Laconic PSI on Authenticated Inputs and Applications

James Bartusek
NYU

Sanjam Garg*
UC Berkeley

Abhishek Jain†
NTT Research and JHU

Guru-Vamsi Policharla
UC Berkeley

Abstract

A common issue with using secure computation in practice is that its security does not place any restrictions on what an adversary can use as input in the protocol. In this work, we focus on the practically-motivated setting of (two-message, labeled) *private set intersection* (PSI), and advocate for a clean and versatile solution to this problem: PSI on authenticated inputs.

Our central contributions are summarized as follows.

- We formulate a novel definition of PSI on authenticated inputs that has the potential for use in several applications, from content moderation in end-to-end encrypted systems to watchlists in anonymous e-cash systems.
- We design a *concretely-efficient* and *laconic* (i.e. the size of the receiver's message is independent of its set size) protocol for PSI on authenticated inputs.
- We build on our PSI protocol to obtain the first laconic *set pre-constrained group signature* scheme, improving on that of Bartusek et al. (Eurocrypt 23).

We also explore various optimizations to our basic protocol, including reducing the receiver's concrete run time, and a tradeoff between crs size and message size.

1 Introduction

Secure computation is a powerful privacy-preserving technology, allowing mutually distrusting parties to compute any joint function on their private inputs without leaking anything other than the output. Once the feasibility of general-purpose secure computation was established in the 80s, researchers have looked to make real-world impact by designing concretely-efficient protocols for application-specific functionalities. One particularly simple but still highly versatile functionality is *private set intersection*, which allows two parties holding private sets to compute the intersection of these sets. Indeed, private set intersection has found several applications including private contact discovery [MPGP09], testing of sequenced human genomes [BBD+11], proximity testing [NTL+11] and botnet detection [NMH+10], among others. Researchers have accompanied this with an extensive line of work constructing PSI protocols (e.g. [FNP04, KS05, PSZ14, KKRT16, RR17]).

Simulation-based security. When proposing a protocol for secure computation, the gold standard notion of security demands that the protocol leak nothing more to an adversary than what they could obtain in an idealized environment. To be concrete, suppose that one party (let's say the server) holds a database D of items, the other party (let's say the client) holds an item x and a payload m, and the parties interact

*Supported in part by the AFOSR Award FA9550-24-1-0156 and research grants from the Bakar Fund, J. P. Morgan Faculty Research Award, Supra Inc., Sui Foundation, and the Stellar Development Foundation.

†Supported in part by NSF CAREER award (1942789), JP Morgan Faculty award, and research grants from Stellar Development Foundation, Cisco and Ethereum Foundation.

in a protocol that is meant to deliver m to the server but only if $x \in D$. Simulation-based security stipulates that an adversary corrupting the server cannot learn anything more than a "simulator" who only has access to an *ideal functionality* that takes the database D from the server, the pair (x, m) from the client, and delivers m to the server if $x \in D$.

Its shortcomings. Simulation-based security is a beautiful paradigm that has played a large part in the success of secure computation as a cryptographic primitive. However, it only goes so far. Indeed, a common concern with secure computation is that it only ensures security of the *computation itself*, and does not provide any guarantees on the *inputs* chosen by the parties.

A prominent example of this shortcoming can be found in the recent "Apple PSI" saga. In 2021, researchers at Apple proposed a system for privacy-preserving content moderation to be integrated into their cloud storage system [BBM+21]. In particular, they wanted to detect the presence of child sexual abuse material (CSAM) on their servers without having to decrypt everyone's uploaded images, and they proposed a PSI protocol that only reveals images if they are contained in a database of CSAM images held by Apple. While they initially planned to release the protocol in a software update, it immediately received widespread criticism [fDT21]. The issue was that nothing about the protocol prevented Apple from using *any* input of their choice for the database of images. In particular, instead of using a database of CSAM images, Apple (or any server running Apple's protocol) could insert, say, politically-motivated images in an attempt at censorship. This is precisely the issue that secure computation generally fails to address.

The backlash was so severe that the proposal was ultimately dropped. However, the tension between users' desire for privacy and law enforcement's desire for content moderation continues to play out in real time. For example, the U.K. government recently issued an order that would allow them access to *all* of Apple's cloud backups, as reported by the Washington Post in February 2025 [Pos25]. Thus, it is still a pressing problem to devise ways to balance this tension, preserving as much privacy as possible in the face of potentially drastic legal changes.

In order to make the original proposed solution more palatable, researchers have been motivated to find ways to build in protections against service providers that may try to maliciously choose their *input* to a protocol, both in the setting of content moderation, and more generally. This includes proposals like "pre-constrained cryptography" [AJJM22, AKN24], in particular with "authenticated set" security [BGJP23], and "privacy-preserving blueprints" [KLN23], which enables secure computation on inputs that are committed elsewhere. In this work, we focus on the case of PSI, and propose a conceptually simple but powerful approach to addressing the issue of input integrity.

1.1 This work

PSI on authenticated inputs. We study the notion of PSI on authenticated inputs, which was introduced originally by Camenisch and Zaverucha [CZ09]. Motivated by several recent works with similar structural requirements [BBM+21, KM21, KLN23, BGJP23], we focus on *two-message* (labeled) PSI, where a receiver encodes a database D into a first message, and then a sender encodes an item x and payload m into a second message.[1] The receiver should then only be able to recover m if $x \in D$. We consider both *element-hiding*, where the receiver only recovers (x, m) if $x \in D$, and *non-element-hiding*, where the sender's message may always reveal x (but still hide m if $x \notin D$). Crucially for the applications we have in mind, we demand that the size of the receiver's message encoding D is much smaller than D itself, that is, the receiver's communication is *laconic*.

But what exactly is meant by authenticated inputs? In a nutshell, the scheme will come parameterized by some verification predicate Ver, and guarantees that the receiver should only be able to recover m if

[1] The sender can input a set by sending multiple messages simultaneously, so we focus on the single-item case for simplicity.

$x \in D$ AND the receiver knows some "witness" σ_x such that $\mathsf{Ver}(x, \sigma_x) = 1$.[2] A typical choice for Ver is the verification procedure for some signature scheme (with the public verification key hard-coded). That is, suppose that a third party Auth samples a signature key pair and publishes the verification key, which defines Ver. Now, running PSI on authenticated inputs with respect to Ver means, intuitively, that the receiver is restricted to using *only* items x such that Auth has issued a signature on x. Thus, this yields a straightforward method for "outsourcing" the trust assumption to a third party. In the case study discussed above, this means that clients would not have to trust Apple (the service provider) to choose "honest" inputs. Instead, this could be outsourced to a trusted third party such as the National Center for Missing and Exploited Children (NCMEC), or even a *combination of* several third parties, as Ver could be defined to only accept given multiple different signatures from different authorities.

A flexible definition. We capture this intuitive requirement with a clean simulation-based security definition that guarantees the integrity of inputs while providing a high level of flexibility on the part of the protocol designer. In particular, we demand that a PSI on authenticated inputs with respect to Ver securely emulates the following ideal functionality.

- Upon receiving a message D from the receiver, record D.
- Upon receiving a message (x, m) from the sender, record (x, m). (In the non-element-hiding case, send x to the receiver.)
- Upon receiving a message (x, σ_x) from the receiver, check that $x \in D$ and that $\mathsf{Ver}(x, \sigma_x) = 1$. If so, send (x, m) to the receiver.

At this point, we would like to stress the fact that the signature σ_x is not required until the *very end of the protocol*, when the receiver attempts to decrypt a sender's message. That is, the receiver can produce the first message of protocol encoding some database D even before knowing signatures on any of the items. Then, if the third party later issues a signature on $x \in D$, the receiver will now be able to decrypt any past (and future) messages encrypted with respect to x. However, we remark that the receiver's first message is still a (computationally) binding commitment to some D (of size at most some upper bound n parameterizing the protocol). This design choice has several important implications:

- **Flexibility.** First, it provides a desirable level of flexibility for the third party and receiver, as outlined above. Concretely, consider the application of watchlists in private payment systems [KLN23].[3] Here, an auditor can quickly insert a suspicious address into their watchlist D, before they have a signature on it. Importantly, this does not yet allow them to de-anonymize transactions involving this address. But if they are able to successfully obtain a signature (effectively a formal sanction) on this address from the relevant authority, they can then go back and de-anonymize all transactions since they updated their input, even if the signature arrives after the fact.
- **Efficiency.** Next, as we discuss further in the technical overview, it is a boon for concrete efficiency, as we do not need to incorporate signatures into the receiver's first message (along with potentially expensive zero-knowledge proofs).
- **Distinguisher-Dependent Simulation.** Finally, proving security with respect to this ideal functionality actually requires a slight weakening of traditional simulation-based security. To see why, note that the simulator will have to extract any signature that the adversary uses to decrypt any of the messages. However, the adversary may simply forward along some signatures to the *distinguisher*, who then uses them to decrypt messages. Thus, to enable extraction, we necessarily must

[2] Our definition is also able to incorporate authentication for sender inputs, but we focus on the receiver's side for this overview.
[3] Watchlists can be thought of as a list of addresses that lose privacy for their transactions owing to suspicious behavior.

allow for the possibility of *distinguisher-dependent* simulation, a concept that was originally introduced in the context of zero-knowledge [DNRS99]. Conceptually, this still captures the same desired security properties as the standard notion of adversary-dependent simulation, at least in the standalone setting. The only setting where we see a real difference is in protocol composition, which we do not address in this work. Finally, it turns out that proving security via distinguisher-dependent simulation is subtle and technically challenging, and we defer further discussion of this point to the technical overview.

Relationship with prior work on authenticated inputs. The earliest formalization of Private Set Intersection on *certified sets* was by Camenisch and Zaverucha [CZ09] where the ideal functionality explicitly models an authority who specifies the inputs of each user. To the best of our knowledge, follow-up works [DJKT09, DT10, DKT10, SSS12, DD15, FM24] improve the efficiency of PSI on authenticated inputs but still demand that the inputs are authenticated at the beginning the protocol, i.e. the party's first-round message must already contain a signature on its input (and the protocol aborts if this is not the case). Thus, our *delayed-signature* setting and its various implications are novel to this work. Moreover, we are the first to achieve a notion of *laconic* secure computation on authenticated inputs. Finally, we note that some other works such as [DGPS24] have extended the notion of PSI on authenticated inputs to general-purpose MPC, but still with the requirement that signatures must be known at the time of generating first-round messages.

Relationship with authenticated-set security. PSI on authenticated inputs provides an insightful lens through which to view recent works on secure computation with input integrity. For example, [BGJP23] recently proposed an improvement to the Apple PSI system that allows the clients to verify that a third party has validated the items in the server's database. Their notion of "authenticated-set" security is conceptually similar to our notion of PSI on authenticated input, but falls short of achieving it. In particular, authenticated-set security allows for the possibility of an *interaction* between the service provider and third party in order to produce the proof of well-formed inputs, and indeed the concretely efficient protocol proposed by [BGJP23] utilizes this interaction. On the other had, PSI on authenticated inputs enables the third party to simply send over signatures to the service provider (at any point in time), significantly reducing the burden on the third party.

1.2 Results

Our main results are, to the best of our knowledge, the first PSI protocols that combine the desirable features of (1) a receiver's message that does not depend on the database size, (2) authentication of inputs, and (3) concrete efficiency. Building upon the techniques of [KZG10, ALOS22, FHAS24, CGPP24], we achieve the following.

Theorem 1 (Informal)**.** *There exists a concretely-efficient* ***non-element-hiding*** *PSI on authenticated inputs with respect to the predicate that checks BLS [BLS01] signatures. The size of the receiver's message, the time of the sender's encryption algorithm, and the time of the receiver's decryption algorithm are all independent of the size of their database D.*

Theorem 2 (Informal)**.** *There exists a concretely-efficient* ***element-hiding*** *PSI on authenticated inputs with respect to the predicate that checks (basic) BB [BB04] signatures. The size of the receiver's message, and the time of the sender's encryption algorithm, are independent of the size of their database D but the time of the receiver's decryption algorithm is $O(D)$.*

Next, we discuss two classes of applications where our techniques and protocols are particularly useful: (1) content moderation in privacy-preserving systems, and (2) watchlists in e-cash systems. Table 1

Scheme	$\lvert msg_R \rvert$	$\lvert msg_S \rvert$	Sen	Dec	$\lvert crs \rvert$	Element Hiding
[KLN23]	$O(D)$	$O(D)$	$O(D)$	$O(D)$	-	✓
[BGJP23]	$O(D)$	$O(1)$	$O(1)$	$O(1)$	-	✓
[GKLS24]	$O(D)$	$O(\log D)$	$O(D)$	$O(D)$	-	✓
This Work	$O(1)$	$O(1)$	$O(1)$	$O(1)/O(D)$	$O(D)$	✗/✓
This Work	$O(\sqrt{D})$	$O(1)$	$O(1)$	$O(1)/O(D)$	$O(\sqrt{D})$	✗/✓

Table 1: Comparison with prior two-message PSI protocols with input integrity guarantees. Here, $\lvert msg_R \rvert$ refers to the size of the receiver's message, msg_S refers to the size of sender's message, Sen is the time required to compute the sender's message, Dec is the time required for the receiver to decrypt the sender's message, and $\lvert crs \rvert$ is the size of the trusted setup system parameters, where "-" denotes a *transparent* setup such as sampling uniformly random group elements, which can be carried out in a transparent manner using a random oracle. In order to emphasize dependence on the database size, we suppress factors that depend only on the security parameter (such as field size).

provides a comparison with solutions proposed by prior works that were motivated by these same applications.

Content moderation in privacy-preserving systems. As mentioned earlier, one of the motivations for this line of work stems from Apple's proposal for utilizing PSI to build content-moderation into its encrypted cloud storage service. The original approach of [BBM+21] was doomed due to its lack of input authentication, which is the central security flaw addressed by our work.

We further provide an application of our techniques to content moderation in *end-to-end encrypted messaging* services. Prior work [BGJP23] put forth the notion of a *set pre-constrained group signature* (SPCGS), which, roughly, satisfies the following properties.

- The group manager publishes a master public key that encodes some database D of messages.
- Clients can register with the group manager to receive a public identity and secret signing key.
- Clients sign messages m with respect to their identity id. The resulting signature hides the identity, even from the group manager, unless $m \in D$, in which case the group manager can de-anonymize the client.

They envisioned using such a group signature scheme in tandem with an end-to-end encrypted messaging system in order to provide a method of content-tracing. That is, all messages originated in system are encrypted under end-to-end encryption along with a group signature on the message. If a user receives a harmful message, they can report it to the system provider / group manager, who is then able to de-anonymize the originator of the message m if indeed $m \in D$. Moreover, they require security against potentially malicious service providers that may try to insert content of their choice into the database.

Following [BGJP23], we describe a generic construction of SPCGS from PSI on authenticated inputs that ports over the strong protections against malicious service providers guaranteed by the PSI security. In particular, our SPCGS securely emulates a functionality $\mathcal{F}_{\text{Anon}}$ (Fig. 7) which guarantees that the service provider can only de-anonymize users who originate content that exists in their database AND has been

authorized by a trusted third party. This improves upon [BGJP23]'s construction, which inherited their potentially interactive authenticated-set style of definition. Finally, we show that by utilizing the right building blocks, we can obtain a *concretely-efficient* instantiation of SPCGS, based on our concretely-efficient PSI on authenticated inputs, Groth-Sahai proofs [GS08], and structure-preserving signatures [Gha17]. This is captured in the following informal theorem - please see Section 6 for the formal SPCGS definitions.

Theorem 3 (Informal)**.** *There exists a concretely-efficient set-preconstrained group signature scheme that securely emulates the functionality $\mathcal{F}_{\mathsf{Anon}}$, in addition to satisfying Traceability, Unframeability, and Client-client anonymity.*

The SPCGS scheme of [BGJP23] required a master public key that grew *linearly* with the size of the database D. Our scheme, in contrast, can achieve *constant-size* or $O(\sqrt{D})$-size master public key, depending upon our PSI protocol.

Watchlists in anonymous e-cash systems. Another natural application of these techniques comes from the world of privacy-preserving *payment* systems, or, anonymous e-cash. Here, the worry is that the anonymity of the system may be abused by criminals engaged in illicit activities. As discussed by [KLN23], one way to mitigate this threat is to incorporate a "watchlist" of users that are suspected of financial crimes, maintained by an external judge. The idea is that all transactions remains encrypted, except for those issued by an ID that is on the watchlist.

Phrased in our language of (labeled) PSI, the database would consist of the IDs on the watchlist, the item would be the ID, and the payload would be the description of the transaction. Here, it is crucial that the PSI satisfies *element-hiding*, as we want all IDs to remain private except for those that match the watchlist. Moreover, authenticating the watchlist is a necessary prerequisite for such systems to be used in practice in order to prevent the e-cash service provider from potentially de-anonymizing any set of IDs of their choice. Thus, our element-hiding PSI with authenticated inputs is a perfect fit for this application.

In related work motivated by the same application, [KLN23] proposed the notion of a "privacy-preserving blueprint", which has continued to be studied in followup works [DEF+24, GKLS24]. At a high level, privacy-preserving blueprints build integrity into secure computation protocols by demanding that the inputs used by the parties are *also* committed inside an additional commitment scheme. That is, parties cannot use inputs that differ from the committed values present in some externally-provided commitment scheme. The focus in this line of work has also been on the two-message PSI setting, where the service provider inputs the database of suspected criminals, and the client produces a transaction with respect to their identity, such that their identity is only leaked to the server if it is on the watchlist.

Intuitively, our notion of PSI on authenticated inputs captures this notion of privacy-preserving blueprint as a special case, by defining the Ver predicate to check a zero-knowledge proof that the protocol input is the same as the committed input. Thus, we expect the clean and general framework offered by PSI on authenticated inputs to be useful in future work in the area. Moreover, the asymptotic complexity of our protocols compare favorably with the prior works [KLN23, GKLS24], as indicated in Table 1.

2 Technical overview

In this overview, we focus on our construction of non-element-hiding laconic PSI on authenticated inputs, as well as the techniques required to show security. We then briefly cover our extensions to (1) optimized concrete efficiency via cuckoo hashing, (2) element-hiding PSI and (3) set pre-constrained group signatures.

2.1 KZG Witness Encryption

We first recall the KZG polynomial commitment scheme [KZG10]. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, and g, h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The scheme relies on a common reference string

(CRS) of the form $\mathsf{crs} = (g, g^{\tau}, \ldots, g^{\tau^n})$, where n is the maximum degree of the polynomial that one wants to commit to. A commitment to a polynomial $f(X)$ is the group element $\mathsf{com} = g^{f(\tau)}$. The opening proof to convince a verifier that the underlying polynomial $f(X)$ evaluates to y^* on x^* is the group element $\pi = g^{q(\tau)}$, where $q(X) = (f(X) - y^*)/(X - x^*)$. The proof is deemed to be valid if

$$e(\mathsf{com}, h) = e(\pi, h^{\tau - x^*}) \cdot e(g^{y^*}, h).$$

The KZG commitment scheme admits an efficient witness encryption to the following relation: "I know an opening proof π of com at (some special point) x^* such that $p(x^*) = y^*$", where (com, x^*, y^*) form the statement and π is the witness [FHAS24]. This can be seen as a special case of a general compiler to transform arbitrary pairing product equations (that have *linear* verification) to a witness encryption [BC16, GKPW24]. Concretely, to encrypt a message $m \in \mathbb{G}_T$ to the statement (com, x^*, y^*), we first sample $\rho \leftarrow\!\!\$\ \mathbb{F}$ and set the ciphertext to be

$$\mathsf{ct} = (h^{(\tau - x^*)\rho}, e(\mathsf{com} \cdot g^{-y^*}, h)^{\rho} \cdot m).$$

Given a witness π, the message can be recovered as $m \leftarrow \mathsf{ct}_2 \cdot e(\pi, \mathsf{ct}_1)^{-1}$. This general idea has been used in several recent works (e.g. [ALOS22, FHAS24, CGPP24, CGPW24]) to build advanced primitives such as laconic OT/PSI and *batched* threshold encryption.

2.2 Boneh-Franklin IBE

We now recall the Boneh-Franklin Identity-Based Encryption (IBE) scheme [BF03] through the lens of applying the witness encryption compiler from [BC16, GKPW24] to the BLS signature scheme [BLS01]. Recall that to verify a BLS signature σ on a message x under a verification key $\mathsf{vk} = h^{\mathsf{sk}}$, we check that

$$e(H(x), \mathsf{vk}) = e(\sigma, h).$$

Again, the BLS signature scheme admits an efficient witness encryption to the following relation: "I know a signature σ on the message x under the verification key vk", where (x, vk) form the statement and σ is the witness. By applying the compiler from [BC16, GKPW24] we recover the Boneh-Franklin IBE scheme, where vk is the master public key, the public key of an identity x is $H(x)$ and $\sigma = H(x)^{\mathsf{sk}}$ is the secret key for x. Concretely, to encrypt a message $m \in \mathbb{G}_T$, to the statement (vk, x), we first sample $\rho \leftarrow\!\!\$\ \mathbb{F}$ and set the ciphertext to be

$$\mathsf{ct} = (h^{\rho}, e(H(x), \mathsf{vk})^{\rho} \cdot m).$$

Given the witness σ, the message can be recovered as $m \leftarrow \mathsf{ct}_2 \cdot e(\sigma, \mathsf{ct})^{-1}$.

2.3 PSI Protocol

Given the above building blocks, we can now describe our construction of non-element-hiding laconic PSI on authenticated inputs. This serves as a useful application in its own right, as well as the foundation on which our various extensions are built. Here we present a version that supports authenticated inputs only for the receiver, which still conveys the main ideas. The full protocol is presented in Fig. 3.

1. An external entity, which we refer to as the *environment*, publishes the verification key $\mathsf{vk} = h^{\mathsf{sk}}$ for the BLS signature scheme.

2. The receiver, given input database D, interpolates a degree-n polynomial $f(X)$ such that $\{f(D[i]) = 0\}_{i \in [n]}$. They then sample $r \leftarrow\!\!\$\ \mathbb{F}$ and send a *blinded* KZG polynomial commitment $\mathsf{com} = g^{r \cdot f(\tau)}$ to the sender.

3. The sender, given input (x, m), encrypts m to the following relation: "I know an opening proof π of com at the point x to the value 0 *and* a signature σ on x under the verification key vk", where $(\mathsf{com}, x, \mathsf{vk})$ form the statement and (π, σ) are the witnesses that can be used to decrypt the ciphertext. Using the techniques described above, this can be achieved by sampling two random field elements $\alpha, \beta \leftarrow\!\!\$\ \mathbb{F}$, and publishing

$$\mathsf{ct}_1 \leftarrow h^{(\tau - x)\alpha}, \quad \mathsf{ct}_2 \leftarrow h^{\beta}, \quad \mathsf{ct}_3 \leftarrow e(\mathsf{com}, h)^{\alpha} \cdot e(H(x), \mathsf{vk})^{\beta} \cdot m,$$

along with the item x.

4. The receiver, given a signature $\sigma_x = H(x)^{\mathsf{sk}}$, computes the KZG opening proof $\pi = g^{r \cdot q(\tau)}$, where $q(X) = f(X)/(X - x)$, and recovers m as

$$m \leftarrow e(\pi, \mathsf{ct}_1)^{-1} \cdot e(\sigma_x, \mathsf{ct}_2)^{-1} \cdot \mathsf{ct}_3.$$

On malicious security. The protocol as described is semi-maliciously secure against both parties. In particular, there is some choice of randomness and input that "explains" any pair of receiver and sender messages. Boosting to full-fledged malicious security introduces very little overhead. To enable extraction from the receiver, we demand that the receiver's polynomial has a zero constant term. This can be verified by having the receiver include a KZG opening that $f(0) = 0$ in their first message, which adds just a single group element. To enable extraction from the sender, we add two efficient proofs of discrete log with respect to the bases $h^{(\tau - x)}$ and h, from which α and β can be obtained. The full details can be found in Section 5.1.

Delayed-input signatures. As promised in the introduction, the receiver does not need to know the signatures σ_x on $x \in D$ when producing their first-round message. As a consequence, they do not need to include *proofs* that they have signatures on the $x \in D$ along with the first message. Stated another way, the receiver could insert any arbitrary (albeit bounded) set of elements in D, but they will still learn *only* the corresponding payloads for items that have been authenticated via a signature by the environment.

This is a subtle but crucial point, which enable us to avoid expensive zero-knowledge proofs for the receiver's message, and keep the structure of the first message extremely short and simple. However, as we will see in the following section, this is a double-edged sword, introducing additional challenges in the proof of security that must be addressed.

2.4 Challenges in Proving Security

Let's consider proving security against a malicious receiver, recalling the ideal functionality outlined earlier in the introduction. In particular, we must design a simulator that extracts the database D and any signatures σ_x on items $x \in D$ that are used by the adversary. Since the adversarial receiver sends a first-round message based on D, extracting the effective committed database is standard practice for simulation-based security. However, extracting signatures presents a very different problem, as the adversary never sends any message based on which signatures it knows. While this does present a conceptual challenge, here it is useful to recall that we formalize security in the *generic group model* (GGM), where all parties must interact with an external oracle in order to implement group operations. This gives us a ray of hope: Perhaps the simulator can observe the GGM queries made by the adversary in order to extract which signatures are being to used to decrypt which payloads.

Distinguisher-dependent simulation. While the ability to observe GGM queries certainly helps, things are still not so simple. First of all, consider that the adversary may simply forward some set of signatures to the distinguisher (whose job is to distinguish between the real and ideal worlds) without actually using

them to attempt decryption. Given this possibility, we must allow the simulator to *depend on the distinguisher*, which has been considered previously for example in the setting of three-message zero-knowledge [DNRS99, JKKR17]. Furthermore, the adversary / distinguisher may be *randomized*, so in any given run they could utilize a different random set of signatures in order to decrypt different payloads. The simulator must get out ahead of this possibility by extracting all signatures that have some noticeable probability of being used by the distinguisher. This requires a careful simulation strategy that runs the adversary-distinguisher pair many times, while still remaining polynomial-time bounded. Finally, and perhaps most importantly, we still need to devise a mechanism for extracting the signatures themselves from GGM queries.

Extracting signatures. In order to convey one of the main challenges, we outline a toy extraction problem that underlies our eventual simulator's strategy. Consider the following sequence of events:

- The *GGM oracle* is initialized, and all parties receive labels for the group generators. In what follows, all group elements will be represented by labels obtained by querying the GGM.
- The *Environment* publishes a BLS verification key h^{sk}. It also initializes the adversary with some auxiliary information. This could include some arbitrary set of signatures, or even sk itself.
- The *Challenger*, given an item x, two message m_0, m_1, and a bit b, samples a random β and sends the adversary $(\mathrm{ct}_1, \mathrm{ct}_2) := (h^{\beta}, e(H(x), h^{\mathsf{sk}})^{\beta} \cdot m_b)$.
- The *Adversary* makes GGM queries in an attempt to guess b.

Intuitively, the adversary is only able to distinguish $b = 0$ from $b = 1$ if it uses the BLS signature $H(x)^{\mathsf{sk}}$ as part of a GGM pairing query along with ct_1 in order to produce a target group element that can be used to unmask m_b. Thus, we hope to prove the following claim.

Claim 1 (Informal)**.** *There exists an* Extractor *observing the adversary's queries such that the probability that the extractor outputs $H(x)^{\mathsf{sk}}$ is at least the distinguishing advantage of the adversary.*

However, the above intuition is not fully correct. For example, the adversary may have been initialized with sk itself, and never needed to include $H(x)^{\mathsf{sk}}$ in one of its queries. Or perhaps the adversary was initialized with some secret sharing of $H(x)^{\mathsf{sk}}$ which it then reconstructed in the target group.

Despite these issues, we establish that such extraction is indeed possible. We note that somewhat similar extraction claims have been shown previously, for example by [FHAS24] in the setting of extracting KZG openings in the algebraic group model (AGM). However, our setting is actually much more demanding. In particular, [FHAS24] crucially rely upon the fact that, in the AGM, all group elements output by the adversary come along with a representation with respect to the set of group generators. However, this is *not* a reasonable heuristic in our setting. To see why, observe that the adversary is initialized with some arbitrary auxiliary information computed by the environment, which could include arbitrary group elements. The adversary may have no idea how these group elements were prepared. Thus, while observing the adversary's GGM queries, the extractor may see certain "mystery" labels for which they do not know the underlying representation.

Despite this difficulty, we devise a strategy that allows the extractor to still reconstruct a signature in Group 1 using a combination of these mystery labels and known exponents (for details, see Appendix A.2). More generally, one can view our techniques as showing how to obtain extractable witness encryption for BLS signatures, even if the extractor cannot observe GGM queries made by the entity that samples the instance (in contrast, [FHAS24] treated this entity as the first stage of the adversary). We also generalize beyond BLS, and consider these ideas to potentially be of independent interest.

2.5 Extensions

Finally, we briefly discuss several extensions of the basic non-element-hiding protocol described above.

Improving concrete efficiency. Note that for the receiver to be able to decrypt arbitrary sender messages, they have to compute KZG openings for potentially all items $x \in D$. Naively, this takes $O(|D|^2)$ time, though [FK23] showed that it can be done in $O(|D| \log^2 |D|)$ time. This can be further reduced to $O(|D| \log |D|)$ time (with good constants) if the database is encoded as the evaluations of the polynomial on some polynomial-size subgroup of *roots of unity*. That is, we need to define the polynomial f such that for any $i \in |D|$, there exists a j such that $f(\omega^j) = D_i$. Moreover, given some element x, the sender must know which j is such that, if $x \in D$, it holds that $f(\omega^j) = x$.

This appears difficult to achieve exactly, since the space of possible database elements is exponential while the subgroup of evaluation points is polynomial. Thus, we relax correctness and use the technique of *cuckoo hashing* to encode the database. That is, we use hashes h_0, h_1 to define a table T of size $(1+\varepsilon)|D|$ for some constant ε, with the guarantee that for *most* items $x \in D$, either $T_{h_0(x)} = x$ or $T_{h_1(x)} = x$. The receiver now defines the polynomial f such that $f(\omega^i) = T_i$ for $i \in (1+\varepsilon)|D|$. For full details, we refer to the protocol Fig. 4.

CRS vs Public Key Size. Unlike previous schemes, our construction requires a trusted setup. One possible solution is to use a common reference string that has already been set up in a *decentralized* manner with contributions from many parties.[4] Naively, the maximum size of the database is now limited to the size of the CRS (a few million). In practice one may want to handle databases with tens of billions of entries. Such a CRS would be in the order of terabytes and running a decentralized protocol to generate this is close to impossible. Using a standard bucketing strategy, we can handle databases of size $O(|\mathsf{crs}|^2)$, at the cost of increasing the Receiver's message to $O(|\mathsf{crs}|)$ here. In practice, this amounts to < 50 MB for a database with a *trillion* entries. This is in stark contrast to the state of the art [BGJP23], which would have required a master public key of size 32 TB, as it is proportional to the size of the database. We also emphasize that the Sign algorithm only needs a small (constant sized) part of the CRS ensuring that it can be run on light clients such as mobile phones.

Element-hiding. Next, we discuss how to ensure that the sender's message hides the item x. In the protocol discussed, we have the sender explicitly include x in their message, so that the receiver knows which KZG opening and signature to use in order to decrypt m. If we simply remove x, then the remaining ciphertext elements $\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3$ do in fact hide x. However, while the receiver can attempt to decrypt with every $x \in D$, they no longer knows which is the "correct" x. To remedy this, we introduce a second witness encryption of the "zero" message under x. The receiver can use this second witness encryption to check which is the correct x, and then use that x to decrypt to intended payload m.

A seemingly more difficult issue arises when we turn to malicious security. In particular, it now becomes possible for a malicious sender to produce ill-formed ciphertexts. A natural solution would be to add a NIZK on top of the sender's message to prove well-formedness. However, even this general-purpose solutions runs into an issue: The sender's computation includes a random oracle hash $H(x)$ of the element, and we ideally want to avoid proving statements that include the code of a random oracle, for both theoretical and practical reasons.

To fix this issue, we replace our use of the BLS signature, which was the source of the random oracle, with a similar scheme that is random oracle free. In particular we use the (basic) Boneh-Boyen scheme [BB04], which we prove secure in the GGM (Appendix A.1). In this scheme, the verification key is still the group element h^{sk}, but now a signature on x is $g^{1/(\mathsf{sk}-m)}$, as opposed to $H(x)^{\mathsf{sk}}$. As it has a similar structure to BLS, in particular linear verification, we can devise efficient witness encryption with respect

[4] For instance, https://ceremony.ethereum.org has 141,416 contributions. The ZCash foundation also carried it out for polynomials of degree $< 2^{21}$ https://zfnd.org/conclusion-of-the-powers-of-tau-ceremony

to BB signatures. Finally, we are able to instantiate the sender's NIZK with a *concretely efficient*, *black-box* proof utilizing the machinery of Groth-Sahai [GS08]. As a result, we achieve a concretely-efficient element-hiding laconic PSI on authenticated inputs, which, as discussed earlier, appears to be a good candidate for addressing the problem of watchlists in anonymous e-cash systems. For full details, please refer to Section 5.3.

Set pre-constrained group signatures. Finally, we use our (non-element-hiding) laconic PSI protocol as a key building block for laconic set-constrained group signatures (SPCGS), which were discussed earlier in the introduction. The details can be found in Section 6. At a high level, in the PSI to SPCGS transformation, the sender's element x is the item that they sign, while the sender's payload is some identifying information ID that they have set up previously via interaction with the group manager / receiver. The idea is that their ID should be leaked to the receiver only if x is in a database of D of, say, harmful content. Thus, in order to achieve this application, we intuitively need some notion of authenticated inputs for the *sender*. That is, the sender must be able to *prove* that their private input ID is one that has been previously set up with the receiver. To incorporate this proof into our protocol in a concretely-efficient and black-box manner, we again make use of that Groth-Sahai proof system [GS08], along with a notion of *structure-preserving* signatures [Gha17].

3 Preliminaries

3.1 Basic cryptographic primitives and assumptions

Digital signatures. A digital signature scheme consists of three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, where $\mathsf{Gen}(1^\lambda)$ outputs a secret key sk and a public verification key vk, $\mathsf{Sign}(\mathsf{sk}, m)$ outputs a signature σ on the message m, and $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ outputs either 1 to indicate that σ is a valid signature on m, or 0 otherwise. It should satisfy correctness, which states that an honestly generated signature should verify with overwhelming probability. A signature scheme is said to be *existentially unforgeable under chosen message attack* (EUF-CMA) if it is difficult to forge a signature on a message that has not been signed before. The formal definition follows.

Definition 1. *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is existentially unforgeable under chosen message attacks if for any PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l} m \notin Q \wedge \\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 \end{array} : \begin{array}{r} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{vk}) \end{array}\right] = \mathsf{negl}(\lambda),$$

where Q *is the set of message queries that* $\mathcal{A}$ *makes to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$*.*

One-way relation. A one-way relation consists of PPT algorithms $(\mathsf{Gen}, \mathsf{Sample})$ and a set of tuples $\mathcal{R}$. $\mathsf{Gen}(1^\lambda)$ outputs public parameters pp, $\mathsf{Sample}(\mathsf{pp})$ samples instance witness pairs (x, w), and $\mathcal{R}$ is a set $\{(\mathsf{pp}, x, w)\}$. For correctness, we require that for any $\mathsf{pp} \in \mathsf{Gen}(1^\lambda)$ and $(x, w) \in \mathsf{Sample}(\mathsf{pp})$, $(\mathsf{pp}, x, w) \in \mathcal{R}$. For security, we require that no PPT adversary, given $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and x such that $(x, w) \leftarrow \mathsf{Sample}(\mathsf{pp})$ can find any w' such that $(\mathsf{pp}, x, w') \in \mathcal{R}$, except with negligible probability.

3.2 Bilinear maps

We briefly recall the definition of an asymmetric bilinear map [Jou04, BF01]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be distinct groups, all of prime order q, and let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a mapping from $\mathbb{G}_1 \times \mathbb{G}_2$ onto the target group $\mathbb{G}_T$. Let g_1, g_2 be generators for $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is an asymmetric bilinear group if the following conditions are met:

- (Efficiency) The group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as well as the mapping $e(\cdot,\cdot)$ are all efficiently computable.
- (Non-degeneracy) $e(g_1, g_2) = g_T$, where g_T is a generator of $\mathbb{G}_T$.
- (Bilinearity) $e(g_1^a, g_2^b) = g_T^{ab}$ for all $a, b \in \mathbb{Z}_q$.

Generic bilinear group model. We will use an extension of the generic group model [Nec94, Sho97] adapted to bilinear groups. We follow the presentation in [BFKT24], in particular defining the label sets as known sets of size equal to the size of the group. Moreover, we define the Add interface to directly support arbitrary linear combinations of group elements, which is without loss of generality due to the double-and-add algorithm.

Definition 2 (Generic Bilinear Group Oracle)**.** *A generic bilinear group oracle is a stateful oracle* GGM*, parameterized by a prime $p \in \mathbb{Z}$, and three sets S_1, S_2, S_T of size p.*

- *On a query* $\mathsf{GGM.Init}()$*, the oracle samples and stores three uniformly random injections $\tau_i : \mathbb{Z}_p \to S_i$ for $i \in \{1, 2, T\}$. Future queries to* GGM.Init *are ignored.*
- *On a query* $\mathsf{GGM.GetGenerator}(i)$*, the oracle returns $\tau_i(1)$.*
- *On a query* $\mathsf{GGM.Add}(i, g_1, g_2, c_1, c_2)$*, return $\perp$ unless $g_1, g_2 \in S_i$ and $c_1, c_2 \in \mathbb{Z}_p$. Otherwise, return $g = \tau_i(c_1\tau_i^{-1}(g_1) + c_2\tau_i^{-1}(g_2))$.*
- *On a query* $\mathsf{GGM.Pair}(g_1, g_2)$*, return $\perp$ unless $g_1 \in S_1, g_2 \in S_2$. Otherwise, return $\tau_T(\tau_1^{-1}(g_1) \cdot \tau_2^{-1}(g_2))$.*

Modeling a group-output random oracle. We will also model the scenario in which parties have access to a cryptographic hash function that outputs group elements, which is needed for example to support BLS signatures [BLS01]. Since the sets S_1, S_2, S_T are public, this is straightforward to implement via a random oracle defined from $\mathbb{Z}_p \to S_i$.

Definition 3. *Let $\boldsymbol{L} = (\boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}_T)$ be three sets of polynomial with elements in $\mathbb{Z}_p[X_1, \ldots, X_n]$. We define the completion of $\boldsymbol{L}$ to be $\boldsymbol{C}(\boldsymbol{L}) = \{\boldsymbol{L}_1 \otimes \boldsymbol{L}_2\} \cup \boldsymbol{L}_T = \{f_i(X_1, \ldots, X_n)\}_i$. A polynomial $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$ is said to be dependent on $\boldsymbol{L} = (\boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}_T)$ if there exist coefficients $\{\kappa_i\}_i \in \mathbb{Z}_p$ such that:*

$$f(X_1, \ldots, X_n) = \sum_{i=1} \kappa_i \cdot g_i(X_1, \ldots, X_D).$$

Otherwise, we say that f is independent of $\boldsymbol{L}$.

3.3 Non-interactive arguments of knowledge

Let $\mathcal{L}$ be an NP language and let $\mathcal{R}$ be the associated binary relation, where a statement $x \in \mathcal{L}$ if and only if there exists a witness w such that $(x, w) \in \mathcal{R}$. A non-interactive argument system for $\mathcal{R}$ consists of algorithms Setup, Prove, Verify, where $\mathsf{Setup}(1^\lambda)$ outputs a string crs, $\mathsf{Prove}(\mathsf{crs}, x, w)$ outputs a proof π, and $\mathsf{Verify}(\mathsf{crs}, x, \pi)$ outputs either 1 to indicate accept or 0 to indicate reject. *Completeness* states that a proof computed from any $(x, w) \in \mathcal{R}$ will verify correctly with overwhelming probability. We also consider the following security properties.

- *Knowledge extraction.* There exists an extractor $(\mathcal{E}_1, \mathcal{E}_2)$ such that $\{\text{crs} : \text{crs} \leftarrow \text{Setup}(1^\lambda)\} \approx_c \{\text{crs} : (\text{crs}, \tau) \leftarrow \mathcal{E}_1(1^\lambda)\}$, and for any PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|r} \text{Verify}(\text{crs}, x, \pi) = 0 & (\text{crs}, \tau) \leftarrow \mathcal{E}_1(1^\lambda) \\ \vee\ (x, w) \in \mathcal{R} & (x, \pi) \leftarrow \mathcal{A}(\text{crs}) \\ & w \leftarrow \mathcal{E}_2(\tau, x, \pi) \end{array}\right] = 1 - \text{negl}(\lambda).$$

- *Zero-knowledge.* There exist algorithms SimSetup and SimProve such that for any $(x, w) \in \mathcal{R}$,

$$\left\{(\text{crs}, x, \pi) : \begin{array}{r} \text{crs} \leftarrow \text{Setup}(1^\lambda), \\ \pi \leftarrow \text{Prove}(\text{crs}, x, w) \end{array}\right\} \approx_c \left\{(\text{crs}, x, \pi) : \begin{array}{r} (\text{crs}, \text{td}) \leftarrow \text{SimSetup}(1^\lambda, x) \\ \pi \leftarrow \text{SimProve}(\text{crs}, \text{td}, x) \end{array}\right\}$$

We say that a non-interactive argument system for a relation $\mathcal{R}$ that satisfies *completeness*, *knowledge extraction*, and *zero-knowledge*, is a *zero-knowledge non-interactive argument of knowledge* (ZK-NIAoK) for $\mathcal{R}$. We say that a ZK-NIAoK is in the *common random string* model, if Setup simply outputs a uniformly random string crs. We say that a ZK-NIAoK is in the *random oracle model* if Setup includes the sampling of a random oracle H, and the knowledge extractor and simulator can sample H and observe and respond to $\mathcal{A}$'s queries to H.

Let $\mathbb{G}$ be a group with generators g_1 and g_2. We will use the fact that the following relations have highly efficient ZK-NIAoKs in the ROM.

- The discrete log relation $\mathcal{R}_{\text{DLog}} = \{((g_1, h), \alpha) : h = g_1^\alpha\}$. A ZK-NIAoK for $\mathcal{R}_{\text{DLog}}$ follows from applying the Fiat-Shamir heuristic [FS87] to Schnorr's sigma protocol [Sch90].
- This can also be extended to the discrete log relation with respect to multiple generators $\mathcal{R}_{\text{Rep}}\{((g_1, g_2, h), (\alpha_1, \alpha_2)) : h = g_1^{\alpha_1} g_2^{\alpha_2}\}$.

3.4 Groth-Sahai proofs

Groth and Sahai [GS08] constructed efficient non-interactive zero-knowledge proof systems (Setup, Prove, Verify) for statements that involve equations over bilinear maps. "GS proofs" prove statements that consist of the following types of equations over variables $X_1, \ldots, X_m \in \mathbb{G}_1, Y_1, \ldots, Y_n \in \mathbb{G}_2, x_1, \ldots, x_{m'}, y_1, \ldots, y_{n'} \in \mathbb{Z}_p$.

- *Pairing product equations.* $\prod_{i=1}^n e(A_i, Y_i) \prod_{i=1}^m e(X_i, B_i) \prod_{i=1}^m \prod_{j=1}^n e(A_i, B_j)^{c_{ij}} = 1_{\mathbb{T}}$, for constants $A_i \in \mathbb{G}_1, B_i \in \mathbb{G}_2, c_{ij} \in \mathbb{Z}_p$, where $1_{\mathbb{T}}$ is the identity element in $\mathbb{T}$. [5]
- *Multi-scalar exponentiation.* $\prod_{i=1}^{n'} A_i^{y_i} \prod_{i=1}^m X_i^{b_i} \prod_{i=1}^m \prod_{j=1}^{n'} X_i^{c_{ij} y_j} = T_1$, for constants $A_i, T_1 \in \mathbb{G}_1, b_i, c_{ij} \in \mathbb{Z}_p$ and analogous statements for multi-scalar exponentiation in $\mathbb{G}_2$.
- *Quadratic equations in* $\mathbb{Z}_p$. $\sum_{i=1}^{n'} a_i y_i + \sum_{i=1}^{m'} x_i b_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} c_{ij} x_i y_j \equiv t \bmod p$, for constant $a_i, b_i, c_{ij}, t \in \mathbb{Z}_p$.

GS proofs are in the *common random string* model, and satisfy the completeness and zero-knowledge properties described in Section 3.3. However, they only satisfy a weaker notion of knowledge extraction which has been referred to as *partial* knowledge extraction [Gro07]. This property states that if the witness consists of both group elements and exponents, only the group elements are extractable.

[5] [GS08] also consider pairing product equations where the target element is not the identity, but their proofs for such equations are in general only witness indistinguishable, as opposed to zero-knowledge.

3.5 Short Structure-Preserving Signatures

We recall the structure-preserving signature scheme by Ghadafi [Gha17]. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, with generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$.

- $\mathsf{Setup}(1^\lambda)$: The signer samples a secret key $\mathsf{sk} := (u, v) \leftarrow\!\$\ \mathbb{F}^2$ and publishes $\mathsf{vk} := (U = h^u, V = h^v)$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: To sign a message $(M, N) = (g^m, h^m)$, sample $r \leftarrow\!\$\ \mathbb{F}$ and set $R := g^r$, $S := ((g^u \cdot M)^r \cdot g)^{\frac{1}{v}}$ and output the signature $\sigma = (R, S) \in \mathbb{G}_1^2$.
- $\mathsf{Verify}(\mathsf{vk}, \sigma, m)$: Parse $\mathsf{vk} = (U, V)$. Accept if the message (M, N) satisfies $\mathsf{dlog}_g M = \mathsf{dlog}_h N$, and $e(S, V) = e(R, U) \cdot e(R, N) \cdot e(g, h)$.

The above scheme was shown to be Existentially Unforgeable under Chosen Message Attacks (EUF-CMA) in [Gha17, Theorem 1] in the generic group model. Against adversaries that are not allowed to query the signing oracle on the same message more than once, the above scheme is Strongly Unforgeable under Chosen Message Attacks (SUF-CMA) [Gha17, Theorem 2].

3.6 KZG Polynomial Commitment Scheme

We recall the KZG polynomial commitment scheme [KZG10]. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map with generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. The KZG polynomial commitment scheme consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda, n)$: A trusted party samples $\tau \leftarrow\!\$\ \mathbb{F}$ and publishes commitment key $\mathsf{ck} = (g, g^\tau, \ldots, g^{\tau^n}, h, h^\tau)$, and verification key $\mathsf{vk} = (g, h, h^\tau)$.
- $\mathsf{Commit}(\mathsf{ck}, f(X))$: To commit to a polynomial $f(X)$ of degree at most n, output $\mathsf{com} = g^{f(\tau)}$.
- $\mathsf{Open}(\mathsf{com}, f(X))$: To compute an opening proof for a commitment com to $f(X)$ at the point x^*, compute $\pi = g^{q(\tau)}$ where $q(X) = (f(X) - f(x^*))/(x - x^*)$, and output π.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{com}, \pi, x^*, y^*)$: To verify the opening of a commitment com to some polynomial at the point x^* to the value y^*, check whether $e(\mathsf{com} \cdot g^{-y^*}, h) = e(\pi, h^{\tau - x^*})$, and accept if so, else reject.

The above commitment was shown to be binding in [KZG10] under the q-SDH assumption. The KZG polynomial commitment is also *extractable* in the algebraic group model [FKL18]. To make the commitment hiding, we must sufficiently *randomize* the commitment. Looking ahead, we will actually use the KZG polynomial commitment scheme as an accumulator to a set $D = \{x_1, \ldots, x_n\}$ of elements, where the accumulator is defined as a commitment to the polynomial $f(X) = \prod_{i \in [n]} (X - x_i)$. A membership proof for an element x^* is then an opening proof for the commitment to 0 at the point x^*. We will also want to make the accumulator *hiding* against an adversary who does not see any opening proofs. To do this, we can simply raise the commitment and opening proofs to a random value.

3.7 Cuckoo hashing

A cuckoo hashing scheme consists of the algorithms $(\mathsf{Setup}, \mathsf{Hash})$, and is parameterized by a universe $\mathcal{U}$ of elements.

- $\mathsf{Setup}(\lambda, n, \epsilon) \to (n', h_0, h_1)$: the setup algorithm takes as input an integer parameter λ, an integer bound n, and $\epsilon \geq 0$, and outputs an integer n' and two hash functions $h_0, h_1 : \mathcal{U} \to [n']$, where n' is a deterministic function of λ, n, and ϵ.

- $\mathsf{Hash}(h_0, h_1, D) \to T$: the (deterministic) hashing algorithm takes hash functions $h_0, h_1 : \mathcal{U} \to [n']$ and a set $D \subseteq \mathcal{U}$, and outputs a table $T = [T_1, \ldots, T_{n'}]$, where each T_i is either an element in D or $\perp$.

The correctness requirements are that

1. For every $x \in \mathcal{U}$, $h_0(x) \neq h_1(x)$. We will assume that this is the case for every pair of even adversarially chosen hash functions.[6]
2. Each non-$\perp$ element of T is distinct.
3. For any n, ϵ and set $D \subseteq \mathcal{U}$ of size n, it holds that with probability $1 - \mathsf{negl}(\lambda)$ over $(m, h_0, h_1) \leftarrow \mathsf{Setup}(\lambda, n, \epsilon)$, $T := \mathsf{Hash}(h_0, h_1, D)$, there exists a set $D' \subseteq D$ such that $|D'| \geq (1 - \epsilon)|D|$ and such that for any $x \in D'$, either $T_{h_0(x)} = x$ or $T_{h_1(x)} = x$.

4 Private Set Intersection on Authenticated Inputs: Definitions

In this section, we define the notion of two-message maliciously-secure labeled private set intersection on authenticated inputs. To reduce clutter, we will let "PSI" refer to two-message maliciously-secure labeled private set intersection. We note that our notion of PSI on authenticated inputs strictly generalizes the standard notion of PSI (i.e. without authenticated inputs), by instantiating the RecVer and SenVer algorithms with algorithms that always accept.

Definition 4 (PSI on authenticated inputs)**.** *A PSI on authenticated inputs is parameterized by two families of verification algorithms* $\{\{\mathsf{RecVer}_{\mathsf{rvk}}\}_{\mathsf{rvk} \in \mathsf{RVK}_\lambda}\}_{\lambda \in \mathbb{N}}$ *and* $\{\{\mathsf{SenVer}_{\mathsf{svk}}\}_{\mathsf{svk} \in \mathsf{SVK}_\lambda}\}_{\lambda \in \mathbb{N}}$*, where for each* $\lambda \in \mathbb{N}$*,* $\mathsf{RVK}_\lambda, \mathsf{SVK}_\lambda$ *are sets of public parameters. Typically, we think of* rvk, svk *as verification keys for some signature schemes, and* RecVer, SenVer *are the verification algorithms, but in principle they could be arbitrarily defined predicates. The protocol consists of four additional algorithms* (Gen, Rec, Sen, Dec) *with the following syntax.*

- $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x) \to \{\top, \perp\}$ *takes as input an item* x *and a signature* σ_x *and either accepts or rejects.*
- $\mathsf{SenVer}_{\mathsf{svk}}((x, m), \sigma_{(x,m)}) \to \{\top, \perp\}$ *takes as input an item and payload pair* (x, m) *and a signature* $\sigma_{(x,m)}$ *and either accepts or rejects.*
- $\mathsf{Gen}(1^\lambda) \to \mathsf{pp}$: *The parameter generation algorithm samples public parameters* pp*.*
- $\mathsf{Rec}(\mathsf{pp}, D) \to \mathsf{msg}_R, \mathsf{st}$: *The receiver's message generation algorithm takes as input the public parameters* pp *and a set of items* D *(usually referred to as a database), and outputs a message* msg_R *and private state* st*.*
- $\mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)}) \to \{\mathsf{msg}_S, \perp\}$*: The sender's message generation algorithm takes as input the public parameters* pp*, the receiver's message* msg_R*, an item* x*, a payload* m*, and a signature* $\sigma_{(x,m)}$*, and outputs either a message* msg_S *or an abort symbol* $\perp$*.*
- $\mathsf{Dec}(\mathsf{st}, \mathsf{msg}_S, \Sigma) \to \{m, \perp\}$: *The decryption algorithm takes as input the receiver's state* st*, the sender's message* msg_S*, and a set of signatures* $\Sigma = \{(x, \sigma_x)\}_x$*, and outputs either a payload* m *or an abort symbol* $\perp$*.*

Remark 1. *One could generalize this definition to also allow the* Rec *algorithm to take as input a set of signatures on items in* D*. However, our construction will achieve the more stringent requirement that signatures are not needed until the* Dec *algorithm is run, and so we write the definition above to reflect this feature.*

[6] For example, h_1 can be defined to first hash x and then check if the hash is equal to $h_0(x)$ and if so add 1.

Moreover, note that in the case of non-element-hiding PSI (i.e. where msg_S *reveals the item* x*), one can without loss of generality take the input* Σ *to* Dec *to consist of only the single signature* (x, σ_x) *required to decrypt that particular message.*

4.1 Correctness

We consider a strong notion of *perfect* correctness that will be crucial for our application to set pre-constrained group signatures.

Definition 5 (Perfect correctness)**.** *A PSI scheme is* perfectly correct *if for any database* D*, item* x*, payload* m*, signature* σ_x *such that* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x) = \top$*, signature* $\sigma_{(x,m)}$ *such that* $\mathsf{SenVer}_{\mathsf{svk}}((x, m), \sigma_{(x,m)}) = \top$*, and* Σ *such that* $(x, \sigma_x) \in \Sigma$*, the following holds with probability* $1 - \mathsf{negl}(\lambda)$ *over* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$*:*

- *For any* $(\mathsf{msg}_R, \mathsf{st}) \in \mathsf{Rec}(\mathsf{pp}, D)$ *and* $\mathsf{msg}_S \in \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)})$*,* $\mathsf{Dec}(\mathsf{st}, \mathsf{msg}_S, \Sigma) \neq m'$ *for any* $m' \neq m$*.*[7]
- *If* $x \in D$*, then with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{msg}_R, \mathsf{st}) \leftarrow \mathsf{Rec}(\mathsf{pp}, D)$*, it holds that for any* $\mathsf{msg}_S \in \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)})$*,* $\mathsf{Dec}(\mathsf{st}, \mathsf{msg}_S, \Sigma) = m$*.*

Next, we consider a relaxed notion of perfect correctness for PSI called *perfect* ϵ*-correctness*, where the parameter ϵ essentially determines an upper bound on the fraction of the set D that might be "dropped" by the Rec algorithm.[8]

Definition 6 (Perfect ϵ-correctness)**.** *A PSI scheme is* perfectly ϵ-correct *for some* $\epsilon \geq 0$ *if for any database* D*, item* x*, payload* m*, signature* σ_x *such that* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x) = \top$*, signature* $\sigma_{(x,m)}$ *such that* $\mathsf{SenVer}_{\mathsf{svk}}((x, m), \sigma_{(x,m)}) = \top$*, and* Σ *such that* $(x, \sigma_x) \in \Sigma$*, the following holds with probability* $1 - \mathsf{negl}(\lambda)$ *over* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$*:*

- *For any* $(\mathsf{msg}_R, \mathsf{st}) \in \mathsf{Rec}(\mathsf{pp}, D, \sigma_D)$ *and* $\mathsf{msg}_S \in \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)})$*,* $\mathsf{Dec}(\mathsf{st}, \mathsf{msg}_S, \Sigma) \neq m'$ *for any* $m' \neq m$*.*
- *With probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{msg}_R, \mathsf{st}) \leftarrow \mathsf{Rec}(\mathsf{pp}, D)$*, there exists a* $D' \subseteq D$ *such that* $|D'| \geq (1 - \epsilon)|D|$ *and if* $x \in D'$*, then for any* $\mathsf{msg}_S \in \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)})$*,* $\mathsf{Dec}(\mathsf{st}, \mathsf{msg}_S, \Sigma) = m$*.*

4.2 Security

We define security using the real world / ideal world framework [Gol04]. We consider the notion of *distinguisher-dependent* simulation, where the simulator is defined based on the description of the adversary as well as the distinguisher [JKKR17].

The environment. We consider the protocol to be run in the presence of an *environment* that samples the verification keys $\mathsf{rvk}, \mathsf{svk}$ potentially along with some auxiliary input, and chooses inputs for the parties running the protocol. Formally, we model the environment as a PPT algorithm $\mathcal{E}(1^\lambda) \rightarrow (\mathsf{rvk}, \mathsf{svk}, z, \mathsf{inp}_R, \mathsf{inp}_S)$, where z is some auxiliary information, $\mathsf{inp}_R = (D, \{x, \sigma_x\}_x)$ is the receiver's input, and $\mathsf{inp}_S = \{x, m, \sigma_{(x,m)}\}_{(x,m)}$ is the sender's input.

The real execution. In the honest execution of the protocol, the receiver R runs the Rec algorithm and then potentially multiple senders S run potentially multiple invocations of the Sen algorithm on their set of inputs. However, the protocol may be executed in the presence of an adversary $\mathcal{A}$ that corrupts one of the parties. Let $C \in \{R, S\}$ refer to the identity of the corrupted party and $H \in \{R, S\}$ refer to the identity of the honest party.[9] Then, the real-world execution proceeds as follows.

[7] Though it could be the case that $\mathsf{Dec}(\mathsf{st}, \mathsf{msg}_S, \Sigma) = \perp$.

[8] Traditionally, one might expect ϵ to be negligible, and thus suppressed in the definition. However, one of our protocols will make use of cuckoo hashing which may introduce an inverse-polynomial ϵ.

[9] Without loss of generality, if the adversary corrupts any sender, we treat all senders as corrupt.

- The environment samples $(\mathsf{rvk}, \mathsf{svk}, z, \mathsf{inp}_R, \mathsf{inp}_S) \leftarrow \mathcal{E}(1^\lambda)$.
- The protocol parameters are generated $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$.
- The PPT adversary $\mathcal{A}$ corrupting party C obtains $(\mathsf{rvk}, \mathsf{svk}, z, \mathsf{inp}_C, \mathsf{pp})$ as input and interacts arbitrarily maliciously with the honest party running H, initialized with input $(\mathsf{rvk}, \mathsf{svk}, \mathsf{inp}_H, \mathsf{pp})$. This interaction produces an honest party output (if it has one), and an adversary's output (which may be an arbitrary function of its view).

We refer to the output of the above experiment using the random variable $\mathsf{REAL}_{\mathcal{E},\mathcal{A}}(1^\lambda)$.

The ideal execution. In the ideal execution, a *simulator* Sim controls the protocol setup and the actions of the corrupted party. Sim has the same input as $\mathcal{A}$ in the real world. However, in the ideal execution, both Sim and the honest party interact with an *ideal functionality* $\mathcal{F}$ in order to send inputs and receive outputs. That is, the ideal-world execution proceeds as follows.

- The environment samples $(\mathsf{rvk}, \mathsf{svk}, z, \mathsf{inp}_R, \mathsf{inp}_S) \leftarrow \mathcal{E}(1^\lambda)$.
- The ideal functionality $\mathcal{F}$ is initialized with rvk, svk.
- The honest party H sends their input inp_H to the ideal functionality $\mathcal{F}$.
- The PPT simulator Sim takes as input $(\mathsf{rvk}, \mathsf{svk}, z, \mathsf{inp}_C)$, interacts with the ideal functionality $\mathcal{F}$, and then outputs an arbitrary function of its view.

We refer to the output of the above experiment, which includes the honest party output obtained from $\mathcal{F}$, along with Sim's output, using the random variable $\mathsf{IDEAL}_{\mathcal{F},\mathcal{E},\mathsf{Sim}}(1^\lambda)$.

Definition 7. *A PSI protocol securely emulates an ideal functionality $\mathcal{F}$ if for any PPT adversary corrupting party C and any PPT distinguisher $\mathcal{D}$, there exists a PPT* Sim *such that for any PPT environment $\mathcal{E}$,*

$$\left| \Pr\left[\mathcal{D}\left(\mathsf{REAL}_{\mathcal{E},\mathcal{A}}(1^\lambda)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{IDEAL}_{\mathcal{F},\mathcal{E},\mathsf{Sim}}(1^\lambda)\right) = 1\right] \right| = \mathsf{negl}(\lambda).$$

Security in idealized models. Looking ahead, we will prove the security of our schemes in the generic bilinear group model (sometimes along with a random oracle $H : \mathbb{Z}_p \to S_1$), as defined in Section 3.2. Concretely, the parameters p, S_1, S_2, S_T are fixed and public (where $p = p(\lambda)$ may depend on the choice of security parameter), and GGM.Init() is invoked at the beginning of time. Then, all parties – the environment $\mathcal{E}$, the honest parties, the adversary $\mathcal{A}$, the simulator Sim, and the distinguisher $\mathcal{D}$ – are granted query access to the interfaces of the oracle GGM, as well as H. Since all parties are PPT, this implies that they all make at most $\mathsf{poly}(\lambda)$ queries to GGM. Syntactically, all group elements in rvk, svk, z, inp_R, inp_S, pp, as well as those manipulated by the Rec, Gen, and Dec algorithms are represented as handles in the sets S_1, S_2, S_T.

Importantly, since the simulator Sim is defined based on the adversary $\mathcal{A}$ and the distinguisher $\mathcal{D}$, it has the ability to run $\mathcal{A}$ and $\mathcal{D}$, and intercept their queries made to GGM and H. However, since the environment $\mathcal{E}$ is run *before* the adversary / simulator are invoked, the simulator will not have the ability to observe the queries made by $\mathcal{E}$. In particular, the adversary / simulator may be initialized with "mystery" handles in the sets S_1, S_2, S_T computed by $\mathcal{E}$ for which it does not know the corresponding exponents (i.e. the inverses under τ_1, τ_2, τ_T).

Ideal functionalities for PSI on authenticated inputs. Now, we describe the ideal functionalities that we aim to emulate in this work: one without element hiding (where all items are revealed), and one with

element hiding (where only the items that match are revealed).[10] These ideal functionalities are defined based on the descriptions of the algorithms RecVer, SenVer, and are presented in Fig. 1 and Fig. 2.

$\mathcal{F}_{\mathsf{PSI}}$: PSI without element hiding

Parameters: Upper bound n on the database size, and parameters rvk, svk.

- Upon receiving a message $(\text{receiver}, \text{sid}, D)$, record (sid, D) if $|D| \leq n$.
- Upon receiving a message $(\text{sender}, \text{sid}, \text{ssid}, x, m, \sigma_{(x,m)})$, check whether $\mathsf{SenVer}_{\mathsf{svk}}((x, m), \sigma_{(x,m)}) = 1$. If so, record $(\text{sid}, \text{ssid}, x, m)$ and send $(\text{sid}, \text{ssid}, x)$ to the receiver.
- Upon receiving a message $(\text{receiver}, \text{sid}, x, \sigma_x)$, check that there exists a record (sid, D), that $x \in D$, that $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x) = 1$, and that there exists a record $(\text{sid}, \text{ssid}, x, m)$. If so, send m to the receiver.

Figure 1: Ideal functionality for PSI on authenticated inputs without element hiding.

$\mathcal{F}_{\mathsf{PSI-EH}}$: PSI with element hiding

Parameters: Upper bound n on the database size, and parameters rvk, svk.

- Upon receiving a message $(\text{receiver}, \text{sid}, D)$, record (sid, D) if $|D| \leq n$.
- Upon receiving a message $(\text{sender}, \text{sid}, \text{ssid}, x, m, \sigma_{(x,m)})$, check whether $\mathsf{SenVer}_{\mathsf{svk}}((x, m), \sigma_{(x,m)}) = 1$. If so, record $(\text{sid}, \text{ssid}, x, m)$ and send $(\text{sid}, \text{ssid})$ to the receiver.
- Upon receiving a message $(\text{receiver}, \text{sid}, x, \sigma_x)$, check that there exists a record (sid, D), that $x \in D$, that $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x) = 1$, and that there exists a record $(\text{sid}, \text{ssid}, x, m)$. If so, send m to the receiver.

Figure 2: Ideal functionality for PSI on authenticated inputs with element hiding.

Security against outsiders. We note that security as defined above generically implies the following desirable property that we call "security against outsiders".

Definition 8 (Security against outsiders)**.** *A PSI scheme satisfies security against outsiders with element-hiding if for any PPT adversary $\mathcal{A}$, pair $(x_0, m_0, \sigma_{(x_0,m_0)}), (x_1, m_1, \sigma_{(x_1,m_1)})$ of item / payload / signature, and database D,*

$$\Pr\left[\mathcal{A}(\mathsf{pp}, \mathsf{msg}_S, \mathsf{msg}_R) = b : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{msg}_R, \mathsf{st}) \leftarrow \mathsf{Rec}(\mathsf{pp}, D) \\ b \leftarrow \{0,1\} \\ \mathsf{msg}_S \leftarrow \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x_b, m_b, \sigma_{(x_b,m_b)}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

[10] We note that one could also define a notion of *strong* element hiding, where the elements x are not leaked even for those $x \in D$ (and only the corresponding payload m is leaked), but we do not consider that notion in this work.

The scheme satisfies security against outsiders without *element-hiding if the above property holds only for pairs* $(x, m_0, \sigma_{(x,m_0)}), (x, m_1, \sigma_{(x,m_1)})$ *(i.e. where the item* x *is the same).*

5 Private Set Intersection on Authenticated Inputs: Construction

In this section we first provide the full protocol for maliciously secure PSI on authenticated inputs and then describe extensions to an asymptotically faster PSI protocol with perfect ϵ-correctness (Section 5.2), and a PSI protocol with element hiding (Section 5.3), all with authenticated inputs.

5.1 PSI protocol

We provide a full description of our PSI protocol in Fig. 3. In the description we allow full flexibility in the SenVer algorithm. Although, for concrete efficiency one would choose a signature scheme with algebraic structure such as structure preserving signatures [AFG+10, Gha17]. This will allow us to construct an efficient NIZK proof using the Groth-Sahai proof system [GS08]. It securely emulates the $\mathcal{F}_{\mathsf{PSI}}$ functionality from Section 4 Theorem 4. The full proof of security can be found in Appendix A.2.

Theorem 4. *Let* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$ *be defined as* $e(H(x), \mathsf{rvk}) \stackrel{?}{=} e(\sigma_x, h)$. *The protocol in Fig. 3 securely emulates (Definition 7) the* $\mathcal{F}_{\mathsf{PSI}}$ *functionality with perfect correctness (Definition 5), with respect to the receiver verification predicate* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$*, against generic non-uniform PPT adversaries corrupting either the sender or the receiver in the random oracle model.*

Receiver-only authenticated inputs. In some settings, we care in particular about the receiver's inputs being authenticated, but can relax this requirement for the sender. In this case, we can instantiate the sender's algorithm in a particularly efficient way. We note that for any choice of $(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$, there exists an input m and a choice of α, β that explains the message. In particular, this means that the scheme is perfectly correct even against malicious senders. Thus, to enable extraction, the proof π only needs to include efficient ZK-NIAoKs of the discrete log relation (see Section 3.3). That is, the sender sets π_1 to be a proof of knowledge of discrete log of ct_1 with respect to $h^{(\tau - x)}$, sets π_2 to be a proof of knowledge of discrete log of ct_2 with respect to h, and sets $\pi = (\pi_1, \pi_2)$.

5.2 Optimized ϵ-correct PSI protocol

The protocol presented in Fig. 3 satisfies perfect correctness but is somewhat expensive for the receiver, as they need to compute KZG opening proofs for $O(|D|)$ number of points. Naively, this would take $O(|D|^2)$ time, though [FK23] showed that it can be done in $O(|D| \log^2 |D|)$ time. This can be further reduced to $O(|D| \log |D|)$ time (with good constants) if the database is encoded as the evaluation points on roots of unity as opposed to as roots of the polynomial. We now describe how Fig. 3 can be modified to improve the run time of the receiver at the cost of sacrificing perfect correctness and slightly larger sender messages.

At a high-level, the receiver will use cuckoo hashing (using say h_0, h_1) to encode the database D into a table T of size $(1+\varepsilon)|D|$ for some constant ε, dropping an ϵ fraction of items (in expectation if necessary). This datastructure allows them to check for membership of an item x in $O(1)$ time with the guarantee that for most items $x \in D$, either $T_{h_0(x)} = x$ or $T_{h_1(x)} = x$. The receiver now interpolates a polynomial f such that $f(\omega^i) = T_i$ for $i \in (1+\varepsilon)|D|$, where ω is the $(1+\varepsilon)|D|$-th root of unity and sends a KZG commitment com to this $f(X)$. The sender now encrypts the message to the statement "I know an opening proof π of com at the point $h_0(x)$ or $h_1(x)$ to the value x *and* a signature σ on x under the verification key vk". The full protocol can be found in Fig. 4.

Π_{PSI}

Parameters: A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with generators $g \in \mathbb{G}_1$, and $h \in \mathbb{G}_2$. Receiver verification key rvk, Sender verification key svk, and a random oracle $H : \{0,1\}^* \to \mathbb{G}_1$.

Gen(1^λ): Sample $\tau \leftarrow\!\!\$\ \mathbb{F}$ and set $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (g, g^\tau, \ldots, g^{\tau^{n+1}}, h, h^\tau)$. Run the setup for the non-interactive zero-knowledge argument of knowledge and set $\mathsf{NIZK}_{\mathsf{crs}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Output $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$.

RecVer$(\mathsf{rvk}, x, \sigma_x)$: If $e(H(x), \mathsf{rvk}) = e(\sigma_x, h)$, output 1, else output 0.

Rec(pp, D): Interpolate a polynomial $f(X)$ of degree at most $n+1$ such that $f(D[i]) = 0$ for all $i \in [n]$, and $f(0) = 0$. Sample a random value $r \leftarrow\!\!\$\ \mathbb{F} \setminus \{0\}$ and compute $\mathsf{com} \leftarrow g^{r \cdot f(\tau)}$, along with opening proofs, using homomorphic operations on $\mathsf{KZG}_{\mathsf{crs}}$:

- $\Pi = \{\pi_i \leftarrow g^{r \cdot q(\tau)}; q(X) \coloneqq f(X)/(X - D[i])\}_{i \in [n]}$
- $\pi_0 \leftarrow g^{r \cdot f(\tau)/\tau}$

Output $\mathsf{msg}_R \coloneqq (\mathsf{com}, \pi_0)$ and $\mathsf{st} = \Pi$.

Sen$(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)})$: Parse $\mathsf{msg}_R = (\mathsf{com}, \pi_0)$. Abort if $\mathsf{com} = \mathbb{1}_{\mathbb{G}_1}$ or $x = 0$ or if $e(\mathsf{com}, h) \neq e(\pi_0, h^\tau)$. Sample $\alpha, \beta \leftarrow\!\!\$\ \mathbb{F}$ and set $\mathsf{msg}_S = (x, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$ defined as:

- $\mathsf{ct}_1 \leftarrow h^{(\tau - x)\alpha}$
- $\mathsf{ct}_2 \leftarrow h^\beta$
- $\mathsf{ct}_3 \leftarrow e(\mathsf{com}, h)^\alpha \cdot e(H(x), \mathsf{rvk})^\beta \cdot m$
- $\pi = \mathsf{NIZK.Prove}\{m, \alpha, \beta, \sigma_{(x,m)} \mid \mathsf{ct}_1 = (h^{\tau - x})^\alpha \wedge \mathsf{ct}_2 = h^\beta \wedge \mathsf{ct}_3 = e(\mathsf{com}, h)^\alpha \cdot e(H(x), \mathsf{rvk})^\beta \cdot m \wedge \mathsf{SenVer}(\mathsf{svk}, (x, m), \sigma_{(x,m)}) = 1\}$

Dec$(\mathsf{st}, \mathsf{msg}_S, \Sigma)$: Parse $\mathsf{ct} = (x, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$ and check whether

- $x \in D$
- $\exists\, \sigma_x \in \Sigma$ such that $\mathsf{RecVer}(\mathsf{rvk}, x, \sigma_x) = 1$
- $\mathsf{NIZK.Verify}((x, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3), \pi) = 1$

If all the above checks pass output $m \leftarrow \mathsf{ct}_3 \cdot e(\Pi_x, \mathsf{ct}_1)^{-1} \cdot e(\sigma_x, \mathsf{ct}_2)^{-1}$, where $\Pi_x \coloneqq g^{q(\tau)}$ for $q(X) \coloneqq f(X)/(X - x)$, and $\sigma_x \in \Sigma$ is a signature on x under rvk. Else output $\perp$.

Figure 3: A PSI protocol emulating $\mathcal{F}_{\mathsf{PSI}}$ with perfect correctness.

Theorem 5. *Let* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$ *be defined as* $e(H(x), \mathsf{rvk}) \stackrel{?}{=} e(\sigma_x, h)$. *The protocol in Fig. 4 securely emulates (Definition 7) the* $\mathcal{F}_{\mathsf{PSI}}$ *functionality with perfect* ϵ*-correctness (Definition 6), with respect to the receiver verification predicate* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$*, against generic non-uniform PPT adversaries corrupting either the sender or the receiver in the random oracle model.*

$\Pi_{\epsilon-\mathsf{PSI}}$

Parameters: A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with generators $g \in \mathbb{G}_1$, and $h \in \mathbb{G}_2$. Receiver verification key rvk, Sender verification key svk, a random oracle $H : \{0,1\}^* \to \mathbb{G}_1$, and PSI correctness parameter ϵ.

Gen$(1^\lambda, \epsilon)$: Sample cuckoo hashes $(n', h_0, h_1) \leftarrow \mathsf{CH.Setup}(\lambda, n, \epsilon)$. Sample $\tau \leftarrow\!\!\$\ \mathbb{F}$ and set $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (g, g^\tau, \ldots, g^{\tau^{n'+1}}, h, h^\tau)$. Run the setup for the non-interactive zero-knowledge argument of knowledge and set $\mathsf{NIZK}_{\mathsf{crs}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Output $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}}, (n', h_0, h_1))$.

RecVer$(\mathsf{rvk}, x, \sigma_x)$: If $e(H(x), \mathsf{rvk}) = e(\sigma_x, h)$, output 1, else output 0, where rvk is the verification key.

Rec(pp, D): Hash the database D into a cuckoo table $T \coloneqq \mathsf{CH.Hash}(h_0, h_1, D)$. Interpolate a polynomial $f_T(X)$ of degree at most $n' + 1$ such that $f_T(\omega^i) = T[i]$ for all $i \in [n']$, $f_T(\omega^{n'+1}) = r$, and $f_T(0) = 0$, where ω is the $(n'+1)$-th root of unity, and $r \leftarrow\!\!\$\ \mathbb{F}$. Compute $\mathsf{com} \leftarrow g^{f_T(\tau)}$ along with opening proofs $\Pi = \{\pi_i \leftarrow g^{q(\tau)}; q(X) = (f_T(X) - T[i])/(X - \omega^i)\}_{i\in[n']}$, and $\pi_0 \leftarrow g^{f(\tau)/\tau}$ using homomorphic operations on $\mathsf{KZG}_{\mathsf{crs}}$. Output $\mathsf{msg}_R \coloneqq (\mathsf{com}, \pi_0)$ and $\mathsf{st} = \Pi$.

Sen$(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)})$: Parse $\mathsf{msg}_R = \mathsf{com}$, abort if $\mathsf{com} = \mathbb{1}_{\mathbb{G}_1}$ or if $x \in \{0, \omega^{n'+1}\}$. Sample $\alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow\!\!\$\ \mathbb{F}$ and set $\mathsf{msg}_S = (x, \mathsf{ct}^0, \mathsf{ct}^1)$, where $\mathsf{ct}^b = (\mathsf{ct}_1^b, \mathsf{ct}_2^b, \mathsf{ct}_3^b, \pi^b)$ for $b \in [0,1]$ is defined as:

- $\mathsf{ct}_1^b \leftarrow h^{(\tau - \omega^{h_b(x)})\alpha}$,
- $\mathsf{ct}_2^b \leftarrow h^\beta$
- $\mathsf{ct}_3^b \leftarrow e(\mathsf{com} \cdot g^{-x}, h)^\alpha \cdot e(H(x), \mathsf{rvk})^\beta \cdot m$
- $\pi^b = \mathsf{NIZK.Prove}\{m, \alpha, \beta, \sigma_{(x,m)} \mid \mathsf{ct}_1 = (h^{\tau - \omega^{h_b(x)}})^\alpha \wedge \mathsf{ct}_2 = h^\beta \wedge \mathsf{ct}_3 = e(\mathsf{com} \cdot g^{-x}, h)^\alpha \cdot e(H(x), \mathsf{rvk})^\beta \cdot m \wedge \mathsf{SenVer}((x,m), \sigma_{(x,m)}) = 1\}$

Dec$(\mathsf{st}, \mathsf{msg}_S, \Sigma)$: Parse $\mathsf{ct} = (x, \mathsf{ct}^0, \mathsf{ct}^1)$ and $\mathsf{ct}^b = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$. Check that $\mathsf{NIZK.Verify}((x, \mathsf{ct}_1^b, \mathsf{ct}_2^b, \mathsf{ct}_3^b), \mathsf{ct}^b.\pi) = 1$ for $b \in [0,1]$. If verification fails or if $x \notin D$, or $\nexists\, \sigma_x \in \Sigma$ such that $\mathsf{RecVer}(x, \sigma_x) = 1$, output $\perp$. Else, output

$$m \leftarrow \mathsf{ct}_3^{\tilde{b}} \cdot e(\Pi_x, \mathsf{ct}_1^{\tilde{b}})^{-1} \cdot e(\sigma_x, \mathsf{ct}_2^{\tilde{b}})^{-1}$$

where $\tilde{b} \in [0,1]$ such that x is stored at $T[h_{\tilde{b}}(x)]$, $\Pi_x \coloneqq g^{r \cdot q(\tau)}$ for $q(X) = (f(X) - x)/(X - \omega^{h_{\tilde{b}}(x)})$, and $\sigma_x \in \Sigma$ is a BLS signature on x under rvk.

Figure 4: A PSI protocol emulating $\mathcal{F}_{\mathsf{PSI}}$ with ϵ-perfect correctness.

5.3 PSI with Element Hiding

We now discuss how the protocol in Fig. 3 can be extended to achieve PSI with element hiding. Informally, we would like to ensure that the sender's message reveals no information about x if $x \notin D$ or if the receiver

does not have a valid signature on x. A similar strategy can be applied to Fig. 4.

The first (obvious) issue is the sender's message in Fig. 3 reveals the item x in the clear! Suppose the sender simply dropped x, then one can argue that x is computationally hidden from the receiver. This requires a formal argument, but intuitively, $(\mathsf{ct}_1, \mathsf{ct}_2)$ are uniformly random group elements. If a generic adversary managed to distinguish between encryptions to x and x', then they must have obtained a handle for ct_3. This would allow us to extract a BLS signature and KZG opening proof for the corresponding item. Which means that either the item was in the receiver's input and the receiver has a signature on it, or we are able to break a computational assumption related to the security of either the KZG commitment scheme or the BLS signature. Although we managed to hide x, we have inadvertently created other issues:

- First, the receiver does not know which item the sender encrypted to. So how does the receiver know which witness (proof-signature pair) to use for decryption? And how do they know that decryption was successful?
- Previously the item x was part of the statement being proved by the sender. But the sender now wishes to hide it, and hence must be a part of the witness. But this would require a proof that the random oracle was correctly evaluated on x. Although one can explicitly instantiate the random oracle with a concrete hash function like SHA256, this would only provide heuristic security guarantees. Moreover, the proof would be significantly more expensive to generate.

For simplicity, let's focus on the semi-honest setting. A natural first attempt is to have the receiver simply run Dec one-by-one for all items $x \in D$, assuming that the sender encrypted with respect to x. But observe that for any item $x' \in D$ that the receiver has a valid signature for, the Dec algorithm will always recover some payload in $\mathbb{G}_T$. Since the payload could in fact be any element in $\mathbb{G}_T$, the receiver has no way of knowing which run of Dec (if any) actually recovered the correct payload. Thus, we need a *robust* Sen algorithm that allows the receiver to learn the item x used by the sender. But this should be possible if and only if the receiver has a signature on x and $x \in D$.

We propose a simple modification to the Sen algorithm in Fig. 3 to achieve this. The sender creates two copies of their message designated for the same item x but with different payloads. The first copy encrypts the payload $m \in \mathbb{G}_T$ as before, but the second copy encrypts a canonical target group element (say $\mathbb{1}_T$). The receiver now runs Dec one-by-one for all items in $x \in D$, assuming that the sender encrypted with respect to x. If the receiver finds some $\hat{x} \in D$ where the recovered payload is $\mathbb{1}_T$, then the receiver simply decrypts the former ciphertext by running Dec assuming that the sender encrypted to $\hat{x}$.

We note here that the receiver's run time now grows with size of the database. If we demand the receiver's run time is also small (say $\mathsf{poly}(\log |D|)$), then as shown in [BGJP23], this implies doubly-efficient private information retrieval (DEPIR) [BIM00, CHR17, BIPW17, LMW23]. Achieving small run-time (concretely efficient) for the receiver in element hiding PSI is an interesting open problem for future work.

With malicious senders, an additional issue arises. It's possible that the sender can pick *bad randomness* such that the receiver outputs a different payload $m' \neq m$. In the full proof of security we show that a) a malicious receiver cannot learn x unless they have a valid signature on x and $x \in D$, and b) it is *computationally infeasible* for the sender to pick *bad randomness* such that the receiver outputs a different payload $m' \neq m$.

Coming to the latter issue, we are now forced to change RecVer algorithm. Following the same strategy as earlier, we need a *random oracle free* signature on the item x, which admits an efficient witness encryption to the relation: "I know a signature σ on item x under verification key rvk". However, this is not the only requirement. Note that the corresponding ciphertext may actually reveal information about x. As it turns out, the BF-IBE came with an additional special property – *anonymity*, which guarantees that the IBE ciphertext does not reveal any information about the intended recipient's identity. In the context of

our PSI protocols, the anonymity translates to hiding the item x from the receiver, and will be crucial to achieving element hiding.

$\Pi_{\mathsf{PSI-EH}}$

Parameters: A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with generators $g \in \mathbb{G}_1$, and $h \in \mathbb{G}_2$. Receiver verification key rvk, and Sender verification key svk.

Gen(1^λ): Sample $\tau \leftarrow\!\!\$\ \mathbb{F}$ and set $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (g, g^\tau, \ldots, g^{\tau^{n+1}}, h, h^\tau)$. Run the setup for the non-interactive zero-knowledge argument of knowledge and set $\mathsf{NIZK}_{\mathsf{crs}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Output $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$.

RecVer$(\mathsf{rvk}, x, \sigma_x)$: If $e(g,h) = e(\sigma_x, \mathsf{rvk} \cdot h^{-x})$, output 1, else output 0. Note that the (unique) valid signature on x is $\sigma_x = g^{1/(\mathsf{rsk}-x)}$.

Rec(pp, D): Interpolate a polynomial $f(X)$ of degree at most $n+1$ such that $f(D[i]) = 0$ for all $i \in [n]$, and $f(0) = 0$. Sample a random value $r \neq 0 \leftarrow\!\!\$\ \mathbb{F}$ and compute $\mathsf{com} \leftarrow g^{r \cdot f(\tau)}$ and opening proofs at all points $\Pi = \{\pi_i \leftarrow g^{r \cdot q(\tau)}; q(X) \coloneqq f(X)/(X - D[i])\}_{i \in [n]}$, $\pi_0 \leftarrow g^{r \cdot f(\tau)/\tau}$. Output $\mathsf{msg}_R \coloneqq (\mathsf{com}, \pi_0)$ and $\mathsf{st} = \Pi$.

Sen$(\mathsf{pp}, \mathsf{svk}, \mathsf{msg}_R, x, m)$: Parse $\mathsf{msg}_R = (\mathsf{com}, \pi_0)$. Abort if $\mathsf{com} = \mathbb{1}_{\mathbb{G}_1}$ or $x = 0$ or if $e(\mathsf{com}, h) \neq e(\pi_0, h^\tau)$. Sample $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 \leftarrow\!\!\$\ \mathbb{F} \setminus \{0\}$ and set $\mathsf{msg}_S = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5, \mathsf{ct}_6, \pi)$, where

- $\mathsf{ct}_1 = h^{(\tau - x)\alpha_1}$,
- $\mathsf{ct}_2 = (\mathsf{rvk} \cdot h^{-x})^{\beta_1}$
- $\mathsf{ct}_3 = e(\mathsf{com}, h)^{\alpha_1} \cdot e(g,h)^{\beta_1} \cdot m$
- $\mathsf{ct}_4 = h^{(\tau - x)\alpha_2}$,
- $\mathsf{ct}_5 = (\mathsf{rvk} \cdot h^{-x})^{\beta_2}$,
- $\mathsf{ct}_6 = e(\mathsf{com}, h)^{\alpha_2} \cdot e(g,h)^{\beta_2}$

along with a NIZK proof π of well-formedness:

$$\pi = \mathsf{NIZK.Prove}\{m, x, \alpha_1, \alpha_2, \beta_1, \beta_2, \sigma_{(x,m)} \mid \mathsf{SenVer}(\mathsf{svk}, (x,m), \sigma_{(x,m)}) = 1 \wedge \mathsf{ct}_1 = (h^{\tau - x})^{\alpha_1} \wedge \mathsf{ct}_2 = (\mathsf{rvk} \cdot h^{-x})^{\beta_1} \wedge \mathsf{ct}_3 \leftarrow e(\mathsf{com}, h)^{\alpha_1} \cdot e(g,h)^{\beta_1} \cdot m \wedge \mathsf{ct}_4 = (h^{\tau - x})^{\alpha_2} \wedge \mathsf{ct}_5 = (\mathsf{rvk} \cdot h^{-x})^{\beta_2} \wedge \mathsf{ct}_6 \leftarrow e(\mathsf{com}, h)^{\alpha_2} \cdot e(g,h)^{\beta_2}\}$$

Dec$(\mathsf{st}, \mathsf{msg}_S, \Sigma)$: Parse $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5, \mathsf{ct}_6, \pi)$. If the NIZK proof does not verify output $\perp$. For each $x \in D$:

- If $\mathsf{ct}_4 = \mathbb{1}_{\mathbb{G}_2}$ or $\mathsf{ct}_5 = \mathbb{1}_{\mathbb{G}_2}$, output $\perp$.
- Check whether $e(\Pi_x, \mathsf{ct}_4) \cdot e(\sigma_x, \mathsf{ct}_5) \stackrel{?}{=} \mathsf{ct}_6$, where $\Pi_x \coloneqq g^{q(\tau)}$ for $q(X) \coloneqq f(X)/(X - x)$ and $\sigma_x \in \Sigma$ is the signature on x under rvk. If the check passes, output $m \leftarrow \mathsf{ct}_3 \cdot e(\Pi_x, \mathsf{ct}_1)^{-1} \cdot e(\sigma_x, \mathsf{ct}_2)^{-1}$. Else, try again with the next item $x \in D$.
- If no item $x \in D$ satisfies the above check, output $\perp$.

Figure 5: A PSI protocol with Element Hiding emulating $\mathcal{F}_{\mathsf{PSI-EH}}$.

Theorem 6. *Let* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$ *be defined as* $e(\sigma, \mathsf{rvk} \cdot g^x) \stackrel{?}{=} e(g, h)$*, where* rvk *is the verification key. The protocol in Fig. 5 securely emulates (Definition 7) the* $\mathcal{F}_{\mathsf{PSI-EH}}$ *functionality (Fig. 2) with respect to the receiver verification predicate* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$*, against generic non-uniform PPT adversaries corrupting either the sender or the receiver.*

A black-box instantiation of the sender's proof. Next, we formulate a *black-box* instantiation of the sender's NIZK in the element-hiding protocol. For this, we assume an additional random group 1 generator denoted ℓ. Then, the sender's message includes the ciphertext components

- $\mathsf{ct}_1 = h^{(\tau - x)\alpha_1}$,
- $\mathsf{ct}_2 = (\mathsf{rvk} \cdot h^{-x})^{\beta_1}$, $\mathsf{ct}'_2 = \ell^{\beta_1}$,
- $\mathsf{ct}_3 = e(\mathsf{com}, h)^{\alpha_1} \cdot e(g, h)^{\beta_1} \cdot m$
- $\mathsf{ct}_4 = h^{(\tau - x)\alpha_2}$,
- $\mathsf{ct}_5 = (\mathsf{rvk} \cdot h^{-x})^{\beta_2}$,
- $\mathsf{ct}_{6,1} = \mathsf{com}^{\alpha_2} \cdot g^{\beta_2} \cdot \ell^{\gamma}$, $\mathsf{ct}_{6,2} = e(\ell, h)^{\gamma}$,

along with a proof π, where $\gamma \leftarrow \mathbb{F}$ is an additional random field element. Note that we have added a component to ct'_2, which will help with extraction, and we have split ct_6 into two parts. In Fig. 5, one can view ct_6 as an encryption of the "0" message in the target group, while here, one can view $(\mathsf{ct}_{6,1}, \mathsf{ct}_{6,2})$ as an encryption of a *random* message ℓ^{γ}, where one component is in group 1 and the other is in the target group. The reason we put one component in group 1 is to allow its exponents to be included in the witness for a GS proof, and the reason we replace the 0 message with a random message is to ensure zero-knowledge. Now, the proof π consists of the following components.

- $\pi_1 = \mathsf{Prove}_{\mathsf{Rep}}((h^{\tau}, h, \mathsf{ct}_1), (\alpha_1, -x\alpha_1))$
- $\pi_2 = \mathsf{Prove}_{\mathsf{DLog}}((\ell, \mathsf{ct}'_2), \beta_1)$.
- $\pi_3 = \mathsf{Prove}_{\mathsf{Rep}}((\mathsf{com}, g, \ell, \mathsf{ct}_{6,1}), (\alpha_2, \beta_2, \gamma))$
- $\pi_4 = \mathsf{Prove}_{\mathsf{DLog}}((e(\ell, h), \mathsf{ct}_{6,2}), \gamma)$
- π_5 is a GS proof with constants $(g, \ell, \mathsf{com}, h, h^{\tau}, \mathsf{rvk}, \mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}'_2, \mathsf{ct}_4, \mathsf{ct}_5, \mathsf{ct}_{6,1})$, variables $(\alpha_1, \alpha'_1, \beta_1, \beta'_1, \alpha_2, \alpha'_2, \beta_2, \beta'_2, \gamma)$, and equations
  - $(h^{\tau})^{\alpha_1} \cdot h^{\alpha'_1} = \mathsf{ct}_1$, $\mathsf{rvk}^{\beta_1} \cdot h^{\beta'_1} = \mathsf{ct}_2$, $\ell^{\beta_1} = \mathsf{ct}'_2$, $(h^{\tau})^{\alpha_2} \cdot h^{\alpha'_2} = \mathsf{ct}_4$, $\mathsf{rvk}^{\beta_2} \cdot h^{\beta'_2} = \mathsf{ct}_5$, $\mathsf{com}^{\alpha_2} \cdot g^{\beta_2} \cdot \ell^{\gamma} = \mathsf{ct}_{6,1}$
  - $\alpha'_1/\alpha_1 = \beta'_1/\beta_1 = \alpha'_2/\alpha_2 = \beta'_2/\beta_2$

It remains to show both zero-knowledge and knowledge soundness, which we sketch below.

- **Zero-knowledge.** We will argue zero-knowledge by showing that, starting with a sender's message $(\mathsf{ct}_1, \ldots, \mathsf{ct}_6)$ from Fig. 5, there is a procedure that produces a distribution indistinguishable from that specified above. The procedure keeps $\mathsf{ct}_1, \ldots, \mathsf{ct}_5$ the same, samples a uniformly random group element ct'_2, samples a uniformly random γ and sets $\mathsf{ct}_{6,1} = \ell^{\gamma}$, $\mathsf{ct}_{6,2} = \mathsf{ct}_6^{-1} \cdot e(\ell, h)^{\gamma}$, and simulates the proofs $\pi_1, \ldots, \pi_5$. To complete the proof, we have that:
  - By a standard application of DDH, ct'_2 is computationally indistinguishable from an independent random group element.
  - $\mathsf{ct}_{6,1}, \mathsf{ct}_{6,2}$ are uniformly random conditioned on $e(\mathsf{ct}_{6,1}, h) \cdot \mathsf{ct}_{6,2}^{-1} = e(\mathsf{com}, h)^{\alpha_2} \cdot e(g, h)^{\beta_1}$, so these are identically distributed to the honest distribution.
  - The simulated proofs $\pi_1, \ldots, \pi_5$ are computationally indistinguishable from the honest proofs, due to the zero-knowledge of the proof systems.

- **Knowledge soundness.** We argue knowledge soundness as follows.
  - If the GS proof verifies, then we know that $\mathrm{ct}_1, \mathrm{ct}_2, \mathrm{ct}_2', \mathrm{ct}_4, \mathrm{ct}_5, \mathrm{ct}_{6,1}$ are well-formed, in the sense that there exists an $(x, \alpha_1, \beta_1, \alpha_2, \beta_2, \gamma)$ such that the honest relations hold.
  - This implies that there is at most one choice of x for which the receiver's check $e(\Pi_x, \mathrm{ct}_4) \cdot e(\sigma_x, \mathrm{ct}_5) \cdot \mathrm{ct}_{6,2} \stackrel{?}{=} e(\mathrm{ct}_{6,1}, h)$ will pass. This can be argued as follows. If the receiver uses an $x' \neq x$, then the check is zero-testing a non-zero polynomial of τ, which will fail. Otherwise, if the receiver uses the correct x, then everything cancels out except for the check $e(\ell^\gamma, h) = e(\ell, h)^\gamma$. Thus, if the sender has used the same γ for both, this will pass, and otherwise it will fail.
  - The two choices of γs can be extracted from π_3 and π_4, so the extractor can determine whether there exists a valid x or not. If not, the extractor sets the input to $\perp$. Otherwise, it continues.
  - Next, the extractor obtains x and α_1 by extracting from π_1, and then β_1 by extracting from π_2. Together these can be used to extract m from ct_3, completing the extraction of the input (x, m).

6 Set Pre-Constrained Group Signatures

In this section, we define and construct SPC group signatures.

6.1 Definitions

A set pre-constrained group signature (SPCGS) scheme Π_{SPCGS} is parameterized by a family of verification algorithms $\{\{\mathsf{ItemVer}_{\mathsf{ivk}}\}_{\mathsf{ivk}\in\mathsf{IVK}_\lambda}\}_{\lambda\in\mathbb{N}}$, where for each $\lambda \in \mathbb{N}$, IVK_λ is a set of public parameters. Typically, we think of ivk as the verification key for some signature scheme, and ItemVer is the verification algorithm, but in principle it could be an arbitrarily defined predicate. The SPCGS then consists of a tuple algorithms Setup, MasterKeyGen, KeyGen, Sign, Verify, Open whose syntax is given below. We refer to the party that runs Gen as the *group manager* GM, and the KeyGen protocol is run by GM and a client C. We will denote signatures meant for ItemVer (i.e. an "authority's" signature) with a tilde $\widetilde{\sigma}_m$ to differentiate them from signatures σ of the group signature scheme itself (i.e. a client's signature).

- $\mathsf{ItemVer}_{\mathsf{ivk}}(m, \widetilde{\sigma}_m) \to \{\top, \perp\}$ takes as input an item m and an authority's signature $\widetilde{\sigma}_m$ and either accepts or rejects.
- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: The setup algorithm takes as input the security parameter 1^λ and outputs public parameter pp.[11]
- $\mathsf{MasterKeyGen}(\mathsf{pp}, D) \to (\mathsf{mpk}, \mathsf{msk})$. The master key generation algorithm takes as input public parameters pp and a set D of items, and outputs a master public key mpk and a master secret key msk.
- $\mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle \to (\mathsf{pk}, \mathsf{sk})$. The KeyGen protocol is run by the group manager GM with input msk and a client C. It delivers an identity pk to both GM and C, and an identity signing key sk to C.
- $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m) \to \sigma$. The signing algorithm takes as input the master public key mpk, an identity signing key sk, and a message m, and outputs a client's signature σ.

[11] When arguing security, we may sometimes consider pp as including access to a random oracle or generic group interface, rather than explicitly writing these as separate interfaces.

- $\mathsf{Verify}(\mathsf{mpk}, m, \sigma) \to \{\top, \bot\}$. The verification algorithm takes as input the master public key mpk, a message m, and a client's signature σ, and outputs either $\top$ or $\bot$, indicating accept or reject.
- $\mathsf{Open}(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m) \to \{\mathsf{pk}, \bot\}$. The opening algorithm takes as input the master secret key msk, a message m, a client's signature σ, and an authority's signature $\widetilde{\sigma}_m$, and outputs either an identity pk or $\bot$.

We imagine using an SPC group signature scheme for encrypted messaging as follows. We assume that there is already a standard end-to-end encrypted messaging system in place, and the server additionally publishes mpk for the SPCGS scheme. Each client runs a KeyGen protocol with the server in order to obtain their identity pk and their secret key sk. Then, whenever they want to send a message m, they additionally compute a signature σ on m, and send the message (m, σ) *under the end-to-end encryption.* Any message received that does not have a properly verifying signature is immediately discarded by the client algorithm. Finally, if an honest client receives a pair (m, σ) for some illegal content m, they can report this to the server, who can run the Open algorithm in order to determine which identity produced the signature σ.

Definition 9 (Correctness)**.** *An* SPC *group signature scheme is correct if for any set of items D and $m \in D$, it holds with probability* $1 - \mathsf{negl}(\lambda)$ *over* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp}, D)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$*, and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$ *that* $\mathsf{Verify}(\mathsf{mpk}, m, \sigma) = \top$.

Next, we define a notion of *traceability*, which ensures that any signature on a message $m \in D$ that is accepted by the verification algorithm will leak the identity of the signer to the master secret key holder (as long as they also have an authority's signature $\widetilde{\sigma}_m$ on m).

Definition 10 (Traceability)**.** *An* SPC *group signature scheme is traceable if for any PPT adversary $\mathcal{A}$ and set of items D,*

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{mpk}, m, \sigma) = \top \\ \wedge\ (m, \mathsf{pk}) \notin \mathcal{M}_{\mathsf{Signed}} \\ \wedge\ \mathsf{ItemVer}(m, \widetilde{\sigma}_m) = \top \\ \wedge\ m \in D' \\ \wedge\ \mathsf{pk} = \mathsf{Open}(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m) \\ \wedge\ \mathsf{pk} \notin \mathcal{I}_{\mathsf{Adv}} \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{MasterKeyGen}(\mathsf{pp}, D) \\ (m, \sigma) \leftarrow \mathcal{A}^{O_{\mathsf{AKG}}, O_{\mathsf{Open}}, O_{\mathsf{HKG}}, O_{\mathsf{Sign}}} \end{array}\right] = \mathsf{negl}(\lambda),$$

where the oracles $O_{\mathsf{AKG}}, O_{\mathsf{Open}}, O_{\mathsf{HKG}}, O_{\mathsf{Sign}}$, and sets $\mathcal{M}_{\mathsf{Signed}}, \mathcal{I}_{\mathsf{Adv}}$ are defined as follows.

- O_{AKG} *(KeyGen initiated by the Adversary) has* msk *hard-coded and, when initialized, acts as the group manager in the* KeyGen *protocol. Define $\mathcal{I}_{\mathsf{Adv}}$ to be the set of identities obtained by* $\mathsf{GM}(\mathsf{msk})$ *as a result of the interactions between $\mathcal{A}$ and O_{AKG}.*
- O_{Open} *has* msk *hard-coded, and on input a message m, client's signature σ and authority's signature $\widetilde{\sigma}_m$, outputs* $\mathsf{Open}(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m)$.
- O_{HKG} *(KeyGen initiated by an Honest party) has* msk *hard-coded and, when queried, runs* $\mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle \to (\mathsf{pk}, \mathsf{sk})$*, and returns* pk *(and not* sk*). Define $\mathcal{I}_{\mathsf{Hon}}$ to be the set of* pk*'s output by O_{HKG}.*
- O_{Sign} *takes a message m and an identity* pk *as input. If* $\mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}$*, let* sk *be the secret key associated with* pk *and return* $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$*. Otherwise, return nothing. Define $\mathcal{M}_{\mathsf{Signed}}$ to be the set of (m, pk) queried to O_{Sign}.*

Next, we define the notion of *unframeability*, which ensures that an adversary cannot produce a verifying signature with respect to some identity pk for which they do not hold the corresponding sk, even if they know the master secret key.

Definition 11 (Unframeability)**.** *An* SPC *group signature scheme satisfies unframeability if for any PPT adversary $\mathcal{A}$ and set of items D,*

$$\Pr\left[\begin{array}{l}\mathsf{Verify}(\mathsf{mpk}, m, \sigma) = \top \\ \wedge\ (\mathsf{mpk}, \mathsf{msk}) \in \mathsf{Gen}(\mathsf{pp}, D) \\ \wedge\ \mathsf{pk} = \mathsf{Open}(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m) \\ \wedge\ (m, \mathsf{pk}) \notin \mathcal{M}_{\mathsf{Signed}} \\ \wedge\ \mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}\end{array} : \begin{array}{r}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}, m, \sigma) \leftarrow \mathcal{A}^{O_{\mathsf{HKG}}, O_{\mathsf{Sign}}}(\mathsf{pp}, D)\end{array}\right] = \mathsf{negl}(\lambda),$$

where the oracles $O_{\mathsf{HKG}}, O_{\mathsf{Sign}}$ and sets $\mathcal{I}_{\mathsf{Hon}}, \mathcal{M}_{\mathsf{Signed}}$ are defined as in Definition 10, and $\widetilde{\sigma}_m$ is any string such that $\mathsf{ItemVer}(m, \widetilde{\sigma}_m) = \top$.

Now, we consider the notion of *anonymity*, which protects the identity of any signer who produces a signature on a message $m \notin D$, even against the group manager. Here, we will formulate a simulation-based definition. The ideal functionality $\mathcal{F}_{\mathsf{Anon}}$ interacts with a *group manager* GM who runs MasterKeyGen, interacts in KeyGen, and runs Open, and potentially multiple *clients*, who interact in KeyGen and runs Sign. Each client's input consists of an identity pk and message m. The client should learn nothing about D and the group manager should only learn the identity pk if $m \in D$ and they have an authority's signature $\widetilde{\sigma}_m$ on m. Following the formalism set up in Section 4.2, we have the following definition.

Definition 12 (Anonymity)**.** *An SPC group signature scheme satisfies anonymity if for any PPT adversary corrupting either the group manager* GM *or a set of clients, and any PPT distinguisher $\mathcal{D}$, there exists a PPT* Sim *such that for any PPT environment $\mathcal{E}$,*

$$\left|\Pr\left[\mathcal{D}\left(\mathsf{REAL}_{\mathcal{E},\mathcal{A}}(1^\lambda)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{Anon}},\mathcal{E},\mathsf{Sim}}(1^\lambda)\right) = 1\right]\right| = \mathsf{negl}(\lambda),$$

where $\mathcal{F}_{\mathsf{Anon}}$ is defined in Fig. 6.

Ideal functionality $\mathcal{F}_{\mathsf{Anon}}$ for defining anonymity in SPCGS

Parameters: Upper bound n on the database size and parameter ivk.

- Upon receiving a message $(\mathsf{GM}, \mathsf{sid}, D)$, record (sid, D) if $|D| \leq n$.
- Upon receiving a message $(\mathsf{client}, \mathsf{sid}, \mathsf{ssid}, \mathsf{pk}, m)$, record $(\mathsf{sid}, \mathsf{ssid}, \mathsf{pk}, m)$ and send $(\mathsf{sid}, \mathsf{ssid}, m)$ to the group manager.
- Upon receiving a message $(\mathsf{GM}, \mathsf{sid}, m, \widetilde{\sigma}_m)$, check that there exists a record (sid, D), that $m \in D$, that $\mathsf{ItemVer}_{\mathsf{ivk}}(m, \widetilde{\sigma}_m) = 1$, and that there exists a record $(\mathsf{sid}, \mathsf{ssid}, \mathsf{pk}, m)$. If so, send pk to the group manager.

Figure 6: Ideal functionality for the defining the property of *anonymity* in set pre-constrained group signatures.

Finally, we consider "client-client" anonymity, which considers the security of signatures against other clients. Here, we can hope for stronger security properties, since clients do not hold the master secret key

and thus might not be able to de-anonymize signatures even on messages $m \in D$. Thus, we give a separate (game-based) definition of anonymity against adversarial clients.

Definition 13 (Client-client anonymity)**.** *An* SPC *group signature scheme satisfies client-client anonymity if for any PPT adversary $\mathcal{A}$, set of items D, and message m, it holds that with probability* $1 - \mathsf{negl}(\lambda)$ *over* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(\mathsf{pp}, D)$, $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$, *and* $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$,

$$\Pr\left[\mathcal{A}^{O_{\mathsf{AKG}}, O_{\mathsf{HKG}}, O_{\mathsf{Sign}}}\left(\mathsf{mpk}, \mathsf{pk}_0, \mathsf{pk}_1, \sigma \right) = b \;:\; \begin{array}{r} b \leftarrow \{0,1\} \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_b, m) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the oracles $O_{\mathsf{AKG}}, O_{\mathsf{HKG}}$, and O_{Sign} are defined as in Definition 10.

6.2 Generic Construction

Adapting the construction from [BGJP23, Section 4.2], we show how to construct an SPC group signature scheme generically from PSI on authenticated inputs, plus a few standard cryptographic tools. Our construction is given in the random oracle model, though we note that if we were willing to assume an additional *simulation-soundness* property of the ZK-NIAoK, then we would not require a random oracle. It is presented in Protocol 7.

Ingredients:

- PSI on authenticated inputs (Gen, Rec, Sen, Dec) (Definition 4) that securely emulates $\mathcal{F}_{\mathsf{PSI}}$ and satisfies *perfect correctness*.
- A one-way relation $(\mathcal{R}, \mathcal{R}.\mathsf{Gen}, \mathcal{R}.\mathsf{Sample})$ (Section 3.1).
- An EUF-CMA secure signature scheme $\mathsf{Sig} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ (Section 3.1).
- A ZK-NIAoK scheme $\mathsf{ZK} = (\mathsf{ZK.Setup}, \mathsf{ZK.Prove}, \mathsf{ZK.Verify})$ in the common random string model for general NP relations (Section 3.3).
- A random oracle H.

Then, we have the following theorem, which follows along the same lines as [BGJP23, Theorem 4]. This proof is given in Appendix A.3.

Theorem 7. *The construction given in Fig. 7 satisfies correctness, traceability, unframeability, anonymity, and client-client anonymity.*

6.3 Concrete Instantiation

Next, we describe a concrete instantiation of laconic SPC group signatures, building on our non-element-hiding construction of laconic PSI (Fig. 3). Our construction is given in Fig. 8, and uses a structure preserving signature Section 3.5 for authentication of sender inputs and a BLS signature for authentication of receiver inputs.

Concrete Instantiation of the NIZK. We use sigma protocols in combination with Groth-Sahai proofs to prove a) knowledge of randomness used to prepare the ciphertext b) knowledge of the client's secret key and c) knowledge of a valid signature on the client's public key. This is necessary due to Groth-Sahai proofs not being able to provide extractions of witness values that are field elements.

Π_{SPCGS}

Setup: $\mathsf{pp}_{\mathsf{PSI}} \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{pp}_{\mathcal{R}} \leftarrow \mathcal{R}.\mathsf{Gen}(1^\lambda)$, and a random oracle H. Let $\mathsf{pp} := (\mathsf{pp}_{\mathsf{PSI}}, \mathsf{pp}_{\mathcal{R}}, H)$. We also define $\mathsf{ItemVer}_{\mathsf{ivk}} := \mathsf{RecVer}_{\mathsf{rvk}}$.

MasterKeyGen(pp, D): run $(\mathsf{msg}_{R,\mathsf{PSI}}, \mathsf{st}_{\mathsf{PSI}}) \leftarrow \mathsf{Rec}(\mathsf{pp}_{\mathsf{PSI}}, D)$ and $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$. Set $\mathsf{mpk} := \mathsf{msg}_{R,\mathsf{PSI}}$, $\mathsf{msk} := (\mathsf{st}_{\mathsf{PSI}}, \mathsf{sk}_{\mathsf{Sig}})$, $\mathsf{svk} := \mathsf{vk}_{\mathsf{Sig}}$, and define $\mathsf{SenVer}_{\mathsf{svk}}((m, \mathsf{pk}), \mathsf{pk}, \sigma_{\mathsf{pk}}) := \mathsf{Sig.Verify}(\mathsf{svk}, \sigma_{\mathsf{pk}})$.

KeyGen$\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$: the client C samples random coins s, computes $(\mathsf{pk}, w) := \mathcal{R}.\mathsf{Sample}(\mathsf{pp}_{\mathcal{R}}; s)$, and sends pk to GM. GM parses msk as $(\mathsf{st}_{\mathsf{PSI}}, \mathsf{sk}_{\mathsf{Sig}})$ and then computes and sends $\sigma_{\mathsf{pk}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{pk})$. C sets $\mathsf{sk} := (s, \sigma_{\mathsf{pk}})$.

Sign$(\mathsf{mpk}, \mathsf{sk}, m)$: parse mpk as $\mathsf{msg}_{R,\mathsf{PSI}}$ and sk as $(s, \sigma_{\mathsf{pk}})$, compute $(\mathsf{pk}, w) := \mathcal{R}.\mathsf{Sample}(\mathsf{pp}_{\mathcal{R}}; s)$, sample random coins r, and compute $\mathsf{ct} := \mathsf{Sen}(\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{pk}, \sigma_{\mathsf{pk}}; r)$. Let $\mathsf{crs} := H(m, \mathsf{ct})$, and compute $\pi \leftarrow \mathsf{ZK.Prove}(\mathsf{crs}, (\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{ct}), (\mathsf{pk}, s, w, \sigma_{\mathsf{pk}}, r))$ for the relation that checks that

- $\mathsf{ct} = \mathsf{Sen}(\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{pk}, \sigma_{\mathsf{pk}}; r)$
- and $(\mathsf{pk}, w) = \mathcal{R}.\mathsf{Sample}(\mathsf{pp}_{\mathcal{R}}; s)$.

Output $\sigma := (\mathsf{ct}, \mathsf{crs}, \pi)$.

Verify$(\mathsf{mpk}, m, \sigma)$: parse mpk as $\mathsf{msg}_{R,\mathsf{PSI}}$ and σ as $(\mathsf{ct}, \mathsf{crs}, \pi)$, check that $H(m, \mathsf{ct}) = \mathsf{crs}$ and if so output $\mathsf{ZK.Verify}(\mathsf{crs}, (\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{ct}), \pi)$.

Open$(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m)$: parse msk as $(\mathsf{st}_{\mathsf{PSI}}, \mathsf{sk}_{\mathsf{Sig}})$ and σ as $(\mathsf{ct}, \mathsf{crs}, \pi)$, and output $\mathsf{Dec}(\mathsf{st}_{\mathsf{PSI}}, \mathsf{ct}, \{(m, \widetilde{\sigma}_m)\})$.

Figure 7: Generic construction of SPC group signatures.

To prove well-formedness the client will use a a standard sigma protocol (across groups) to prove the following relation:

$$\{\alpha, \beta, b, \mid \mathsf{ct}_1 = (h^{\tau - x})^\alpha \wedge \mathsf{ct}_2 = h^\beta \wedge \mathsf{ct}_3 \leftarrow e(\mathsf{com}, h)^\alpha \cdot e(H(m), \mathsf{rvk})^\beta \cdot e(\ell, h)^b\}.$$

This allows us to extract the randomness used in the ciphertext, and the secret witness b corresponding to the client's public key $\mathsf{pk} = \ell^b$. Next, we use Groth-Sahai proofs for the following relation

$$\begin{aligned}\pi = \mathsf{NIZK.Prove}\{\alpha, \beta, b, \mathsf{pk}, \sigma_{\mathsf{pk}}, h_\alpha, \mathsf{rvk}_\beta \mid \mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}, \sigma_{\mathsf{pk}}) = 1 \\ \wedge \mathsf{ct}_1 = (h^{\tau - x})^\alpha \wedge \mathsf{ct}_2 = h^\beta \wedge \mathsf{ct}_3 \leftarrow e(\mathsf{com}, h_\alpha) \cdot e(H(m), \mathsf{rvk}_\beta) \cdot e(\mathsf{pk}, h) \\ \wedge \mathsf{pk} = \ell^b \wedge h_\alpha = h^\alpha \wedge \mathsf{rvk}_\beta = \mathsf{rvk}^\beta\}\end{aligned}$$

where Sig.Verify is the verification equation in Section 3.5. This allows us to extract a valid signature on the client's public key $\mathsf{pk} = \ell^b$.

Π_{SPCGS}

Setup: A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with (random) generators $g, \ell \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. Receiver verification key rvk. Random oracle $H : \{0,1\}^* \to \mathbb{G}_1$. Parameters from PSI Gen, which include $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (g^\tau, \ldots, g^{\tau^{n+1}}, h^\tau)$ and $\mathsf{NIZK}_{\mathsf{crs}} \leftarrow \mathsf{NIZK.Setup}(\lambda)$ for GS proofs (see Section 3.4). An additional random generator ℓ serving as the public parameters for the discrete log one-way function.

ItemVer$(\mathsf{rvk}, x, \sigma_x)$: If $e(H(x), \mathsf{rvk}) = e(\sigma_x, h)$, output 1, else output 0.

MasterKeyGen(pp, D): Interpolate a polynomial $f(X)$ of degree at most n such that $f(D[i]) = 0$ for all $i \in [n]$. Sample a random value $r \neq 0 \leftarrow\!\!\$\ \mathbb{F}$ and compute $\mathsf{com} \leftarrow g^{r \cdot f(\tau)}$ and opening proofs at all points $\Pi_{D[i]} = \{\pi_i \leftarrow g^{r \cdot q(\tau)}; q(X) := f(X)/(X - D[i])\}_{i \in [n]}$, $\pi_0 \leftarrow g^{r \cdot f(\tau)/\tau}$. Generate parameters $(\mathsf{sk}_{\mathsf{Sig}}, \mathsf{vk}_{\mathsf{Sig}})$ for Sig, a structure-preserving signature (see Section 3.5) on elements in $\mathbb{G}_1$. Output $\mathsf{mpk} := (\mathsf{com}, \pi_0, \mathsf{vk}_{\mathsf{Sig}})$ and $\mathsf{msk} = (\Pi, \mathsf{sk}_{\mathsf{Sig}})$.

KeyGen$\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C} \rangle$: C samples $b \leftarrow\!\!\$\ \mathbb{F}$, computes ℓ^b, and sends ℓ^b to GM. GM parses $\mathsf{msk} = (\Pi, \mathsf{sk}_{\mathsf{Sig}})$ and sends $\sigma_{\mathsf{pk}} \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_{\mathsf{Sig}}, \ell^b)$. C sets $\mathsf{pk} = \ell^b$ and $\mathsf{sk} := (b, \sigma_{\mathsf{pk}})$.

Sign$(\mathsf{mpk}, \mathsf{sk}, m)$: Parse $\mathsf{mpk} = (\mathsf{com}, \pi_0, \mathsf{vk}_{\mathsf{Sig}})$ and $\mathsf{sk} = (b, \sigma_{\mathsf{pk}})$. Abort if $\mathsf{com} = \mathbb{1}_{\mathbb{G}_1}$, $x = 0$, or $e(\mathsf{com}, h) \neq e(\pi_0, h^\tau)$. Sample $\alpha, \beta \leftarrow\!\!\$\ \mathbb{F}$ and set $\sigma_m = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$, where

- $\mathsf{ct}_1 \leftarrow h^{(\tau - m)\alpha}$
- $\mathsf{ct}_2 \leftarrow h^\beta$
- $\mathsf{ct}_3 \leftarrow e(\mathsf{com}, h)^\alpha \cdot e(H(m), \mathsf{rvk})^\beta \cdot e(\ell^b, h)$

along with a NIZK proof π of well-formedness:

$$\pi = \mathsf{NIZK.Prove}\{\alpha, \beta, b, \sigma_{\mathsf{pk}} \mid \mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \ell^b, \sigma_{\mathsf{pk}}) = 1 \\ \wedge\, \mathsf{ct}_1 = (h^{\tau - x})^\alpha \wedge \mathsf{ct}_2 = h^\beta \wedge \mathsf{ct}_3 \leftarrow e(\mathsf{com}, h)^\alpha \cdot e(H(m), \mathsf{rvk})^\beta \cdot e(\ell^b, h)\}$$

Output $\sigma_m = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$.

Verify$(\mathsf{mpk}, m, \sigma_m)$: Parse $\sigma_m = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$ and check that the NIZK proof π passes verification.

Open$(\mathsf{msk}, m, \widetilde{\sigma}_m)$: Parse $\sigma_m = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \pi)$ and output

$$e(\mathsf{pk}, h) \leftarrow \mathsf{ct}_3 \cdot e(\Pi_m, \mathsf{ct}_1)^{-1} \cdot e(\widetilde{\sigma}_m, \mathsf{ct}_2)^{-1}.$$

Figure 8: Efficient instantiation of SPC group signatures.

References

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor,

CRYPTO 2010, volume 6223 of *LNCS*, pages 209–236. Springer, Berlin, Heidelberg, August 2010.

[AJJM22] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Pre-constrained encryption. In Mark Braverman, editor, *ITCS 2022*, volume 215, pages 4:1–4:20. LIPIcs, January / February 2022.

[AKN24] Shweta Agrawal, Simran Kumari, and Ryo Nishimaki. Don't trust setup! new directions in pre-constrained cryptography. Cryptology ePrint Archive, Paper 2024/1294, 2024.

[ALOS22] Diego F. Aranha, Chuanwei Lin, Claudio Orlandi, and Mark Simkin. Laconic private set-intersection from pairings. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 111–124. ACM Press, November 2022.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Berlin, Heidelberg, May 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BBD+11] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011*, pages 691–702. ACM Press, October 2011.

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

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Berlin, Heidelberg, December 2016.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001.

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[BFKT24] Jan Bobolz, Pooya Farshim, Markulf Kohlweiss, and Akira Takahashi. The brave new world of global generic groups and UC-secure zero-overhead SNARKs. Cryptology ePrint Archive, Paper 2024/818, 2024.

[BGJP23] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 35–66. Springer, Cham, April 2023.

[BIM00] Amos Beimel, Yuval Ishai, and Tal Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 55–73. Springer, Berlin, Heidelberg, August 2000.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 662–693. Springer, Cham, November 2017.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

[CGPP24] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024.

[CGPW24] Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516, 2024.

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 694–726. Springer, Cham, November 2017.

[CZ09] Jan Camenisch and Gregory M. Zaverucha. Private intersection of certified sets. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 108–127. Springer, Berlin, Heidelberg, February 2009.

[DD15] Sumit Kumar Debnath and Ratna Dutta. Secure and efficient private set intersection cardinality using bloom filter. In Javier Lopez and Chris J. Mitchell, editors, *ISC 2015*, volume 9290 of *LNCS*, pages 209–226. Springer, Cham, September 2015.

[DEF+24] Bernardo David, Felix Engelmann, Tore Kasper Frederiksen, Markulf Kohlweiss, Elena Pagnin, and Mikhail Volkhov. Updatable privacy-preserving blueprints. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 105–139. Springer, Singapore, December 2024.

[DGPS24] Moumita Dutta, Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Compute, but verify: Efficient multiparty computation over authenticated inputs. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 133–166. Springer, Singapore, December 2024.

[DJKT09] Emiliano De Cristofaro, Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Privacy-preserving policy-based information transfer. In Ian Goldberg and Mikhail J. Atallah, editors, *PETS 2009*, volume 5672 of *LNCS*, pages 164–184. Springer, Berlin, Heidelberg, August 2009.

[DKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, Berlin, Heidelberg, December 2010.

[DNRS99] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, October 1999.

[DT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, Berlin, Heidelberg, January 2010.

[fDT21] Center for Democracy and Technology. International coalition calls on Apple to abandon plan to build surveillance capabilities into iPhones, iPads, and other products, 2021.

[FHAS24] Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for KZG commitments and efficient laconic OT. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 423–453. Springer, Singapore, December 2024.

[FK23] Dankrad Feist and Dmitry Khovratovich. Fast amortized KZG proofs. Cryptology ePrint Archive, Report 2023/033, 2023.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

[FM24] Francesca Falzon and Evangelia Anna Markatou. Re-visiting authorized private set intersection: A new privacy-preserving variant and two protocols. Cryptology ePrint Archive, Paper 2024/1579, 2024.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Berlin, Heidelberg, May 2004.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[Gha17] Essam Ghadafi. More efficient structure-preserving signatures - or: Bypassing the type-III lower bounds. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *ESORICS 2017, Part II*, volume 10493 of *LNCS*, pages 43–61. Springer, Cham, September 2017.

[GKLS24] Scott Griffy, Markulf Kohlweiss, Anna Lysyanskaya, and Meghna Sengupta. Privacy-preserving blueprints via succinctly verifiable computation over additively-homomorphically encrypted data. Cryptology ePrint Archive, Report 2024/675, 2024.

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 352–386. Springer, Cham, August 2024.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Berlin, Heidelberg, December 2007.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan Katz and Hovav

Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 158–189. Springer, Cham, August 2017.

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[KLN23] Markulf Kohlweiss, Anna Lysyanskaya, and An Nguyen. Privacy-preserving blueprints. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 594–625. Springer, Cham, April 2023.

[KM21] Anunay Kulshrestha and Jonathan R. Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 893–910. USENIX Association, August 2021.

[KS05] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Berlin, Heidelberg, August 2005.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 595–608. ACM Press, June 2023.

[MPGP09] Ghita Mezzour, Adrian Perrig, Virgil D. Gligor, and Panos Papadimitratos. Privacy-preserving relationship path discovery in social networks. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 189–208. Springer, Berlin, Heidelberg, December 2009.

[Nec94] Vassiliy Ilyich Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994.

[NMH+10] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *USENIX Security 2010*, pages 95–110. USENIX Association, August 2010.

[NTL+11] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS 2011*. The Internet Society, February 2011.

[Pos25] Washington Post. U.K. orders Apple to let it spy on users' encrypted accounts. 2025.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 797–812. USENIX Association, August 2014.

[RR17] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

[SSS12] Emil Stefanov, Elaine Shi, and Dawn Song. Policy-enhanced private set intersection: Sharing information while enforcing privacy policies. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 413–430. Springer, Berlin, Heidelberg, May 2012.

A Security Proofs

A.1 Strong Existential Unforgeability of the Boneh-Boyen basic signature scheme in the GGM

In [BB04, BB08], Boneh and Boyen present two signature schemes. The first is a (simpler) basic scheme that only satisfies weak chosen message attacks under q-SDH assumption – where the adversary has to pick all messages they want a signature on before seeing the public key. The second scheme satisfies strong existential unforgeability under chosen message attacks. In this section, we show that the simpler scheme also satisfies strong existential unforgeability under chosen message attacks in the GGM.

We begin by recalling the signature scheme. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, with generators g and h for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.

- Gen: The key generation algorithm samples a secret key $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$. The verification key is set to be $\mathsf{vk} = h^{\mathsf{sk}}$.
- $\mathsf{Sign}(m, \mathsf{sk})$: If $\mathsf{sk} - m = 0$, output $\perp$. Otherwise, the signature on a message $m \in \mathbb{Z}_p$ is computed as $\sigma = g^{1/(\mathsf{sk}-m)}$.
- $\mathsf{Ver}(m, \sigma, \mathsf{vk})$: The verification algorithm accepts if $e(\sigma, \mathsf{vk} \cdot h^m) = e(g, h)$.

Theorem 8. *The basic Boneh-Boyen signature scheme satisfies SUF-CMA security in the Generic Group Model.*

Proof. We will argue security in the symbolic GGM (Definition 14), and from Lemma 1 we know that this switch is indistinguishable to a computationally bounded (i.e. polynomial-query) adversary. At the beginning of the game the challenger will initialize $\mathsf{GGM.Init}(\mathbf{K})$ corresponding to the discrete log of the verification key, and obtains a label for it $\mathsf{vk} = \tau_2(\mathbf{K})$. The adversary $\mathcal{A}^{\mathsf{symGGM(vk)}}$ is given vk as input and for every signature request made by the adversary on a message m_i, the challenger will respond with $\sigma_i = \tau_1((\mathbf{K} - m_i)^{-1})$.[12]

We now argue that a signature on a message $m^* \notin \{m_i\}_{i=1}^q$ is *linearly independent* of all group elements in the adversary's view, and hence it is impossible for the adversary to forge a signature. From the verification equation, it is easy to see that there is exactly one possible signature for any given message m viz.

[12] Technically, this requires a straightforward extension of the symbolic GGM to supports *rational functions* of formal variables.

$\sigma = g^{1/(\mathbf{K}-m)}$. Now note that any signature that the adversary forges must be a linear combination of the signatures that the challenger has provided. That is

$$\sigma^* = \tau_1\left(\frac{1}{\mathbf{K}-m^*}\right) = \tau_1\left(\sum_{i=1}^{q} z_i \frac{1}{\mathbf{K}-m_i} + z_0\right)$$

where $m^* \notin \{m_i\}_{i=1}^q$ and z_i's are from $\mathbb{Z}_p$. This can be rewritten as

$$(\mathbf{K}-m^*)\left(\sum_{i=1}^{q} z_i L_i(\mathbf{K})\right) + (z_0(\mathbf{K}-m^*)-1)\left(\prod_{i=1}^{q}(\mathbf{K}-m_i)\right) = 0,$$

where $L_i(\mathbf{X}) = \prod_{i\neq j}(\mathbf{K}+m_j)$. Note that $L_i(-m_j) = 0$ for all $i \neq j$, and $L_i(m_i)$ is some non-zero value. Now observe that for the equality to hold z_0 must be zero, which then further simplifies to:

$$(\mathbf{K}-m^*)\left(\sum_{i=1}^{q} z_i L_i(\mathbf{K})\right) = \left(\prod_{i=1}^{q}(\mathbf{K}-m_i)\right).$$

Since the r.h.s evaluates to 0 at all points $\{m_i\}_{i=1}^q$, the only allowed solutions are of the form $z_i = 1$ for some i and $z_j = 0$ for all $j \neq i$, which in turn forces $m^* = m_i$. Thus there is no admissible forgery and hence the scheme is SUF-CMA secure. □

A.2 Security proofs of PSI protocols

In this section, we present proofs of security for our three PSI schemes. First, Theorem 4 establishes security of non-element-hiding perfectly-correct PSI (Fig. 3). Next, Theorem 5 establishes security of non-element-hiding ϵ-correct PSI (Fig. 4). Finally, Theorem 6 establishes security of element-hiding PSI (Fig. 5).

Theorem 4. *Let* $\mathsf{RecVer}_{\mathsf{rvk}}(x,\sigma_x)$ *be defined as* $e(H(x),\mathsf{rvk}) \stackrel{?}{=} e(\sigma_x,h)$. *The protocol in Fig. 3 securely emulates (Definition 7) the* $\mathcal{F}_{\mathsf{PSI}}$ *functionality with perfect correctness (Definition 5), with respect to the receiver verification predicate* $\mathsf{RecVer}_{\mathsf{rvk}}(x,\sigma_x)$, *against generic non-uniform PPT adversaries corrupting either the sender or the receiver in the random oracle model.*

Proof. It is easy to see that the scheme satisfies perfect correctness Definition 5, as for any database D and randomness $r \neq 0$ chosen by an honest receiver, and any item x, chosen by an honest sender, the sender's message $\mathsf{msg}_S = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3)$, fully determines the message m that was sent as the payload. In particular, $m := \mathsf{ct}_3 \cdot e(\mathsf{com},h)^{-\alpha} \cdot e(H(x),\mathsf{rvk})^{-\beta}$, where $\alpha = \mathsf{dlog}_{h^{t-x}}(\mathsf{ct}_1)$ and $\beta = \mathsf{dlog}_h(\mathsf{ct}_2)$.

Furthermore, note that Π_x and σ_x are the unique group elements that satisfy $e(H(x),\mathsf{rvk}) = e(\sigma_x,h)$, and $e(\mathsf{com},h) = e(\Pi_x, h^{(t-x)})$, respectively. Hence, if $x \in D$, which implies the receiver can compute Π_x and there exists $\sigma_x \in \Sigma$ such that $\mathsf{RecVer}(x,\sigma_x) = 1$, the receiver can always recover the message m sent by the sender by following the honest decryption algorithm. If $x \notin D$, or if there does not exist $\sigma_x \in \Sigma$ such that $\mathsf{RecVer}(x,\sigma_x) = 1$, then the receiver always outputs $\perp$.

Security. We define a sequence of simulators Sim_μ, where μ is any integer-valued function of the security parameter,[13] such that for any PPT distinguisher $\mathcal{D}$ and adversary $\mathcal{A}$ corrupting one of the parties C, we show that there exists a μ^* such that Sim_{μ^*} (with oracle access to $\mathcal{D}$ and $\mathcal{A}$) simulates all interaction between $\mathcal{A}$ and $\mathcal{F}_{\mathsf{PSI}}$ and produces a view $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},C}[1^\lambda, \mathsf{inp}_R, \mathsf{inp}_S, z]$ indistinguishable from the view of the adversary in the real world execution $\mathsf{REAL}_{\mathcal{A},C}[1^\lambda, \mathsf{inp}_R, \mathsf{inp}_S, z]$, where inp_R and inp_S are the inputs to the receiver and sender respectively, and z is auxiliary input to the adversary. In the remainder of the

[13] Looking ahead, μ will roughly correspond to the number of rewinding attempts that the simulator makes.

proof we will drop the subscript μ where obvious for brevity. We do not program the random oracle and only rely on Sim being able to observe all queries made.

To aid in the analysis we will use a *symbolic GGM model* [Sho97] in intermediate hybrids. Recall, that in the GGM defined in Definition 2, the GGM is first initialized using GGM.Init() and it samples injections from field elements to labels in the three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. In the symbolic GGM, we modify this as follows:

Definition 14 (Symbolic Bilinear Group Oracle)**.** *A symbolic bilinear group oracle is a stateful oracle* symGGM *parametrized by a prime $p \in \mathbb{Z}$, and three sets S_1, S_2, S_T of size p.*

- *Every party interacting with the oracle is assigned the role of* challenger *or* adversary*.*
- GGM.Init($\mathbf{X}$)*: As before, the oracle maintains three sets of labels S_1, S_2, S_T as before but they are now initially set to be empty and* lazily *sampled (while avoiding collisions). The domain for sampling labels is $\mathbb{Z}_p[\mathbf{X}_1, \ldots, \mathbf{X}_n]$ – the set of n-variate polynomials with coefficients in $\mathbb{Z}_p$, where $(\mathbf{X}_1, \ldots, \mathbf{X}_n)$ are the formal variables from previous* Init *requests. Any time* GGM.Init($\cdot$) *is called by the challenger, the domain is expanded to include the new variables, whilst remaining consistent with the previous queries viz. if* GGM.Init($\mathbf{X}_1$) *is the first query, then the oracle maps polynomials $\mathbb{Z}_p[\mathbf{X}_1] \to S_i$ for $i \in \{1, 2, T\}$, as queries are made. If followed by* GGM.Init($\mathbf{X}_2$)*, then the label sampling domain is expanded to $\mathbb{Z}_p[\mathbf{X}_1, \mathbf{X}_2]$, but the labels for any previously sampled polynomials in $\mathbb{Z}_p[\mathbf{X}_1]$ remain the same. If this query is made by the adversary, the oracle returns $\perp$.*
- *On a query* GGM.GetGenerator(i)*, the oracle returns the label sampled corresponding to the constant polynomial 1 in the set S_i.*
- *On a query* GGM.Add(i, g_1, g_2, c_1, c_2)*, return $\perp$, unless $g_1, g_2 \in S_i$, and $c_1, c_2 \in \mathbb{Z}_p$. Otherwise, retrieve the polynomials $f_1(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$, and $f_2(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$, corresponding to the labels g_1 and g_2 respectively. Compute $f = c_1 f_1 + c_2 f_2$ and return the label corresponding to f in S_i, if it already exists. Otherwise, sample a new label for f and return it. Optionally, if the challenger initialized some formal variable $\mathbf{X}$ say, it is allowed to use $\mathbf{X}$ in place of a scalars c_1/c_2. As a result, this now allows the challenger to also initialize higher powers of formal variables $\mathbf{X}^2, \mathbf{X}^3, \ldots$.*
- *On a query* GGM.Pair(g_1, g_2)*, return $\perp$ unless $g_1 \in S_1$, and $g_2 \in S_2$. Otherwise, retrieve the polynomials $f_1(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$, and $f_2(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$, corresponding to the labels g_1 and g_2 respectively. Compute $f = f_1 \cdot f_2$ and return the label corresponding to f in S_T, if it already exists. Otherwise, sample a new label for f and return it.*

Lemma 1. *Let $\boldsymbol{L} = (\boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}_T)$ be three sets of polynomials with elements in $\mathbb{Z}_p[\mathbf{X}_1, \ldots, \mathbf{X}_n]$, such that $|\boldsymbol{L}_1| + |\boldsymbol{L}_2| + |\boldsymbol{L}_T| = s$, and the maximum degree of the polynomials, across the three sets is d. We denote $\boldsymbol{L}_{\mathsf{GGM}} := (\tau_1(\boldsymbol{L}_1(\mathbf{x})), \tau_2(\boldsymbol{L}_2(\mathbf{x})), \tau_T(\boldsymbol{L}_1(\mathbf{x})))$ to be the labels corresponding to the field elements obtaining by evaluating the polynomials in $\boldsymbol{L}$ at $\mathbf{x}$ in* GGM*, where $\mathbf{x} \leftarrow\!\$\ \mathbb{F}^n$. Similarly, we will use $\boldsymbol{L}_{\mathsf{symGGM}} := ((\tau_1(\boldsymbol{L}_1), \tau_2(\boldsymbol{L}_2), \tau_T(\boldsymbol{L}_1)))$ to refer to the labels corresponding to the formal polynomials in the symbolic GGM. At the end of any interactive protocol between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, we have that*

$$\left|\Pr[\mathcal{A}^{\mathsf{GGM}}(\boldsymbol{L}_{\mathsf{GGM}}) \to 1] - \Pr[\mathcal{A}^{\mathsf{symGGM}}(\boldsymbol{L}_{\mathsf{symGGM}}) \to 1]\right| \leq \frac{2(q+s+2)^2 d}{p},$$

where $\mathcal{A}$ makes at most q queries (involving Add *or* Pair*) to the (symbolic) bilinear group oracle* (sym)GGM*.*

Proof. Note that when $\mathcal{A}$ interacts with symGGM, the labels corresponding to distinct polynomials are unique by definition. However, in the GGM, there is a possibility, over the choice of randomness $\mathbf{x}$, that

two distinct polynomials evaluate to the same value, and hence have the same label in GGM. If such an event does not occur, then an interaction with symGGM is perfectly indistinguishable from an interaction with GGM. We now bound the probability that such an event occurs. After q queries, there are at most $q+s+2$ polynomials that symGGM has sampled labels for. Thus for any pair of polynomials f, g sampled by symGGM, the probability that $f(\mathbf{x}) = g(\mathbf{x})$ is at most d/p by the Schwartz-Zippel lemma. Across the three different groups, there are no more than $3\binom{q+s+2}{2}$ possible pairs. Thus

$$\left|\Pr[\mathcal{A}^{\mathsf{GGM}}(\boldsymbol{L}_{\mathsf{GGM}}) \to 1] - \Pr[\mathcal{A}^{\mathsf{symGGM}}(\boldsymbol{L}_{\mathsf{symGGM}}) \to 1]\right| \leq \binom{q+s+2}{2}\frac{3d}{p} \leq \frac{2(q+s+2)^2 d}{p}$$

□

Corrupt Receiver. The environment (after interacting with the symbolic GGM) outputs labels corresponding to inputs to the protocol and auxiliary information that is to be given as input to the corrupt receiver $(\mathsf{rvk} := \tau_1(s), \mathsf{svk}, \mathsf{inp}_R, \mathsf{inp}_S, \mathsf{aux})$. The simulator initializes formal variables and samples the crs of the NIZK proofs and the KZG polynomial commitment scheme as described in the protocol, and provides these labels to $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$ to $\mathcal{A}$. $\mathcal{A}$ outputs $\mathsf{msg}_R := (\tau_1(c), \tau_1(q_0))$, where c and q_0 are the discrete logarithms of com and π_0 respectively. By observing the GGM queries made by $\mathcal{A}$, Sim can express $\tau_1(c)$ as:

$$\tau_1(c) = \tau_1(\sum_{i=1}^{n+1} f_i \boldsymbol{\tau}^i + \sum_i a_{1,i} R_{1,i} + a_0)$$

where $\{\tau_1(R_{1,i})\}_i$ is the set of labels corresponding to all $\mathbb{G}_1$ elements that are not in $\mathsf{KZG}_{\mathsf{crs}}$. If any of the checks in Fig. 3 fail, then Sim will output $\perp$ as done in the real protocol.

If they do pass, Sim will extract $\mathcal{A}$'s input D as the roots of the polynomial $f(\boldsymbol{\tau}) = \sum_{i=1}^{n+1} f_i \boldsymbol{\tau}^{i-1}$ and sends $(\mathsf{receiver}, \mathsf{sid}, D)$ to the ideal functionality $\mathcal{F}_{\mathsf{PSI}}$. Sim now receives a tuple of the form $(\mathsf{sid}, \mathsf{ssid}, x)$ from $\mathcal{F}_{\mathsf{PSI}}$, for each message sent by the sender.

Next, in order to simulate the honest sender's message, Sim needs to retrieve the payload from the ideal functionality. But to do so it needs to know signatures on the corresponding item. Sim will run a signature extraction strategy (described below) on $(\mathcal{A}, \mathcal{D})$ to extract signatures on the database D.

The high level idea is to start by sending "dummy" ciphertexts to the adversary on all items. It can be shown that if an adversary can distinguish a dummy ciphertext from one in the real protocol, then there exists an algorithm that can extract a signature on the corresponding item by observing the GGM queries made by the distinguisher and adversary $(\mathcal{D}, \mathcal{A})$. Of course, for items $x \notin D$, we do not need to extract a signature as it is computationally infeasible for a corrupt receiver to learn any information about the underlying payload.

Note that since $(\mathcal{D}, \mathcal{A})$ are randomized algorithms, they may not always succeed in distinguishing, and thus we may not always be able to extract the signature. Therefore, the simulator may need to re-run $(\mathcal{D}, \mathcal{A})$ multiple times to extract the signature. One must be careful in deciding *how many times* the simulator runs $(\mathcal{D}, \mathcal{A})$. Clearly, we cannot have the simulator run until it extracts a signature as $(\mathcal{D}, \mathcal{A})$ may not even know a signature to begin with. At this point, we allow the simulator to *depend on the distinguisher*. More formally, we define a sequence of simulators $\{\mathsf{Sim}_\mu\}_\mu$ indexed by the number of times (polynomial in the security parameter) they re-run the signature extraction strategy. Hence, for any $(\mathcal{D}, \mathcal{A})$ which can distinguish the simulation from a real execution of the protocol with non-negligible probability, there exists *some simulator* Sim_{μ^*} which extracts a signature with overwhelming probability. We formalize this as follows:

Let $\mathcal{I} = \{(x_i, m_i)\}_i$ be the item-message pairs that form the sender input. Note that items and messages may be repeated. At any point in time, let $\mathcal{L}$ denote the set of item-signature pairs that the simulator has extracted so far. $\mathcal{L}$ is initially set to be empty.

For inputs where $x_i \in D$, but $(x_i, \cdot) \notin \mathcal{L}$ the sender messages will be computed as:

$$C_0' = \{x_i, \tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x_i)), \tau_2(\boldsymbol{\beta}_i), \tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i), \pi_i\}_{x_i \in D \wedge (x_i,\cdot) \notin \mathcal{L}}$$

where $\mathcal{M} : [n_s] \to [n_x]$ is defined as $\mathcal{M}(i) \to j$ iff the discrete log of $H(x_i)$ is equal to $\mathbf{X}_j$, $\{\boldsymbol{\alpha}_i\}$, and $\{\boldsymbol{\beta}_i\}_i$ are formal variables initialized by the simulator, and π_i is a NIZK proof of the relation in Fig. 3 computed using NIZK.SimProve. To simulate the proofs, Sim will use NIZK.SimProve to simulate proofs for the messages in C_0'. Intuitively, this is the set of sender inputs for which we want to extract signatures. We can also define an analogous sender's message in the real-world that would be computed as:

$$C_0 = \{x_i, \tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x_i)), \tau_2(\boldsymbol{\beta}_i), \tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i), \pi_i\}_{x_i \in D \wedge (x_i,\cdot) \notin \mathcal{L}}$$

For the remaining inputs, we use a similar strategy except we do not initialize formal variables for α_i, β_i's and instead treat them as constants for simplicity. There are now two cases – for items $x \notin D$, we use $m_i = 0$. But for inputs $x \in D$ and $(x, \cdot) \in \mathcal{L}$, the simulator will retrieve the corresponding payload m_i from the ideal functionality and faithfully follow the protocol in Fig. 3 to produce the sender message.

$$C_1 = \{x_i, \tau_2(\alpha_i(\boldsymbol{\tau} - x_i)), \tau_2(\beta_i), \tau_T(c\alpha_i + s\mathbf{X}_{\mathcal{M}(i)}\beta_i), \pi_i\}_{x_i \notin D} \\ \cup \{x_i, \tau_2(\alpha_i(\boldsymbol{\tau} - x_i)), \tau_2(\beta_i), \tau_T(c\alpha_i + s\mathbf{X}_{\mathcal{M}(i)}\beta_i + m_i), \pi_i\}_{x_i \in D \wedge (x_i,\cdot) \in \mathcal{L}}$$

The adversary then makes a series of queries to the GGM and outputs a bit b. More succinctly:

1. $\mathcal{E}^{\mathsf{GGM}} \to (\tau_2(s), \mathsf{aux}, \mathcal{I} = \{(x_i, m_i)\}_i)$
2. The simulator initializes $\boldsymbol{\tau}$, and computes $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (\tau_1(1), \tau_1(\boldsymbol{\tau}), \tau_1(\boldsymbol{\tau}^2), \ldots, \tau_1(\boldsymbol{\tau}^{n+1}), \tau_2(1), \tau_2(\boldsymbol{\tau}))$ along with the $\mathsf{NIZK}_{\mathsf{crs}}$ and sets $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$.
3. Run $\mathcal{A}_1^{\mathsf{GGM}}(\tau_2(s), \mathsf{aux}, \mathsf{pp}) \to (\tau_1(c), \tau_1(q_0), \mathsf{st})$.
4. Let the transcript of the protocol be defined as $\mathsf{tr} := (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$.
5. By observing the GGM queries made by $\mathcal{A}_1$ to compute $\tau_1(c)$, extract D and compute C_0 and C_1 as described above.
6. We can then define $\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0 \cup C_1) \to \{0,1\}$, and $\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0' \cup C_1) \to \{0,1\}$, as two binary-valued random variables. Note that this also implicitly includes the distinguisher. They correspond to running the adversary on real-world/simulated ciphertexts. Looking ahead, we will show that for all transcripts tr sampled as above, if $|\Pr[\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0 \cup C_1) = 1] - \Pr[\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0' \cup C_1) = 1]|$ is non-negligible, then we can extract a signature on *some* item $x \in D$ but $(x, \cdot) \notin \mathcal{L}$.

Next we show how to extract signatures from the queries made by $\mathcal{A}_2$. To simplify our analysis in the symbolic GGM we pay special attention to certain formal variables that will aid in our analysis as follows:

- $\tau_1 : \sum_{i=1}^{n_x} z_i\mathbf{X}_i + z_0 \to S_1$, where z_0 and $\{z_i\}_i$ are multivariate polynomials in all other formal variables.
- $\tau_2 : \sum_{i=1}^{n_s} z_i\boldsymbol{\beta}_i + z_0 \to S_2$, where z_0 and $\{z_i\}_i$ are multivariate polynomials in all other formal variables.
- $\tau_T : \sum_{i=1}^{n_x}\sum_{j=1}^{n_s} z_{3,i,j}\mathbf{X}_i\boldsymbol{\beta}_j + \sum_{i=1}^{n_x} z_{2,i}\mathbf{X}_i + \sum_{i=1}^{n_s} z_{1,i}\boldsymbol{\beta}_i + z_0 \to S_T$, where z_0, $\{z_{1,i}\}_i$, $\{z_{2,i}\}_i$, and $\{z_{3,i,j}\}_{i,j}$ are multivariate polynomials in all other formal variables.

Note that we explicitly disallow the environment from making any queries that involve $\{\boldsymbol{\beta}_i\}_i$, since we model them as only coming into existence once the protocol starts (which is after the environment runs). But it could make random oracle queries and hence aux could depend on $\{\mathbf{X}_i\}_i$.

Next, we introduce some notation. Consider the set of τ_T queries made by $\mathcal{A}$, meaning queries made to GGM that result in a target group element. Let $\{Q_k\}_k$ denote this list, where we include $\tau_T(1)$ and $\{\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)\}_i$ explicitly in this set (where m_i is set to 0 for simulated ciphertexts). We will write each query Q_k in two different forms.

- First, we pay special attention to the labels $\{\tau_1(\mathbf{X}_i)\}_i$, $\{\tau_2(\boldsymbol{\beta}_i)\}_i$, and $\{\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)\}_i$, and write Q_k in the form

$$Q_k = \tau_T\left(\sum_{i,j} a_{k,i,j} R_{1,i} R_{2,j} + \sum_i b_{k,i} R_{T,i} + \sum_{i,j} c_{k,i,j}\mathbf{X}_i R_{2,j} + \sum_{i,j} d_{k,i,j} R_{1,j}\boldsymbol{\beta}_i + \sum_{i,j} e_{k,i,j}\mathbf{X}_j\boldsymbol{\beta}_i + \sum_i f_{k,i}\left(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i\right)\right),$$

  where $\{\tau_1(R_{1,i})\}_i$ contains all labels not equal to $\{\tau_1(\mathbf{X}_i)\}_i$, $\{\tau_2(R_{2,i})\}_i$ contains all labels not equal to $\{\tau_2(\boldsymbol{\beta}_i)\}_i$, and $\{\tau_T(R_{T,i})\}_i$ contains all labels not equal to $\{\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)\}_i$. Note that each $R_{1,i}$, $R_{2,i}$, and $R_{T,i}$ are multivariate polynomials in $\boldsymbol{\tau}$, and $\{\mathbf{X}_i\}_i$. That is, we can actually write each Q_k as

$$Q_k = \tau_T\left(g_k + \sum_{i,j} d_{k,i,j} R_{1,j}\boldsymbol{\beta}_i + \sum_{i,j} e_{k,i,j}\mathbf{X}_j\boldsymbol{\beta}_i + \sum_i f_{k,i}\left(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i\right)\right),$$

  where g_k is some multivariate polynomial in $\{\mathbf{X}_i\}_i$ and $\boldsymbol{\tau}$ but has no dependence on $\{\boldsymbol{\beta}_i\}$. Going forward, we will be especially interested in the coefficients $\{d_{k,i,j}\}_{i,j}, \{e_{k,i,j}\}_{i,j}, \{f_{k,i}\}_i$.

- Next, we can write the result of each query Q_k as the polynomial

$$\sum_{i=1}^{n_s} z_{k,1,i}\boldsymbol{\beta}_i + z_{k,0}$$

  stored by the GGM, where each of $z_{k,1,i}$ and $z_{k,0}$ are affine functions of $\mathbf{X}_i$.

Signature Extraction. We are finally ready to describe the extraction algorithm Ext that the simulator invokes as part of its *signature extraction strategy*. Sim will sample a transcript $\mathsf{tr} = (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$ as described above, and records all GGM queries made by the adversary. If $\tau_T(c) \neq \tau_T(q_0\tau)$, Sim outputs $\perp$.

If the check passes however, Sim passes a record of all the GGM queries made by $\mathcal{A}$ to Ext along with tr and the list of extracted signatures $\mathcal{L}$ (initially set to be empty and updated over time). Additionally, Ext is given black box access to $\mathcal{A}$, and the GGM. We now describe the extractor:

- Ext runs $\mathcal{A}_2^{\mathsf{GGM}}\left(\mathsf{tr}, C_0' \cup C_1\right) \to b$, where C_0' and C_1 are computed as above. During this process, it maintains a list of τ_T queries made by $\mathcal{A}_2$ in the first form described above. That is, record the list of coefficients $(\{d_{k,i,j}\}_{i,j}, \{e_{k,i,j}\}_{i,j}, \{f_{k,i}\}_i)$ for each query made by $\mathcal{A}$. Note that this is possible since Ext knows $\{\tau_1(\mathbf{X}_i)\}_i$, $\{\tau_2(\boldsymbol{\beta}_i)\}_i$, and $\{\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)\}_i$.

- Any time a new query Q_k is made, Ext will then loop over all previous queries $Q_{k'}$ and run the algorithm given in Claim 3 on the coefficients $(\{d_{k,i,j}\}_{i,j}, \{e_{k,i,j}\}_{i,j}, \{f_{k,i}\}_i)$ and $(\{d_{k',i,j}\}_{i,j}, \{e_{k',i,j}\}_{i,j}, \{f_{k',i}\}_i)$. If the algorithm returns a label $\tau_1(s\mathbf{X})$ corresponding to a signature on some item x such that $x \in D$ but $(x, \cdot) \notin \mathcal{L}$, then the extractor halts and outputs $\tau_1(s\mathbf{X})$. Otherwise it outputs $\perp$.

Recall that because the adversary and distinguisher are randomized algorithms, the extractor may need to run multiple times to successfully extract a signature. The simulator is indexed by a parameter μ and it runs Ext (described above) as follows:

1. Initialize $\mathcal{L} = \emptyset$.
2. For $i \in [\mu]$ run Ext on inputs tr and $\mathcal{L}$.:
   - If Ext outputs a valid signature $\tau_1(s\mathbf{X})$ on some item $x \in D$ but $(x, \cdot) \notin \mathcal{L}$, update $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, \tau_1(s\mathbf{X}))\}$, and restart step 2.
   - If Ext outputs $\perp$, continue.
3. Output $\mathcal{L}$ as the list of extracted signatures.

In other words, the simulator attempts to extract a signature up to μ times before moving on. If it does succeed, it updates the list of extracted signatures and restarts the process.

We will show later that for any distinguisher $\mathcal{D}$ and adversary $\mathcal{A}$ that can distinguish the simulation from a real execution of the protocol with non-negligible probability, there exists a μ^* such that Sim_{μ^*} can extract a signature with overwhelming probability.

Theorem 9. *Let $\mathcal{L}$ be the set of item-signature pairs the simulator has extracted so far. For all transcripts* $\mathsf{tr} = (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$ *sampled as:*

1. $\mathcal{E}^{\mathsf{GGM}} \rightarrow (\tau_2(s), \mathsf{aux}, \mathcal{I})$
2. *Initialize $\boldsymbol{\tau}$, and compute* $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (\tau_1(1), \tau_1(\boldsymbol{\tau}), \tau_1(\boldsymbol{\tau}^2), \ldots, \tau_1(\boldsymbol{\tau}^{n+1}), \tau_2(1), \tau_2(\boldsymbol{\tau}))$ *along with the* $\mathsf{NIZK}_{\mathsf{crs}}$ *and sets* $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$.
3. *Run* $\mathcal{A}_1^{\mathsf{GGM}}(\tau_2(s), \mathsf{aux}, \mathsf{pp}) \rightarrow (\tau_1(c), \tau_1(q_0), \mathsf{st})$,

Let D be the receiver's input that is extracted by the simulator as described above. We can then define the following sets:

- $C_0 = \{x_i, \tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x_i)), \tau_2(\boldsymbol{\beta}_i), \tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i), \pi_i\}_{x_i \in D \wedge (x_i, \cdot) \notin \mathcal{L}}$
- $C_0' = \{x_i, \tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x_i)), \tau_2(\boldsymbol{\beta}_i), \tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i), \pi_i\}_{x_i \in D \wedge (x_i, \cdot) \notin \mathcal{L}}$
- $C_1 = \{x_i, \tau_2(\alpha_i(\boldsymbol{\tau} - x_i)), \tau_2(\beta_i), \tau_T(c\alpha_i + s\mathbf{X}_{\mathcal{M}(i)}\beta_i), \pi_i\}_{x_i \notin D} \cup \{x_i, \tau_2(\alpha_i(\boldsymbol{\tau} - x_i)), \tau_2(\beta_i), \tau_T(c\alpha_i + s\mathbf{X}_{\mathcal{M}(i)}\beta_i + m_i), \pi_i\}_{x_i \in D \wedge (x_i, \cdot) \in \mathcal{L}}$

where $\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}\}_i$ are formal variables initialized by the simulator, and $\{\alpha_i, \beta_i\} \leftarrow\!\!\$\ \mathbb{F}^2$, π_i's are NIZK proofs computed using $\mathsf{NIZK.SimProve}$ *for the relation in Fig. 3. We now have that for all transcripts* tr *sampled as above:*

$$\Pr\left[\tau_1(s\mathbf{X}) \leftarrow \mathsf{Ext}^{\mathsf{GGM},\mathcal{A}}(\mathsf{tr})\right] \geq \left|\Pr[1 \leftarrow \mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0 \cup C_1)] - \Pr[1 \leftarrow \mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0' \cup C_1)]\right|,$$

where x is some item such that $x \in D$, and $(x, \cdot) \notin \mathcal{L}$. The probabilities are taken over all of the randomness of the GGM*, the adversary, and the randomness used in preparing C_0, C_0', C_1.*

Proof. First, we define an event E_{dist} over the execution of $\mathcal{A}_2(\text{tr}, C_0 \cup C_1)$, where E_{dist} is initially set to 0. Each message in C_0, contains a term of the form $\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)$. Given the sequence of queries $\{Q_k\}_k$ made by the adversary, for each query Q_k consider the list of tuples $\{z_{k,1,i}, f_{k,i}\}_{i\in[n_s]}$. If there exists any pair of tuples such that $z_{k,1,i} = z_{k',1,i}$ for some $i \in [n_s]$ (that is they are identical multivariate polynomials) and $f_{k,i} \neq f_{k',i}$, flip $E_{\text{dist}} = 1$. Then, we have the following claim.

Claim 2. *As long as $E_{\text{dist}} = 0$, the view of the adversary $\mathcal{A}_2$ is* independent *of the payloads $\mathcal{M}_0 := \{m_i \mid (x_i, m_i) \in \mathcal{I} \wedge x_i \in D \wedge (x_i, \cdot) \notin \mathcal{L}\}$, where $\mathcal{I}, D$, and $\mathcal{L}$ are defined as in Theorem 9.*

Before coming to the proof, we state an important corollary.

Corollary 1. *The probability that $E_{\text{dist}} = 1$ at the end of running $\mathcal{A}_2(\text{tr}, C_0 \cup C_1)$ is* independent *of $\mathcal{M}_0$. Moreover, defining p^* to be this probability, we have that*

$$\left|\Pr[1 \leftarrow \mathcal{A}_2^{\text{GGM}}(\text{tr}, C_0 \cup C_1)] - \Pr[1 \leftarrow \mathcal{A}_2^{\text{GGM}}(\text{tr}, C_0' \cup C_1)]\right| \leq p^*.$$

To see Claim 2, note that each tuple $(z_{k,1,i}, f_{k,i})$ can be derived from $\{d_{k,i,j}\}_j$, $\{R_{1,j}\}_j\{e_{k,i,j}\}_j$, $\{f_{k,i}\}$, and s, and that the event that E_{dist} flips to 1 can be derived from these tuples. The coefficient s and the values of $\{R_{1,i}\}_i$ are fixed by the environment before $\mathcal{M}_0$ is introduced into the experiment, and Claim 2 implies that, as long as $E_{\text{dist}} = 0$, the coefficients $\{\{d_{k,i,j}\}_{i,j}, \{e_{k,i,j}\}_{i,j}, \{f_{k,i}\}_i\}_k$ are independent of m, since they are derived from the adversary's view. Thus, the event that E_{dist} flips to 1 can be derived from information that is independent of the value of $\mathcal{M}_0$. The second statement follows because Claim 2 shows that if $E_{\text{dist}} = 0$ at the end of the experiment, the adversary's distinguishing advantage is 0.

Proof. (of Claim 2) As long as $E_{\text{dist}} = 0$, we have that for every pair of queries $(Q_k, Q_{k'})$, with corresponding tuples $\{\{z_{k,1,i}, f_{k,i}\}_i\}$ and $\{\{z_{k',1,i}, f_{k',i}\}_i\}$, we have that $\{(z_{k,1,i} \neq z_{k',1,i}) \vee (f_{k,i} = f_{k',i})\}_{i\in[n_s]}$.

Note that due to the uniform randomness of the injections sampled by GGM, the only information that can be obtained from a sequence of GGM queries is whether each pair of queries yields the same label or not. Our goal is to show that for each pair within the *combined* set of queries made by the adversary and environment, the event that they yield the same label (which we refer to as equality) is independent of $\mathcal{M}_0$. We go case by case:

- Case 1: Two environment queries. These queries occur before $\mathcal{M}_0$ is introduced in the experiment, and thus their equality cannot depend on $\mathcal{M}_0$.
- Case 2: An environment query and an adversary query with $\{z_{k,1,i} = 0\}_{i\in[n_s]}$. Since $\tau_T(1)$ is included by default in the list of adversary queries, the tuple $\{z_{1,1,i} = 0, f_{1,i} = 0\}_i$ is included in the list used to define E_{dist}, so it must be the case that the adversary query also has $\{f_{k,i} = 0\}_i$. Thus, this query does not involve any terms of the form $\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)$, and the environment's query comes before $\mathcal{M}_0$, so their equality cannot depend on $\mathcal{M}_0$.
- Case 3: An environment query and an adversary query with $z_{k,1,i} \neq 0$ for some $i \in [n_s]$. Since no environment query has a non-zero coefficient on any term that includes $\{\boldsymbol{\beta}_i\}_i$, these queries are different regardless of the value of $\mathcal{M}_0$.
- Case 4: Two adversary queries with $z_{k,1,i} \neq z_{k,1,i}$ for some $i \in [n_s]$. Since they have different coefficients on $\boldsymbol{\beta}_i$, they are different regardless of the value of $\mathcal{M}_0$.
- Case 5: Two adversary queries with the same $\{z_{k,1,i} = z_{k',1,i}\}_i$ and $\{f_{k,i} = f_{k',i}\}_i$. Since the $f_{k,i}$ values are the same, the dependence on $\{\tau_T(c\boldsymbol{\alpha}_i + s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i + m_i)\}_i$ cancels out, and the event that these queries are equal depends only on other terms, which do not involve the value of $\mathcal{M}_0$.

□

Now to complete the proof of the theorem, it suffices to show that

$$\Pr\left[\tau_1(s\mathbf{X}) \leftarrow \mathsf{Ext}^{\mathsf{GGM},\mathcal{A}}(\mathsf{tr})\right] \geq p^*.$$

The following claim shows that once $E_{\mathsf{dist}} = 1$, the extractor can succeed in extracting a signature $\tau_1(s\mathbf{X})$ on some item $x \in D$ where $(x, \cdot) \notin \mathcal{L}$. Note that the corollary above establishes that the probability $E_{\mathsf{dist}} = 1$ is independent of $\mathcal{M}_0$, so it doesn't matter what $\mathcal{M}_0$ the extractor initializes the adversary with.

Claim 3. *Suppose there exists a pair of queries $Q_k, Q_{k'}$ such that $z_{k,1,i} = z_{k',1,i}$ and $f_{k,i} \neq f_{k',i}$. Then given the coefficients $(\{d_{k,i,j}\}_{i,j}, \{e_{k,i,j}\}_{i,j}, \{f_{k,i}\}_i)$ and $(\{d_{k',i,j}\}_{i,j}, \{e_{k',i,j}\}_{i,j}, \{f_{k',i}\}_i)$ it is possible to compute $\tau_1(s\mathbf{X}_{\mathcal{M}(i)})$, which is the BLS signature on the item x_i.*

Proof. We first observe that, since the $R_{1,i}, R_{2,i}, R_{T,i}$ values do not involve any $\{\boldsymbol{\beta}_i\}_i$ terms,

$$\sum_{i,j} a_{k,i,j} R_{1,i} R_{2,j}, \quad \sum_i b_{k,i} R_{T,i}, \quad \text{and} \quad \sum_{i,j} c_{k,i,j} \mathbf{X}_i R_{2,j}$$

cannot contribute to the coefficient $z_{k,1,i}$ on $\boldsymbol{\beta}_i$.

Thus, it must be the case that:

$$\left(\sum_j d_{k,i,j} R_{1,j}\right)\boldsymbol{\beta}_i + \left(\sum_j e_{k,i,j}\mathbf{X}_j\right)\boldsymbol{\beta}_i + f_{k,i} s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i = \left(\sum_j d_{k',i,j} R_{1,j}\right)\boldsymbol{\beta}_i + \left(\sum_j e_{k',i,j}\mathbf{X}_j\right)\boldsymbol{\beta}_i + f_{k',i} s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i$$

$$\rightarrow (f_{k',i} - f_{k,i})^{-1}\left(\sum_j (d_{k,i,j} - d_{k',i,j}) R_{1,j} + \sum_j (e_{k,i,j} - e_{k',i,j})\mathbf{X}_j\right)\boldsymbol{\beta}_i = s\mathbf{X}_{\mathcal{M}(i)}\boldsymbol{\beta}_i$$

$$\rightarrow \tau_1\left((f_{k',i} - f_{k,i})^{-1}\left(\sum_j (d_{k,i,j} - d_{k',i,j}) R_{1,j} + \sum_j (e_{k,i,j} - e_{k',i,j})\mathbf{X}_j\right)\right) = \tau_1(s\mathbf{X}_{\mathcal{M}(i)})$$

and the LHS is a formula that the extractor can use for obtaining the signature $\tau_1(s\mathbf{X}_{\mathcal{M}(i)})$ on the item x_i. □

□

We now describe a sequence of hybrids that are indistinguishable from each other starting from the protocol in Fig. 3 in the real world and ending in the ideal world where Sim handles all interaction between the parties and the ideal functionality.

$\mathbf{H}_0$: The real world where the adversary interacts with the honest parties who execute the protocol as described in Fig. 3.

$\mathbf{H}_1$: In this hybrid, the simulator plays the role of all honest parties and the trusted party that samples the crs by faithfully following the protocol from Fig. 3. This hybrid has an identical distribution to the previous hybrid.

$\mathbf{H}_2$: Next we switch from the GGM to the symbolic GGM, where the role of the "challenger" is assigned to the simulator and the role of the "adversary" is assigned to environment, adversary and distinguisher. Now, Sim initializes the symbolic GGM with formal variables when sampling randomness and the random oracle query responses are also treated as formal variables. From Lemma 1 this hybrid is computationally indistinguishable from the previous hybrid.

$\mathbf{H}_3$: Sim now changes the the way sender messages are computed. It samples $(\mathsf{NIZK}_{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{SimSetup}(1^\lambda)$ and runs $\{\mathsf{GGM.Init}(\boldsymbol{\tau}^i)\}_{i\in[n+1]}$, to obtains handles to the $\mathsf{KZG}_{\mathsf{crs}}$. It then provides $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$ to $\mathcal{A}$. For all sender messages, Sim will use NIZK.SimProve, instead of NIZK.Prove to produce the NIZK proof. This hybrid is computationally indistinguishable from the previous hybrid from the zero-knowledge property of the NIZK.

$\mathbf{H}_4$: In this hybrid Sim further changes how sender messages are computed. Let the receiver message contain $\mathsf{msg}_R = (\tau_1(c), \tau_1(q_0))$, along with a NIZK proof. If the proof does not verify, or $\tau_T(c) \neq \tau_T(q_0\tau)$, Sim outputs $\perp$. Otherwise, it uses $\mathcal{A}$'s GGM queries to express $\tau_1(c)$ and $\tau_1(q_0)$ as follows:

$$\tau_1(c) = \tau_1(\sum_{i=1}^{n+1} f_i\boldsymbol{\tau}^i + \sum_i a_{1,i}R_{1,i} + a_0)$$

where $\{\tau_1(R_{1,i})\}_i$ is the set of labels corresponding to all $\mathbb{G}_1$ elements that are not in $\mathsf{KZG}_{\mathsf{crs}}$. Sim sets the receiver input D to be the roots of the polynomial $f(\boldsymbol{\tau}) = \sum_{i=1}^{n+1} f_i\boldsymbol{\tau}^{i-1}$. For all sender messages that encrypt to an item $x \notin D$, Sim replaces the sender's payload with 0.

Lemma 2. *Let* $\mathsf{tr} = (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$ *be the transcript of the protocol between the simulator, the environment* $\mathcal{E}$*, and the corrupt receiver* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*. Suppose* $\tau_1(c)$ *was computed by* $\mathcal{A}_1$ *as:*

$$\tau_1(c) = \tau_1(\sum_{i=1}^{n+1} f_i\boldsymbol{\tau}^i + \sum_i a_{1,i}R_{1,i} + a_0)$$

and the verification checks in Fig. 3 pass. We then define the set $D \subset \mathbb{Z}_p$ *to be the roots of the polynomial* $f(\boldsymbol{\tau}) = \sum_{i=1}^{n+1} f_i\boldsymbol{\tau}^{i-1}$*. Then for all transcripts* tr *sampled as above, for all* $x \notin D$*, for all payloads* m*, and for all non-uniform PPT adversaries* $\mathcal{B}$*, we have that*

$$\left|\Pr[\mathcal{B}(\mathsf{tr}, \mathsf{SimSen}(\mathsf{p}, \mathsf{msg}_R, x, m = 0)) \to 1] - \Pr[\mathcal{B}(\mathsf{tr}, \mathsf{Sen}(\mathsf{p}, \mathsf{msg}_R, x, m)) \to 1]\right| = 0.$$

Proof. Suppose $\tau_1(q_0)$ was computed by $\mathcal{A}_1$ as:

$$\tau_1(q_0) = \tau_1(\sum_{i=1}^{n+1} q_i\boldsymbol{\tau}^i + \sum_i b_{1,i}R_{1,i} + b_0).$$

Since $\tau_T(c) = \tau_T(q_0\tau)$ we have that:

$$\sum_{i=1}^{n+1} f_i\boldsymbol{\tau}^i + \sum_i a_{1,i}R_{1,i} + a_0 = \boldsymbol{\tau}(\sum_{i=1}^{n+1} q_i\boldsymbol{\tau}^i + \sum_i b_{1,i}R_{1,i} + b_0).$$

Because $R_{1,i}$ does not depend on $\boldsymbol{\tau}$, it must be the case that $\{a_{1,i} = 0\}_i$ and $a_0 = 0$, and hence $\tau_1(c) = \sum_{i=1}^{n+1} f_i\tau_1(\boldsymbol{\tau}^i) = \boldsymbol{\tau}f(\boldsymbol{\tau})$.

Now recall that the sender message contains:

- $\mathsf{ct}_1 = \tau_2(\boldsymbol{\alpha}(\boldsymbol{\tau} - x))$
- $\mathsf{ct}_2 = \tau_2(\boldsymbol{\beta})$
- $\mathsf{ct}_3 = \tau_T(c\boldsymbol{\alpha} + s\mathbf{X}\boldsymbol{\beta} + m)$

where $\mathbf{X}$ is the discrete log of $H(x)$, and $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are formal variables corresponding to the randomness used in the sender's message. Since $x \neq 0$ and $x \notin D$ that there does not exist a polynomial $q(\boldsymbol{\tau})$ of degree at most $n+1$ such that $\boldsymbol{\alpha}(\boldsymbol{\tau} - x)q(\boldsymbol{\tau}) = \boldsymbol{\alpha}\boldsymbol{\tau}f(\boldsymbol{\tau})$. Hence, the polynomial $c\boldsymbol{\alpha}$ must be linearly independent of the set $\{\boldsymbol{\alpha}(\boldsymbol{\tau} - x)\boldsymbol{\tau}^i\}_{i=0}^{n+1}$. By definition of the symbolic GGM, every polynomial is assigned a unique label. Since ct_3 is linearly independent of all terms in the *completion set* of the adversary's view, the adversary cannot distinguish between the real and simulated sender messages. □

From Lemma 2, this hybrid is perfectly indistinguishable from the previous hybrid.

$\mathbf{H}_5$: In the previous hybrid Sim changed the way sender messages were computed for items $x \notin D$. In this hybrid, Sim will change how sender messages are computed for items $x \in D$. It does so by first running a "signature extraction strategy" on the adversary to compute $\mathcal{L}$ – a list of item-signature pairs. More formally, the simulator will first compute a transcript $\mathsf{tr} = (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$ as described above with $\mathcal{L}$ initially set to the emptyset. The simulator is indexed by a parameter μ and it runs Ext (described above) as follows:

1. Initialize $\mathcal{L} = \emptyset$.
2. For $i \in [\mu]$ run Ext on inputs tr and $\mathcal{L}$:
   - If Ext outputs a valid signature $\tau_1(s\mathbf{X})$ on some item $x \in D$ but $(x, \cdot) \notin \mathcal{L}$, update $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, \tau_1(s\mathbf{X}))\}$, and restart step 2.
   - If Ext outputs $\perp$, continue.
3. Output $\mathcal{L}$ as the list of extracted signatures.

In other words, the simulator attempts to extract a signature up to μ times before moving on. If it does succeed, it updates the list of extracted signatures and restarts the process.

Over the choice of all possible transcripts tr, all possible sender inputs $\mathcal{I}$, and all possible subsets $S \subset \mathcal{I}$, let the probability p_{min} be a lower-bound on the probability p^* in Corollary 1. Using the Chernoff bound, we can see that over μ^* trials, Ext will extract a signature with probability at least $1 - 2e^{-\mu^* p_{\mathsf{min}}}$, and for μ^* large enough, this probability is overwhelming.

Once Sim has extracted the list of item-signatures pairs $\mathcal{L}$, it simulates the senders message as $C_0' \cup C_1$ which are computed as described in Theorem 9. Note that Sim already has the payloads as it receives $\mathcal{I}$ directly from the environment. It merely ignores some of them when computing the sender's message. At this point, it must be the case that the advantage of the adversary in Corollary 1 must be negligible, and hence this hybrid is computationally indistinguishable from the previous hybrid.

$\mathbf{H}_6$: We switch the symbolic GGM back to the GGM, which is computationally indistinguishable by Lemma 1.

$\mathbf{H}_7$: In the final hybrid, Sim no longer controls the sender and only handles interaction between the ideal functionality and $\mathcal{A}$. After it extracts D from $\mathcal{A}_1$'s message, it sends $(\mathsf{receiver}, D)$ to the ideal functionality. When the sender sends a message to the ideal functionality Sim now receives $(\mathsf{sid}, \mathsf{ssid}, x)$ from $\mathcal{F}_{\mathsf{PSI}}$. To simulate the sender messages, Sim will run a signature extraction strategy as in the previous hybrid to compute $\mathcal{L}$. Sim will then $C_0' \cup C_1$ just as in the previous hybrid, but now it no longer has access to the payloads. Instead, it will use the extracted signatures in $\mathcal{L}$ to retrieve the payloads corresponding to all items $x \in D$, such that $(x, \sigma_x) \in \mathcal{L}$. This is done by sending $(\mathsf{receiver}, \mathsf{sid}, x, \sigma_x)$ to the ideal functionality. For the remaining sender messages, Sim does not use the payload in the sender's input. This hybrid has an identical distribution to the previous hybrid and concludes the proof.

Corrupt Sender. The only message a corrupt sender receives is the first round message from the receiver. Since it has a uniformly random distribution, Sim simply replaces msg_R with a uniformly random group element in $\mathbb{G}_1$. The distribution of the simulation and real world in the symbolic GGM. We incur a negligible loss in switching from the GGM to symbolic GGM and hence distribution of the ideal world execution is computationally indistinguishable from the real world execution. To extract the sender's payload, Sim runs the extractor of the NIZK to recover the payload m and a signature $\sigma_{(x,m)}$ on (x, m), and sends $(\mathsf{sender}, \mathsf{sid}, x, m, \sigma_{(x,m)})$ to $\mathcal{F}_{\mathsf{PSI}}$. □

Theorem 5. *Let* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$ *be defined as* $e(H(x), \mathsf{rvk}) \stackrel{?}{=} e(\sigma_x, h)$. *The protocol in Fig. 4 securely emulates (Definition 7) the* $\mathcal{F}_{\mathsf{PSI}}$ *functionality with perfect* ϵ*-correctness (Definition 6), with respect to the receiver verification predicate* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$*, against generic non-uniform PPT adversaries corrupting either the sender or the receiver in the random oracle model.*

Proof. The proof strategy is very similar to that of Theorem 4 except for two main differences.

First, we only achieve perfect ϵ-correctness, owing to the fact that there will be some collisions in the cuckoo hashing step, and hence some items may be dropped by the Rec algorithm. However, cuckoo hashing does guarantee that with $1-\mathsf{negl}(\lambda)$ probability, there exists a set $D' \subseteq D$ such that $|D'| \geq (1-\epsilon)|D|$ and for any $x \in D'$, either $T[h_0(x)] = x$ or $T[h_1(x)] = x$. This allows us to finish the argument using a similar strategy.

Next, the simulator Sim now extracts the receiver's input differently. In particular, instead of solving for the roots of the polynomial $f(X)$, Sim now evaluates f on $\{\omega^i\}_{i\in[n']}$ sets the evaluations to be the receivers input D. □

Theorem 6. *Let* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$ *be defined as* $e(\sigma, \mathsf{rvk} \cdot g^x) \stackrel{?}{=} e(g, h)$*, where* rvk *is the verification key. The protocol in Fig. 5 securely emulates (Definition 7) the* $\mathcal{F}_{\mathsf{PSI-EH}}$ *functionality (Fig. 2) with respect to the receiver verification predicate* $\mathsf{RecVer}_{\mathsf{rvk}}(x, \sigma_x)$*, against generic non-uniform PPT adversaries corrupting either the sender or the receiver.*

Proof. **Corrupt Receiver.** The proof of security against a corrupt receiver will follow a similar strategy as that in Theorem 4. Here we focus on the main differences.

The environment outputs $(\mathsf{rvk} := \tau_2(s), \mathsf{svk}, \mathsf{inp}_R, \mathsf{inp}_S, \mathsf{aux})$. The simulator initializes formal variables while sampling the public parameters $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$ and provides it to $\mathcal{A}$ who outputs $\mathsf{msg}_R = (\tau_1(c), \tau_1(q_0))$. The simulator then extracts the receiver's input D as done in Theorem 4.

Next, we need to simulate the honest sender's messages. As before we need to extract signatures but note that in the element-hiding case, the simulator does not even have access to the list of items in the sender's input. To accurately simulate sender inputs, we still need to extract signatures on items in D but here we additionally need to use *dummy* values for both the item and the payload. As before, let $\mathcal{L}$ be a list of item-signature pairs extracted so far. We will now need to show that the adversary cannot distinguish a *dummy* ciphertext from a real ciphertext unless they have a signature on *some* item $x \in D$ but $(x, \cdot) \notin \mathcal{L}$, in which case we will extract the signature and restart the simulation.

Let $\mathsf{inp}_S = \{(x_i, m_i)\}_i$ be the sender's input chosen by the environment and $\mathcal{L}$ is initialed to empty. Initially, the simulator only receives the size of the sender's input. As the simulator extracts signatures, it will retrieve the corresponding payload from the ideal functionality.

Let $\mathcal{I}_1$ be the subset of the sender's input where $x_i \in D$ and $(x_i, \cdot) \in \mathcal{L}$, and $\mathcal{I}_0 = \mathsf{msg}_S \setminus \mathcal{I}_1$. In the real world, for all sender inputs in $\mathcal{I}_0$, the sendee's message will be computed as:

$$C_0 = \{\tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x_i)), \tau_2(\boldsymbol{\beta}_i(s - x_i)), \tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i + m_i), \pi_i\}_{(x_i,m_i)\in\mathcal{I}_0},$$

where $\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i\}_i$ are formal variables. For convenience, we will redefine $\boldsymbol{\beta}'_i = \boldsymbol{\beta}_i(s - x_i)$ so that

$$C_0 = \{\tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x_i)), \tau_2(\boldsymbol{\beta}'_i), \tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s - x_i)^{-1} + m_i), \pi_i\}_{(x_i, m_i) \in \mathcal{I}_0}.$$

In contrast, the simulator will compute these sender messages as

$$C'_0 = \{\tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau} - x'_i)), \tau_2(\boldsymbol{\beta}'_i), \tau_T(\boldsymbol{\beta}''_i), \pi_i\}_{(x_i, m_i) \in \mathcal{I}_0},$$

where $\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}'_i, \boldsymbol{\beta}''_i\}_i$ are formal variables initialized by the simulator, and π_i is a NIZK proof of the relation in Fig. 5 computed using NIZK.SimProve.

For all inputs in $\mathcal{I}_1$ the simulator can retrieve the payload from the ideal functionality, and we simulate sender's messages as:

$$C_1 = \{\tau_2(\alpha_i(\boldsymbol{\tau} - x_i)), \tau_2(\beta_i(s - x_i)), \tau_T(c\alpha_i + \beta_i + m_i), \pi_i\}_{(x_i, m_i) \in \mathcal{I}_1},$$

where we don't initialize any formal variables for simplicity.

We now define a distinguishing game between the real world and the ideal world.

- $\mathcal{E}^{\mathsf{GGM}} \to (\tau_2(s), \mathsf{aux}, \mathsf{msg}_S)$
- The simulator initializes $\boldsymbol{\tau}$ and computes the public parameters $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$.
- Run $\mathcal{A}_1^{\mathsf{GGM}}(\tau_2(s), \mathsf{aux}, \mathsf{pp}) \to (\tau_1(c), \tau_1(q_0), \mathsf{st})$.
- Let the transcript of the protocol be defined as $\mathsf{tr} \coloneqq (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$.
- The simulator extracts the receiver's input D and computes C_0 and C_1 as described above using.
- We can then define $\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0 \cup C_1) \to \{0, 1\}$ and $\mathcal{A}_3^{\mathsf{GGM}}(\mathsf{tr}, C'_0 \cup C_1) \to \{0, 1\}$ as two binary valued random variables, corresponding to running the distinguisher on the real world/simulated ciphertexts. Note that this also implicitly includes the distinguisher. Looking ahead we will show that for all transcripts tr sampled as above, if $\left|\Pr[\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0 \cup C_1) = 1] - \Pr[\mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C'_0 \cup C_1) = 1]\right|$ is non-negligible, then we can extract a signature on some item $x \in D$ but $(x, \cdot) \notin \mathcal{L}$.

Next we show how to extract signatures from the queries made by $\mathcal{A}_2$. Consider the set of τ_T queries made by $\mathcal{A}$, meaning queries made to GGM that result in a target group element. Let $\{Q_k\}_k$ denote this list, where we include $\tau_T(1)$ and $\{\tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s - x_i)^{-1} + m_i)\}_i$ explicitly in this set (where m_i is set to 0, and x_i is a random scalar for simulated ciphertexts). We will write each query Q_k in two different forms.

- First, we pay special attention to the labels $\{\tau_2(\boldsymbol{\beta}'_i)\}_i$, and $\{\tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s - x)^{-1} + m_i)\}_i$, and write Q_k in the form

$$Q_k = \tau_T\left(\sum_{i,j} a_{k,i,j} R_{1,i} R_{2,j} + \sum_i b_{k,i} R_{T,i} + \sum_{i,j} c_{k,i,j} R_{1,j} \boldsymbol{\beta}'_i + \sum_i d_{k,i}\left(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s - x_i)^{-1} + m_i\right)\right),$$

  where $\{\tau_1(R_{1,i})\}_i$ contains all labels from S_1, $\{\tau_2(R_{2,i})\}_i$ contains all labels not equal to $\{\tau_2(\boldsymbol{\beta}'_i)\}_i$ from S_2, and $\{\tau_T(R_{T,i})\}_i$ contains all labels not equal to $\{\tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s - x_i)^{-1} + m_i)\}_i$ from S_T. We can then write each Q_k as

$$Q_k = \tau_T\left(g_k + \sum_{i,j} c_{k,i,j} R_{1,j} \boldsymbol{\beta}'_i + \sum_i d_{k,i}\left(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s - x_i)^{-1} + m_i\right)\right),$$

  where g_k is some multivariate polynomial in $\boldsymbol{\tau}$ but has no dependence on $\{\boldsymbol{\beta}'_i\}$. Going forward, we will be especially interested in the coefficients $\{c_{k,i,j}\}_{i,j}, \{d_{k,i}\}_i$.

- Alternatively, we can write the result of each query Q_k as the polynomial

$$\sum_i z_{k,1,i}\boldsymbol{\beta}'_i + z_{k,0}$$

stored by the GGM, where each of $z_{k,1,i}$ and $z_{k,0}$ are multivariate polynomials in the remaining formal variables.

Signature Extraction. We are now ready to describe the extraction algorithm Ext that the simulator invokes as part of its *signature extraction strategy*. Sim will sample a transcript $\mathsf{tr} = (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$ as described above, and records all GGM queries made by the adversary. If $\tau_T(c) \neq \tau_T(q_0\tau)$, Sim outputs $\perp$.

If the check passes however, Sim passes a record of all the GGM queries made by $\mathcal{A}$ to Ext along with tr and the list of extracted signatures $\mathcal{L}$ (initially set to be empty and updated over time). Additionally, Ext is given black box access to $\mathcal{A}$, and the GGM. We now describe the extractor:

- Ext runs $\mathcal{A}_2^{\mathsf{GGM}}\left(\mathsf{tr}, C'_0 \cup C_1\right) \to b$, where C'_0 and C_1 are computed as above. During this process, it maintains a list of τ_T queries made by $\mathcal{A}_2$ in the first form described above. That is, record the list of coefficients $(\{c_{k,i,j}\}_{i,j}, \{d_{k,i}\}_i)$ for each query made by $\mathcal{A}$. Note that this is possible since Ext knows $\{\tau_2(\boldsymbol{\beta}'_i)\}_i$, and $\{\tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s-x_i)^{-1} + m_i)\}_i$.
- Any time a new query Q_k is made, Ext will then loop over all previous queries $Q_{k'}$ and run the algorithm given in Claim 3 on the coefficients $(\{c_{k,i,j}\}_{i,j}, \{d_{k,i}\}_i)$ and $(\{c_{k',i,j}\}_{i,j}, \{d_{k',i}\}_i)$. If the algorithm returns a label $\tau_1(1/(s-x))$ corresponding to a signature on some item x such that $x \in D$ but $(x,\cdot) \notin \mathcal{L}$, then the extractor halts and outputs $\tau_1(1/(s-x))$. Otherwise it outputs $\perp$.

As in Theorem 4, the simulator indexed by a parameter μ will run Ext multiple times to extract signatures.

Theorem 10. *Let $\mathcal{L}$ be the set of item-signature pairs the simulator has extracted so far. For all transcripts* $\mathsf{tr} = (\tau_2(s), \mathsf{aux}, \mathsf{pp}, \tau_1(c), \tau_1(q_0), \mathsf{st})$ *sampled as:*

1. $\mathcal{E}^{\mathsf{GGM}} \to (\tau_2(s), \mathsf{aux}, \mathcal{I})$
2. *Initialize* $\boldsymbol{\tau}$*, and compute* $\mathsf{KZG}_{\mathsf{crs}} \leftarrow (\tau_1(1), \tau_1(\boldsymbol{\tau}), \tau_1(\boldsymbol{\tau}^2), \ldots, \tau_1(\boldsymbol{\tau}^{n+1}), \tau_2(1), \tau_2(\boldsymbol{\tau}))$ *along with the* $\mathsf{NIZK}_{\mathsf{crs}}$ *and sets* $\mathsf{pp} = (\mathsf{KZG}_{\mathsf{crs}}, \mathsf{NIZK}_{\mathsf{crs}})$.
3. *Run* $\mathcal{A}_1^{\mathsf{GGM}}(\tau_2(s), \mathsf{aux}, \mathsf{pp}) \to (\tau_1(c), \tau_1(q_0), \mathsf{st})$,

Let D be the receiver's input that is extracted by the simulator as described above. Let $\mathcal{I}_1 \subset \mathsf{msg}_S$ be the subset of the sender's input where $x_i \in D$ and $(x_i,\cdot) \in \mathcal{L}$, and $\mathcal{I}_0 = \mathsf{msg}_S \setminus \mathcal{I}_1$. We can then define the following sets:

- $C_0 = \{\tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau}-x_i)), \tau_2(\boldsymbol{\beta}'_i), \tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}'_i(s-x_i)^{-1} + m_i), \pi_i\}_{(x_i,m_i)\in\mathcal{I}_0}$
- $C'_0 = \{\tau_2(\boldsymbol{\alpha}_i(\boldsymbol{\tau}-x'_i)), \tau_2(\boldsymbol{\beta}'_i), \tau_T(\boldsymbol{\beta}''_i), \pi_i\}_{(x_i,m_i)\in\mathcal{I}_0}$
- $C_1 = \{\tau_2(\alpha_i(\boldsymbol{\tau}-x_i)), \tau_2(\beta_i(s-x_i)), \tau_T(c\alpha_i + \beta_i + m_i), \pi_i\}_{(x_i,m_i)\in\mathcal{I}_1}$

where $\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\beta}''_i\}_i$ are formal variables, $\{\alpha_i, \beta_i\} \leftarrow\!\!\$\ \mathbb{F}^2$, and π_i's are NIZK proofs computed using $\mathsf{NIZK.SimProve}$ *for the relation in Fig. 5. We now have that for all transcripts* tr *sampled as above:*

$$\Pr\left[\tau_1((s-x)^{-1}) \leftarrow \mathsf{Ext}^{\mathsf{GGM},\mathcal{A}}(\mathsf{tr})\right] \geq \left|\Pr[1 \leftarrow \mathcal{A}_2^{\mathsf{GGM}}(\mathsf{tr}, C_0 \cup C_1)] - \Pr[1 \leftarrow \mathcal{A}_2^{\mathsf{GGM}}\left(\mathsf{tr}, C'_0 \cup C_1\right)]\right|,$$

where x is some item such that $x \in D$, but $(x,\cdot) \notin \mathcal{L}$. The probabilities are taken over all of the randomness of the GGM*, the adversary, and the randomness used in preparing C_0, C'_0, C_1.*

Proof. First, we define an event E_{dist} over the execution of $\mathcal{A}_2(\text{tr}, C_0 \cup C_1)$, where E_{dist} is initially set to 0. Given the sequence of queries $\{Q_k\}_k$ made by the adversary, for each query Q_k consider the list of tuples $\{z_{k,i}, d_{k,i}\}_i$. If there exists any pair of tuples such that $z_{k,i} = z_{k',i}$ for some i (that is they are identical multivariate polynomials) and $f_{k,i} \neq f_{k',i}$, flip $E_{\text{dist}} = 1$. Then, we have the following claim.

Claim 4. *As long as $E_{\text{dist}} = 0$, the view of the adversary $\mathcal{A}_2$ in $\mathcal{A}_2^{\text{GGM}}(\text{tr}, C_0 \cup C_1)$ is identical to its view in $\mathcal{A}_2^{\text{GGM}}(\text{tr}, C_0' \cup C_1)$.*

Proof. (of Claim 4) The only difference in the two experiments is that we switch $\tau_T(c\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i'(s - x_i)^{-1} + m_i)$ with $\tau_T(\boldsymbol{\beta}_i'')$. Let this component be denoted by Γ_i. We will show that as long as $E_{\text{dist}} = 0$, this change has no effect on whether two adversary queries yield *equivalent* labels and thus, give no information about which experiment the adversary is participating in.

As long as $E_{\text{dist}} = 0$, we have that for every pair of queries $(Q_k, Q_{k'})$, with corresponding tuples $\{\{z_{k,i}, f_{k,i}\}_i\}$ and $\{\{z_{k',i}, f_{k',i}\}_i\}$, we have that $\{(z_{k,i} \neq z_{k',i}) \vee (f_{k,i} = f_{k',i})\}_i$.

- Case 1: Two environment queries. These queries occur before $\{\Gamma_i\}_i$ is introduced in the experiment, and thus their equality cannot depend on $\{\Gamma_i\}_i$.
- Case 2: An environment query and an adversary query with $\{z_{k,i} = 0\}_i$. Since $\tau_T(1)$ is included by default in the list of adversary queries, the tuple $\{z_{1,i} = 0, f_{1,i} = 0\}_i$ is included in the list used to define E_{dist}, so it must be the case that the adversary query also has $\{f_{k,i} = 0\}_i$. Thus, this query does not involve $\{\Gamma_i\}_i$.
- Case 3: An environment query and an adversary query with $z_{k,i} \neq 0$ for some i. Since no environment query has a non-zero coefficient on any term that includes $\{\boldsymbol{\beta}_i'\}_i$, these queries are different regardless of the value of $\{\Gamma_i\}_i$.
- Case 4: Two adversary queries with $z_{k,i} \neq z_{k,i}$ for some i. Since they have different coefficients on $\boldsymbol{\beta}_i'$, they are different regardless of the value of $\{\Gamma_i\}_i$.
- Case 5: Two adversary queries with the same $\{z_{k,i} = z_{k',i}\}_i$ and $\{f_{k,i} = f_{k',i}\}_i$. Since the $f_{k,i}$ values are the same, the dependence on $\{\Gamma_i\}_i$ cancels out, and the event that these queries are equal depends only on other terms, which do not involve the value of $\{\Gamma_i\}_i$.

□

The following claim shows that once $E_{\text{dist}} = 1$, the extractor can succeed in extracting a signature $\tau_1((s - x)^{-1})$ on some item $x \in D$ where $(x, \cdot) \notin \mathcal{L}$. Note that the corollary above establishes that the probability $E_{\text{dist}} = 1$ is independent of $\{\Gamma_i\}_i$.

Claim 5. *Suppose we run $\mathcal{A}_2(\text{tr}, C_0 \cup C_1)$ there exists a pair of queries $Q_k, Q_{k'}$ such that $z_{k,i} = z_{k',i}$ and $f_{k,i} \neq f_{k',i}$. Then given the coefficients $(\{c_{k,i,j}\}_{i,j}, \{d_{k,i}\}_i)$ and $(\{c_{k',i,j}\}_{i,j}, \{d_{k',i}\}_i)$ it is possible to compute $\tau_1((s - x_i)^{-1})$, which is the Boneh-Boyen signature on the item x_i.*

Proof. Since the coefficients on $\boldsymbol{\beta}_i'$ are the same, we have that

$$\left(\sum_j c_{k,i,j} R_{1,j}\right) + d_{k,i}(s - x_i)^{-1} = \left(\sum_j c_{k',i,j} R_{1,j}\right) + d_{k',i}(s - x_i)^{-1}$$

$$\rightarrow (d_{k',i} - d_{k,i})^{-1} \left(\sum_j (c_{k,i,j} - c_{k',i,j}) R_{1,j}\right) = (s - x_i)^{-1}$$

and the LHS is a formula that the extractor can use for obtaining the signature $\tau_1((s-x_i)^{-1})$ on the item x_i. □

Now, let p be the probability that E_{dist} flips from 0 to 1 when running $\mathcal{A}_2(\text{tr}, C_0 \cup C_1)$. In particular, this means that with probability p, there exists a query k^* such that, prior to Q_{k^*}, $E_{\text{dist}} = 0$, and after Q_{k^*}, $E_{\text{dist}} = 1$. Moreover, up until the point that the coefficients $d_{k^*,i}, \{c_{k^*,i,j}\}_j$ are produced, the adversary's view (which includes these coefficients) is identical to the adversary's view in $\mathcal{A}_2(\text{tr}, C_0' \cup C_1)$. Now, we know that

$$(d_{k',i} - d_{k,i})^{-1}\left(\sum_j (c_{k,i,j} - c_{k',i,j})R_{1,j}\right) = (s-x_i)^{-1},$$

It must *also* be the case that the same identity holds in a run of $\mathcal{A}_2(\text{tr}, C_0' \cup C_1)$, since otherwise this would contradict the fact that the experiments are perfectly indistinguishable until the label for the k^*'th query is returned. Ext will thus run the algorithm given in Claim 5 on all pairs of queries and attempt to extract a signature. Also note that it is easy to check whether the extracted signature is valid, using queries that do not involve $\{\Gamma_i\}_i$. Thus, we have shown that the simulator's strategy succeeds in outputting $\tau_1(\boldsymbol{\rho}(s-x)^{-1})$ with probability at least p, which, combined with Claim 4, completes the proof of the theorem. □

The sequence of hybrids will be very similar to Theorem 4, except we simulate sender messages differently and use Theorem 10 to argue that the simulator can successfully extract signatures and simulate the view the adversary in the ideal world.

Corrupt Sender. Proving security against a corrupt sender is more involved than Theorem 4 as we no longer achieve perfect correctness. While the simulator can extract the payload and item chosen by the sender by running the extractor of the NIZK, the receiver in the real world may not recover the same payload if the sender uses *bad randomness*.

We begin by proving that it is computationally infeasible for any non-uniform PPT adversary to find sender input $(x, m, \sigma_{(x,m)})$ and randomness $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ such that an honest receiver outputs $m' \notin \{m, \perp\}$. This holds true even if the adversary has full control over the receiver input D, receiver verification key rvk, and signatures Σ on D.

Claim 6. *Consider the following game between a challenger C and an adversary $\mathcal{A}$:*

1. *C runs* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ *and provides $\mathcal{A}$ with the resulting* pp.
2. *$\mathcal{A}$ outputs a list of items D, receiver verification key* rvk *and valid signatures Σ on D.*
3. *C runs* $(\mathsf{st}, \mathsf{msg}_R) \leftarrow \mathsf{Rec}(\mathsf{pp}, D)$ *and provides $\mathcal{A}$ with* msg_R.
4. *$\mathcal{A}$ produces sender input $(x, m, \sigma_{(x,m)})$ and sender randomness $(\alpha_1, \alpha_2, \beta_1, \beta_2)$.*

$\mathcal{A}$ wins the above game if $\mathsf{Dec}(\mathsf{st}, \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)}; (\alpha_1, \alpha_2, \beta_1, \beta_2)), \Sigma) \notin \{m, \perp\}$. *For the algorithms* $(\mathsf{Gen}, \mathsf{Rec}, \mathsf{Sen}, \mathsf{Dec})$ *described in Fig. 5, for all non-uniform PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins the above game is negligible, where the probability is taken over the randomness used in the* Gen *and* Rec *algorithms.*

Proof. Suppose there exists a non-uniform generic PPT adversary $\mathcal{A}$ that can win the above game with non-negligible probability. We will then use such an $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ that can solve for τ given $\mathsf{KZG}_{\text{crs}} = (g, g^\tau, \ldots, g^{\tau^n}, h^\tau)$ with non-negligible probability. This in turn leads to a contradiction as it computationally infeasible to solve for τ given $\mathsf{KZG}_{\text{crs}}$ in the generic group model.

$\mathcal{A}'$ works as follows: $\mathcal{A}'$ receives $\mathsf{KZG}_{\text{crs}}$ as input. It runs $\mathsf{NIZK}_{\text{crs}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and provides $\mathcal{A}$ with $\mathsf{pp} = (\mathsf{KZG}_{\text{crs}}, \mathsf{NIZK}_{\text{crs}})$. $\mathcal{A}$ outputs $(D, \mathsf{vk}_R = (U, V), \Sigma)$. $\mathcal{A}'$ then runs $(\mathsf{st}, \mathsf{msg}_R) \leftarrow \mathsf{Rec}(\mathsf{pp}, D)$ and

provides $\mathcal{A}$ with msg_R. $\mathcal{A}$ outputs $(x, m, \sigma_{(x,m)})$ and $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ such that, with non-negligible probability $\mathsf{Dec}(\mathsf{st}, \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)}), (\alpha_1, \alpha_2, \beta_1, \beta_2)) \notin \{m, \perp\}$. Parse $\mathsf{msg}_S = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5, \mathsf{ct}_6, \pi)$. Since Dec returns some m', it must be the case that there exists $x' \neq x$, $x' \in D$, with signature $\sigma_{x'} \in \Sigma$ and KZG opening proof $\Pi_{x'}$ such that:

$$e(\Pi_{x'}, h^{(\tau - x)\alpha_2}) \cdot e(\sigma_{x'}, (\mathsf{rvk} \cdot h^{-x})^{\beta_2}) = e(\mathsf{com}, h)^{\alpha_2} \cdot e(g, h)^{\beta_2}. \tag{1}$$

Note that the KZG verification equation – $e(\Pi_x, h^{(\tau - x)}) = e(\mathsf{com}, h)$ has a *unique witness* for all $x \neq \tau$, i.e. Π_x is fully determined by x for a given com and $\Pi_x \neq \Pi_{x'}$ for $x \neq x' \neq \tau$. Thus,

$$x \neq x' \implies e(\Pi_{x'}, h^{(\tau - x)}) \neq e(\mathsf{com}, h) \implies \forall\, \alpha_2 \neq 0, e(\Pi_{x'}, h^{(\tau - x)\alpha_2}) \neq e(\mathsf{com}, h)^{\alpha_2}. \tag{2}$$

If Eq. (1) holds and $x \neq x'$, $\alpha_2 \neq 0$, and $\beta_2 \neq 0$, then Eq. (2) implies that:

$$e(\Pi_{x'}, h^{(\tau - x)\alpha_2}) \neq e(\mathsf{com}, h)^{\alpha_2}$$

$$e(\sigma_{x'}, (\mathsf{rvk} \cdot h^{-x})^{\beta_2}) \neq e(g, h)^{\beta_2}$$

But note that $\sigma_{x'}$ is a valid signature on x' and $\Pi_{x'}$ is a valid KZG opening proof of com at the point x' to 0. This implies

$$e(\Pi_{x'}, h^{(\tau - x')\alpha_2}) = e(\mathsf{com}, h)^{\alpha_2}$$

$$e(\sigma_{x'}, (\mathsf{rvk} \cdot h^{-x})^{\beta_2}) = e(g, h)^{\beta_2}$$

By substituting $e(\mathsf{com}, h)^{\alpha_2}$ and $e(g, h)^{\beta_2}$ with the expressions above, we can rewrite Eq. (1) as:

$$e(\sigma_{x'}, h^{\delta_x \beta_2}) \cdot e(\Pi_{x'}, h^{\delta_x \alpha_2}) = 0, \tag{3}$$

where $\delta_x = x - x' \neq 0$. Switching to the symbolic GGM, we set $\boldsymbol{r}$, and $\boldsymbol{\tau}$ to be the formal variables corresponding to r and τ respectively. Then, Eq. (3) implies that the adversary has found $(\alpha_2, \beta_2, x, x', D)$, and a non-zero formal polynomial $s_x(\boldsymbol{\tau})$ such that

$$\beta_2 s_x(\boldsymbol{\tau}) + \alpha_2 \boldsymbol{r} q(\boldsymbol{\tau}) = 0$$

where D is the roots of a polynomial $f(\boldsymbol{\tau})$, and $q(\boldsymbol{\tau}) = f(\boldsymbol{\tau})/(\boldsymbol{\tau} - x')$. But this is impossible as $\boldsymbol{r}$ is linearly independent of any degree $n+1$ polynomial $s_x(\boldsymbol{\tau})$, and hence the probability that $\mathcal{A}$ wins the above game is negligible. □

Next we show that if $x \in D$, and $\sigma_x \in \Sigma$ is a valid signature on x under rvk, then an honest receiver recovers the payload m sent by the sender with overwhelming probability.

Claim 7. *Consider the following game between a challenger C and an adversary $\mathcal{A}$:*

1. *C runs* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ *and provides $\mathcal{A}$ with the resulting* pp.
2. *$\mathcal{A}$ outputs a list of items D, receiver verification key* rvk *and valid signatures Σ on D.*
3. *C runs* $(\mathsf{st}, \mathsf{msg}_R) \leftarrow \mathsf{Rec}(\mathsf{pp}, D)$ *and provides $\mathcal{A}$ with* msg_R.
4. *$\mathcal{A}$ produces sender input $(x, m, \sigma_{(x,m)})$ and sender randomness $(\alpha_1, \alpha_2, \beta_1, \beta_2)$, for some $x \in D$.*

$\mathcal{A}$ wins the above game if $\mathsf{Dec}(\mathsf{st}, \mathsf{Sen}(\mathsf{pp}, \mathsf{msg}_R, x, m, \sigma_{(x,m)}; (\alpha_1, \alpha_2, \beta_1, \beta_2)), \Sigma) \neq m$. *For the algorithms* (Gen, Rec, Sen, Dec) *described in Fig. 5, for all non-uniform PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins the above game is negligible, where the probability is taken over the randomness used in the* Gen *and* Rec *algorithms.*

Proof. Since $x \in D$, the receiver has a valid opening proof Π_x and signature σ_x on x. During decryption, the check $e(\Pi_x, \mathsf{ct}_4) \cdot e(\sigma_x, \mathsf{ct}_5) \stackrel{?}{=} \mathsf{ct}_6$ will always pass, on input x but it may also pass on some $x' \neq x$. If using (Π_x, σ_x) to decrypt, we are guaranteed that the receiver always outputs m. Using a similar strategy as in the proof of Claim 6, we can show that it is computationally infeasible for $\mathcal{A}$ to produce sender input and randomness such that the check also passes for some $x \neq x'$. Thus, the probability that $\mathcal{A}$ wins the above game is negligible. □

From Claim 6 and Claim 7, we can show that the payload m and item x that is extracted by the simulator, is the same as the payload and item that the receiver would have output in the real world. This completes the proof of security against a corrupt sender. □

A.3 Generic construction of SPCGS

Theorem 7. *The construction given in Fig. 7 satisfies correctness, traceability, unframeability, anonymity, and client-client anonymity.*

Proof. We adapt the proof of [BGJP23, Theorem 4] with only very minor changes. First, correctness follows immediately from the perfect correctness of the PSI protocol and the completeness of the ZK-NIAoK. Next, the fact that the PSI securely emulates $\mathcal{F}_{\mathsf{PSI}}$ immediately translates to Π_{SPCGS} securely emulating the $\mathcal{F}_{\mathsf{Anon}}$ functionality. We show the other properties in turn.

Traceability. Consider a hybrid experiment where the random oracle H is lazily sampled as follows. The oracle O_{Sign} will respond with a signature $(\mathsf{ct}, \mathsf{crs}, \pi)$ that is computed by running the zero-knowledge simulator on instance $(\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{ct})$ to produce (crs, π). Then, $H(m, \mathsf{ct})$ is set to crs. Any other fresh query (m, ct) to H will be answered by running the knowledge extraction simulator to produce (crs, τ), recording (crs, τ), and setting $H(m, \mathsf{ct}) = \mathsf{crs}$. Note that the knowledge extraction and zero-knowledge properties of the ZK-NIAoK imply that this hybrid is indistinguishable from the real experiment, as long as each (m, ct) queried by O_{Sign} is a fresh query. However, if a ct produced by O_{Sign} has been previously queried by the adversary, then this would contradict the security against outsiders of PSI, since it would imply the ability to predict a msg_S output by the Sen algorithm with noticeable probability.

Now, we show that any adversary that succeeds with noticeable probability in this hybrid experiment can be used to break the security of the one-way relation. Consider a reduction that obtains $(\mathsf{pp}_{\mathcal{R}}, \mathsf{pk}^*)$ from the one-way relation challenger, picks a uniformly random O_{HKG} query to answer with pk^*, and then if the adversary outputs a winning $(m, \sigma) := (m, (\mathsf{ct}, \mathsf{crs}, \pi))$ according to Definition 10, runs the knowledge extractor on $(\tau, (\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{ct}), \pi)$, where τ is the trapdoor recorded along with crs, to produce a witness $(\mathsf{pk}, s, w, \sigma_{\mathsf{pk}}, r)$, and returns w to the challenger. We claim that, conditioned on $(m, \sigma, \mathsf{pk} := \mathsf{Open}(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m))$ satisfying the conditions of Definition 10, it holds that (i) $(\mathsf{pp}_{\mathcal{R}}, \mathsf{pk}, w) \in \mathcal{R}$ with overwhelming probability, and (ii) $\mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}$. Observe that this would complete the proof, since $\mathsf{pk}^* \in \mathcal{I}_{\mathsf{Hon}}$, which is a polynomial size set, so $\mathsf{pk} = \mathsf{pk}^*$ with noticeable probability.

We first argue that, conditioned on the above, τ must exist and thus $(\mathsf{pk}, s, w, \sigma_{\mathsf{id}}, r)$ must be a valid witness with overwhelming probability. By the description of the hybrid game, this follows if (m, ct) was at some point queried to H by the adversary. Now, we know that (m, ct) must have been queried at some point during the experiment, otherwise the crs associated with (m, ct) would be uniformly random and independent of the adversary's view. So it remains to show that (m, ct) could not have been queried to H by the O_{Sign} oracle. Indeed, note that $\mathsf{ct} \notin \mathsf{Sen}(\mathsf{p}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{pk}', \sigma_{\mathsf{pk}'})$ for any $\mathsf{pk}' \neq \mathsf{pk}$, by the perfect correctness of PSI and the fact that $\mathsf{pk} = \mathsf{Dec}(\mathsf{st}_{\mathsf{PSI}}, \mathsf{ct}, \{(m, \widetilde{\sigma}_m)\})$. But we know that any query (m', pk') to O_{Sign} must be such that $(m', \mathsf{pk}') \neq (m, \mathsf{pk})$, since $(m, \mathsf{pk}) \notin \mathcal{M}_{\mathsf{Signed}}$. This establishes point (i). Next, we can claim that $\mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}$. This follows because $\mathsf{pk} \notin \mathcal{I}_{\mathsf{Adv}}$ plus the EUF-CMA security of the signature

scheme, since the knowledge extractor must have produced a valid signature on pk with noticeable probability, and the adversary only has access to signatures under $\mathsf{vk}_{\mathsf{Sig}}$ via O_{AKG} or O_{HKG}.

Unframeability. This proof is similar to the proof of traceability. We consider the same hybrid experiment, where H is lazily sampled, and each crs is computed by running the knowledge extraction simulator except for those crs that result from queries by O_{Sign}, which are computed by running the zero-knowledge simulator. We also consider the same reduction to the hardness of the one-way relation, where the reduction obtain $(\mathsf{pp}_{\mathcal{R}}, \mathsf{pk}^*)$, picks a uniformly random O_{HKG} query to answer with pk^*, and then extracts a witness w from the adversary's proof π in $(m, \sigma) := (m, (\mathsf{ct}, \mathsf{crs}, \pi))$. Here, it just suffices to show that conditioned on $(m, \sigma, \mathsf{pk} := \mathsf{Open}(\mathsf{msk}, m, \sigma, \widetilde{\sigma}_m))$ satisfying the conditions of Definition 11, it holds that $(\mathsf{pp}_{\mathcal{R}}, \mathsf{pk}, w) \in \mathcal{R}$ with overwhelming probability. Indeed, condition (ii) in the previous proof, that $\mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}$, is already enforced by the conditions of Definition 11. This first condition again holds due to the perfect correctness of PSI and the fact that $(\mathsf{mpk}, \mathsf{msk}) \in \mathsf{Gen}(\mathsf{pp}, D)$, which implies that (m, ct) could not have been queried to H by O_{Sign}, and the knowledge extraction of the ZK-NIAoK.

Client-client anonymity. Consider a hybrid experiment where H is lazily sampled as follows. The signature $\sigma = (\mathsf{ct}, \mathsf{crs}, \pi)$ computed on m using sk_b in the description of the game will be computed by running the zero-knowledge simulator on instance $(\mathsf{pp}_{\mathsf{PSI}}, \mathsf{msg}_{R,\mathsf{PSI}}, m, \mathsf{ct})$ to produce (crs, π). Then $H(m, \mathsf{ct})$ is set of crs. The adversary's queries to the O_{Sign} oracle will be answered in the same way. Every other fresh query (m, ct) to H will just be answered with a uniformly random string crs. This hybrid is indistinguishable from the real experiment by the zero-knowledge property of the ZK-NIAoK, as long as each (m, ct) queried by O_{Sign} is a fresh query. We saw in the proof of traceability that this follows from the security against outsiders of PSI. Finally, in this hybrid, client-client anonymity follows directly from the security against outsiders of PSI. □