists a knowledge extractor $\mathcal{E}_{\mathcal{A}}$. In G_2, the challenger uses $\mathcal{E}_{\mathcal{A}}$ to modify the experiment as follows.

After $\mathcal{A}$ outputs its challenge signatures, for each valid message–signature pair (msg_i, σ_i), i.e., such that $\mathsf{NIZK}_{\mathsf{S}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{S}}, (msg_i, \sigma_{\mathsf{Sig}}, ct), \pi_{\mathsf{S}}) = 1$ where $\sigma_i = (\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}})$, the challenger uses $\mathcal{E}_{\mathcal{A}}$ to obtain a corresponding witness $w = (c, y, \mathsf{pid}, \omega_{\mathsf{PC}}, R)$ and checks whether $\mathcal{R}^{\mathsf{S}}_{k,\mathsf{ck},\mathsf{pk}_{\mathsf{PKE}},\mathsf{pk}_{\mathsf{BS}}}((msg_i, \sigma_{\mathsf{Sig}}, ct), w) = 1$. If this check fails, the challenger aborts the game (outputting 0). We denote that failure event by fail_{12}.

If the game does not abort, then, writing $\mathsf{pid} = ((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}})$, it holds that: ω_{PC} is a valid PC opening of com at $(\mathsf{M2F}(\sigma_{\mathsf{Sig}}), y)$; ct is a correctly formed PKE encryption of the pair (y, pid) with randomness R; σ_{BS} is a valid blind signature on $(com, \mathsf{pk}_{\mathsf{Sig}})$; and σ_{Sig} is a valid Sig signature on $c \| msg_i$ under $\mathsf{pk}_{\mathsf{Sig}}$. The challenger records the extracted tuple $(c, y, \mathsf{pid}, \omega_{\mathsf{PC}}, R)$ for each output signature. Note that, since the pseudo-identity

pid is no longer transmitted in the clear, this extraction step is what gives the challenger access to the pseudo-identities used by the adversary.

We now construct an adversary $\mathcal{B}$ against the knowledge-soundness of $\mathsf{NIZK_S}$. $\mathcal{B}$ perfectly simulates G_2 for $\mathcal{A}$, using the $\mathsf{NIZK_S}$ parameters received from its own challenger. Honest signatures are computed using the real $\mathsf{NIZK_S.Prove}$ algorithm, since $\mathcal{B}$ knows the actual witnesses. If the failure event occurs, $\mathcal{B}$ outputs $((msg_i, \sigma_{\mathsf{Sig}}, ct), \pi_{\mathsf{S}})$ as its solution in the knowledge-soundness game and wins. Otherwise, $\mathcal{B}$ outputs random data. Hence, the probability that $\mathcal{B}$ wins the knowledge-soundness game is at least the probability of the failure event in G_2.

Since G_1 and G_2 behave identically unless the failure event occurs and $\mathsf{NIZK_S}$ is knowledge-sound, we have

$$\begin{aligned}\Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] &\leqslant \Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{12} \textit{ in } G_2(\mathcal{A}, 1^\lambda)] \\ &\leqslant \Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] + \mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK_S}}(\lambda). \quad (3)\end{aligned}$$

G_3: In this game, for each adversarial query to $\mathcal{O}^{\mathsf{KeyGen}}$ that results in the state accept and does not cause the game to abort, the challenger parses the witness w extracted in G_1 as $(f, \mathsf{sk_{Sig}}, \mathsf{pk_{Sig}}, R)$, computes a commitment $com \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, f)$, and then obtains a blind signature σ_{BS} for $(com, \mathsf{pk_{Sig}})$ as follows:

$$(umsg_1, \mathsf{st}) \leftarrow \mathsf{BS.U}_1(\mathsf{pk_{BS}}, (com, \mathsf{pk_{Sig}}); R) \quad (4)$$

$$\sigma_{\mathsf{BS}} \leftarrow \mathsf{BS.U}_2(\mathsf{st}, smsg_1), \quad (5)$$

where $smsg_1$ is the server-side BS message computed as part of the oracle's reply to the query.

At the start of the game, the challenger initializes two empty lists L_{BS} and L_{com}. During execution, each pair (f, com) is stored in L_{com}, and each corresponding blind signature $((com, \mathsf{pk_{Sig}}), \sigma_{\mathsf{BS}})$ is stored in L_{BS}. In addition to values computed in the interactive $\mathcal{O}^{\mathsf{KeyGen}}$ oracle, the challenger *includes values computed during honest executions* of the KeyGen protocol when $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{HonestKG}}$.

When the adversary $\mathcal{A}$ outputs its signatures, the challenger—using the witnesses extracted in G_2—compiles a list

$$\mathcal{L}_{\mathsf{pid}} = \{\mathsf{pid}_i \mid (c_i, y_i, \mathsf{pid}_i, \omega_{\mathsf{PC},i}, R_i) \text{ is the witness extracted from } \sigma_i\}$$

of all pseudo-identities recovered from the $\mathsf{NIZK_S}$ proofs of the output signatures. Recall that each pseudo-identity $\mathsf{pid} = ((com, \mathsf{pk_{Sig}}), \sigma_{\mathsf{BS}})$ consists of a commitment/signature-key pair together with a blind signature over it, and that, by the changes of G_2, every recovered σ_{BS} is a valid blind signature on $(com, \mathsf{pk_{Sig}})$. The challenger checks whether each $\mathsf{pid} \in \mathcal{L}_{\mathsf{pid}}$ appears in the list $\mathcal{L}_{\mathsf{BS}}$, and aborts the game (outputting 0) if this check fails. We denote this failure event by fail_{23}.

We now construct an adversary $\mathcal{C}$ against the strong one-more unforgeability of BS as follows. $\mathcal{C}$ simulates G_3 for $\mathcal{A}$, using the BS parameters received from its own challenger. When replying to $\mathcal{A}$'s $\mathcal{O}^{\mathsf{KeyGen}}$ or $\mathcal{O}^{\mathsf{HonestKG}}$ queries, $\mathcal{C}$ uses its own omUnforg server-side signing oracle to compute $smsg_1$. It then constructs the list $\mathcal{L}_{\mathsf{BS}}$ exactly as in G_3.

When $\mathcal{A}$ outputs its signatures, the list $\mathcal{L}_{\mathsf{BS}}$ contains exactly as many message–signature pairs as the number of $\mathcal{C}$'s queries to its signing oracle. If the failure event fail_{23} occurs, this means $\mathcal{A}$ has provided at least one $\mathsf{pid} = ((com, \mathsf{pk_{Sig}}), \sigma_{\mathsf{BS}})$ where σ_{BS} is a valid blind signature over $(com, \mathsf{pk_{Sig}})$, that is not in $\mathcal{L}_{\mathsf{BS}}$. In this case, $\mathcal{C}$ outputs $\mathcal{L}_{\mathsf{BS}} \cup \{\mathsf{pid}\}$ in the omUnforg experiment and wins. If the failure event does not occur, $\mathcal{C}$ outputs $\mathcal{L}_{\mathsf{BS}}$, losing its game. Hence, the probability that $\mathcal{C}$ wins is at least the probability of fail_{23} in G_3.

Since G_2 and G_3 are identical when the failure event does not occur, we have:

$$\begin{aligned}\Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] &\leqslant \Pr[G_3(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{23} \textit{ in } G_3(\mathcal{A}, 1^\lambda)] \\ &\leqslant \Pr[G_3(\mathcal{A}, 1^\lambda) \to 1] + \mathsf{negl}^{\mathrm{omUnforg}}_{\mathsf{BS}}(\lambda). \quad (6)\end{aligned}$$

G_4: In this game, the challenger, for each valid output signature $\sigma_i = (\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}})$, checks whether the PKE decryption of ct equals the pair (y, pid) extracted from π_{S} in G_2. If this check fails, the challenger aborts (outputting 0). We denote that failure event by fail_{34}.

If the check does not fail, the challenger replaces ct with the extracted pair (y, pid) and π_{S} with the extracted opening ω_{PC}; it also skips PKE decryption in Trace, working directly with the extracted (y, pid).

Due to the changes introduced in G_2, every ciphertext ct in each output signature is a correctly computed PKE encryption of the extracted pair (y, pid) with randomness R. Hence, the failure event fail_{34} corresponds exactly to a violation of PKE correctness. Since G_3 and G_4 behave identically unless the failure event occurs, we have

$$\Pr[G_3(\mathcal{A}, 1^\lambda) \to 1] \leqslant \Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{34} \text{ in } G_4(\mathcal{A}, 1^\lambda)] \leqslant \Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] + \mathsf{negl}^{\mathrm{corr}}_{\mathsf{PKE}}(\lambda). \quad (7)$$

G_5: This game is almost identical to G_4. Before running the Trace algorithm on the signatures output by the adversary, and after performing the checks introduced in the previous games, the challenger checks whether there exist two output message–signature pairs (msg_i, σ_i) and (msg_j, σ_j), with $\sigma_i = (\sigma_{\mathsf{Sig},i}, ct_i, \pi_{\mathsf{S},i})$ and $\sigma_j = (\sigma_{\mathsf{Sig},j}, ct_j, \pi_{\mathsf{S},j})$, such that:

$$\begin{aligned} \mathsf{pid}_i &= \mathsf{pid}_j, \\ msg_i &\neq msg_j, \\ \mathsf{M2F}(\sigma_{\mathsf{Sig},i}) &= \mathsf{M2F}(\sigma_{\mathsf{Sig},j}), \end{aligned}$$

where $\mathsf{pid}_i, \mathsf{pid}_j$ are the pseudo-identities extracted in G_2. That is, it checks whether there is a collision on the x-coordinates of the secret sharing for a fixed pseudo-identity. If such a pair is found, the challenger aborts the game (outputting 0). We denote this failure event by fail_{45}.

By the changes of G_2, the extracted witnesses guarantee that $\sigma_{\mathsf{Sig},i}$ is a valid Sig signature on $c_i \| msg_i$ and $\sigma_{\mathsf{Sig},j}$ a valid Sig signature on $c_j \| msg_j$, both under the same public key $\mathsf{pk}_{\mathsf{Sig}}$ contained in $\mathsf{pid}_i = \mathsf{pid}_j$. Since $msg_i \neq msg_j$, we have $c_i \| msg_i \neq c_j \| msg_j$, so these two valid pairs together with $\mathsf{M2F}(\sigma_{\mathsf{Sig},i}) = \mathsf{M2F}(\sigma_{\mathsf{Sig},j})$ constitute a winning tuple for the MFcr game of Sig. It follows immediately that we can construct an adversary $\mathcal{B}$ such that the probability of fail_{45} occurring is at most the probability of $\mathcal{B}$ winning the MFcr game.

Since G_4 and G_5 behave identically unless the failure event occurs, we have:

$$\Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] \leqslant \Pr[G_5(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{45} \text{ in } G_4(\mathcal{A}, 1^\lambda)] \leqslant \Pr[G_5(\mathcal{A}, 1^\lambda) \to 1] + \mathsf{negl}^{\mathrm{MFcr}}_{\mathsf{Sig}}(\lambda). \quad (8)$$

G_6: In G_6, before running Trace, the challenger, for each valid output signature $\sigma_i = (\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}})$ with extracted tuple $(c, y, \mathsf{pid}, \omega_{\mathsf{PC}}, R)$ and $\mathsf{pid} = ((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}})$, checks whether $y = f(x)$, where f is the polynomial corresponding to commitment com retrieved from L_{com} and $x = \mathsf{M2F}(\sigma_{\mathsf{Sig}})$. The challenger aborts the game (outputting 0) if this check fails. We denote this failure event by fail_{56}.

We now construct an adversary $\mathcal{D}$ against the binding property of PC. $\mathcal{D}$ perfectly simulates G_6 for $\mathcal{A}$, using the commitment key ck received from its own challenger. If the failure event occurs, then one of the signatures satisfies $f(x) \neq y$, even though the extracted opening ω_{PC} is a valid opening for com at (x, y). In that case, $\mathcal{D}$ computes $\omega_{real} = \mathsf{PC.Open}(\mathsf{ck}, f, x, f(x))$, and outputs the tuple $(com, x, y, f(x), \omega_{\mathsf{PC}}, \omega_{real})$. Otherwise, it fails its own experiment. Hence, the probability that $\mathcal{D}$ wins is at least the probability of fail_{56} in G_6.

Since G_5 and G_6 are identical unless this failure event occurs, we have:

$$\Pr[G_5(\mathcal{A}, 1^\lambda) \to 1] \leqslant \Pr[G_6(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{56} \text{ in } G_6(\mathcal{A}, 1^\lambda)] \leqslant \Pr[G_6(\mathcal{A}, 1^\lambda) \to 1] + \mathsf{negl}^{\mathrm{Bind}}_{\mathsf{PC}}(\lambda). \quad (9)$$

G_7: In this game, the challenger checks whether the pseudo-identity pid extracted in G_2 from any valid output signature corresponds to an honest user. (Note that, due to G_3, pseudo-identities may belong only to adversarially controlled users or honest users.) We denote the event that any such signature occurs by fail_{67}. Recall that the traceability experiment $\mathbf{Exp}^{\mathrm{Trace}}_{\Pi}$ rejects the adversary's output whenever any output

message msg_i appears in the list L_{hon} of messages previously signed via $\mathcal{O}^{\mathsf{Sign}}$; consequently, the message in any signature triggering fail_{67} is one that was never queried to $\mathcal{O}^{\mathsf{Sign}}$ for any honest user.

Since, by the changes of G_2, every valid k-AGS signature contains a valid Sig signature on a message $c\|msg$ under the corresponding $\mathsf{pk}_{\mathsf{Sig}}$, the failure event fail_{67} implies that the adversary has produced a fresh Sig message–signature pair $(c\|msg, \sigma_{\mathsf{Sig}})$ for some honestly generated key pair $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}})$. Hence, there exists an adversary $\mathcal{B}$ against the sUFCMA security of Sig, such that

$$\Pr[G_6(\mathcal{A}, 1^\lambda) \to 1] \leqslant \Pr[G_7(\mathcal{A}, 1^\lambda) \to 1] + \Pr[\mathsf{fail}_{67} \text{ in } G_7(\mathcal{A}, 1^\lambda)] \leqslant \Pr[G_7(\mathcal{A}, 1^\lambda) \to 1] + q_{hon} \cdot \mathsf{negl}^{\mathrm{sUFCMA}}_{\mathsf{Sig}}(\lambda), \tag{10}$$

where q_{hon} is the maximum number of registered honest users—$\mathcal{B}$ first "guesses" for which honest user the failure event happens.

Analysis of G_7: Let L'_{BS} and L'_{com} be the subsets of L_{BS} and L_{com}, respectively, that contain only entries added as a result of adversarial KeyGen sessions. If all the additional checks introduced in the previous games are passed, the following holds:

1. The number of message–signature pairs in L'_{BS} and the number of pairs in L'_{com} are both equal to the number of adversarial KeyGen sessions that resulted in accept. Moreover, for each $(f, \cdot) \in L'_{com}$, it holds that $f(0) \in L'_{\mathsf{id}}$ (due to G_1).

2. The number of distinct pseudo-identities pid extracted from the signatures output by the adversary is at most the number of message–signature pairs in L'_{BS}. Moreover, each such pid is in L'_{BS}, and each commitment in a pseudo-identity has a corresponding entry in L'_{com} (due to G_3 and the abort event in G_7).

3. For each distinct pid, there are no two signatures that imply the same x-coordinate for the secret sharing (due to G_5).

4. For every signature with commitment $\mathsf{pid}.com$, it holds that $y = f(x)$, where f is the polynomial corresponding to com from L'_{com} (due to G_6).

5. The number of signatures output by the adversary is at least $k \cdot usr + 1$, where usr is the number of successful adversarial KeyGen sessions (due to the check in the original game G_0).

From these properties, it follows that there exists some public identity pid which is used in at least $k + 1$ signatures, and each signature "has" a distinct x-coordinate. In G_7, the challenger, while running Trace, finds such a pid, parses the corresponding signatures to obtain distinct (due to item 3) shares $(x_1, y_1), \ldots, (x_{k+1}, y_{k+1})$, and interpolates a polynomial f' over these points. Clearly (due to item 4), f' is equal to the polynomial f retrieved from L'_{com} for commitment $\mathsf{pid}.com$, and $f(0) \in L_{\mathsf{id}}$ (by item 1). Hence, if the game has not already aborted, Trace will always output some $\mathsf{id} \in L_{\mathsf{id}}$, thereby causing the adversary to lose. We have:

$$\Pr[G_7(\mathcal{A}, 1^\lambda) \to 1] = 0.$$

Finally, by combining inequalities (2)–(10), we conclude the proof of the theorem. □

A.2 k-Anonymity

Theorem 10 *Let $\mathcal{A}$ be an adversary against the k-Anonymity of the k-AGS scheme $\Pi[\mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$. Let $\mathsf{NIZK}_{\mathsf{KG}}$ be a subversion-zero-knowledge NIZK scheme, PC a hiding polynomial commitment scheme, and BS a secure blind signature scheme. Then, there exist negligible functions $\mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}}(\lambda)$, $\mathsf{negl}^{\mathrm{Hiding}}_{\mathsf{PC}}(\lambda)$, and $\mathsf{negl}^{\mathrm{Blind}}_{\mathsf{BS}}(\lambda)$ such that*

$$\mathsf{Adv}^{\mathrm{kAnon}}_{\Pi}(\mathcal{A}, \lambda) \leqslant 4 \cdot \mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda) \;+\; 4 \cdot \mathsf{negl}^{\mathrm{Hiding}}_{\mathsf{PC}}(\lambda) \;+\; \mathsf{negl}^{\mathrm{Blind}}_{\mathsf{BS}}(\lambda).$$

Proof. Since the k-Anonymity experiment is defined with the group manager as the adversary, and the group manager holds the master opening key $\mathsf{mok} = \mathsf{sk}_{\mathsf{PKE}}$, encryption towards the group manager is transparent to it. We therefore immediately consider a modification of Π in which a signature directly contains the share's y-coordinate and the pseudo-identity pid—which in Π are jointly encrypted under $\mathsf{pk}_{\mathsf{PKE}}$—together with the polynomial commitment opening ω_{PC}, in place of the ciphertext ct and the proof π_{S}. Concretely, the modified signature is $(\sigma_{\mathsf{Sig}}, c, y, \mathsf{pid}, \omega_{\mathsf{PC}})$ with $\mathsf{pid} = ((com, \mathsf{pk}_{\mathsf{Sig}}), \sigma_{\mathsf{BS}})$. Since the group manager could itself decrypt ct to recover (y, pid), this modification only gives the adversary more information; consequently, for k-anonymity we do not rely on any properties of PKE or $\mathsf{NIZK}_{\mathsf{S}}$.

We construct a sequence of indistinguishable games $G_0, \ldots, G_7$, where G_0 is the experiment $\mathbf{Exp}_{\Pi}^{\mathrm{kAnon}}(1^\lambda, \mathcal{A})$ with bit $b = 0$, and G_7 is the experiment $\mathbf{Exp}_{\Pi}^{\mathrm{kAnon}}(1^\lambda, \mathcal{A})$ with bit $b = 1$.

G_1: To define G_1, we first construct a subverter $\mathcal{S}$ for the NIZK scheme $\mathsf{NIZK}_{\mathsf{KG}}$ from $\mathcal{A}$. The subverter $\mathcal{S}$ starts $\mathcal{A}$ and uses the $\mathsf{crs}_{\mathsf{KG}}$ output by $\mathcal{A}$ (contained in the master public key mpk) as its own output. It includes all other outputs of $\mathcal{A}$ in its state st, together with $\mathcal{A}$'s internal state and outputs it along with the random tape used. By the definition of subversion zero-knowledge, there exists a negligible function $\mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda)$ and a PPT simulation trapdoor extractor $\mathcal{E}_{\mathcal{S}}$ such that, for every $\lambda \in \mathbb{N}$, every $\mathcal{R} \in \mathsf{RelGen}(1^\lambda)$, and every PPT adversary $\mathcal{B}$,

$$\mathsf{Adv}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}((\mathcal{S}, \mathcal{B}), \lambda) \leqslant \mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda).$$

In G_1, right after starting $\mathcal{A}$, the challenger checks that the $\mathsf{crs}_{\mathsf{KG}}$ is well formed using $\mathsf{NIZK}_{\mathsf{KG}}.\mathsf{CVerify}$, and extracts a trapdoor td using $\mathcal{E}_{\mathcal{S}}$. The KeyGen oracle is then modified as follows:

In both KeyGen sessions, when computing the first client-side message in response to a corresponding oracle query, the challenger no longer generates the proof

$$\pi_{\mathsf{KG}} = \mathsf{NIZK}_{\mathsf{KG}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{KG}}, (umsg_1, \mathsf{id}), (f, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, R))$$

as in the original scheme Π. Instead, it uses the NIZK simulator:

$$\pi_{\mathsf{KG}} = \mathsf{NIZK}_{\mathsf{KG}}.\mathsf{Sim}(\mathsf{crs}_{\mathsf{KG}}, \mathsf{td}, (umsg_1, \mathsf{id})).$$

For the convenience of establishing "indistinguishability" of the games G_0 and G_1, we introduce an intermediate game G_1', in which the proof is simulated only in the first KeyGen session, while the second remains unchanged.

We now construct two adversaries, $\mathcal{B}_1$ and $\mathcal{B}_2$, and corresponding pairs $(\mathcal{S}, \mathcal{B}_1)$ and $(\mathcal{S}, \mathcal{B}_2)$ against the subversion zero-knowledge of $\mathsf{NIZK}_{\mathsf{KG}}$ and extractor $\mathcal{E}_{\mathcal{S}}$. In both $\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}(1^\lambda, (\mathcal{S}, \mathcal{B}_1))$ and $\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}(1^\lambda, (\mathcal{S}, \mathcal{B}_2))$, the subverter $\mathcal{S}$ first runs $\mathcal{A}$, outputs $\mathsf{crs}_{\mathsf{KG}}$, and stores the remaining outputs (including $\mathcal{A}$'s state $\mathsf{st}_{\mathcal{A}}$) as its own state $\mathsf{st}_{\mathcal{S}}$.

Each adversary then reconstructs the master public key from $\mathsf{st}_{\mathcal{S}}$, resumes $\mathcal{A}$ with state $\mathsf{st}_{\mathcal{A}}$, and continues the simulation. The distinction is as follows. $\mathcal{B}_1$ simulates G_0, except that in the first KeyGen session it forwards the statement–witness pair $((umsg_1, \mathsf{id}), (f, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, R))$ to its own challenger and uses the returned proof π as a reply to $\mathcal{A}$'s query. $\mathcal{B}_2$ behaves analogously, but does so in the second KeyGen session, while the first session is handled as in G_1 (with a simulated proof).

Finally, both adversaries output whatever $\mathcal{A}$ outputs.

It follows that $(\mathcal{S}, \mathcal{B}_1)$ perfectly simulates G_0 if the bit $b = 0$ in its ZK experiment, since the proof received form the challenger is a real one, and G_1' if $b = 1$, since the proof is simulated. Analogously, $(\mathcal{S}, \mathcal{B}_2)$ perfectly simulates G_1' if $b = 0$, and G_1 if $b = 1$. Hence,

$$\begin{aligned}
&\Pr[G_0(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] \\
&= \Pr[G_0(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_1'(\mathcal{A}, 1^\lambda) \to 1] + \Pr[G_1'(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] \\
&= \Pr[\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}(1^\lambda, (\mathcal{S}, \mathcal{B}_1)) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}(1^\lambda, (\mathcal{S}, \mathcal{B}_1)) \to 1 \mid b = 1] \\
&\quad + \Pr[\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}(1^\lambda, (\mathcal{S}, \mathcal{B}_2)) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}(1^\lambda, (\mathcal{S}, \mathcal{B}_2)) \to 1 \mid b = 1] \\
&= \mathsf{Adv}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}((\mathcal{S}, \mathcal{B}_1), \lambda) + \mathsf{Adv}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}, \mathcal{E}_{\mathcal{S}}}((\mathcal{S}, \mathcal{B}_2), \lambda) \leqslant 2 \cdot \mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda). \qquad (11)
\end{aligned}$$

G_2: In this game, in both KeyGen sessions initiated by adversarial queries to the oracle, the challenger modifies the way the first client-side message is computed. Whenever the adversary submits a query that contains a user identity id, the challenger instead chooses a fresh identity id^{new}, and uses it to generate a secret polynomial f. It then computes a polynomial commitment com to f, generates a fresh signing key pair $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, and the first message $umsg_1$ of the blind signature scheme on the pair $(com, \mathsf{pk}_{\mathsf{Sig}})$. At the same time, the challenger still uses the adversarially chosen id to compute a simulated proof. The resulting user secret key consists of the polynomial f associated with the fresh identity id^{new}, the signing key $\mathsf{sk}_{\mathsf{Sig}}$, and a public identity pid formed by the pair $(com, \mathsf{pk}_{\mathsf{Sig}})$ together with a blind signature σ_{BS} on $(com, \mathsf{pk}_{\mathsf{Sig}})$. These values, being part of the secret key, are also used to answer signing queries through the Sign oracle. In other words, instead of running $\mathsf{KeyGen.C}_1(\mathsf{id})$, the challenger executes the modified algorithm $\tilde{\mathsf{C}}_1$ given below.

$\tilde{\mathsf{C}}_1(\mathsf{id})$

1 : $\mathsf{id}^{new} \leftarrow \mathbb{Z}_p$
2 : $\mathsf{ck}, \mathsf{pk}_{\mathsf{BS}}, \mathsf{crs}_{\mathsf{KG}} \Leftarrow \mathsf{mpk}$
3 : $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$
4 : $f \xleftarrow{\$} \{f \in \mathbb{Z}_p[x] : \deg(f) = k \wedge f(0) = \mathsf{id}^{new}\}$
5 : $com \leftarrow \mathsf{PC.Commit}(\mathsf{ck}, f)$
6 : $(umsg_1, \mathsf{st}_{\mathsf{BS}}; R) \leftarrow \mathsf{BS.U}_1(\mathsf{pk}_{\mathsf{BS}}, (com, \mathsf{pk}_{\mathsf{Sig}}))$
7 : $\pi_{\mathsf{KG}} \leftarrow \mathsf{NIZK}_{\mathsf{KG}}.\mathsf{Sim}(\mathsf{crs}_{\mathsf{KG}}, \mathsf{td}, (umsg_1, \mathsf{id}))$
8 : $\mathsf{st} \leftarrow (\mathsf{id}^{new}, f, \mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk}_{\mathsf{Sig}}, \mathsf{st}_{\mathsf{BS}})$
9 : **return** $\mathsf{st}, umsg_1, \pi_{\mathsf{KG}}$

We also define an intermediate game G_2', where a fresh identity id^{new} is used only in the first KeyGen session, while in the second session the challenger proceeds with the adversarially chosen identity id.

We now construct two adversaries, $\mathcal{C}_1$ and $\mathcal{C}_2$, against the hiding property of the polynomial commitment scheme PC.

Adversary $\mathcal{C}_1$ simulates the game G_1 for $\mathcal{A}$. It begins by parsing the master public key mpk, output by $\mathcal{A}$, in order to extract the commitment key ck. When $\mathcal{A}$ initiates the first KeyGen session with an identity id, $\mathcal{C}_1$ samples a fresh signing key pair $(\mathsf{sk}_{\mathsf{Sig},1}, \mathsf{pk}_{\mathsf{Sig},1}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, outputs $(\mathsf{ck}, k, (0, \mathsf{id}))$ and its own internal state as input to its hiding experiment, and receives in return a commitment com. Using this com together with $\mathsf{pk}_{\mathsf{Sig},1}$, $\mathcal{C}_1$ runs the blind-signature protocol to obtain a blind signature on the pair $(com, \mathsf{pk}_{\mathsf{Sig},1})$, computes the simulated zero-knowledge proof, and forms the corresponding public identity $\mathsf{pid}_1 = ((com, \mathsf{pk}_{\mathsf{Sig},1}), \sigma_{\mathsf{BS}})$ for the first user. For the second KeyGen session, $\mathcal{C}_1$ proceeds exactly as in G_2, i.e., it executes the KeyGen protocol using the adversary's chosen identity id, but always replaces the zero-knowledge proof with a simulated one.

When $\mathcal{A}$ queries the signing oracle for the first user with message m, $\mathcal{C}_1$ needs to compute a signature share and opening ω_{PC}, which are derived from the polynomial f underlying com. To do this, $\mathcal{C}_1$ first runs the loop $c \leftarrow 0$; $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig},1}, c \| m)$; **while** $\mathsf{M2F}(\sigma_{\mathsf{Sig}}) = 0$: $c \leftarrow c + 1$; $\sigma_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig},1}, c \| m)$ to obtain $x = \mathsf{M2F}(\sigma_{\mathsf{Sig}})$, and then queries its own Open oracle at x. For the second user, $\mathcal{C}_1$ answers signing queries according to the scheme.

If in $\mathcal{C}_1$'s hiding experiment the challenge bit $b = 0$, then the commitment com and its openings correspond to a polynomial f such that $f(0) = \mathsf{id}$. In this case, $\mathcal{C}_1$ provides a perfect simulation of G_1. If $b = 1$, then the commitment corresponds to a polynomial f with $f(0)$ equal to a random value independent of id, which yields a perfect simulation of G_2'. In either case, $\mathcal{C}_1$ outputs the same result as $\mathcal{A}$.

The adversary $\mathcal{C}_2$ is defined in a completely analogous way, but it instead relates G_2' and G_2. Specifically, $\mathcal{C}_2$ simulates G_2' for $\mathcal{A}$, meaning that in the first KeyGen session it uses a fresh identity id^{new}, while in the second KeyGen session it follows the protocol with the adversary's chosen identity. However, $\mathcal{C}_2$ is also given a polynomial commitment by its challenger, and uses it to form the corresponding public identity and to answer signing queries (through its Open oracle), just as $\mathcal{C}_1$ does for the first user. If $b = 0$, $\mathcal{C}_2$'s simulation is indistinguishable from G_2'; if $b = 1$, it simulates G_2. Again, $\mathcal{C}_2$ outputs whatever $\mathcal{A}$ outputs.

Putting these together, we obtain:

$$\begin{aligned}
&\Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] \\
&= \Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_2'(\mathcal{A}, 1^\lambda) \to 1] + \Pr[G_2'(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] \\
&= \Pr[\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{C}_1) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{C}_1) \to 1 \mid b = 1] \\
&\quad + \Pr[\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{C}_2) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\text{Hiding}}_{\mathsf{PC}}(1^\lambda, \mathcal{C}_2) \to 1 \mid b = 1] \\
&= \mathsf{Adv}^{\text{Hiding}}_{\mathsf{PC}}(\mathcal{C}_1, \lambda) + \mathsf{Adv}^{\text{Hiding}}_{\mathsf{PC}}(\mathcal{C}_2, \lambda) \leqslant 2 \cdot \mathsf{negl}^{\text{Hiding}}_{\mathsf{PC}}(\lambda), \qquad (12)
\end{aligned}$$

where the last inequality follows from the hiding property of PC.

G_3: In this game, the challenger slightly changes the way in which user identities and the associated polynomials are generated. After $\mathcal{A}$ is initialized, but before it is given access to the KeyGen oracle, the challenger samples two fresh identities id_0^{new} and id_1^{new}, two fresh signing key pairs $(\mathsf{sk}_{\mathsf{Sig},0}, \mathsf{pk}_{\mathsf{Sig},0})$ and $(\mathsf{sk}_{\mathsf{Sig},1}, \mathsf{pk}_{\mathsf{Sig},1})$, and two polynomials f_0, f_1 of degree k over $\mathbb{Z}_p$ that evaluate to the corresponding identity at 0. It then computes two commitments, $com_0 = \mathsf{PC.Commit}(\mathsf{ck}, f_0)$ and $com_1 = \mathsf{PC.Commit}(\mathsf{ck}, f_1)$, using the commitment key ck that $\mathcal{A}$ provided in the master public key. These quantities are fixed for the rest of the game. Whenever the adversary queries the KeyGen oracle, the challenger no longer generates the polynomial commitment and the signing key pair on the fly, but instead uses the precomputed ones: for the first user it uses $(com_0, \mathsf{pk}_{\mathsf{Sig},0})$ and for the second user it uses $(com_1, \mathsf{pk}_{\mathsf{Sig},1})$. In both cases, the challenger engages in the blind-signature protocol with $\mathcal{A}$ to produce a blind signature on the corresponding pair and thus form the public identity $\mathsf{pid}_i = ((com_i, \mathsf{pk}_{\mathsf{Sig},i}), \sigma_i)$. The same two precomputed polynomials and signing keys are used by the challenger to answer signing queries via the Sign oracle.

This modification is purely technical: it only changes the point in time when the two polynomials and their commitments are sampled, without altering the distribution of the resulting values or the view of the adversary. Therefore,

$$\Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] = \Pr[G_3(\mathcal{A}, 1^\lambda) \to 1].$$

G_4: In this game, the challenger sets the bit b to 1. In other words, the challenger still uses id_i^{new} together with the corresponding polynomial f_i and commitment com_i in the KeyGen session initiated by the adversary $\mathcal{A}$ for user $i \in \{0, 1\}$, obtaining the secret key sk_i as the outcome of that session. At the same time, the roles of the keys in the signing oracle are swapped: when $\mathcal{A}$ requests a signature for user $i = 0$, the challenger responds using sk_1, and when $\mathcal{A}$ requests a signature for user $i = 1$, it responds using sk_0.

We now construct an adversary $\mathcal{D}$ against the blindness of BS. The adversary $\mathcal{D}$ simulates the execution of G_3 for $\mathcal{A}$. After initializing $\mathcal{A}$ and generating the fresh identities, signing key pairs, and corresponding commitments com_0 and com_1, $\mathcal{D}$ outputs $(\mathsf{pk}_{\mathsf{BS}}, (com_0, \mathsf{pk}_{\mathsf{Sig},0}), (com_1, \mathsf{pk}_{\mathsf{Sig},1}))$ along with its internal state to its own experiment, where $\mathsf{pk}_{\mathsf{BS}}$ is parsed from the master public key generated by $\mathcal{A}$, and $((com_0, \mathsf{pk}_{\mathsf{Sig},0}), (com_1, \mathsf{pk}_{\mathsf{Sig},1}))$ serves as the challenge message pair in the blindness experiment.

Whenever $\mathcal{A}$ initiates a KeyGen session for user i, $\mathcal{D}$ queries its own oracles in order to compute the blind-signature messages. Since $\mathcal{D}$ does not receive the blind signatures (σ_0, σ_1) from its experiment until both KeyGen sessions are completed, it postpones the formation of the secret keys until that point. Once the signatures are obtained, $\mathcal{D}$ sets $\mathsf{sk}_i.\mathsf{pid} = ((com_i, \mathsf{pk}_{\mathsf{Sig},i}), \sigma_i)$ and uses the resulting $\mathsf{sk}_i = (f_i, \mathsf{sk}_{\mathsf{Sig},i}, \mathsf{pid}_i)$ to respond to signing queries of $\mathcal{A}$ for user i.

If in $\mathcal{D}$'s experiment the bit b is 0, then in the first KeyGen session the pair $(com_0, \mathsf{pk}_{\mathsf{Sig},0})$ is blind signed, and the corresponding secret key sk_0 belongs to the first session, while sk_1 belongs to the second. In this case, $\mathcal{D}$ perfectly simulates the game G_3 for $\mathcal{A}$.

If instead the bit b in $\mathcal{D}$'s experiment is 1, then in the first KeyGen session $(com_1, \mathsf{pk}_{\mathsf{Sig},1})$ is blind signed, and thus sk_1 belongs to the first session while sk_0 belongs to the second. At the same time, the use of the secret keys in the signing oracle is not swapped: when $\mathcal{A}$ queries signing for user i, the challenger still responds using sk_i. Consequently, the secret key generated in the first KeyGen session is used for signing on

behalf of the second user, and vice versa. This matches exactly the behavior of the challenger in G_4. Hence, in this case $\mathcal{D}$ perfectly simulates G_4 for $\mathcal{A}$.

It follows that

$$\begin{aligned}
&\Pr[G_3(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] \\
&= \Pr[\mathbf{Exp}^{\mathrm{Blind}}_{\mathsf{BS}}(1^\lambda, \mathcal{D}) \to 1 \mid b = 0] - \Pr[\mathbf{Exp}^{\mathrm{Blind}}_{\mathsf{BS}}(1^\lambda, \mathcal{D}) \to 1 \mid b = 1] \\
&= \mathsf{Adv}^{\mathrm{Blind}}_{\mathsf{BS}}(\mathcal{D}, \lambda) \leqslant \mathsf{negl}^{\mathrm{Blind}}_{\mathsf{BS}}(\lambda). \qquad (13)
\end{aligned}$$

G_5, G_6, G_7: In these games, the modifications introduced in games G_3, G_2, and G_1 are successively reverted. Thus, G_7 corresponds exactly to the experiment $\mathbf{Exp}^{\mathrm{kAnon}}_{\Pi}(1^\lambda, \mathcal{A})$ with the challenge bit fixed to $b = 1$. Therefore, we obtain

$$\Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_7(\mathcal{A}, 1^\lambda) \to 1] \leqslant 2 \cdot \mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{KG}}}(\lambda) + 2 \cdot \mathsf{negl}^{\mathrm{Hiding}}_{\mathsf{PC}}(\lambda). \qquad (14)$$

Finally, by combining the bounds established in the transitions from G_0 through G_7, we conclude the proof of the theorem. □

A.3 Non-GM Unlinkability

Theorem 11 *Let $\mathcal{A}$ be an adversary against the non-GM unlinkability of the k-AGS scheme $\Pi[\mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$, making at most q_{Sign} queries to the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. Let $\mathsf{NIZK}_{\mathsf{S}}$ be a zero-knowledge NIZK scheme, PKE an* INDCPA*-secure public-key encryption scheme, and Sig a key-anonymous (*kAnonSig*) signature scheme. Then there exist negligible functions $\mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{S}}}(\lambda)$, $\mathsf{negl}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(\lambda)$, and $\mathsf{negl}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(\lambda)$ such that*

$$\mathsf{Adv}^{\mathrm{nGMUnlink}}_{\Pi}(\mathcal{A}, \lambda) \leqslant 2\mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{S}}}(\lambda) + 2 \cdot \mathsf{negl}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(\lambda) + \mathsf{negl}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(\lambda).$$

Proof. Since the adversary holds neither the master secret key msk nor the master opening key mok, and the two target users $\mathsf{id}_0, \mathsf{id}_1$ are registered honestly by the experiment—so that the adversary never observes their KeyGen executions—the only channel through which the secret keys $\mathsf{sk}_0, \mathsf{sk}_1$ influence the adversary's view is the response of the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. A signature $\sigma = (\sigma_{\mathsf{Sig}}, ct, \pi_{\mathsf{S}})$ carries information about the signing user in three components: the conventional signature σ_{Sig}, produced under the user's signing key $\mathsf{sk}_{\mathsf{Sig}}$; the PKE ciphertext ct, which encrypts the share's y-coordinate together with the pseudo-identity pid; and the $\mathsf{NIZK}_{\mathsf{S}}$ proof π_{S}, whose witness $(c, f(x), \mathsf{pid}, \omega_{\mathsf{PC}}, R)$ contains the pseudo-identity, the share, the counter, the commitment opening, and the encryption randomness. We first neutralise π_{S} and ct using the zero-knowledge property of $\mathsf{NIZK}_{\mathsf{S}}$ and the INDCPA security of PKE; once this is done, the pseudo-identities $\mathsf{pid}_0, \mathsf{pid}_1$ are no longer observable, and we may switch the signing key embedded in σ_{Sig} from one target user to the other using the key anonymity of Sig.

We construct a sequence of games $G_0, \ldots, G_7$, where G_0 is the experiment $\mathbf{Exp}^{\mathrm{nGMUnlink}}_{\Pi}(1^\lambda, \mathcal{A})$ with bit $b = 0$, and G_7 is the experiment $\mathbf{Exp}^{\mathrm{nGMUnlink}}_{\Pi}(1^\lambda, \mathcal{A})$ with bit $b = 1$. Recall that for $b = 0$ every $\mathcal{O}^{\mathsf{Sign}}$ query—regardless of the slot index $i \in \{0, 1\}$—is answered with sk_0, whereas for $b = 1$ a query with slot index i is answered with sk_i. Recall also that the experiment forbids querying the same message to $\mathcal{O}^{\mathsf{Sign}}$ more than once.

G_1: In this game, every $\mathsf{NIZK}_{\mathsf{S}}$ proof produced by the $\mathcal{O}^{\mathsf{Sign}}$ oracle is replaced by a simulated one. Concretely, whenever the oracle would invoke

$$\pi_{\mathsf{S}} \leftarrow \mathsf{NIZK}_{\mathsf{S}}.\mathsf{Prove}\big(\mathsf{crs}_{\mathsf{S}},\ (msg, \sigma_{\mathsf{Sig}}, ct),\ (c, f(x), \mathsf{pid}, \omega_{\mathsf{PC}}, R)\big),$$

the challenger instead computes

$$\pi_{\mathsf{S}} \leftarrow \mathsf{NIZK}_{\mathsf{S}}.\mathsf{Sim}\big(\mathsf{crs}_{\mathsf{S}}, \mathsf{td}_{\mathsf{S}}, (msg, \sigma_{\mathsf{Sig}}, ct)\big),$$

using the simulation trapdoor $\mathsf{td_S}$, which the challenger holds because in the non-GM unlinkability experiment the master keys—and in particular $\mathsf{crs_S}$—are generated honestly by the experiment itself. The polynomial-commitment opening ω_{PC} is no longer needed and the corresponding computation is skipped.

Games G_0 and G_1 differ only in the distribution of the $\mathsf{NIZK_S}$ proofs in the $\mathcal{O}^{\mathsf{Sign}}$ outputs. Hence,

$$\Pr[G_0(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] \leqslant \mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK_S}}(\lambda). \tag{15}$$

G_2: In this game, every PKE ciphertext produced by the $\mathcal{O}^{\mathsf{Sign}}$ oracle is replaced by an encryption of a fixed dummy plaintext m_0 from the message space of PKE. Concretely, whenever the oracle would invoke $(ct; R) \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, (f(x), \mathsf{pid}))$, the challenger instead computes $(ct; R) \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, m_0)$.

We construct an adversary $\mathcal{B}$ against the INDCPA security of PKE. $\mathcal{B}$ receives $\mathsf{pk_{PKE}}$ from its own challenger, generates the remaining master keys honestly, and simulates G_1 for $\mathcal{A}$. For each $\mathcal{O}^{\mathsf{Sign}}$ query, $\mathcal{B}$ submits the pair $\big((f(x), \mathsf{pid}),\, m_0\big)$ to its left-or-right encryption oracle and uses the returned ciphertext in the signature. Due to the modification introduced in G_1, the $\mathsf{NIZK_S}$ proof is simulated and does not depend on the plaintext underlying ct, so the simulation is consistent; no decryption oracle is ever required. If the challenger's bit is 0, $\mathcal{B}$ perfectly simulates G_1, and if it is 1, G_2. Hence,

$$\Pr[G_1(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] \leqslant \mathsf{negl}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(\lambda). \tag{16}$$

G_3: In this game, the challenger replaces, for each target user $i \in \{0, 1\}$, the pseudo-identity pid_i in the user's secret key sk_i with an independently generated dummy pseudo-identity $\widetilde{\mathsf{pid}}_i$: it samples a fresh polynomial, commits to it, samples a fresh independent signing key pair, and issues a blind signature on the resulting pair using $\mathsf{msk} = \mathsf{sk_{BS}}$. After the modifications of G_1 and G_2, the $\mathcal{O}^{\mathsf{Sign}}$ oracle no longer reads pid_i at all—the proof π_{S} is simulated and the ciphertext ct encrypts a fixed dummy plaintext—and the pseudo-identity is never otherwise exposed, since the adversary obtains neither sk_i nor any transcript of the target users' KeyGen executions. The adversary's view is therefore identical in G_2 and G_3:

$$\Pr[G_2(\mathcal{A}, 1^\lambda) \to 1] = \Pr[G_3(\mathcal{A}, 1^\lambda) \to 1].$$

G_4: In this game, the challenger answers every $\mathcal{O}^{\mathsf{Sign}}$ query with slot index $i = 1$ by computing the conventional signature σ_{Sig} under the signing key $\mathsf{sk}_{\mathsf{Sig},1}$ of the second target user, while queries with slot index $i = 0$ are still answered under $\mathsf{sk}_{\mathsf{Sig},0}$. All remaining components of the signature are produced as in G_3 (the ciphertext ct encrypts the dummy plaintext, and π_{S} is simulated).

We construct an adversary $\mathcal{B}$ against the key anonymity (kAnonSig) of Sig. $\mathcal{B}$ generates the master keys $(\mathsf{mpk}, \mathsf{msk}, \mathsf{mok})$ honestly itself and registers the two target users, forming their secret keys with independently generated dummy pseudo-identities exactly as in G_3; in particular, $\mathcal{B}$ does not need the actual signing keys of the target users. It hands $(\mathsf{mpk}, \mathsf{id}_0, \mathsf{id}_1)$ to $\mathcal{A}$ and answers the $\mathcal{O}^{\mathsf{KeyGen}}$ and $\mathcal{O}^{\mathsf{HonestKG}}$ queries honestly, which it can do since it holds msk. To answer an $\mathcal{O}^{\mathsf{Sign}}$ query (i, msg), $\mathcal{B}$ runs the signing procedure as in G_3, except that the conventional signature is obtained from $\mathcal{B}$'s own kAnonSig signing oracles: $\mathcal{B}$ queries $\mathcal{O}_0^{\mathsf{Sign}}$ if $i = 0$ and $\mathcal{O}_1^{\mathsf{Sign}}$ if $i = 1$, submitting the messages $c\|msg$ for $c = 0, 1, 2, \ldots$ until the returned signature σ_{Sig} satisfies $\mathsf{M2F}(\sigma_{\mathsf{Sig}}) \neq 0$. Since the experiment forbids querying the same message to $\mathcal{O}^{\mathsf{Sign}}$ twice, all messages msg across $\mathcal{A}$'s queries are distinct, and therefore no string $c\|msg$ is ever submitted to both $\mathcal{O}_0^{\mathsf{Sign}}$ and $\mathcal{O}_1^{\mathsf{Sign}}$; $\mathcal{B}$ thus respects the restriction of the kAnonSig experiment. Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

By construction, $\mathcal{O}_0^{\mathsf{Sign}}$ always signs under $\mathsf{sk}_{\mathsf{Sig},0}$, while $\mathcal{O}_1^{\mathsf{Sign}}$ signs under $\mathsf{sk}_{\mathsf{Sig},\beta}$ for the hidden bit β of the kAnonSig experiment. When $\beta = 0$, all conventional signatures are produced under $\mathsf{sk}_{\mathsf{Sig},0}$, so $\mathcal{B}$ perfectly simulates G_3; when $\beta = 1$, slot-1 queries are answered under $\mathsf{sk}_{\mathsf{Sig},1}$, so $\mathcal{B}$ perfectly simulates G_4. Hence

$$\Pr[G_3(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] \leqslant \mathsf{negl}^{\mathrm{kAnonSig}}_{\mathsf{Sig}}(\lambda). \tag{17}$$

G_5: In this game, the dummy pseudo-identities introduced in G_3 are reverted: for each target user $i \in \{0,1\}$, the challenger restores the genuine pseudo-identity $\mathsf{pid}_i = ((com_i, \mathsf{pk}_{\mathsf{Sig},i}), \sigma_{\mathsf{BS},i})$, where com_i commits to the user's real polynomial f_i and $\mathsf{pk}_{\mathsf{Sig},i}$ is the public key corresponding to $\mathsf{sk}_{\mathsf{Sig},i}$. As argued in G_3, the pseudo-identities are not observable by the adversary once G_1 and G_2 are in force, so this change does not affect the adversary's view:

$$\Pr[G_4(\mathcal{A}, 1^\lambda) \to 1] = \Pr[G_5(\mathcal{A}, 1^\lambda) \to 1].$$

G_6: In this game, the modification of G_2 is reverted: every PKE ciphertext produced by $\mathcal{O}^{\mathsf{Sign}}$ is again a genuine encryption of $(f_i(x), \mathsf{pid}_i)$, where i is the index of the user actually answering the query—namely $i = 0$ for slot-0 queries and $i = 1$ for slot-1 queries. By the same reduction as in the transition from G_1 to G_2, applied in the reverse direction,

$$\Pr[G_5(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_6(\mathcal{A}, 1^\lambda) \to 1] \leqslant \mathsf{negl}^{\mathrm{INDCPA}}_{\mathsf{PKE}}(\lambda). \tag{18}$$

G_7: In this game, the modification of G_1 is reverted: every $\mathsf{NIZK}_{\mathsf{S}}$ proof produced by $\mathcal{O}^{\mathsf{Sign}}$ is again computed with the real prover. After the reverts of G_5 and G_6, each statement $(msg, \sigma_{\mathsf{Sig}}, ct)$ is a true statement with the genuine witness $(c, f_i(x), \mathsf{pid}_i, \omega_{\mathsf{PC}}, R)$ available to the challenger, so the real prover applies. By the same hybrid argument as in the transition from G_0 to G_1,

$$\Pr[G_6(\mathcal{A}, 1^\lambda) \to 1] - \Pr[G_7(\mathcal{A}, 1^\lambda) \to 1] \leqslant q_{\mathsf{Sign}} \cdot \mathsf{negl}^{\mathrm{ZK}}_{\mathsf{NIZK}_{\mathsf{S}}}(\lambda). \tag{19}$$

At this point every modification has been reverted, and slot-i queries are answered with sk_i for $i \in \{0,1\}$; hence G_7 is exactly the experiment $\mathbf{Exp}^{\mathrm{nGMUnlink}}_{\Pi}(1^\lambda, \mathcal{A})$ with the challenge bit fixed to $b = 1$.

Finally, combining the bounds (15)–(19), we conclude the proof of the theorem. □

B Security Proofs of k-ASPCGS Generic Transform

Theorem 12 *Let $\mathcal{A}$ be an adversary against the traceability of a k-ASPCGS scheme $\Pi[k, n, \mathcal{M}, \mathcal{ID} = \mathbb{Z}_p]$. Let NIZK be a knowledge-sound NIZK scheme, let SPCE be a correct set pre-constrained encryption scheme, and let kAGS be a traceable k-anonymous group signature scheme. Then there exists negligible functions $\mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}}(\lambda)$, $\mathsf{negl}^{\mathrm{corr}}_{\mathsf{SPCE}}(\lambda)$, and $\mathsf{negl}^{\mathsf{Trace}}_{\mathsf{kAGS}}(\lambda)$ such that*

$$\mathsf{Adv}^{\mathrm{Trace}}_{\Pi}(\mathcal{A}, \lambda) \leqslant \mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}}(\lambda) + \mathsf{negl}^{\mathrm{corr}}_{\mathsf{SPCE}}(\lambda) + \mathsf{negl}^{\mathsf{Trace}}_{\mathsf{kAGS}}(\lambda).$$

Proof. (Sketch) We construct a sequence of indistinguishable games G_0, G_1, G_2, where G_0 corresponds exactly to the experiment $\mathbf{Exp}^{\mathrm{Trace}}_{\Pi}(1^\lambda, \mathcal{A})$, and G_2 is equivalent to the traceability game for the underlying kAGS scheme.

G_1: By the knowledge-soundness property of NIZK, for the adversary $\mathcal{A}$ there exists a knowledge extractor $\mathcal{E}_{\mathcal{A}}$. In G_1, the challenger uses $\mathcal{E}_{\mathcal{A}}$ to modify the experiment as follows.

After $\mathcal{A}$ outputs challenge signatures, for each valid message-signature pair $(msg, \sigma) \in \{(msg_i, \sigma_i), i = 1, \ldots, s\}$, i.e., such that $\mathsf{NIZK.Verify}(\mathsf{crs}, (msg, ct), \pi) = 1$ where $\sigma = (ct, \pi)$, the challenger uses $\mathcal{E}_{\mathcal{A}}$ to obtain a corresponding witness $w = (\sigma_{\mathsf{kAGS}}, R)$ and checks whether $\mathcal{R}^{kASPCGS}_{k, \mathsf{pk}_{\mathsf{SPCE}}, \mathsf{mpk}_{\mathsf{kAGS}}}((msg, ct), w) = 1$. If this check fails, the challenger aborts (outputting 0). We denote that failure event by fail_{01}.

If the game does not abort, it holds that σ_{kAGS} is a correct kAGS signature of msg, and ct is a correctly computed SPCE encryption of that signature with respect to msg. The challenger records all extracted signatures in a list.

We now construct an adversary $\mathcal{B}$ against the knowledge-soundness of NIZK. $\mathcal{B}$ perfectly simulates G_1 for $\mathcal{A}$, using the NIZK parameters received from its own challenger. If the failure event occurs, $\mathcal{B}$ outputs $((msg, ct), \pi)$ as its solution in the knowledge-soundness game and wins. Otherwise, $\mathcal{B}$ outputs random data.

Hence, the probability that $\mathcal{B}$ wins the knowledge-soundness game is at least the probability of the failure event in G_1.

Since G_0 and G_1 behave identically unless the failure event occurs and NIZK is knowledge-sound, we have

$$\begin{aligned}\Pr[G_0(\mathcal{A},1^\lambda)\to 1] &\leqslant \Pr[G_1(\mathcal{A},1^\lambda)\to 1] + \Pr[\mathsf{fail}_{01} \textit{ in } G_1(\mathcal{A},1^\lambda)]\\ &\leqslant \Pr[G_1(\mathcal{A},1^\lambda)\to 1] + \mathsf{Adv}^{\mathrm{knSound}}_{\mathsf{NIZK}}(\mathcal{B},\lambda) \leqslant \Pr[G_1(\mathcal{A},1^\lambda)\to 1] + \mathsf{negl}^{\mathrm{knSound}}_{\mathsf{NIZK}}(\lambda). \quad (20)\end{aligned}$$

G_2: In this game, when running the Trace algorithm, the challenger checks each signature $\sigma_i = (ct, \pi)$ as follows:

- If the corresponding message msg_i is in D, the challenger checks whether an SPCE decryption of ct is equal to a previously extracted kAGS signature.
- If $msg_i \notin D$, the challenger checks whether an SPCE decryption of ct is equal to $\perp$ or to a previously extracted kAGS signature.

If this check fails, the challenger aborts (outputting 0). We denote that failure event by fail_{12}.

Note that when the Trace algorithm is called, it has already been checked that the number of signed messages in D exceeds $k \cdot |\mathcal{L}_{\mathsf{id}}|$. Thus, if G_2 does not abort on the checks introduced above, then during Trace, after decrypting ciphertexts contained in $\{\sigma_1, \ldots, \sigma_s\}$, the challenger obtains more than $k \cdot |\mathcal{L}_{\mathsf{id}}|$ correct kAGS signatures.

Due to the changes introduced in G_1, every ciphertext ct in each signature σ_i is a correctly computed SPCE encryption of an extracted kAGS signature with respect to msg. Hence, the failure event fail_{12} corresponds exactly to a violation of SPCE correctness.

Since G_1 and G_2 behave identically unless the failure event occurs, we have

$$\begin{aligned}\Pr[G_1(\mathcal{A},1^\lambda)\to 1] &\leqslant \Pr[G_2(\mathcal{A},1^\lambda)\to 1] + \Pr[\mathsf{fail}_{12} \text{ in } G_2(\mathcal{A},1^\lambda)]\\ &\leqslant \Pr[G_2(\mathcal{A},1^\lambda)\to 1] + \mathsf{negl}^{\mathrm{corr}}_{\mathsf{SPCE}}(\lambda). \quad (21)\end{aligned}$$

To bound $\Pr[G_2(\mathcal{A},1^\lambda)\to 1]$, we construct an adversary $\mathcal{B}$ against the traceability of kAGS. $\mathcal{B}$ receives $(\mathsf{mpk}_{\mathsf{kAGS}}, \mathsf{mok}_{\mathsf{kAGS}})$ from its kAGS challenger; it generates $(\mathsf{pk}_{\mathsf{SPCE}}, \mathsf{sk}_{\mathsf{SPCE}}) \leftarrow \mathsf{SPCE.Gen}(1^\lambda, \mathcal{D})$ and $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}^{kASPCGS}_{k,\mathsf{pk}_{\mathsf{SPCE}},\mathsf{mpk}_{\mathsf{kAGS}}})$ itself, and assembles the $kASPCGS$ master public key $\mathsf{mpk} = (\mathsf{pk}_{\mathsf{SPCE}}, \mathsf{mpk}_{\mathsf{kAGS}}, \mathsf{crs})$ and the master opening key $\mathsf{mok} = (\mathsf{sk}_{\mathsf{SPCE}}, \mathsf{mok}_{\mathsf{kAGS}})$ which it provides to $\mathcal{A}$. $\mathcal{B}$ forwards $\mathcal{A}$'s $\mathcal{O}^{\mathsf{KeyGen}}$ and $\mathcal{O}^{\mathsf{HonestKG}}$ queries to its own oracle without modification, since the key-generation protocol of Π is exactly that of kAGS. Thus, the number of users $\mathcal{B}$ registers with its challenger is exactly the number $\mathcal{A}$ registers. To answer a query to $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{B}$ forwards it to its own oracle, and then SPCE encrypts the received signature with respect to the queried message and computes a corresponding proof.

When $\mathcal{A}$ outputs the challenge signatures, $\mathcal{B}$ runs the checks from G_1 (aborting if any check fails), then compiles a list of kAGS signatures by SPCE-decrypting the ciphertexts contained in $\mathcal{A}$'s signatures. It also runs the additional decryption checks from G_2. $\mathcal{B}$ outputs the resulting list of kAGS signatures to its challenger. As argued, all these signatures are valid, and there are at least $k \cdot |\mathcal{L}_{\mathsf{id}}|$ of them.

$\mathcal{A}$ wins G_2 if and only if $\mathsf{kAGS.Trace}$ does not output identity of any registered user when run on the decrypted signatures. Likewise, $\mathcal{B}$ wins its own game if and only if $\mathsf{kAGS.Trace}$ does not output any registered user when run on its output. Since $\mathcal{B}$'s output is exactly the decrypted signatures, $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. Therefore, by the traceability of kAGS, we have

$$\Pr[G_2(\mathcal{A},1^\lambda)\to 1] = \mathbf{Exp}^{\mathrm{Trace}}_{\mathsf{kAGS}}(1^\lambda, \mathcal{B}) \leqslant \mathsf{negl}^{\mathsf{Trace}}_{\mathsf{kAGS}}(\lambda). \quad (22)$$

Finally, by combining inequalities (20)–(22), we conclude the proof. □