End-to-End Secure Messaging with Traceability *Only* for Illegal Content

James Bartusek
UC Berkeley

Sanjam Garg
UC Berkeley and NTT Research

Abhishek Jain
JHU

Guru-Vamsi Policharla
UC Berkeley

Abstract

As end-to-end encrypted messaging services become widely adopted, law enforcement agencies have increasingly expressed concern that such services interfere with their ability to maintain public safety. Indeed, there is a direct tension between preserving user privacy and enabling content moderation on these platforms. Recent research has begun to address this tension, proposing systems that purport to strike a balance between the privacy of "honest" users and traceability of "malicious" users. Unfortunately, these systems suffer from a lack of protection against malicious or coerced service providers.

In this work, we address the privacy vs. content moderation question through the lens of pre-constrained cryptography [Ananth et al., ITCS 2022]. We introduce the notion of *set pre-constrained* (SPC) *group signatures* that guarantees security against *malicious key generators*. SPC group signatures offer the ability to trace users in messaging systems who originate pre-defined illegal content (such as child sexual abuse material), while providing security against malicious service providers.

We construct concretely efficient protocols for SPC group signatures, and demonstrate the real-world feasibility of our approach via an implementation. The starting point for our solution is the recently introduced Apple PSI system, which we significantly modify to improve security and expand functionality.

1 Introduction

End-to-end encrypted services offer users the ability to communicate information, with the guarantee that even the service provider itself cannot access the raw information that it is storing or transmitting. Billions of people worldwide are now using end-to-end encrypted systems such as WhatsApp and Signal.

However, the strong data privacy guarantees offered by end-to-end encryption (E2EE) technology have not been universally celebrated. Law enforcement and national security agencies have argued that such services interfere with their ability to prosecute criminals and maintain public safety [18, 41]. In particular, E2EE appears to directly conflict with the goals of *content moderation*, which refers to the ability to screen, monitor, or trace the origin of user-generated content.

One prominent example of the use of content moderation is in fighting the proliferation of child sexual abuse material, or CSAM. In the United States, the proposed EARN IT act [42] would enable legal action to be taken against internet service providers that fail to remove CSAM material from their service. It has been argued that the proposed legislation would inhibit the use of E2EE, which prevents service providers from detecting in the first place if they are hosting or transmitting CSAM [35]. In fact, a 2019 open letter to Facebook signed by then U.S. Attorney General William Barr along with international partners explicitly requested that Facebook not proceed with its planned implementation of E2EE, due to its tension with CSAM detection [40].

One can imagine that this "encryption debate" polarizes to two conceivable outcomes: a world with E2EE but without any content moderation, or a world without E2EE but with content moderation. Since neither of these outcomes seems to be truly satisfactory, it becomes vital to explore the space in between, or more fundamentally, to identify if any such space even exists. Indeed, the past few years have seen researchers paying increased attention to this very question, as covered for example by a recent report [29] released by the Center for Democracy and Technology, and a recent talk about the question of CSAM detection vs. E2EE given at Real World Crypto 2022 [39].

In this work, we explore the viability of using cryptographic techniques to balance the need for both user privacy and illegal content moderation in messaging systems. Along the way, we also study content moderation in the context of encryption systems used by cloud service providers. This might be of independent interest.

Prior solutions. In the setting of encrypted messaging systems, the principle goal of illegal content moderation is to identify the existence of illegal content in the system and uncover the identity of the originator of such content. The desirable privacy goals are to (i) hide the messages exchanged in the system, even from the server, and (ii) preserve the anonymity of the originator of any *harmless* content that is forwarded through the system. Note that this latter property is crucial in many real-world scenarios, e.g., whistleblowers may desire to use the protection provided by E2EE without the threat of being de-anonymized. A recent proposal [30] in this direction fails to adequately balance these goals, allowing a malicious server to de-anonymize *any* user, thereby completely violating the fundamental guarantee of E2EE.

We also note that some recent works have attempted to address the fundamentally different but related question of content moderation for *misinformation*, and we refer to Section 1.3 and Section 1.4 for discussion on this.

The problem. The main problem with existing proposals is that they suffer from a glaring lack of protection against a server who wishes to use the system beyond its prescribed functionality. This is a serious problem, not only because the server itself might have malicious intent, but also because of the threat of coercion from powerful actors that may want to use the technology for surveillance or censorship.

This lack of built-in protection fundamentally damages the transparency of E2EE, reducing the incentive for users to adopt the systems for their communication. While these works have indeed tried to strike a balance between privacy and content moderation, we believe that, for the deterrence of *pre-defined*,[1] *illegal* content (such as CSAM), they have over-compromised on privacy. In this work, we seek to build systems that offer similar content tracing functionality, while offering greater transparency and rigorous cryptographic guarantees about the possible scope of server behavior.

1.1 Summary of Our Contributions

We present novel definitions and efficient protocols for illegal content moderation in the setting of encrypted messaging.

Set Pre-Constrained (SPC) **Group Signatures.** We propose a new notion of *set pre-constrained group signatures* which can be implemented in an end-to-end secure messaging application. This allows tracing users who send illegal content while ensuring privacy for everyone else.

- *Definition*: In SPC group signatures, a database D (of illegal content) can be encoded within the group's public key. The key requirement is that the signer of any message $m \in D$ can be de-anonymized by the group manager but signers of messages $m \notin D$ remain anonymous *even to the*

[1]By pre-defined, we mean any content that has been classified as "illegal", for example by a governmental body, *before* the parameters of the cloud storage or messaging system are sampled. Updating parameters to include new content classified as illegal is an interesting question in this context, which we discuss further in Section 1.3.

group manager. Our definitions model *malicious* group managers and ensure that the group's public key encodes a database D that is authorized by a third-party such as the US National Center for Missing and Exploited Children (or more generally, multiple third parties). Furthermore, the public key is *publicly-verifiable*, so all clients in the system can verify for themselves (without knowing D) whether the group manager's public key encodes an acceptable D.[2]

- *Construction*: We provide a *concretely efficient* construction of SPC group signatures based on standard bilinear map assumptions, in the Random Oracle model. In this construction, we allow the group manager's public key to grow with the size of D. Crucially, however, the running time of the signing algorithm (with oracle access to the public key) as well as verification and tracing is *independent* of the size of D.

SPC **Encryption.** Along the way to constructing SPC group signatures, we define and construct efficient set pre-constrained (SPC) *encryption* schemes. Our construction builds and improves upon the recent Apple PSI protocol [9]: (1) We identify a gap in their proof of security against a malicious server and show how to efficiently build on top of their protocol in order to close this gap. (2) Further, we augment their construction to achieve a stronger notion of security that provides guarantees on the integrity of the database embedded in the public key (analogous to SPC group signatures).

Our SPC encryption scheme has public keys of size linear in the database D and constant encryption and decryption times. We demonstrate that this asymptotic efficiency trade-off is likely the "best-possible" in that further improvements would imply the elusive notion of doubly-efficient private information retrieval [11, 10], which is not known to exist under standard cryptographic assumptions.

Evaluation. We implement our SPC group signature scheme and provide benchmarks in Section 5. We find that signing and verification take tens of milliseconds, and signature size is in the order of a few kilobytes[3]. When instantiated over the BN254 curve, the communication overhead for typical image sizes of 400 KB is under 1% and the additional computation incurs a $\sim$ 15% overhead on top of message delivery time. We view these results as strong initial evidence that illegal content moderation in E2EE messaging systems – with security against malicious servers – can indeed be performed in the real world.

While our current focus is on illegal content moderation, we believe that the efficiency properties of our SPC group signature and encryption schemes make them attractive tools for other applications that involve membership testing against a private "blocklist". Examples include privacy-preserving DNS blocklisting [25] where the blocklist could be proprietary, and anonymous credential systems where it is desirable to hide revocation attributes.

1.2 Our Approach

In this work, we aim to build a messaging system that satisfies, at the very least, the following set of requirements.

1. The system is end-to-end encrypted. In particular, the server cannot learn anything at all about the content transmitted in the system unless it receives some side information from a user participating in the system.

2. The originator of any piece of content remains anonymous to any user that receives the forwarded content.

[2]In the body, we generalize our definition to consider general *functionalities* F as opposed to just the set-membership function specified by D. However, all of our constructions in this work target the special case of sets D, and we restrict our attention to such functionalities in the overview.

[3]More precisely, for the BN254 curve, this translates to 3.5 Kilobytes per SPC group signature.

3. If a user receives some illegal content, they can report it to the server, who can then determine the identity of the user who originated the content. This holds even if the content has been forwarded an arbitrary number of times before being reported.

4. The originator of any *harmless* content remains anonymous, *even from the perspective of the server* who may receive a report about the content.

Naïve Approaches. To demonstrate the challenges in realizing all four properties, we first consider some existing approaches.

As a first attempt, we could try simply using end-to-end encryption. While this may satisfy the first two properties, it clearly does the support the third constraint, which we refer to as *traceability*.

A natural next attempt would be to use a *group signature* scheme [14, 8] underneath E2EE in order to recover this property of traceability. In a group signature scheme, there is a group manager that generates a master public key mpk and a master secret key msk. A new client enters the system by interacting with the group manager in order to receive a client-specific secret key sk. Any client can use their sk to produce a signature σ on a message m, which can be verified by anyone that knows mpk. On the one hand, the identity of the signer remains anonymous from anyone that knows σ but not msk. On the other hand, knowing msk allows the group manager to determine which client produced σ. Thus, we can satisfy the first three goals above by having the messaging service provider additionally take on the role of the group manager. Each user in the system would then obtain a signing key sk from the server, and then attach a signature to any piece of content that they send (where the signature is also transmitted under the encryption). Unfortunately, this solution does not prevent the server from colluding with a user to identify the originator of *any* piece of content received by that user. That is, this solution appears to be fundamentally at odds with the crucial fourth requirement, or *anonymity*, stated above.

Despite some prior attempts at recovering a notion of anonymity in group signature (see Section 1.3 from some more discussion), we conclude that existing frameworks are insufficient for capturing the security that we demand. In order to address this issue, we must somehow *constrain* the ability of the group manager to de-anonymize anyone in the system.

SPC **group signatures: Definitions.** This motivates our first contribution, which is the definition of a *set pre-constrained group signature*, or SPC group signature. In this primitive, the group manager's master public key will be computed with respect to some set D of *illegal content* (which should remain hidden from clients even given the master public key). The novel security property we desire is that the anonymity of a client who produces a signature on some message $m \notin D$ remains intact, *even from the perspective of the group manager.*

More concretely, we ask for the following (informally stated) set of security properties.

- *Traceability*: the identity of a client who signs a message $m \in D$ should be recoverable given the signature and the master secret key.

- *Client-server anonymity*: the identity of a client who signs a message $m \notin D$ should be hidden, even given the master secret key.

- *Set-hiding:* the master public key should not reveal the set D.[4]

- *Unframeability:* no party, not even the master secret key holder, should be able to produce a signature that can be attributed to an honest client.

[4]Note that if we want to prevent even the group manager from seeing / storing the illegal content, we can set D to be hashes of the content itself.

- *Client-client anonymity*: the identity of a client who signs any message m should be hidden from the perspective of any party who does not have the master secret key.

At this point, we must stop to consider the meaningfulness of the above security definitions as stated. In particular: who decides D? Clearly, if D is set to be the whole universe of messages, then this is no more secure than a standard group signature. And if an adversarial group manager is trying to break the client-server anonymity of the above scheme, what is preventing them from generating their master public key with respect to this "trivial" set D?

In order to constrain D in a meaningful way, we introduce a predicate $\mathcal{P}$ into the definition of client-server anonymity. The description of $\mathcal{P}$ will be fixed at setup time along with some public parameters pp known to everybody in the system and secret parameters sp known only to the group manager (we will discuss below the reason we include secret parameters). We will model client-server anonymity using an ideal functionality $\mathcal{F}_{\mathsf{anon}}$ that takes a set of items D as input from the group manager and a sequence of pairs of identities and messages $(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_k, m_k)$ from the client (who represents all clients in the system). If $\mathcal{P}(\mathsf{pp}, \mathsf{sp}, D) = 0$, the functionality aborts, and otherwise it delivers $\{m_i\}_{i \in [k]}, \{\mathsf{pk}_i\}_{i : m_i \in D}$ to the group manager.

This gives us a generic framework for specifying how to constrain the possible D used by the group manager. In particular, we are able to delegate the responsibility of constraining D to a *third-party* (e.g. the National Center for Missing and Exploited Children, or NCMEC), who is tasked with setting up the parameters $(\mathsf{pp}, \mathsf{sp})$ for the predicate $\mathcal{P}$. That is, we can gracefully split the responsibility of implementing / maintaining the encrypted messaging system (by e.g. WhatsApp) and the responsibility of specifying what constitutes illegal content (by e.g. NCMEC or a collection of such agencies).

Perhaps the most natural example of $\mathcal{P}$ is the "subset" predicate, which is parameterized by a set D^* of "allowed" messages (e.g. the entire database of illegal content as defined by NCMEC), and accepts only if $D \subseteq D^*$. In this case, since D^* itself represents illegal content, we do not want to make it public. Thus, we set $\mathsf{sp} = D^*$, and $\mathsf{pp} = |D^*|$. We refer to security with respect to this subset predicate as *authenticated-set security.*

In our full definition, we explicitly consider the third-party Auth as a participant in the system, who begins by setting up a pp and sp of their choice. Then, we require security against an adversary that corrupts *either* the client (and thus cannot learn anything about D), the group manager (and thus can only learn $\{\mathsf{pk}_i\}_{i : m_i \in D}$ for some "valid" D), or the third-party Auth (and thus cannot learn anything about *any* of the identities pk_i). Note that security is only vacuous if the adversary manages to corrupt both the group manager and Auth at the very beginning of the protocol, and thus is able to set sp and D as it wishes. While this seems like a potential limitation, our framework is general enough to support a *de-centralized* Auth. That is, we could consider many third-parties $\mathsf{Auth}_1, \ldots, \mathsf{Auth}_\ell$ who each specify a database D_i^*, and set $\mathcal{P}$ to accept D only if (for example) $D \subseteq D_1^* \cap \cdots \cap D_\ell^*$. Thus, in order to compromise the system, an adversary would have to corrupt the group manager and *all* third-party authorities *simultaneously*, while the key generation procedure is occurring.

SPC group signatures: Construction. We next investigate the feasibility and efficiency of constructing SPC group signatures. To do so, we abstract out the basic "pre-constraining" property we need from the group signature scheme, and re-state it in the context of an *encryption scheme.*

That is, we first define a scheme for what we call *set pre-constrained encryption*, or SPC encryption, with the following properties.

- The public key pk is generated with respect to some database D of items.
- The public key pk should not reveal D, since D may consist of sensitive or harmful content.

- Any user, given pk, can encrypt a message m with respect to an item x such that the key generator (using sk) can recover m if $x \in D$, but learns *nothing* about m if $x \notin D$.

We note that our terminology is inspired by the recent work of Ananth et al. [3] who proposed the notion of pre-constrained encryption. However, our definitions and constructions are quite different; see Section 1.4 for further discussion.

Our security definition for set pre-constrained encryption mirrors the anonymity definition explained above, where the key generator for the encryption scheme now plays of role of the group generator. Specifically, we can still parameterize security by a predicate $\mathcal{P}$ and parameters $(\mathsf{pp}, \mathsf{sp})$ set up by a third-party Auth.

Now, we describe a generic construction of an SPC group signature scheme from an SPC encryption scheme plus standard crytographic tools: a one-way function F, a digital signature scheme, and a zero-knowledge non-interactive argument of knowledge.

The group manager will take as input some set D and sample a public key for the SPC encryption scheme computed with respect to D. It will also include a verification key for the signature scheme in its master public key. A client can join the system by sampling a secret s, setting $\mathsf{id} = F(s)$ to be their public identity, and obtaining a signature on id from the group manager. Now, to sign a message m, the client first encrypts their identity id with respect to item m using the SPC encryption scheme, producing a ciphertext ct. Then, they produce a zero-knowledge proof π that

"I know some id, a signature on id, and s such that $\mathsf{id} = F(s)$, such that ct is an SPC encryption of id with respect to m"

Observe that given any valid signature (ct, π) on a message $m \in D$, the group manager should be able to recover the id that produced (ct, π) by decrypting ct. We refer to this property as *traceability*. One subtle issue that emerges here is that π can only attest that ct is in the space of valid ciphertexts encrypting id under item m, and cannot show that ct was *sampled* correctly. Thus, we will need to require that the SPC encryption is *perfectly correct*, that is, ct is perfectly binding to id when $m \in D$.

Next, we see that any signature (ct, π) on a message m hides id from any other client, which gives us the client-client anonymity property. More specific to our case, we can also show that any signature (ct, π) on a message $m \notin D$ hides id, *even from the server*, which we capture using our simulation-based security definition.

Finally, we highlight the notion of *unframeability*, which requires that a malicious server cannot produce a signature (ct, π) that can be opened to the id of any honest client. Intuitively, this follows because the server will not know the pre-image s of id, and so cannot produce a valid proof π.

SPC Encryption: Construction. With this generic compiler in hand, we provide a *concretely efficient* construction of SPC encryption, and then a *concretely efficient* instantiation of the generic compiler described above. This results in a *practical* proposal for SPC group signatures, which is our main constructive result.

Our construction of SPC encryption builds on top of the Apple PSI protocol [9]. This protocol already satisfies the basic syntax that we require, namely, the ability to embed a set D in the public key pk of an encryption scheme. However, their security notion is much weaker than the authenticated-set security we desire, and described above. Nevertheless, we can capture the security they do claim to achieve using our generic framework, and we refer to it as *bounded-set security*. In more detail, in their scheme, the key generator is completely free to choose the set D, as long as the size of D is below some public bound n. That is, $\mathsf{pp} = n$, sp is empty, and $\mathcal{P}(n, , D) = 1$ if $|D| \leq n$.

Building on their basic scheme, we provide three new contributions.

- We observe that the proof of security (for bounded-set security) given in the Apple PSI paper [9] only holds when the bound n is large enough with respect to other system parameters. This results in a large gap between correctness (the number of items that an honest server programs into its public key) and security (the number of items that a malicious server can potentially program into its public key). We show how to remedy this in a concretely efficient manner, completely closing this gap and achieving essentially no difference between the correctness and security bounds.
- We build on top of the protocol in a different manner in order to establish an efficient protocol that satisfies our novel (and much stronger) definition of *authenticated-set security*.
- We show how to tweak these schemes in order to obtain the *perfect correctness* guarantee needed to make our compiler from SPC encryption to SPC group signatures work. Interestingly, we lose an "element-hiding" property of the scheme in this process. Luckily, we don't require this property for our compiler, since elements correspond to messages in the SPC group signature scheme, which we are not worried about leaking to the server in the event of a user report.

An in-depth overview of the Apple PSI protocol and the technical ideas involved in our improved constructions are given in Section 3.1.

Finally, we derive a *concretely efficient* instantiation of the SPC encryption to SPC group signature compiler, which makes use of structure-preserving signatures [1] and the Groth-Sahai proof system [24]. We provide an overview of the technical ideas involved in our constructions in Section 4.3. We also implement the resulting SPC group signature scheme and provide further discussion and benchmarking in Section 5.

SPC Encryption: Limitations. As a separate contribution, we investigate generic *asymptotic* efficiency properties of SPC encryption. We identify three desirable "succinctness" properties with respect to the database size n: succinct public-key size, succinct encryption time, and succinct decryption time, where in each case, succinctness refers to poly-logarithmic complexity in n. The Apple-PSI-based protocols have non-succinct public-key size, but succinct encryption and succinct decryption. A natural question is whether it is also possible to achieve *succinct* public key. We observe the following, and provide more details in Section 3.4.

- There are techniques in the literature [2] that can achieve succinct public key and succinct encryption with either (i) non-succinct decryption with element-hiding, or (ii) succinct decryption without element-hiding, from standard cryptographic assumptions. However, these constructions are impractical and not suitable for real-world deployment.
- An "optimal" SPC encryption scheme with succinct public key, succinct encryption, succinct decryption, and element-hiding implies the elusive notion of *doubly-efficient private-information retrieval* [11, 10], which is not known to exist under any standard cryptographic assumption.

Thus, while the Apple PSI paper is not explicit about why they settled for a protocol with a non-succinct public key, our analysis validates this choice.

1.3 Discussion

CSAM deterrence vs. misinformation. As mentioned above, CSAM deterrence and combating misinformation are two of the most prominent applications of online content moderation. While both applications indeed fall under the umbrella of content moderation, they each introduce unique challenges from a cryptosystem perspective. The pre-constraining techniques that we make use of in this paper are designed specifically for the deterrence of illegal content, such as CSAM. On the other hand, the "traceback" systems introduced in prior works such as [45, 32, 36] are arguably geared more towards the application of combating misinformation.

Perhaps the biggest distinction between these applications from a cryptographic perspective is their amenability to *pre-definition*. As already discussed, illegal content must be pre-defined in some sense, for example by a governmental body. It is crucial to take advantage of this pre-definition in designing cryptosystems for illegal content deterrence. Indeed, since the description of the illegal content itself can be baked into the parameters of the system, we can hope to obtain rigorous guarantees about *which* content is being tracked and monitored by the system administrator.

On the other hand, it is not even clear in the first place how to define misinformation, or even who has the authority to define it. Plus, new content that could potentially be classified as misinformation is constantly being created and distributed. Thus, it is less clear how to obtain rigorous security guarantees against potentially malicious servers in the setting of misinformation deterrence. A potential approach could be to allow new content (such as new misinformation or abuse) to be added to the "constrained" set, so that the originators of prior messages containing this content could be traced. This feature is reminiscent of "retrospective" access to encrypted data as considered in [22] in a somewhat different context. They show that such access requires the use of powerful (and currently very inefficient) cryptographic tools, and it would be interesting to see if the same implications hold in the setting of tracing in end-to-end encrypted messaging systems.

Deniability vs. unframeability. Another difference between illegal content and misinformation from a cryptographic perspective is reflected in the technical tension between the notions of *deniability* and *unframeability*. Deniability essentially asks that messages between users can be simulated without any user-specific secrets, where indistinguishability from real messages holds from the perspective of an entity with *full information*, including user and even server secrets. This can certainly be a desirable property of encrypted messaging systems, especially when there is a threat of coercion from powerful outside sources. However, this property conflicts with *unframeability* against malicious servers, since it enables servers to produce these simulated messages [44]. While deniability has been a sought-after feature of encrypted systems with traceback functionality [36], it actually appears to be *counter-productive* in systems that are meant to detect originators of CSAM or other illegal content. Indeed, it is important that not only can the server identify the originator, but also that the server can convince law enforcement of the identity of the content originator. On the other hand, we view unframeability against malicious servers as a crucial property of CSAM deterrence systems, since users can face dramatic consequences if framed for the generation or dissemination of illegal content. Thus, our techniques are tailored to obtain the strongest notion of unframeability and no deniability,[5] while prior work [36] that focused on combating misinformation took the opposite approach.

On security against malicious servers. In this work, we took steps towards ensuring privacy and anonymity against malicious (or even honest-but-curious) servers in encrypted systems with support for content moderation. As mentioned earlier in the introduction, it is absolutely vital to explore the space of solutions to the "encrytion debate" that don't give up fully on either end-to-end encryption or content moderation. There is much more work to be done in this space, and we view our techniques as one tool in an ever-expanding toolbox of techniques meant to address the broad question of privacy vs. content moderation.

In particular, while we remove the need to trust *service providers* (think, WhatsApp), the notion of authenticated-set security essentially moves this trust to a third party (think NCMEC). We consider this progress, since it splits the responsibility of providing a messaging service and defining illegal content.

[5] Though we note that one could potentially alter our group signature scheme to obtain deniability at the cost of unframeability, by including in the zero-knowledge argument a clause along the lines of "OR I know the master secret key".

Moreover, as discussed earlier, our scheme would immediately extend to support *multiple* third parties that can each attest to the validity of the server's public parameters, further splitting the trust. However, we acknowledge that there is opportunity to further improve the transparency and trust in such content moderation systems.

Additional challenges and future directions. We conclude our discussion with a few directions for future work. First, a desirable property of encrypted illegal content moderation systems is the ability to *update* public parameters to include new illegal content. As discussed in the Apple PSI paper [9], a simple way to handle updates is to redo setup and release the updated public key as part of system update. Achieving more efficient updates, however, is an interesting direction for future work. For example, if an update only corresponds to locations that are changed, it may start leaking the positions that correspond to database elements. This suggests the need for creative solutions, for example the use of differential privacy techniques to hide this leakage.

Next, we did not consider *thresholding* in this work, which would protect the privacy of content or anonymity of users until *multiple matches* were found in the database. While this is straightforward to incorporate into SPC encryption, it is not as immediate for SPC group signatures, at least if the goal is to maintain concrete efficiency. We leave an exploration of this to future work.

Next, we chose to use Groth-Sahai proof systems in order to demonstrate that SPC group signatures could be constructed with reasonable efficiency. However, there are other tools available, such as efficient SNARGs (succinct non-interactive arguments) that may result in better verification time at the cost of increased signer work. We leave further investigation of this to future work.

Finally, we mention broader considerations that would come with using our system in the real world. In the system, the actual database D would likely not consist of the actual CSAM images themselves, but rather *hashes* of CSAM images computed using a perceptual hash function, such as Apple's NeuralHash [4]. This introduces the possibility of adversarial use of the hash function, for example targeted collision-finding. We view this as an important attack vector to consider, especially when using these hash functions in conjunction with cryptographic protocols meant to provide privacy against malicious servers. Exploration of this topic is outside the scope of the current work, and we refer the reader to [38] and references therein for current research on the topic.

1.4 Related Work

Pre-Constrained Cryptography. Our work borrows the terminology of pre-constrained cryptography from Ananth et al. [3] because of sharing a similar vision – that of putting pre-specified restrictions on the key generation authority. Our definitions and constructions, however, are different from [3]. First, we note that the notion of (set) pre-constrained group signatures is *new* to our work, while Ananth et al. [3] only focus on (pre-constrained) encryption systems. In the setting of pre-constrained encryption, the notion of malicious security in [3] is weaker than ours and allows the authority to choose *any* "constraint" from a class of constraints. This weaker notion is not meaningful in our setting, as it allows the service provider (think, WhatsApp) to use an *arbitrary set* of their choice. Ananth et al. propose constructions for different flavors of pre-constrained encryption; the one that comes closest to our setting relies on indistinguishability obfuscation [6], and is only of theoretical interest. In contrast, we provide concretely efficient constructions for our setting.

Traceback Systems. While our work focuses on moderation for *pre-defined illegal content*, there has also been much recent work on the adjacent question of moderation for *misinformation* or *abusive* content. Solutions for this problem typically build "traceback" mechanisms into end-to-end encrypted systems [45, 32, 36, 28], extending the reach of so-called "message franking" systems [26, 16, 44]. These solutions rely on user reporting to identify the existence of harmful content. Once a report is received by the server, the server and reporting user can work together to identify the originator of the harmful message. Unfortunately, these systems suffer from various drawbacks [21]: (1) They allow colluding server and users to de-anonymize the originator of *any* message, even if the content is harmless. (2) Initial solutions in this space additionally require the help of users on the traceback path to identify the originator, and do not maintain their anonymity. While the latter drawback was addressed in the recent work of [36], no known solution provides security guarantees against malicious servers. Our system addresses both of these shortcomings, for our specific setting of illegal content moderation.

Group Signatures. Finally, we mention a related line of work on group signatures with *message-dependent opening* (GS-MDO) [17, 31]. Here, trust is split between the group manager and an additional entity called the "admitter". The identity of a group member that produces a signature on a message m can be revealed only if the group manager and admitter combine their private information. Unlike SPC group signatures, GS-MDO does *not* require any "commitment" to, or "pre-constraining" of, the set of messages that can be de-anonymized. This means that even after the system parameters are set up, the group manager and admitter can in principle work together to de-anonymize *every* signature while still acting "semi-honestly" w.r.t. the protocol specification. In particular, clients of the system will not have the peace of mind guaranteed by public parameters that are publicly "authenticated" to only allow de-anonymization of a particular set of illegal content specificied by some trusted (collection of) third party(ies).

2 Preliminaries

The security parameter is denoted by $\lambda \in \mathbb{N}$. A function $f : \mathbb{N} \to \mathbb{N}$ is said to be polynomial if there exists a constant c such that $f(n) \le n^c$ for all $n \in \mathbb{N}$, and we write $\mathsf{poly}(\cdot)$ to denote such a function. A function $f : \mathbb{N} \to [0,1]$ is said to be negligible if for every $c \in \mathbb{N}$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$, and we write $\mathsf{negl}(\cdot)$ to denote such a function. A probability is *noticeable* if it is not negligible, and *overwhelming* if it is equal to $1 - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\lambda)$. For a set $\mathcal{S}$, we write $s \leftarrow \mathcal{S}$ to indicate that s is sampled uniformly at random from $\mathcal{S}$. For a random variable $\mathcal{D}$, we write $d \leftarrow \mathcal{D}$ to indicate that d is sampled according to $\mathcal{D}$. An algorithm $\mathcal{A}$ is PPT (probabilistic polynomial-time) if its running time is bounded by some polynomial in the size of its input. For two ensembles of random variables $\{\mathcal{D}_{0,\lambda}\}_{\lambda\in\mathbb{N}}$, $\{\mathcal{D}_{1,\lambda}\}_{\lambda\in\mathbb{N}}$, we write $\mathcal{D}_0 \approx_c \mathcal{D}_1$ to indicate that for all PPT $\mathcal{A}$, it holds that

$$\left| \Pr_{d\leftarrow\mathcal{D}_{0,\lambda}}[\mathcal{A}(d) = 1] - \Pr_{d\leftarrow\mathcal{D}_{1,\lambda}}[\mathcal{A}(d) = 1] \right| \le \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.1 Basic cryptographic primitives and assumptions

We will use a standard symmetric-key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with keypspace $\mathcal{K}$ that satisfies *random key robustness*, which states that for any message m, $\Pr_{k,k'\leftarrow\mathcal{K}}[\mathsf{Dec}(k', \mathsf{Enc}(k, m)) = \bot] = 1 - \mathsf{negl}(\lambda)$. We will also make use of a standard digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ that is *existentially unforgeable under chosen message attack* (EUF-CMA):

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.

The Diffie-Hellman problem. Let $\mathbb{G}$ be a group of prime order q with generator g. The tuple $(g, h_1, h_2, h_3) \in \mathbb{G}^4$ is called a *Diffie-Hellman tuple* if there exists $\alpha \in \mathbb{Z}_q$ such that $h_1 = g^\alpha$ and $h_3 = h_2^\alpha$. The *decisional Diffie-Hellman* (DDH) assumption in group $\mathbb{G}$ asserts that it is computationally difficult to distinguish a random Diffie-Hellman (DH) tuple from uniformly random group elements.

We now describe a random self-reduction for Diffie-Hellman tuples, due to [34]. Consider fixed group elements (g, h_1, h_2, h_3), and uniformly random $\beta, \gamma \leftarrow \mathbb{Z}_q$. Let $h_2' = g^\beta \cdot h_2^\gamma$, and $h_3' = h_1^\beta \cdot h_3^\gamma$ Then the following two properties hold.

1. If (g, h_1, h_2, h_3) is a Diffie-Hellman tuple, then (h_2', h_3') are uniform conditioned on (g, h_1, h_2', h_3') being a Diffie-Hellman tuple.

2. If (g, h_1, h_2, h_3) is not a Diffie-Hellman tuple, then (h_2', h_3') are fresh uniformly random group elements.

2.2 Simulation-based security

In this work, we will sometimes prove security via simulation, following the standard real/ideal world paradigm [20]. In this paradigm, a cryptographic scheme specifies an interactive protocol Π that takes place between some parties $P_1, \ldots, P_n$ initialized with inputs $x_1, \ldots, x_n$. The protocol Π is meant to emulate some ideal functionality $\mathcal{F}$ that takes an input $x_1, \ldots, x_n$ from each party and delivers an output $y_1, \ldots, y_n$ to each party.

The real execution. In the real execution, the protocol Π is executed in the presence of an adversary $\mathcal{A}$ that corrupts some subset $M \subset [n]$ of n parties. $\mathcal{A}$ takes as input the security parameter 1^λ, a set of input $\{x_i\}_{i \in M}$, and an auxiliary input z. The honest parties $[n] \setminus M$ follow the instructions of Π, while $\mathcal{A}$ sends messages on behalf of parties in M. If $\mathcal{A}$ is *malicious*, these messages may be computed following an arbitrary polynomial-time strategy, while if $\mathcal{A}$ is *semi-honest*, these messages must be computed following the instructions of Π. The interaction of $\mathcal{A}$ in the protocol Π defines a random variable $\mathsf{REAL}_{\Pi,\mathcal{A}}[1^\lambda, \vec{x}, z, M]$ that descibed the output of $\mathcal{A}$, which may be an arbitrary funtion of its view.[6]

The ideal execution. In the ideal execution, a simulator Sim controlling some subset $M \subset [n]$ of n parties interacts with a trusted party $\mathcal{I}_\mathcal{F}$ implementing the functionality $\mathcal{F}$. Sim takes as input the security parameter 1^λ, a set of inputs $\{x_i\}_{i \in M}$, and an auxiliary input z. Each honest party $P_i \in [n] \setminus M$ sends their input x_i to $\mathcal{I}$, while Sim sends an input x_i' on behalf of each party $P_i \in M$. Let $x_1', \ldots, x_n'$ be the entire set of inputs received by $\mathcal{I}$. Next, $\mathcal{I}$ computes $(y_1, \ldots, y_n) = \mathcal{F}(x_1', \ldots, x_n')$ and delivers $\{y_i\}_{i \in M}$ to Sim. Finally, Sim outputs an arbitrary function of its view, which defines a random variable $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}}[1^\lambda, \vec{x}, z, M]$.

Definition 2. *An n-party protocol Π securely emulates an ideal functionality $\mathcal{F}$ in the presence of malicious (resp. semi-honest) adversaries corrupting a subset of parties $M \subset [n]$ if for any PPT malicious (resp. semi-honest) $\mathcal{A}$ corrupting parties M, there exists a PPT Sim such that for any set of inputs $\vec{x}$, and auxiliary input z,*

$$\mathsf{REAL}_{\Pi,\mathcal{A}}[1^\lambda, \vec{x}, z, M] \approx_c \mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim}}[1^\lambda, \vec{x}, z, M].$$

[6]Typically, this output would also include the outputs of the honest parties. However, we will not require correctness against malicious parties in this work, and so do not include the honest party outputs.

The random oracle model. In the random oracle model (ROM), parties are given *oracle access* to some function H that is sampled uniformly at random from the space of all functions $H : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are finite non-empty sets. That is, parties can query their oracle on an input $x \in \mathcal{X}$ and receive in return $H(x) \in \mathcal{Y}$. When proving security of a protocol Π in the random oracle model, the simulator Sim is able to "control" the oracle, observing queries made by the adversary and simulating responses.

2.3 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 $\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}$, 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 $\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\} \approx_c \{\mathsf{crs} : (\mathsf{crs}, \tau) \leftarrow \mathcal{E}_1(1^\lambda)\}$, and for any PPT adversary $\mathcal{A}$,

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

- *Zero-knowledge.* There exists a simulator $\mathcal{S}$ such that for any $(x, w) \in \mathcal{R}$,

$$\left\{(\mathsf{crs}, x, \pi) : \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) \end{array}\right\} \approx_c \left\{(\mathsf{crs}, x, \pi) : (\mathsf{crs}, \pi) \leftarrow \mathcal{S}(1^\lambda, x)\right\}$$

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. We will use the fact that the following relations all have highly efficient ZK-NIAoKs in the ROM. Let $\mathbb{G}$ be a group of order q with generator g.

- The relation $\mathcal{R}_{\mathsf{DLog}} = \{((g, h), \alpha) : h = g^\alpha\}$. A ZK-NIAoK for $\mathcal{R}_{\mathsf{DLog}}$ follows from applying the Fiat-Shamir heuristic [19] to Schnorr's sigma protocol [43].

- The relation $\mathcal{R}_{\mathsf{DH}} = \{((g, h_1, h_2, h_3), \alpha) : (h_1 = g^\alpha) \wedge (h_3 = h_2^\alpha)\}$. A ZK-NIAoK for $\mathcal{R}_{\mathsf{DH}}$ follows from applying the Fiat-Shamir heuristic to Chaum and Pederson's sigma protocol [13].

- For any n and $k \leq n$, the relation $\mathcal{R}_{\mathsf{DLog}_n^k} = \{((g, h_1, \ldots, h_n), (S, \{\alpha_i\}_{i \in S})) : (|S| = k) \wedge (\forall i \in S, h_i = g^{\alpha_i})\}$. A ZK-NIAoK for $\mathcal{R}_{\mathsf{DLog}_n^k}$ follows from applying the Fiat-Shamir heuristic to the protocol of [15]. Moreover, an efficient *succinct* argument system for this language whose size is logarithmic in n, was shown recently by [5].

2.4 Cuckoo hashing

A cuckoo hashing scheme consists of the algorithms (Setup, 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.[7]
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 $(m, h_0, h_1) \leftarrow \mathsf{Setup}(\lambda, n, \epsilon)$, 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 $T_{h_1(x)} = x$, where $T := \mathsf{Hash}(h_0, h_1, D)$.

2.5 Bilinear maps

Let $\mathcal{G}$ be a bilinear group generator that on input 1^λ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{T}, e, g_1, g_2)$, where

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{T}$ are groups of order p, where p is a λ-bit prime.
- g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and e is a non-degenerate bilinear map. That is, $e(g, g)$ is a generator of $\mathbb{T}$, and for all $a, b \in \mathbb{Z}_p$, it holds that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- Group operations, evaluation of the bilinear map, and group membership are all efficiently computable.
- The DDH assumption is assumed to hold in each of $\mathbb{G}_1$ and $\mathbb{G}_2$. In other words, the SXDH (symmetric external Diffie-Hellman) assumption is assumed to hold.

Structure-preserving signatures. Structure-preserving signatures [1] are digital signatures schemes where the messages, verification key, and signatures are group elements, and the verification equation evaluates "pairing product equations". The formal definition follows.

Definition 3. *A structure-preserving signature defined with respect to a group with bilinear map* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{T}, e, g_1, g_2)$ *satisfies the following properties.*

- *The messages, verification key, and signatures are all group elements in* $\mathbb{G}_1$ *or* $\mathbb{G}_2$*.*
- *The* Verify *algorithm takes as input group elements in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, verifies group membership, and evaluates pairing product equations of the form*

$$\prod_i \prod_j e(A_i, B_j)^{a_{i,j}} = 1,$$

where $A_i \in \mathbb{G}_1$, $B_j \in \mathbb{G}_2$ *are inputs, and* $a_{i,j}$ *are constants in* $\mathbb{Z}_p$.

[7]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.

Groth-Sahai proofs. Groth and Sahai [24] 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 $\mathbb{T}$. [8]

- *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 $\mathbb{G}_2$.

They also consider quadratic equations in $\mathbb{Z}_p$, but these will not be used in our work.

GS proofs are in the *common random string* model, and satisfy the completeness and zero-knowledge properties described in Section 2.3. However, they only satisfy a weaker notion of knowledge extraction which has been referred to as *partial* knowledge extraction [23]. This property states that if the witness consists of both group elements and exponents, only the group elements are extractable.

2.6 Doubly-efficient private information retrieval

We use a slightly weaker definition of doubly-efficient private information retrieval compared to that found in the literature [11, 10] and henceforth refer to it as DEPIR. A DEPIR scheme (with preprocessing) consists of four algorithms (KeyGen, Process, Query, Resp, Decode).

- $\mathsf{Gen}(1^\lambda, \mathsf{DB}) \to (k_c, k_s)$: takes as input a database DB and outputs a pair of keys (k_c, k_s).
- $\mathsf{Query}(k_c, i) \to (q, st)$: takes k_c and index $i \in [n]$ as input and outputs a query q and local state st.
- $\mathsf{Resp}(q, k_s) \to a$: takes q and k_s and returns the server answer a.
- $\mathsf{Decode}(\mathsf{st}, a) \to b$: takes local state st and a and outputs a bit.

The main difference in our definition is that the key generation procedure Gen depends on the database. As part of our efficiency requirements we demand that the size of $k_c = \mathrm{poly}(\lambda, \log n)$, Gen runs in time $\mathrm{poly}(\lambda, n)$ and Query, Resp and Decode run in time $\mathrm{poly}(\lambda, \log n)$. We note that this definition is interesting only when the key k_c is succinct, otherwise the entire database can be stored as part of the key.

A DEPIR scheme Π is secure if these exists a polynomial time simulator $\mathcal{S}$ such that $\mathsf{REAL}_{\mathcal{A},\Pi} \approx \mathsf{IDEAL}_{\mathcal{A},\mathcal{S}}$ where:

$\mathsf{REAL}_{\mathcal{A},\Pi}$ is the output of $\mathcal{A}$ in the following interaction

1. $\mathcal{A} \to \mathsf{DB}$; $\mathcal{A}$ obtains $k_c, k_s \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{DB})$.

[8] [24] 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.

2. $\mathcal{A} \to i \in [N]$ and obtains $q \leftarrow \mathsf{Query}(k_c, i)$.
3. Repeat step 2 until $\mathcal{A}$ generates an output.

$\mathsf{IDEAL}_{\mathcal{A},\mathcal{S}}$ is the output of $\mathcal{A}$ in the following interaction

1. $\mathcal{A} \to \mathsf{DB}$; $\mathcal{A}$ obtains $k_c, k_s \leftarrow \mathcal{S}(\mathsf{DB})$.
2. $\mathcal{A} \to i \in [N]$ and obtains $q \leftarrow \mathcal{S}()$.
3. Repeat step 2 until $\mathcal{A}$ generates an output.

3 Set Pre-Constrained Encryption

In this section, we define and construct set pre-constrained (SPC) encryption. We start by providing an overview in Section 3.1. We then present formal definitions of SPC encryption in Section 3.2, and constructions in Section 3.3. In Section 3.4 we demonstrate that an *optimal* version of SPC encryption implies doubly-efficient private information retrieval.

3.1 Overview

The basic Apple PSI protocol. We start by recalling the basic Apple PSI protocol, viewed as an *encryption scheme*. "Basic" here refers to the protocol without the extra threshold or synthetic match functionalities, which we will not consider explicitly in this work.

A key technique used in Apple's protocol is the Naor-Reingold Diffie-Hellman random self reduction [34]. Let $\mathbb{G}$ be a cyclic group of order q with generator g, and let h_1, h_2, h_3 be three other group elements. Suppose that $\beta, \gamma \leftarrow \mathbb{Z}_q$ are sampled as uniformly random exponents, and $h_2' := g^\beta \cdot h_2^\gamma, h_3' := h_1^\beta \cdot h_3^\gamma$. Then it holds that (i) if (g, h_1, h_2, h_3) is a Diffie-Hellman tuple (that is, there exists α such that $g^\alpha = h_1$ and $h_2^\alpha = h_3$), then (g, h_1, h_2', h_3') is a Diffie-Hellman tuple, and (ii) if (g, h_1, h_2, h_3) is *not* a Diffie-Hellman tuple, then (h_2', h_3') are fresh uniformly random group elements.

Now, this self-reduction can be used to construct a set pre-constrained encryption scheme for a single-item set $\{x\}$ as follows. Let H be a hash function that hashes items to group elements (H will be treated as a random oracle in the security proof). The key generator, on input an item x, will sample $\alpha \leftarrow \mathbb{Z}_q$ and publish $(A = g^\alpha, B = H(x)^\alpha)$ as the public key. Note that $(g, A, H(x), B)$ is a Diffie-Hellman tuple, while for any $x' \neq x$, $(g, A, H(x'), B)$ is *not* a Diffie-Hellman tuple. This suggests a natural encryption scheme. Given the public key, an item y, and a message m, the encryption algorithm will run the Naor-Reingold self-reduction on $(g, A, H(y), B)$ to produce group elements (Q, S), and then treat S as a secret key for encrypting the message m. That is, the ciphertext will consist of $(Q, \mathsf{SEnc}_S(m))$, where SEnc is a symmetric-key encryption scheme. If $y \neq x$, then S will be uniformly random, even from the key generator's perspective, so m remains hidden. On the other hand, if $y = x$, then (g, A, Q, S) is a Diffie-Helman tuple, and the element $S = Q^\alpha$ can be computed by the key generator and used to recover m.

This scheme can easily be extended to support larger set sizes, by having the key generator publish $(A, H(x_1)^\alpha, \ldots, H(x_n)^\alpha)$ as the public key, where $x_1, \ldots, x_n$ is its input set. However, the naive extensions of the encryption and decryption algorithms described above will have running time that grows with the size n of the set. The authors of the Apple PSI system make use of a technique called *cuckoo hashing* to significantly reduce this running time. Concretely, the key generator will hash the set $(x_1, \ldots, x_n)$ into a table T of size $n' = (1 + \epsilon)n$ for some constant ϵ, using randomly sampled hash keys h_0, h_1. The guarantee is that with high probability, for most x_i, either $T_{h_0(x_i)} = x_i$ or $T_{h_1(x_i)} = x_i$. Note that T will have $n' - n$ empty entries, which we denote with $\perp$. The key generator will then publish $(A, B_1, \ldots, B_{n'})$ as the public key, where for each $i \in [n']$, if $T_i = x$ then $B_i = H(x)^\alpha$, while if $T_i = \perp$ then $B_i = g^r$ for a random exponent r. Now, to encrypt a message m with respect to an item y, one only has to produce two pairs

$(Q_0, \mathsf{SEnc}_{S_0}(m)), (Q_1, \mathsf{SEnc}_{S_1}(m))$, where (Q_b, S_b) is the result of applying the Naor-Reingold self-reduction to $(g, A, H(y), B_{h_b(y)})$.

This results in a set pre-constrained encryption scheme that can handle pre-constraining sets of size n with a public key of about $n' = (1+\epsilon)n$ group elements, and encryption and decryption algorithms whose running times do not grow with the size of n. One can show (in the random oracle model) that this scheme already satisfies set-hiding under the DDH assumption, and can be made to satisfy element-hiding from DDH, as long as the two pairs $(Q_0, \mathsf{SEnc}_{S_0}(m)), (Q_1, \mathsf{SEnc}_{S_1}(m))$ that constitute the ciphertext are randomly permuted, and $B_1, \ldots, B_{n'}$ are all distinct group elements.

Achieving bounded-set and authenticated-set security. Next, we show that augmenting the above template with *simple* and *efficient* zero-knowledge arguments suffices to achieve first bounded-set and next authenticated-set security. While the potential utility of adding zero-knowledge arguments to Apple's PSI system has previously been discussed informally [12, 37], we view our formalization and efficient realization of rigorous security definitions as a necessary and important contribution in this space.

The Apple PSI paper [9] actually already claims to achieve *bounded-set* security, which guarantees that a malicious key generator can only decrypt messages that are encrypted with respect to some set of items of size at most B. However, it is left unclear what B is, and how it depends on other parameters in the system. In fact, their proof completely breaks down if $B < n'$. In particular, their proof relies on extracting the input set X of the key generator by observing random oracle queries, potentially adding one item x to X for each group element B_i in the public key. If the resulting X is such that $|X| > B$, then the ideal functionality aborts, and the malicious key generator would not receive encryptions from the client. However, this behavior does not reflect what would happen in the real world, where the client would not be able to tell how large the key generator's "effective input" actually is.

This issue in the proof occurs with good reason, since a malicious key generator can indeed publish $(A, H(x_1)^\alpha, \ldots, H(x_{n'})^\alpha)$ for n' items $x_1, \ldots, x_{n'}$ without being detected. However, correctness for honest key generators is only guaranteed to hold for up to n items (due to the cuckoo hashing). Thus, in the best case, we could hope for a scheme that achieves bounded-set security with bound n.

We show how to achieve this by instructing the key generator to append to their key $(A, B_1, \ldots, B_{n'})$ a zero-knowledge non-interactive proof of knowledge that they know the discrete logarithm α of A and at least $n' - n$ discrete logarithms $\{r_i\}$ of the elements $B_1, \ldots, B_{n'}$. Highly efficient proofs supporting these languages are known [15, 5]. Intuitively, the $n' - n$ group elements B_i for which the generator knows r_i such that $B_i = g^{r_i}$ are "useless" for decrypting encrypted messages. To see why, recall that, due to the Naor-Reingold self-reduction, B_i can only be used to decrypt with respect to an item x such that $(g, A, H(x), B_i)$ forms a Diffie-Hellman tuple. However, if the generator knows an x, α, and r_i such that this holds, they can break the discrete logarithm problem, since $H(x) = g^{r/\alpha}$ and $H(x)$ can be programmed by a reduction. Thus, only at most n of the elements $(B_1, \ldots, B_{n'})$ will actually be useful for decrypting messages, which we leverage to show bounded-set security with a bound of n.

Next, we consider our notion of *authenticated-set* security, which introduces a third party that chooses the set D. In our scheme, the third party first sends D to the key generator. Then, the key generator prepares a public key $(A, B_1, \ldots, B_{n'})$. In the honest case, for each i it either holds that there exists $x \in D$ such that $(g, A, H(x), B_i)$ form a Diffie-Hellman tuple, or the generator knows r_i such that $B_i = g_i^r$. Now, these are claims that the generator can prove efficiently in zero-knowledge to the third party. The third party will then checks these proofs, and if all verify, will sign the set of group elements $(A, B_1, \ldots, B_{n'})$ under its public verification key. We show in Section 3.3 that this is sufficient for achieving authenticated-set security.

3.2 Definitions

A set pre-constrained encryption (SPCE) scheme $\Pi_{\mathsf{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ consists of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and is parameterized by a universe $\mathcal{U}$ of elements, a message space $\mathcal{M}$, a set size n, and a correctness parameter ϵ. $\mathcal{U}, \mathcal{M}, n, \epsilon$ may actually be infinite families parameterized by the security parameter λ, though we suppress mention of this for ease of notation.

- $\mathsf{Gen}(1^\lambda, D) \to (\mathsf{pk}, \mathsf{sk})$: the parameter generation algorithm takes as input a security parameter 1^λ and a set $D \subseteq \mathcal{U}$ of size at most n, and outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, x, m) \to \mathsf{ct}$: the encryption algorithm takes as input a public key pk, an item $x \in \mathcal{U}$, and a message $m \in \mathcal{M}$, and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \{m, \bot\}$: the decryption algorithm takes as input a secret key sk and a ciphertext ct and outputs either a message $m \in \mathcal{M}$ or a bot symbol $\bot$.

We note that any SPC encryption scheme can be utilized for encrypted cloud storage as follows. The server initially publishes pk, and whenever the client wants to upload some content x, they would sample an (element-hiding) SPC encryption of (x, m), where m is arbitrary "associated data" (e.g. the name of the client). Then, if $x \in D$, the server would be able to use sk to recover the associated data m. Otherwise (x, m) will remain hidden from the server.

Efficiency. By default, all algorithms in an SPC encryption scheme should be polynomial-time in the size of their inputs, and n,$|x|$,$|m|$ should be polynomial-size in λ. However, we will want to consider a more fine-grained notion of efficiency with respect to the size n of the set D, which may be a large polynomial. We say that the scheme has *succinct public-key* if $|\mathsf{pk}| = \mathsf{poly}(\lambda, \log n)$, *succinct encryption* if the running time of Enc is $\mathsf{poly}(\lambda, \log n)$, and *succinct decryption* if the running time of Dec is $\mathsf{poly}(\lambda, \log n)$.

Correctness. We define notions of correctness for an SPC encryption scheme. We first consider the following notion of ϵ-correctness, where the parameter ϵ essentially determines an upper bound on the fraction of the set D that is "dropped" by the Gen algorithm.[9]

Definition 4. *An* SPC *encryption scheme (*$\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$*) is ϵ-correct for some $\epsilon \geq 0$ if for any $\lambda \in \mathbb{N}$ and $D \subseteq \mathcal{U}$, it holds that with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, D)$*, there exists a $D' \subseteq D$ such that $|D'| \geq (1 - \epsilon)|D|$ and for any $x \in D'$ and $m \in \mathcal{M}$,*

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x, m)) = m] = 1 - \mathsf{negl}(\lambda).$$

Next, we define the stronger notion of *perfect* ϵ-correctness that will be useful for our application of SPC encryption to building SPC group signatures in Section 4.

Definition 5. *An* SPC *encryption scheme (*$\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$*) is perfectly ϵ-correct for some $\epsilon \geq 0$ if for any $\lambda \in \mathbb{N}$ and $D \subseteq \mathcal{U}$, it holds that with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, D)$*, there exists a $D' \subseteq D$ such that $|D'| \geq (1 - \epsilon)|D|$ and for any $m \in \mathcal{M}$, it holds that*

- *For every $x \in \mathcal{U}$ such that $x \in D'$,*
$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x, m)) = m] = 1.$$
- *For every $x \in \mathcal{U}$ such that $x \notin D'$,*
$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x, m)) \in \{m, \bot\}] = 1.$$

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

Security. We define security using the simulation framework, via an ideal functionality described in Fig. 1. The ideal functionality $\mathcal{F}_{\mathsf{SPCE}}^{\mathcal{P}}$ takes place between a *server*, who runs Gen and Dec, a *client*, who runs Enc, and a third party Auth, whose role will be described below. In full generality, the server's input is a function F, but in our applications, we will always parse F as a description of a database D of items. The client's input is a sequence of items and messages $(x_1, m_1), \ldots, (x_k, m_k)$. The client should learn nothing about D, Auth should learn nothing about the messages $m_1, \ldots, m_k$, and the server should learn only $\{m_i\}_{i:x_i \in D}$ (and potentially the elements $\{x_i\}_{i \in [k]}$).

To make security against the server meaningful, we must place some restriction on D. We do this (in a modular way) by parameterizing the functionality with a predicate $\mathcal{P}$. This predicate may depend on some public parameters pp (known to both client and server) and some secret parameters sp (known only to the server). It is the job of Auth to set up these parameters. We allow a malicious adversary to corrupt either the server, the client, or Auth. We note that one could also consider collusions between any pair of parties, but in each case security becomes vacuous, so we do not consider this in our proofs of security.

Below, we describe the instantiations of $\mathcal{P}$ that we will consider in this work: one will define what we call *bounded-set security* and the other will define what we call *authenticated-set security*.

$\mathcal{F}_{\mathsf{SPCE}}^{\mathcal{P}}$

Public information.

- Parties: server S, client C, and authority Auth.
- Parameters: universe $\mathcal{U}$, message space $\mathcal{M}$.

The functionality.

- Obtain input (pp, sp) from Auth. Deliver pp to both C and S, and sp to S.
- Obtain input $F = (F_S, F_{\mathsf{Auth}})$ from S and deliver F to Auth. Abort and deliver $\perp$ to all parties if $\mathcal{P}(\mathrm{pp}, \mathrm{sp}, F) = 0$.
- Obtain input $(x_1, m_1), \ldots, (x_k, m_k)$ from client, where each $x_i \in \mathcal{U}$ and each $m_i \in \mathcal{M}$.
- Deliver $F_S(\{x_i, m_i\}_{i \in [k]})$ to server and $F_{\mathsf{Auth}}(\{x_i, m_i\}_{i \in [k]})$ to Auth.

Figure 1: Ideal functionality for SPC encryption. $\mathcal{P}$ is a predicate that takes as input some public parameters pp, secret parameters sp, and a pair of functions $F = (F_S, F_{\mathsf{Auth}})$, and outputs a bit.

Bounded-set security. Here, we define two predicates $\mathcal{P}[\mathsf{BS}]$ and $\mathcal{P}[\mathsf{BS\text{-}EH}]$, where BS stands for bounded-set, and EH stands for element-hiding. For each, the public parameters pp are parsed as an integer n, there are no secret parameters sp, and F is parsed as the description of a database $D \subseteq \mathcal{U}$. The predicate then outputs 1 if and only if $|D| \leq n$. For $\mathcal{P}[\mathsf{BS}]$,

$$F_S(\{x_i, m_i\}_{i \in [k]}) = \{x_i\}_{i \in [k]}, \{m_i\}_{i:x_i \in D}, \quad F_{\mathsf{Auth}}(\{x_i, m_i\}_{i \in [k]}) = \{x_i\}_{i \in [k]},$$

and for $\mathcal{P}[\mathsf{BS\text{-}EH}]$,

$$F_S(\{x_i, m_i\}_{i \in [k]}) = k, \{m_i\}_{i:x_i \in D}, \quad F_{\mathsf{Auth}}(\{x_i, m_i\}_{i \in [k]}) = k.$$

Authenticated-set security. Here, we define two predicates $\mathcal{P}[\mathsf{AS}]$ and $\mathcal{P}[\mathsf{AS\text{-}EH}]$, where AS stands for authenticated-set. For each, the public parameters pp are parsed as an integer n, the secret parameters are parsed as a database $D^* \subseteq \mathcal{U}$ of size n, and F is parsed as a database $D \subseteq \mathcal{U}$. The predicate then outputs 1 if and only if $D \subseteq D^*$. For $\mathcal{P}[\mathsf{AS}]$,

$$F_S(\{x_i, m_i\}_{i\in[k]}) = \{x_i\}_{i\in[k]}, \{m_i\}_{i:x_i\in D}, \quad F_{\mathsf{Auth}}(\{x_i, m_i\}_{i\in[k]}) = \{x_i\}_{i\in[k]},$$

and for $\mathcal{P}[\mathsf{AS\text{-}EH}]$,

$$F_S(\{x_i, m_i\}_{i\in[k]}) = k, \{m_i\}_{i:x_i\in D}, \quad F_{\mathsf{Auth}}(\{x_i, m_i\}_{i\in[k]}) = k.$$

Finally, we define the notion of security against outsiders.

Definition 6 (Outsider-Security). *An* SPC *encryption scheme* (Gen, Enc, Dec) *satisfies security against outsiders with element-hiding if for any PPT adversary $\mathcal{A}$, any pair $(x_0, m_0), (x_1, m_1) \in \mathcal{U} \times \mathcal{M}$, and any $\lambda \in \mathbb{N}, D \subseteq \mathcal{U}$,*

$$\Pr\left[\mathcal{A}(\mathsf{pk}, \mathsf{ct}) = b : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, D) \\ b \leftarrow \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

The scheme satisfies security against an outsider without *element-hiding if the above only holds for any pair* $(x, m_0), (x, m_1)$.

3.3 Construction

We begin by giving templates for SPC encryption based on Apple's PSI protocol [9]. These schemes will have succinct encryption and succinct decryption, but *non-succinct* public-key. We will first describe a scheme $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}$ (Protocol 2) that satisfies ϵ-correctness and security against outsiders with element-hiding. Then, we describe a related scheme $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}$ (Protocol 3) that satisfies *perfect* ϵ-correctness but only security against outsiders *without* element-hiding, and is tailored to support encrypting messages that are group elements. This latter scheme will be useful for our construction of set pre-constrained group signatures in Section 4.

Following these basic templates, we will then show how to (efficiently) upgrade each to obtain bounded-set and authenticated-set security, resulting in schemes $\Pi_{\mathsf{SPCE}}^{\mathsf{BS-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{BS-PC}}, \Pi_{\mathsf{SPCE}}^{\mathsf{AS-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{AS-PC}}$.

Ingredients:

- A cyclic group $\mathbb{G}$ of prime order q in which the DDH problem is assumed to be hard.
- A symmetric-key encryption scheme (RobEnc, RobDec) with keyspace $\mathcal{K}$ that satisfies *random key robustness* (Section 2.1).
- Hash functions $H : \mathcal{U} \to \mathbb{G} \setminus \{0\}$ and $G : \mathbb{G} \to \mathcal{K}$ modeled as random oracles, where G maps the uniform distribution over $\mathbb{G}$ to (negligibly close to) the uniform distribution over $\mathcal{K}$.
- A cuckoo hashing scheme (CH.Setup, CH.Hash) (Section 2.4).

Theorem 1 (Proof in Appendix A). $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ *satisfies* ϵ-correctness *and* security against outsiders with element-hiding, *and* $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ *satisfies* perfect ϵ-correctness *and* security against outsiders without element-hiding.

Achieving bounded-set security. It can in fact be shown that $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ (resp. $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$) already securely emulates $\mathcal{F}_{\mathsf{SPCE}}^{\mathcal{P}[\mathsf{BS\text{-}EH}]}$ (resp. $\mathcal{F}_{\mathsf{SPCE}}^{\mathcal{P}[\mathsf{BS}]}$) with set size $\mathsf{pp} = n'$ where n' is such that $(n', \cdot, \cdot) \leftarrow$ CH.Setup(λ, n, ϵ). However, n' may be much larger than n, which means a large gap between correctness

$\Pi^{\text{Basic-EH}}_{\text{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$

Parameters: universe $\mathcal{U}$, message space $\mathcal{M}$, set size n, correctness parameter ϵ, and security parameter λ.
Setup: description of the group $\mathbb{G}$ with generator g, and random oracles H, G.
$\mathsf{Gen}(1^\lambda, D)$:

- Run $(n', h_0, h_1) \leftarrow \mathsf{CH.Setup}(\lambda, n, \epsilon)$ and then $T := \mathsf{CH.Hash}(h_0, h_1, D)$.
- Sample $\alpha \leftarrow \mathbb{Z}_q$ and set $A := g^\alpha$.
- Define $\widetilde{T}$ as follows. For each $i \in [n']$, if $T_i = \bot$ then sample $r_i \leftarrow \mathbb{Z}_q$ and set $\widetilde{T}_i := g^{r_i}$, and otherwise set $\widetilde{T}_i := H(T_i)^\alpha$.
- Output $\mathsf{pk} := (h_0, h_1, A, \widetilde{T})$ and $\mathsf{sk} := \alpha$.

$\mathsf{Enc}(\mathsf{pk}, x, m)$:

- Parse pk as $(h_0, h_1, A, \widetilde{T})$ and abort if there are any duplicate entries in $\widetilde{T}$.
- For $b \in \{0,1\}$, sample $\beta_b, \gamma_b \leftarrow \mathbb{Z}_q$, and compute $Q_b := g^{\beta_b} \cdot H(x)^{\gamma_b}, S_b := A^{\beta_b} \cdot \widetilde{T}^{\gamma_b}_{h_b(x)}, \mathsf{ct}_b := \mathsf{RobEnc}(G(S_b), m)$. Sample $b \leftarrow \{0,1\}$ and output $\mathsf{ct} := (Q_b, \mathsf{ct}_b, Q_{1-b}, \mathsf{ct}_{1-b})$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:

- Parse sk as α and ct as $(Q_0, \mathsf{ct}_0, Q_1, \mathsf{ct}_1)$.
- For $b \in \{0,1\}$ and compute $m_b := \mathsf{RobDec}(G(Q_b^\alpha), \mathsf{ct}_b)$. If exactly one of m_0 or m_1 is not $\bot$, then output this message, and otherwise output $\bot$.

Figure 2: Basic SPC encryption with element-hiding

(an honest server would be able to decrypt with respect to $(1-\epsilon)n$ items) and security (a dishonest server would potentially be able to decrypt with respect to up to n' items).

Below, we show that a simple and efficient tweak to the basic schemes results in schemes $\Pi^{\text{BS-EH}}_{\text{SPCE}}, \Pi^{\text{BS-PC}}_{\text{SPCE}}$ (Protocol 4) that completely close this gap. That is, for any n, the schemes $\Pi^{\text{BS-EH}}_{\text{SPCE}}, \Pi^{\text{BS-PC}}_{\text{SPCE}}$ securely emulate $\mathcal{F}^{\mathcal{P}[\text{BS-EH}]}_{\text{SPCE}}, \mathcal{F}^{\mathcal{P}[\text{BS}]}_{\text{SPCE}}$ with $\mathsf{pp} = n$.

Observe that the correctness properties of $\Pi^{\text{Basic-EH}}_{\text{SPCE}}, \Pi^{\text{Basic-PC}}_{\text{SPCE}}$ are preserved by the this transformation, due to the completeness of the ZK-NIAoKs, and the security against outsiders properties are also preserved, due to the zero-knowledge of the ZK-NIAoKs. We prove the following theorem in Appendix A.

Theorem 2 (Proof in Appendix A). *$\Pi^{\text{BS-EH}}_{\text{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ securely emulates (Definition 2) $\mathcal{F}^{\mathcal{P}[\text{BS-EH}]}_{\text{SPCE}}$ with $\mathsf{pp} = n$ in the presence of a malicious adversary corrupting either the server, the client, or* Auth, *and $\Pi^{\text{BS-PC}}_{\text{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ securely emulates $\mathcal{F}^{\mathcal{P}[\text{BS}]}_{\text{SPCE}}$ with $\mathsf{pp} = n$ in the presence of a malicious adversary corrupting either the server, the client, or* Auth.

Achieving authenticated-set security. Next, we describe schemes $\Pi^{\text{AS-EH}}_{\text{SPCE}}, \Pi^{\text{AS-PC}}_{\text{SPCE}}$ (Protocol 5) that satisfy authenticated set security. In order to achieve this notion, we will relax Gen to be an *interactive protocol* between the server and Auth, with the following syntax.

$$\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$$

Parameters: same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}$, except that the message space is the set of group elements in $\mathbb{G}$.
Setup: same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}$.
$\mathsf{Gen}(1^\lambda, D)$: same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}$, except that h_0, h_1, T are included in sk.
$\mathsf{Enc}(\mathsf{pk}, x, m)$:

- Parse pk as $(h_0, h_1, A, \widetilde{T})$ and abort if there are any duplicate entries in $\widetilde{T}$.
- For $b \in \{0,1\}$, sample $\beta_b, \gamma_b \leftarrow \mathbb{Z}_q$, and compute $Q_b := g^{\beta_b} \cdot H(x)^{\gamma_b}, S_b := A^{\beta_b} \cdot \widetilde{T}_{h_b(x)}^{\gamma_b}$. Output $\mathsf{ct} := (x, Q_0, S_0 \cdot m, Q_1, S_1 \cdot m)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$:

- Parse sk as (h_0, h_1, T, α) and ct as $(x, Q_0, S_0', Q_1, S_1')$.
- If there exists exactly one $b \in \{0,1\}$ such that $T_{h_i(x)} = x$, then output $m = S_b'/Q_b^\alpha$. Otherwise, output $\perp$.

Figure 3: Basic SPC encryption with perfect correctness

$$\Pi_{\mathsf{SPCE}}^{\mathsf{BS-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon], \Pi_{\mathsf{SPCE}}^{\mathsf{BS-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$$

Parameters: Same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}$. Note that the parameters λ, n, and ϵ determine a maximum hash table size n', where $(n', \cdot, \cdot) \leftarrow \mathsf{Setup}(\lambda, n, \epsilon)$.

Setup: Let $(\mathsf{Prove}_{\mathsf{DLog}}, \mathsf{Verify}_{\mathsf{DLog}})$ be a ZK-NIAoK for $\mathcal{R}_{\mathsf{DLog}}$ and let $(\mathsf{Prove}_{\widetilde{\mathsf{DLog}}}, \mathsf{Verify}_{\widetilde{\mathsf{DLog}}})$ be a ZK-NIAoK for $\mathcal{R}_{\mathsf{DLog}_{n'}^{n'-n}}$ (Section 2.3). Both of these proof systems are in the random oracle model and have no additional setup, so there is no additional setup required for $\Pi_{\mathsf{SPCE}}^{\mathsf{BS-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{BS-PC}}$.

$\mathsf{Gen}(1^\lambda, D)$: Same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}$, except that proofs $\pi_A \leftarrow \mathsf{Prove}_{\mathsf{DLog}}((g, A), \alpha)$ and $\pi_{\widetilde{T}} \leftarrow \mathsf{Prove}_{\widetilde{\mathsf{DLog}}}((g, \widetilde{T}), \{r_i\}_{i:T_i=\perp})$ are computed and appended to the public key pk.

$\mathsf{Enc}(\mathsf{pk}, x, m)$: Same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}$, except that the algorithm aborts if either of π_A or $\pi_{\widetilde{T}}$ fails to verify, or the number of group elements in $\widetilde{T}$ is greater than n'.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: same as $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}$.

Figure 4: SPC encryption with bounded-set security

- $\mathsf{Gen}\langle\mathsf{Server}, \mathsf{Auth}(D)\rangle(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: the parameter generation protocol takes place between a server and Auth with input a set $D \subseteq \mathcal{U}$, and outputs to the server a public key pk and a secret key sk.

Observe that the correctness properties of $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}, \Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}$ are preserved by the transformation, due

$$\Pi^{\text{AS-EH}}_{\text{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon], \Pi^{\text{AS-PC}}_{\text{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$$

Parameters: same as $\Pi^{\text{Basic-EH}}_{\text{SPCE}}, \Pi^{\text{Basic-PC}}_{\text{SPCE}}$.

Setup: let $(\text{Sig.Gen}, \text{Sig.Sign}, \text{Sign.Verify})$ be a EUF-CMA secure signature scheme (Section 2.1). Before the protocol begins, Auth will sample $(\text{vk}_{\text{Auth}}, \text{sk}_{\text{Auth}}) \leftarrow \text{Sig.Gen}(1^\lambda)$ and broadcast vk_{Auth} to all parties. Also, let $(\text{Prove}_{\text{DLog}}, \text{Verify}_{\text{DLog}})$ be a ZK-NIAoK for $\mathcal{R}_{\text{DLog}}$ and $(\text{Prove}_{\text{DH}}, \text{Verify}_{\text{DH}})$ be a ZK-NIAoK for $\mathcal{R}_{\text{DH}}$ (Section 2.3). Both of these proof systems are in the random oracle model, so require no additional setup beyond $\Pi^{\text{Basic-EH}}_{\text{SPCE}}, \Pi^{\text{Basic-PC}}_{\text{SPCE}}$.

$\text{Gen}\langle\text{Server}, \text{Auth}(D)\rangle(1^\lambda)$:

- Auth sends D to Server.
- Server first runs the Gen algorithm of $\Pi^{\text{Basic-EH}}_{\text{SPCE}}, \Pi^{\text{Basic-PC}}_{\text{SPCE}}$ on input $(1^\lambda, D)$ to obtain output $(h_0, h_1, A, \widetilde{T}), \alpha$, along with table T and randomness $\{r_i\}_{i:T_i=\perp}$. Next, compute $\pi_A \leftarrow \text{Prove}_{\text{DLog}}((g, A), \alpha)$. Finally, for each $i \in [n']$, where n' is the size of $T, \widetilde{T}$:
  - If $T_i = \perp$, compute $\pi_i \leftarrow \text{Prove}_{\text{DLog}}((g, \widetilde{T}_i), r_i)$.
  - If $T_i \neq \perp$, compute $\pi_i \leftarrow \text{Prove}_{\text{DH}}((g, A, H(T_i), \widetilde{T}_i), \alpha)$.

  Send $(A, T, \widetilde{T}, \pi_A, \{\pi_i\}_{i\in[n']})$ to Auth.
- Auth runs $\text{Verify}_{\text{DLog}}((g, A), \pi_A)$ and for each $i \in [n']$: if $T_i = \perp$, runs $\text{Verify}_{\text{DLog}}((g, \widetilde{T}_i), \pi_i)$ and if $T_i \neq \perp$, check that $T_i \in D$ and runs $\text{Verify}_{\text{DH}}((g, A, H(T_i), \widetilde{T}_i), \pi_i)$. If all checks pass, compute $\sigma \leftarrow \text{Sig.Sign}(\text{sk}, (A, \widetilde{T}))$, and return σ.
- Server outputs $\text{pk} := (h_0, h_1, A, \widetilde{T}, \sigma)$ and $\text{sk} := \alpha$.

$\text{Enc}(\text{pk}, x, m)$: same as $\Pi^{\text{Basic-EH}}_{\text{SPCE}}, \Pi^{\text{Basic-PC}}_{\text{SPCE}}$, except that it first runs $\text{Sig.Verify}(\text{vk}_{\text{Auth}}, (A, \widetilde{T}), \sigma)$[a] and aborts if the signature fails to verify. $\text{Dec}(\text{sk}, \text{ct})$: same as $\Pi^{\text{Basic-EH}}_{\text{SPCE}}, \Pi^{\text{Basic-PC}}_{\text{SPCE}}$.

[a]Note that this verification only needs to be done once per user and not every time Enc is run, since the public key does not change.

Figure 5: SPC encryption with authenticated-set security

to the completeness of the ZK-NIAoKs and correctness of Sig, and the security against outsiders properties are also preserved, due to the zero-knowledge of the ZK-NIAoKs.

Theorem 3 (Proof in Appendix A). $\Pi^{\text{AS-EH}}_{\text{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ *securely emulates (Definition 2)* $\mathcal{F}^{\mathcal{P}[\text{AS-EH}]}_{\text{SPCE}}$ *in the presence of a malicious adversary corrupting either the server, the client, or* Auth, *and* $\Pi^{\text{AS-PC}}_{\text{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ *securely emulates* $\mathcal{F}^{\mathcal{P}[\text{AS}]}_{\text{SPCE}}$ *in the presence of a malicious adversary corrupting either the server, the client, or* Auth.

3.4 SPC encryption with a succinct public-key

SPC encryption from laconic labeled PSI. Alamati et al. [2] construct laconic private set intersection which allows detection of encrypted messages belonging to a set of illegal images. We believe their approach can be extended to construct succinct public-key, succinct encryption, non-succinct decryption, element-hiding SPC encryption. Furthermore, their approach could also be tweaked to obtain succinct public-key, succinct encryption, succinct decryption, non-element-hiding SPC encryption. We do not explore this direction as the approaches are not concretely efficient and unsuitable for real-world deployment.

Optimal SPC encryption implies DEPIR. Recall that our constructions of SPC encryption satisfy *succinct encryption* and *succinct decryption*, but have a *non-succinct public-key*. If in addition, we demand a succinct public key (while maintaining the element-hiding property), the resulting SPC encryption scheme can used to realize doubly-efficient private information retrieval with preprocessing (Section 2.6). Given a database $D \in \{0,1\}^n$, define $\widetilde{D} \coloneqq \{D[i]||i\}_{i\in[n]}$.

- PIR.Gen on input D executes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SPC.Gen}(1^\lambda, \widetilde{D})$ and outputs $k_c = \mathsf{pk}$ and $k_s = \mathsf{sk}$.
- PIR.Query samples a random bit $\mathsf{st} = b$ as the local state and outputs $q = \mathsf{SPC.Enc}(\mathsf{pk}, i||b, 0_{\mathcal{M}})$.
- PIR.Resp computes $\mathsf{SPC.Dec}(\mathsf{sk}, q)$ and outputs $a = 0$ if it obtains $0_{\mathcal{M}}$ and otherwise $a = 1$ if it obtains $\perp$.
- PIR.Decode outputs $b \oplus a$.

To see that this scheme is secure against semi-honest servers, note that the view of the server consists of SPC ciphertexts, which only reveal whether or not the corresponding item exists in the database. Furthermore, the query is prepared by choosing a random bit b as the item, which means that the output of SPC.Dec is uniformly random and independent of the row being queried. Thus the server does not learn any information about the client's query.

4 SPC Group Signatures

In this section, we define and construct SPC group signatures. We present formal definition of SPC group signatures in Section 4.1, and constructions in Section 4.2 and Section 4.3.

4.1 Definitions

A set pre-constrained group signature (SPCGS) scheme $\Pi_{\mathsf{SPCGS}}[\mathcal{M}, \mathcal{P}, n, \epsilon]$ consists of algorithms Gen, Sign, Verify, Open, along with an interactive protocol KeyGen. 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. It is parameterized by a message space $\mathcal{M}$, an identity (or public key) space $\mathcal{P}$, a set size n, and a correctness parameter ϵ.

- $\mathsf{Gen}(1^\lambda, D) \to (\mathsf{mpk}, \mathsf{msk})$. The parameter generation algorithm takes as input a security parameter 1^λ and a set $D \subseteq \mathcal{M}$ of size at most n, 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 $\mathsf{pk} \in \mathcal{P}$ 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 $m \in \mathcal{M}$, and outputs a signature σ.

- $\mathsf{Verify}(\mathsf{mpk}, m, \sigma) \to \{\top, \bot\}$. The verification algorithm takes as input the master public key mpk, a message $m \in \mathcal{M}$, and a signature σ, and outputs either $\top$ or $\bot$, indicating accept or reject.

- $\mathsf{Open}(\mathsf{msk}, \sigma) \to \{\mathsf{pk}, \bot\}$. The opening algorithm takes as input the master secret key msk and a signature σ, and outputs either an identity $\mathsf{pk} \in \mathcal{P}$ 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.

We now port the definitions of bounded-set and authenticated-set security against malicious servers (as previously defined for SPC encryption) to the group signature setting. Further, we follow standard definitions of *traceability*, *anonymity*, and *unframeability* for group signatures.

Definition 7 (Correctness). *An* SPC *group signature scheme* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open})$ *is correct if for any $\lambda \in \mathbb{N}, D \subseteq \mathcal{M}$, and message $m \in \mathcal{M}$, it holds with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, D), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$*, and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$ *that* $\mathsf{Verify}(\mathsf{mpk}, m, \sigma) = 1$.

Security. We formulate several notions of security for an SPC group signature scheme. First, 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.

Definition 8 (Traceability). *An* SPC *group signature scheme* $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open})$ *is ϵ-traceable for some $\epsilon \geq 0$ if for any PPT adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, and $D \subseteq \mathcal{M}$, it holds that with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, D)$*, there exists a $D' \subseteq D$ with $|D'| \geq (1 - \epsilon)|D|$, such that*

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

where the oracles $O_{\mathsf{AKG}}, O_{\mathsf{Open}}, O_{\mathsf{HKG}}, O_{\mathsf{Sign}}$, set $\mathcal{I}_{\mathsf{Adv}}$ and predicate $\mathsf{IsValid}[\mathsf{mpk}]$ *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 signature σ, outputs* $\mathsf{Open}(\mathsf{msk}, \sigma)$.

- 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* $\mathsf{pk} \notin \mathcal{I}_{\mathsf{Hon}}$*, return nothing, and otherwise let* sk *be the secret key associated with* pk *and return* $\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, m)$*. Define $\mathcal{J}$ to be the set of* (m, pk) *queried to O_{Sign}.*

- $\mathsf{IsValid}[\mathsf{mpk}](m, \sigma, \mathsf{pk})$ *outputs* $(\mathsf{Verify}(\mathsf{mpk}, m, \sigma) = 1) \ \wedge\ ((m, \mathsf{pk}) \notin \mathcal{J})$*. That is, it accepts if the adversary produced a valid message signature pair that was not a query to its signing oracle.*

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 9 (Unframeability). *An* SPC *group signature scheme* (Gen, KeyGen, Sign, Verify, Open) *satisfies unframeability if for any PPT adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, and $D \subseteq \mathcal{M}$,*

$$\Pr\left[\begin{array}{lcr} \mathsf{Verify}(\mathsf{mpk}, m, \sigma) = 1 & & (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, D) \\ \wedge\ ((m, \mathsf{pk}) \notin \mathcal{J}) & : & (m, \sigma) \leftarrow \mathcal{A}^{O_{\mathsf{HKG}}, O_{\mathsf{Sign}}}(\mathsf{msk}) \\ \wedge\ (\mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}) & & \mathsf{pk} \leftarrow \mathsf{Open}(\mathsf{msk}, \sigma) \end{array}\right] = \mathsf{negl}(\lambda),$$

where the oracles $O_{\mathsf{HKG}}, O_{\mathsf{Sign}}$ and sets $\mathcal{I}_{\mathsf{Hon}}, \mathcal{J}$ are defined as in Definition 8.

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 follow our simulation-based notion of security for SPC encryption. The ideal functionality $\mathcal{F}_{\mathsf{anon}}^{\mathcal{P}}$ takes place between a *group manager* GM who runs Gen, interacts in KeyGen, and runs Open, a *client*, who interacts in KeyGen and runs Sign, and an authority Auth, whose role will be described below. In full generality, the group manager's input is a function F, but in our applications, we will always parse F as a description of a database D of messages. The client's input is a sequence of identities and messages $(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_k, m_k)$. The client should learn nothing about D, Auth should learn nothing about the identities $\{\mathsf{pk}_i\}_{i \in [k]}$, and the group manager should learn nothing about the identities $\{\mathsf{pk}_i\}_{i : m_i \notin D}$, except perhaps how many "repeats" there are (if we don't require the property of *unlinkability*).

To make security against the server meaningful, we must place some restriction on D. We do this (in a modular way) by parameterizing the functionality with a predicate $\mathcal{P}$. This predicate may depend on some public parameters pp (known to both client and group manager) and some secret parameters sp (known only to the group manager). It is the job of Auth to set up these parameters.

Below, we describe the instantiations of $\mathcal{P}$ that we will consider in this work: one will define what we call *bounded-set security* and the other will define what we call *authenticated-set security.*

Bounded-set security. Here, we define two predicates $\mathcal{P}[\mathsf{BS}]$ and $\mathcal{P}[\mathsf{BS\text{-}link}]$, where link stands for linkability. For each, the public parameters pp are parsed as an integer n, there are no secret parameters sp, and F is parsed as a description of a database $D \subseteq \mathcal{M}$. The predicate then outputs 1 if and only if $|D| \leq n$. For $\mathcal{P}[\mathsf{BS}]$,

$$F_{\mathsf{GM}}(\{\mathsf{pk}_i, m_i\}_{i \in [k]}) = \{\mathsf{pk}_i\}_{i : m_i \in D}, \{m_i\}_{i \in [k]}, \quad F_{\mathsf{Auth}}(\{\mathsf{pk}_i, m_i\}_{i \in [k]}) = \{m_i\}_{i \in [k]},$$

and for $\mathcal{P}[\mathsf{BS\text{-}link}]$,

$$F_{\mathsf{GM}}(\{\mathsf{pk}_i, m_i\}_{i \in [k]}) = \{\mathsf{pk}_i\}_{i : m_i \in D}, \mathsf{Aux}(\{\mathsf{pk}_i\}_{i : m_i \notin D}), \{m_i\}_{i \in [k]},$$

where for any multiset S, $\mathsf{Aux}(S)$ consists of the number of distinct elements in S along with how many times each appears, and

$$F_{\mathsf{Auth}}(\{\mathsf{pk}_i, m_i\}_{i \in [k]}) = \{m_i\}_{i \in [k]}.$$

Authenticated-set security. Here, we define two predicates $\mathcal{P}[\mathsf{AS}]$ and $\mathcal{P}[\mathsf{AS\text{-}EH}]$. For each, the public parameters pp are parsed as an integer n, the secret parameters are parsed as a database $D^* \subseteq \mathcal{M}$ of size n, and F is parsed as a database $D \subseteq \mathcal{M}$. The predicate then outputs 1 if and only if $D \subseteq D^*$. For $\mathcal{P}[\mathsf{AS}]$,

$$F_{\mathsf{GM}}(\{\mathsf{pk}_i, m_i\}_{i \in [k]}) = \{\mathsf{pk}_i\}_{i : m_i \in D}, \{m_i\}_{i \in [k]}, \quad F_{\mathsf{Auth}}(\{\mathsf{pk}_i, m_i\}_{i \in [k]}) = \{m_i\}_{i \in [k]},$$

$\mathcal{F}_{\mathsf{anon}}^{\mathcal{P}}$

Public information.

- Parties: Group manager and client.
- Parameters: message space $\mathcal{M}$, identity space $\mathcal{I}$

The functionality.

- Obtain input $(\mathsf{pp}, \mathsf{sp})$ from Auth. Deliver pp to both group manager and client, and sp to group manager.
- Obtain input $F = (F_{\mathsf{GM}}, F_{\mathsf{Auth}})$ from group manager and deliver F to Auth. Abort and deliver $\perp$ to all parties if $\mathcal{P}(\mathsf{pp}, \mathsf{sp}, F) = 0$.
- Obtain input $(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_k, m_k)$ from client, where each $\mathsf{pk}_i \in \mathcal{I}$ and each $m_i \in \mathcal{M}$.
- Deliver $F_{\mathsf{GM}}(\{\mathsf{pk}_i, m_i\}_{i\in[k]})$ to group manager and $F_{\mathsf{Auth}}(\{\mathsf{pk}_i, m_i\}_{i\in[k]})$ to Auth.

Figure 6: Ideal functionality for SPC group signatures with anonymity. $\mathcal{P}$ is a predicate that takes as input some public parameters pp, and a pair of functions $F = (F_S, F_{\mathsf{Auth}})$, and outputs a bit.

and for $\mathcal{P}[\text{AS-link}]$,

$$F_{\mathsf{GM}}(\{\mathsf{pk}_i, m_i\}_{i\in[k]}) = \{\mathsf{pk}_i\}_{i:m_i\in D}, \mathsf{Aux}(\{\mathsf{pk}_i\}_{i:m_i\notin D}), \{m_i\}_{i\in[k]},$$

where for any multiset S, $\mathsf{Aux}(S)$ consists of the number of distinct elements in S along with how many times each appears, and

$$F_{\mathsf{Auth}}(\{\mathsf{pk}_i, m_i\}_{i\in[k]}) = \{m_i\}_{i\in[k]}.$$

Finally, we consider "client-client" anonymity and unlinkability, 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 separate (game-based) definitions of anonymity and unlinkability against adversarial clients.

Definition 10 (Anonymity). *An* SPC *group signature scheme* (Gen, KeyGen, Sign, Verify, Open) *satisfies client-client anonymity if for any PPT adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, $D \subseteq \mathcal{M}$, and $m \in \mathcal{M}$, it holds that with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, D)$, $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$, $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$,

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

where the oracles O_{AKG}, O_{HKG}, and O_{Sign} are defined as in Definition 8.

Definition 11 (Unlinkability). *An* SPC *group signature scheme* (Gen, KeyGen, Sign, Verify, Open) *satisfies client-client unlinkability if for any PPT adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, $D \subseteq \mathcal{M}$, and messages $m_0, m_1 \in \mathcal{M}$, it holds that with probability* $1 - \mathsf{negl}(\lambda)$ *over* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, D)$, $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}\langle \mathsf{GM}(\mathsf{msk}), \mathsf{C}\rangle$, $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow$

$\mathsf{KeyGen}\langle\mathsf{GM}(\mathsf{msk}),\mathsf{C}\rangle$,

$$\Pr\left[\mathcal{A}^{O_{\mathsf{AKG}},O_{\mathsf{HKG}},O_{\mathsf{Sign}}}\left(\begin{array}{c}\mathsf{mpk},\mathsf{pk}_0,\\ \mathsf{pk}_1,\sigma_0,\sigma_1\end{array}\right)=b\ :\ \begin{array}{r}\sigma_0\leftarrow\mathsf{Sign}(\mathsf{mpk},\mathsf{sk}_0,m_0)\\ b\leftarrow\{0,1\}\\ \sigma_1\leftarrow\mathsf{Sign}(\mathsf{mpk},\mathsf{sk}_b,m_1)\end{array}\right]\leq\frac{1}{2}+\mathsf{negl}(\lambda),$$

where the oracles O_{AKG}, O_{HKG}, and O_{Sign} are defined as in Definition 8.

4.2 Generic construction

We show how to construct an SPC group signature scheme generically from an SPC encryption scheme that satisfies certain properties, 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:

- An SPC encryption scheme $\Pi_{\mathsf{SPCE}}=(\mathsf{SPCE.Gen},\mathsf{SPCE.Enc},\mathsf{SPCE.Dec})$ that satisfies *perfect ϵ-correctness*, *security against outsiders*, and either *bounded-set* security or *authenticated-set* security (Section 3).
- A one-way relation $(\mathcal{R},\mathcal{R}.\mathsf{Gen},\mathcal{R}.\mathsf{Sample})$ (Section 2.1). Let $\mathcal{P}$ denote the set of instances.
- An EUF-CMA secure signature scheme $\mathsf{Sig}=(\mathsf{Sig.Gen},\mathsf{Sig.Sign},\mathsf{Sig.Verify})$ with message space $\mathcal{P}$ (Section 2.1).
- 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 2.3).
- A random oracle H.

Theorem 4 (Proof in Appendix A)**.** $\Pi_{\mathsf{SPCGS}}[\mathcal{M},\mathcal{I},n,\epsilon]$ *satisfies correctness, ϵ-traceability, unframeability, client-client anonymity, and client-client unlinkability. Moreover, if* Π_{SPCE} *satisfies bounded-set security, then* Π_{SPCGS} *securely emulates* $\mathcal{F}_{\mathsf{anon}}^{\mathcal{P}[\mathsf{BS}]}$*, and if* Π_{SPCE} *satisfies authenticated-set security, then* Π_{SPCGS} *securely emulates* $\mathcal{F}_{\mathsf{anon}}^{\mathcal{P}[\mathsf{AS}]}$.

4.3 An efficient instantiation

We now describe a concretely efficient instantiation of the above generic template, based on constructions of SPC encryption schemes from Section 3.

Note that we will need a concretely efficient instantiation of a zero-knowledge argument system that can be used to prove statements that involve verifying signatures and the correctness of SPC encryption. Our goal here is to avoid *non-black-box* use of the cryptography needed for signatures and SPC encryption. Thus, we use *bilinear maps*, and make use of the Groth-Sahai proof system [24], which can efficiently prove statements that involve certain operations in pairing groups.[10] We combine the GS proof system with the use of *structure-preserving signatures* [1], which support messages, verification keys, and signatures that consist solely of group elements. Details about the resulting scheme follow, and we also provide implementations and benchmarking, which we cover in Section 5.

[10]We remark that, although GS proofs only satisfy *partial* knowledge extraction (see Section 2.5), this is sufficient for our construction. Indeed, the signatures extracted in order to show ϵ-traceability and unframeability only consist of group elements, and the one-way relation witness extracted during the proof of unframeability is also a group element.

$\Pi_{\mathsf{SPCGS}}[\mathcal{M}, \mathcal{P}, n, \epsilon]$

Parameters: message space $\mathcal{M}$, identity space $\mathcal{P}$, set size n, correctness parameter ϵ, and security parameter λ.

Setup: $\mathsf{pp} \leftarrow \mathcal{R}.\mathsf{Gen}(1^\lambda)$ and a random oracle H.

$\mathsf{Gen}(1^\lambda, D)$: run $(\mathsf{pk}_{\mathsf{SPCE}}, \mathsf{sk}_{\mathsf{SPCE}}) \leftarrow \mathsf{SPCE.Gen}(1^\lambda, D)$[a] and $(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$. Set $\mathsf{mpk} := (\mathsf{pk}_{\mathsf{SPCE}}, \mathsf{vk}_{\mathsf{Sig}})$ and $\mathsf{msk} := (\mathsf{sk}_{\mathsf{SPCE}}, \mathsf{sk}_{\mathsf{Sig}})$.

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

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

- $\mathsf{ct} = \mathsf{SPCE.Enc}(\mathsf{pk}_{\mathsf{SPCE}}, m, \mathsf{pk}; r)$,
- $(\mathsf{pk}, w) = \mathcal{R}.\mathsf{Sample}(\mathsf{pp}; s)$,
- and $\mathsf{Sig.Verify}(\mathsf{vk}_{\mathsf{Sig}}, \mathsf{pk}, \sigma_{\mathsf{id}})$.

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

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

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

[a]If the SPC encryption scheme satisfies authenticated-set security, this will be an interactive procedure between GM and Auth.

Figure 7: Generic construction of SPC group signatures.

Formally, our construction will make use of a bilinear map $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{T}, e, g_1, g_2)$ where the SXDH assumption is assumed to hold, as described in Section 2.5. The group signature scheme $\Pi_{\mathsf{SPCGS}}[\mathcal{M}, \mathbb{G}_1, n, \epsilon]$ will have an arbitrary message space $\mathcal{M}$ and identities consisting of group elements in $\mathbb{G}_1$. The four ingredients are instantiated as follows.

- SPC encryption: Either the scheme $\Pi^{\mathsf{BS-PC}}_{\mathsf{SPCE}}[\mathcal{M}, \mathbb{G}_1, n, \epsilon]$ or $\Pi^{\mathsf{AS-PC}}_{\mathsf{SPCE}}[\mathcal{M}, \mathbb{G}_1, n, \epsilon]$ from Section 3.
- One-way relation: The Diffie-Hellman relation in $\mathbb{G}_1$, where $\mathcal{R}$ is the set of tuples $(g, g^\alpha, g^\beta, g^{\alpha\cdot\beta}) \in \mathbb{G}_1^4$. $\mathcal{R}.\mathsf{Gen}$ outputs $(g, g^\alpha) = (g, h)$, and $\mathcal{R}.\mathsf{Sample}$ chooses randomness β and outputs (g^β, h^β). This relation is one-way from the hardness of the computational Diffie-Hellman problem in $\mathbb{G}_1$.

- Signature scheme: The structure-preserving signature scheme (Section 2.5) from [1].
- ZK-NIAoK: The Groth-Sahai proof system (Section 2.5).

The full details of our concretely efficient SPCGS scheme can be found in Protocol 8. We can use either of the two SPC encryption schemes – $\Pi^{\mathsf{BS-PC}}_{\mathsf{SPCE}}[\mathcal{M}, \mathbb{G}_1, n, \epsilon]$ or $\Pi^{\mathsf{AS-PC}}_{\mathsf{SPCE}}[\mathcal{M}, \mathbb{G}_1, n, \epsilon]$. For ease of exposition, we suppress mention of the generation and verification of the proofs in $\mathsf{pk}_{\mathsf{SPCE}}$ that are used to obtain bounded-set and authenticated-set security.

In order to write out all of the constraints that will be proved, we will need to describe details of the structure-preserving signature. For a formal description see [1]. The public parameters are three group elements $(F, K, T) \in \mathbb{G}_1^3$ and the verification key is a group element $Y \in \mathbb{G}_2$. A valid signature is five group elements $(\sigma_A, \sigma_C, \sigma_R, \sigma_D, \sigma_S) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$ that satisfy the following equations,

- $e(\sigma_A, Y \cdot \sigma_D) = e(K \cdot M, g_2) \cdot (T, \sigma_S)$
- $e(\sigma_C, g_2) = e(F, \sigma_D)$
- $e(\sigma_R, g_2) = e(g_1, \sigma_S)$.

Importantly, observe that all three verification equations can be cast as pairing product equations (see Section 2.5).

The group signature consists of two parts – an SPCE ciphertext and a GS proof showing that the client encrypted its identity. Since a malicious client can encrypt arbitrary identities, we also require a proof that

1. The client knows a signature on its identity pk that verifies under the server's verification key.
2. The client knows sk and hpk such that $\mathsf{pk} = g^{\mathsf{sk}}$ and $\mathsf{hpk} = h^{\mathsf{sk}}$, where h is a random public element in $\mathbb{G}_1$.

Intuitively, the former prevents a malicious client from using a public key that has not been registered with the server. The latter prevents the server from forging messages on behalf of the client, provided the computational Diffie-Hellman assumption holds. The group signatures are verified by verifying the attached GS proofs and the signatures can be opened by the GM using the decryption algorithm of SPCE.

5 Implementation

We evaluate the performance of our SPC group signature scheme using a prototype implementation in C++ which can be found at https://github.com/guruvamsi-policharla/pc-sigs. Internally, our construction implements an SPC encryption scheme allowing us to also infer the performance of SPC encryption. Our finite field and group arithmetic is implemented using the mcl library for pairing-based cryptography [27]. We use the BN curve $p(z) = 36z^4 + 36z^3 + 24z^2 + 6z + 1$ over 3 different primes to highlight the trade-off between performance and the number of bits of security. For details on the BN curve, see [7]. The parameters used can be found in the README file of the mcl github repository [27], and [33] contains a discussion about the security levels offered by different parameter values.

Benchmarks. Our experiments were performed using single threaded execution on a machine with a 1.8 GHz Intel Core i7 Processor and 8 GB of RAM. As alluded to earlier, we focus on benchmarking the SPCGS scheme which internally uses an SPCE scheme along with Groth-Sahai proofs to handle malicious senders. For a comparison of the SPCGS scheme across different security parameters, see Table 1. We emphasize that in our SPCGS scheme the Sign, Verify and Open algorithms can be computed in *constant*

$\Pi_{\mathsf{SPCGS}}[\mathcal{M}, \mathbb{G}_1, n, \epsilon]$

Public information: Description of bilinear groups, hash functions $H : \mathcal{U} \rightarrow \mathbb{G}_1 \setminus \{0\}$ and $G : \mathbb{G}_1 \rightarrow \mathcal{K}$, crs $\leftarrow$ GS.Setup for GS proofs, and $h \leftarrow \mathbb{G}_1$.

Gen:

- GM runs $(\mathsf{pk}_{\mathsf{SPCE}}, \mathsf{sk}_{\mathsf{SPCE}}) \leftarrow \mathsf{SPCE.Gen}$ and publishes $\mathsf{pk}_{\mathsf{SPCE}} = (h_0, h_1, A, \widetilde{T})$.
- GM runs $(\mathsf{vk}_s = Y, \mathsf{sk}_s) \leftarrow \mathsf{Sig.Gen}$ and publishes vk_s.

KeyGen:

- Client obtains a signature from GM on its identity $\mathsf{pk} \in \mathbb{G}_1$ as $(\sigma_A, \sigma_C, \sigma_R, \sigma_S, \sigma_S) \leftarrow \mathsf{Sig.Sign}_{\mathsf{sk}_s}(\mathsf{pk})$.

Sign:

- Encrypt identity with respect to a message $m \in \mathcal{M}$ as $\mathsf{ct} := (m, Q_0, S_0 \cdot \mathsf{pk}, Q_1, S_1 \cdot \mathsf{pk}) = \mathsf{SPCE.Enc}(\mathsf{pk}_{\mathsf{SPCE}}, m, \mathsf{pk})$.
- Compute a ZK-NIAoK for the following constraints:
  - $\{\mathsf{pk}, \beta_0, \gamma_0 : (S_0 \cdot \mathsf{pk}) = \mathsf{pk} \cdot A^{\beta_0} \cdot \widetilde{T}_{h_0(m)}^{\gamma_0}\}$.
  - $\{\mathsf{pk}, \beta_1, \gamma_1 : (S_1 \cdot \mathsf{pk}) = \mathsf{pk} \cdot A^{\beta_1} \cdot \widetilde{T}_{h_1(m)}^{\gamma_1}\}$.
  - $\{\mathsf{pk}, \beta_0, \gamma_0 : Q_0 = g_1^{\beta_0} \cdot H(m)^{\gamma_0}\}$.
  - $\{\mathsf{pk}, \beta_1, \gamma_1 : Q_1 = g_1^{\beta_1} \cdot H(m)^{\gamma_1}\}$.
  - $\{\mathsf{pk}, \mathsf{hpk}, \mathsf{sk} : \mathsf{pk} = g_1^{\mathsf{sk}} \wedge \mathsf{hpk} = h^{\mathsf{sk}}\}$.
  - $\{\sigma_C, \sigma_D : e(\sigma_C, g_2) = e(F, \sigma_D)\}$.
  - $\{\sigma_A, \sigma_D, \sigma_S, \mathsf{pk} :$
    $e(\sigma_A, Y \cdot \sigma_D) = e(K \cdot \mathsf{pk}, g_2) \cdot e(T, \sigma_S)\}$.
  - $\{\sigma_R, \sigma_S : e(\sigma_R, g_2) = e(g_1, \sigma_S)\}$.
- Output the signature (ct, π), where π denotes the GS proofs.

Verify:

- If $\mathsf{GS.Verify}(\pi, \mathsf{crs})$ passes, accept.
- Else output $\perp$.

Open:

- Output $\mathsf{SPCE.Dec}(\mathsf{sk}_{\mathsf{SPCE}}, \mathsf{ct})$.

Figure 8: Efficient instantiation of SPC group signatures.

time, independent of database size. This is reflected in the benchmarks which remain invariant to changes in the database size.

Fig. 9 breaks down the time taken by various components of the SPCGS scheme on the BN254 curve. *Encrypt* ($\sim 200\mu s$), and *Open* ($\sim 80\mu s$), correspond to the Encryption, and Decryption times respectively of our SPCE scheme over the BN254 curve. As expected, these are very small in comparison to the overall time, confirming that a majority of time is spent in the creation (*Create* in Fig. 9) and verification (*Verify* in Fig. 9) of Groth-Sahai proofs.

Integration overhead. Any end-to-end encrypted messaging system can incorporate traceability of illegal content by having its users attach SPC group signatures on images they originate. All honest users that receive images check if the signature is valid using SPCGS.Verify. If the content is harmful, then *any* user who receives the message, even if forwarded, can report the image to the server who will then trace the user by executing SPCGS.Open. A report consists of the SPC group signature along with the image. All clients using the messaging service store the master public key locally on their device.

The algorithms SPCGS.Sign and SPCGS.Verify are called every time an image is sent and both of these can be executed in tens of milliseconds. In practice, these operations will likely be performed on weaker mobile devices. However, we note that both of these algorithms can be trivially parallelized. Since the proof construction and verification consists of independent equations for each constraint, with a total of 9 constraints, we expect close to an *order of magnitude* improvement in Sign and Verify on latest consumer-grade hardware supporting multi-threading. Even after allowing for a 5x slowdown on mobile platforms, our constructions remain practical as images are sent sporadically by the average user and the overhead introduced is imperceptible in such cases.

Note that, in the context of secure messaging, message delivery typically takes a few hundred milliseconds on cellular networks[11]. Therefore, the additional latency introduced in our system is $\approx 15\%$ (when using BN254). In terms of communication, our protocols incur an additional overhead of ≈ 3.5 kilobytes ($< 1\%$) for a typical image ≈ 400 KB when using BN254. If the database has 10 million entries when using BN254, the public key is ≈ 320 MB in size but only has to be downloaded once by clients at the beginning of the protocol. Although, the public-key size is large, it is identical to that of the PSI system proposed by Apple [9] and should be viewed through the lens of system updates.

Re-using proofs. One can also consider a weaker anonymity notion that allows any party looking at two SPC group signatures to identify whether they were sent by the same client (Protocol 6). In this setting, clients can reuse some commitments and proofs thereby achieving a 2-5x reduction in prover and verifier times. More specifically, only the first 4 constraints in Protocol 8 change every time a new message is sent. As a result, the proofs prepared for the remaining constraints can be reused. The anonymous SPC group signature uses 34 elements each of $\mathbb{G}_1$ and $\mathbb{G}_2$, whereas the anonymous linkable SPC group signature only uses 8 elements each of $\mathbb{G}_1$ and $\mathbb{G}_2$.

Curve	**Scheme**	**Sign** (ms)	**Verify** (ms)	**Open** (μs)
BN254	Anon	9.0 ± 0.3	35.5 ± 0.3	81 ± 2
	Anon-Link	1.69 ± 0.05	8.9 ± 0.1	46 ± 6
BN381	Anon	21.7 ± 0.2	94.2 ± 0.4	194 ± 3
	Anon-Link	4.0 ± 0.1	24.3 ± 0.1	104 ± 4
BN512	Anon	71.1 ± 0.5	300.2 ± 5.2	622 ± 18
	Anon-Link	13.1 ± 0.2	75.6 ± 0.3	323 ± 10

Table 1: Benchmarks of SPCGS algorithms implemented on different curves.

[11]We use the same latency as related work [32] where they assume a network latency of 80ms on an 8 Mbps connection. For reference, a typical image ≈ 400 KB in size takes 200-300ms to be delivered.

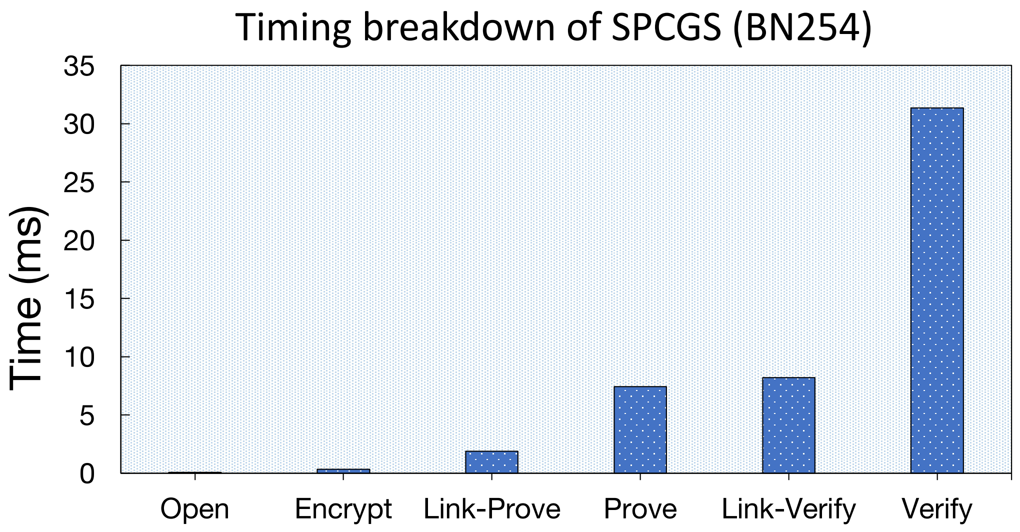

Figure 9: Breakdown of time taken in each component of the SPC group signature scheme on the BN254 curve. Prove and Verify take 3.9x and 3.8x longer than Link-Prove and Link-Verify respectively.

Acknowledgments

First, second and fourth authors were supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. The first and the fourth author were also supported by a grant from the CLTC. The third author was supported in part by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, AFOSR Award FA9550-19-1-0200, Office of Naval Research Grant N00014-19-1-2294, JP Morgan Faculty Award, and research gifts from Ethereum, Stellar and Cisco. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA.

References

[1] 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, Heidelberg, August 2010.

[2] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In *Theory of Cryptography Conference*, pages 94–125. Springer, 2021.

[3] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Pre-constrained encryption. In *Innovations in Theoretical Computer Science (ITCS)*, 2022.

[4] Apple. CSAM Detection Technical Summary, 2021.

[5] Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 65–91, Virtual Event, August 2021. Springer, Heidelberg.

[6] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[7] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International Workshop on Selected Areas in Cryptography*, pages 319–331. Springer, 2005.

[8] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

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

[10] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In *Theory of Cryptography Conference*, pages 662–693. Springer, 2017.

[11] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *Theory of Cryptography Conference*, pages 694–726. Springer, 2017.

[12] Ran Canetti and Gabriel Kaptchuk. The broken promise of apple's announced forbidden-photo reporting system – and how to fix it, 2021.

[13] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[14] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.

[15] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

[16] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Heidelberg, August 2018.

[17] Keita Emura, Goichiro Hanaoka, Yutaka Kawai, Takahiro Matsuda, Kazuma Ohara, Kazumasa Omote, Yusuke Sakai, and David Megias. Group signatures with message-dependent opening: Formal definitions and constructions. *Sec. and Commun. Netw.*, 2019, jan 2019.

[18] Federal Bureau of Investigation. Going Dark: Are Technology, Privacy, and Public Safety on a Collision Course? [speech], 2014.

[19] 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, Heidelberg, August 1987.

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

[21] Matthew Green. Thinking about "traceability", 2021.

[22] Matthew Green, Gabriel Kaptchuk, and Gijs Van Laer. Abuse resistant law enforcement access systems. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 553–583, Cham, 2021. Springer International Publishing.

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

[24] 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, Heidelberg, April 2008.

[25] Paul Grubbs, Arasu Arun, Ye Zhang, Joseph Bonneau, and Michael Walfish. Zero-knowledge middleboxes. In *USENIX Security Symposium*, pages 4255–4272. USENIX Association, 2022.

[26] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, August 2017.

[27] herumi. mcl. https://github.com/herumi/mcl, 2021.

[28] Rawane Issa, Nicolas Alhaddad, and Mayank Varia. Hecate: abuse reporting in secure messengers with sealed sender. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2335–2352, 2022.

[29] Seny Kamara, Mallory Knodel, Emma Llansó, Greg Nojeim, Lucy Qin, Dhanaraj Thakur, and Caitlin Vogus. Outside looking in. 2021.

[30] Anunay Kulshrestha and Jonathan Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[31] Benoît Libert and Marc Joye. Group signatures with message-dependent opening in the standard model. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 286–306. Springer, 2014.

[32] Linsheng Liu, Daniel S. Roche, Austin Theriault, and Arkady Yerukhimovich. Fighting fake news in encrypted messaging with the fuzzy anonymous complaint tally system (facts). Cryptology ePrint Archive, Report 2021/1148, 2021.

[33] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of nfs advances on the security of pairing-based cryptography. In *International Conference on Cryptology in Malaysia*, pages 83–108. Springer, 2016.

[34] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51:231–262, 2004.

[35] L.H. Newman. The EARN IT Act Is a Sneak Attack on Encryption. *Wired*, 2020.

[36] Charlotte Peale, Saba Eskandarian, and Dan Boneh. Secure complaint-enabled source-tracking for encrypted messaging. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1484–1506, 2021.

[37] Benny Pinkas. The private set intersection (psi) protocol of the apple csam detection system, 2021.

[38] Jonathan Prokos, Tushar M. Jois, Neil Fendley, Roei Schuster, Matthew Green, Eran Tromer, and Yinzhi Cao. Squint hard enough: Evaluating perceptual hashing with machine learning. Cryptology ePrint Archive, Report 2021/1531, 2021.

[39] Matthew Green. An evaluation of the risks of client-side scanning, 2022.

[40] U.S. Department of Justice. Attorney General Barr Signs Letter to Facebook From US, UK, and Australian Leaders Regarding Use of End-To-End Encryption, 2019.

[41] U.S. Department of Justice. International Statement: End-To-End Encryption and Public Safety, 2020.

[42] U.S. Senate Committee on the Judiciary. Graham, Blumenthal, Hawley, Feinstein Introduce EARN IT Act to Encourage Tech Industry to Take Online Child Sexual Exploitation Seriously, 2020.

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

[44] Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 222–250. Springer, Heidelberg, August 2019.

[45] Nirvan Tyagi, Ian Miers, and Thomas Ristenpart. Traceback for end-to-end encrypted messaging. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 413–430, 2019.

A Proofs

Theorem 1. $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ *satisfies* ϵ-correctness *and* security against outsiders with element-hiding, *and* $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ *satisfies* perfect ϵ-correctness *and* security against outsiders without element-hiding.

Proof. First, for correctness, note that the elements of $\widetilde{T}$ will be distinct with overwhelming probability, by the second correctness property of cuckoo hashing (Section 2.4), and the randomness of the oracle H. Now, ϵ-correctness of $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ follows from the correctness of cuckoo hashing (Section 2.4), the two properties of the Diffie-Hellman self reduction described in Section 2.1, and the random key robustness of $(\mathsf{RobEnc}, \mathsf{RobDec})$, while perfect ϵ-correctness of $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ follows from the correctness of cuckoo hashing (Section 2.4) and the first property of the Diffie-Hellman self reduction described in Section 2.1.

Next, we show the security of $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ against outsiders with element-hiding. The security of $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-PC}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ against outsiders without element-hiding follows from essentially the same argument. Fix any $(x_0, m_0), (x_1, m_1)$, and consider the following distributions.

- $\mathcal{D}_0$: the distribution over $(\mathsf{pk}, \mathsf{ct}) = (h_0, h_1, A, \widetilde{T}, Q_0, \mathsf{ct}_0, Q_1, \mathsf{ct}_1)$ where ct is an encryption of m_0 with respect to item x_0.
- $\mathcal{D}_1$: same as $\mathcal{D}_0$ except that if $T_i = x_0$ for some i, then $\widetilde{T}_i = H(x_0)^\alpha$ is replaced with a uniformly random group element. $\mathcal{D}_0 \approx_c \mathcal{D}_1$ follows from a reduction to the DDH assumption. In the reduction, all $H(x)$ for $x \neq x_0$ are set to a random group element g^r, where r is sampled by the reduction. This enables the reduction to simulate the rest of pk without knowing α.

- $\mathcal{D}_2$: same as $\mathcal{D}_1$ except that if $T_i = x_1$ for some i, then $\widetilde{T}_i = H(x_1)^\alpha$ is replaced with a uniformly random group element. $\mathcal{D}_1 \approx_c \mathcal{D}_2$ follows from DDH via the same reduction as above.

- $\mathcal{D}_3$: same as $\mathcal{D}_2$ except that $\mathsf{ct}_0, \mathsf{ct}_1$ are sampled as encryptions of m_1 with respect to item x_1. $\mathcal{D}_2 \approx \mathcal{D}_3$ follows from property 2 of the Diffie-Hellman self reduction described in Section 2.1, along with semantic security of $(\mathsf{RobEnc}, \mathsf{RobDec})$.

- $\mathcal{D}_4$: same as $\mathcal{D}_3$ except that if $T_i = x_0$ for some i, then $\widetilde{T}_i = H(x_0)^\alpha$. $\mathcal{D}_3 \approx \mathcal{D}_4$ follows from DDH via the same reduction as above.

- $\mathcal{D}_5$: same as $\mathcal{D}_4$ except that if $T_i = x_1$ for some i, then $\widetilde{T}_i = H(x_1)^\alpha$. $\mathcal{D}_4 \approx \mathcal{D}_5$ follows from DDH via the same reduction as above. $\mathcal{D}_5$ is $(\mathsf{pk}, \mathsf{ct})$ where ct is an encryption of m_1 with respect to item x_1, completing the proof.

$\square$

Theorem 2. $\Pi^{\mathsf{BS-EH}}_{\mathsf{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ *securely emulates (Definition 2)* $\mathcal{F}^{\mathcal{P}[\mathsf{BS\text{-}EH}]}_{\mathsf{SPCE}}$ *with* $\mathsf{pp} = n$ *in the presence of a malicious adversary corrupting either the server, the client, or* Auth, *and* $\Pi^{\mathsf{BS-PC}}_{\mathsf{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ *securely emulates* $\mathcal{F}^{\mathcal{P}[\mathsf{BS}]}_{\mathsf{SPCE}}$ *with* $\mathsf{pp} = n$ *in the presence of a malicious adversary corrupting either the server, the client, or* Auth.

Proof. Below, we show that $\Pi^{\mathsf{BS-EH}}_{\mathsf{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ securely emulates $\mathcal{F}^{\mathsf{BS\text{-}EH}}_{\mathsf{SPCE}}$ with a set size bound of $\mathsf{pp} = n$. The corresponding claim for $\Pi^{\mathsf{BS-PC}}_{\mathsf{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ follows from a similar argument.

Security against malicious server. First, we describe the simulator for a malicious server $\mathcal{S}^*$.

- Observe $\mathcal{S}^*$'s queries to the random oracle H, and lazily sample the responses by choosing uniformly random group elements. Record the set of queries as $(y_1, H(y_1) = g^{s_1}, s_1), \ldots, (y_q, H(y_Q) = g^{s_q}, s_q)$, where $s_1, \ldots, s_q$ are uniformly random exponents chosen by the simulator.

- Obtain $\mathsf{pk} = (h_0, h_1, A, \widetilde{T}, \pi_A, \pi_{\widetilde{T}})$ from $\mathcal{S}^*$. Abort if either of π_A or $\pi_{\widetilde{T}}$ fails to verify, if the number of group elements in $\widetilde{T}$ is greater than n', or if $\widetilde{T}$ contains any duplicate group elements. Next, initialize D as the empty set. For each tuple (y_j, g^{s_j}, s_j), check if either $\widetilde{T}_{h_0(y_j)} = A^{s_j}$ or $\widetilde{T}_{h_1(y_j)} = A^{s_j}$. If so, add y_j to D. If at the end of this process $|D| > n$, then abort.

- Query D to the ideal functionality and receive $k, (m_1, \ldots, m_{k'})$, for some $k' \leq k$.

- We will define two ways of generating a pair of group elements (Q, S). $(Q, S) \leftarrow \mathcal{R}$ means that both Q and S are sampled as uniformly random and independent group elements, while $(Q, S) \leftarrow \mathcal{DH}$ means that Q and S are sampled uniformly at random conditioned on (g, A, Q, S) being a a DH tuple. That is, $Q = g^\beta$ and $S = A^\beta$ for a uniformly sampled exponent β.

- For each $i \in [k']$, sample $b_i \leftarrow \{0,1\}$, $(Q_{i,b}, S_{i,b}) \leftarrow \mathcal{DH}$, $(Q_{i,1-b}, S_{i,1-b}) \leftarrow \mathcal{R}$, and then $\mathsf{ct}_{i,b} \leftarrow \mathsf{RobEnc}(G(S_{i,b}), m_i)$, $\mathsf{ct}_{i,1-b} \leftarrow \mathsf{RobEnc}(G(S_{i,1-b}), 0)$, and set $\mathsf{ct}_i := (Q_{i,0}, \mathsf{ct}_{i,0}, Q_{i,1}, \mathsf{ct}_{i,1})$.

- For each $i \in [k'+1, \ldots, k]$, sample $(Q_{i,0}, S_{i,0}), (Q_{i,1}, S_{i,1}) \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,0} \leftarrow \mathsf{RobEnc}(G(S_{i,0}), 0)$, $\mathsf{ct}_{i,1} \leftarrow \mathsf{RobEnc}(G(S_{i,1}), 0)$, and set $\mathsf{ct}_i := (Q_{i,0}, \mathsf{ct}_{i,0}, Q_{i,1}, \mathsf{ct}_{i,1})$.

- Deliver $\{\mathsf{ct}_i\}_{i \in [k]}$ to $\mathcal{S}^*$, and output whatever $\mathcal{S}^*$ outputs.

Now, we argue directly that $\mathcal{S}^*$'s simulated view is computationally indistinguishable from its real view. We will argue this in two steps.

- Claim 1: the probability p_{abort} that the extracted D is such that $|D| > n$ (which would cause the simulator to abort) is negligible.
- Claim 2: the simulated ciphertexts are indistinguishable from the real ciphertexts conditioned on the simulator not aborting.

Proof of Claim 1: Assume that p_{abort} is noticeable. Note that p_{abort} will still be noticeable when running the extractors for the two ZK-NIAoKs. In this case, the extractor will obtain α such that $A = g^{\alpha}$ and an r such that there exists indices i, j such that $\widetilde{T} = g^{r} = A^{s_j} = g^{\alpha \cdot s_j}$, which implies that $r \cdot \alpha^{-1} = s_j \bmod q$. However, such an extractor (along with $\mathcal{S}^*$) can be used to break the hardness of the discrete logarithm problem, via a reduction that programs the challenge group element g^{s_j} as an output of the random oracle H.

Proof of Claim 2: Consider any item, message pair (x, m) encrypted by the client. Suppose first that $x \in D$, where D is the input extracted by the simulator. In this case, one of $(g, A, H(x), \widetilde{T}_{h_0(x)})$ or $(g, A, H(x), \widetilde{T}_{h_1(x)})$ is a DH tuple and the other is not, since $h_0(x) \neq h_1(x)$ and there are no duplicate entries in $\widetilde{T}$ (if Enc does not abort). Thus, by the two properties of the Diffie-Hellman self-reduction given in Section 2.1, and the semantic security of (RobEnc, RobDec), the simulator's encryption described above is computationally indistinguishable from the real encryption of (x, m). Next, suppose that $x \notin D$. If x was one of the queries obtained by the simulator prior to receiving pk, then we know that neither of $(g, A, H(x), \widetilde{T}_{h_0(x)})$ or $(g, A, H(x), \widetilde{T}_{h_1(x)})$ is a DH tuple. Otherwise, $H(x)$ is sampled after pk is fixed, meaning that with overwhelming probability, neither of $(g, A, H(x), \widetilde{T}_{h_0(x)})$ or $(g, A, H(x), \widetilde{T}_{h_1(x)})$ is a DH tuple. Thus, by the second property of the Diffie-Hellman self-reduction, and the semantic security of (RobEnc, RobDec), the simulator's encryption described above is computationally indistinguishable from the real encryption of (x, m).

Security against malicious client. The simulator for a malicious client samples $A, \widetilde{T}$ as $n' + 1 = (1 + \epsilon)n + 1$ uniformly random group elements, and runs the ZK-NIAoK simulators to obtain simulated proofs $\pi_A, \pi_{\widetilde{D}}$. To show that this is indistinguishable from the real pk produced from the set D, we first simulate the ZK-NIAoK proofs. Then, for each $x \in D$, if x is inserted into T as position $h_b(x)$, switch $\widetilde{T}_{h_b(x)}$ to a uniformly random group element. Each of these switches is computationally indistinguishable by reduction to the DDH assumption.

Security against a malicious Auth. This follows immediately from the security against outsiders with element-hiding (Definition 6) of $\Pi^{\mathsf{Basic-EH}}_{\mathsf{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ and the zero-knowledge of the ZK-NIAoK.

$\square$

Theorem 3. *$\Pi^{\mathsf{AS-EH}}_{\mathsf{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ securely emulates (Definition 2) $\mathcal{F}^{\mathcal{P}[\mathsf{AS\text{-}EH}]}_{\mathsf{SPCE}}$ in the presence of a malicious adversary corrupting either the server, the client, or* Auth, *and $\Pi^{\mathsf{AS-PC}}_{\mathsf{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ securely emulates $\mathcal{F}^{\mathcal{P}[\mathsf{AS}]}_{\mathsf{SPCE}}$ in the presence of a malicious adversary corrupting either the server, the client, or* Auth.

Proof. Below, we show that $\Pi^{\mathsf{AS-EH}}_{\mathsf{SPCE}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ securely emulates $\mathcal{F}^{\mathsf{AS-EH}}_{\mathsf{SPCE}}$. The corresponding claim for $\Pi^{\mathsf{AS-PC}}_{\mathsf{SPCE}}[\mathcal{U}, \mathbb{G}, n, \epsilon]$ follows from a similar argument.

Security against malicious server. First, we describe the simulator for a malicious server $\mathcal{S}^*$.

- Sample $(\mathsf{vk}_{\mathsf{Auth}}, \mathsf{sk}_{\mathsf{Auth}}) \leftarrow \mathsf{Sig.Gen}(1^{\lambda})$
- Obtain input $(\mathsf{pp}, \mathsf{sp}) = (n, D^*)$ from the ideal functionality.
- Run $\mathcal{S}^*$ on input D^* until it outputs a message $(A, T, \widetilde{T}, \pi_A, \{\pi_i\}_{i \in [n']})$.

- Run the honest Auth algorithm and abort if Auth aborts. Otherwise, let σ be the output of Auth.
- Run $\mathcal{S}^*$ on input σ until it outputs a public key $(h_0, h_1, A, \widetilde{T}, \sigma)$. Abort if there are any duplicate entries in $\widetilde{T}$ or if any σ fails to verify. Let D be the set of elements x such that there exist (i, b) such that $T_i = x$ and $h_b(x) = i$, and query D to the ideal functionality. If $D \not\subseteq D^*$ the ideal functionality will abort, and the experiment ends.
- Receive k, $(m_1, \ldots, m_{k'})$ for some $k' \leq k$.
- For each $i \in [k']$, sample $b_i \leftarrow \{0,1\}$, $(Q_{i,b}, S_{i,b}) \leftarrow \mathcal{DH}$, $(Q_{i,1-b}, S_{i,1-b}) \leftarrow \mathcal{R}$, where $\mathcal{R}, \mathcal{DH}$ are as defined in the proof of Theorem 2, and then $\mathsf{ct}_{i,b} \leftarrow \mathsf{RobEnc}(G(S_{i,b}), m_i)$, $\mathsf{ct}_{i,1-b} \leftarrow \mathsf{RobEnc}(G(S_{i,1-b}), 0)$, and set $\mathsf{ct}_i := (Q_{i,0}, \mathsf{ct}_{i,0}, Q_{i,1}, \mathsf{ct}_{i,1})$.
- For each $i \in [k'+1, \ldots, k]$, sample $(Q_{i,0}, S_{i,0}), (Q_{i,1}, S_{i,1}) \leftarrow \mathcal{R}$, $\mathsf{ct}_{i,0} \leftarrow \mathsf{RobEnc}(G(S_{i,0}), 0)$, $\mathsf{ct}_{i,1} \leftarrow \mathsf{RobEnc}(G(S_{i,1}), 0)$, and set $\mathsf{ct}_i := (Q_{i,0}, \mathsf{ct}_{i,0}, Q_{i,1}, \mathsf{ct}_{i,1})$.
- Deliver $\{\mathsf{ct}_i\}_{i \in [k]}$ to $\mathcal{S}^*$, and output whatever $\mathcal{S}^*$ outputs.

Now, we argue directly that $\mathcal{S}^*$'s simulated view is computationally indistinguishable from its real view. First, we can condition on the event E_{same} that the elements $(A, \widetilde{T})$ in the public key that $\mathcal{S}^*$ eventually outputs are equal to those in $\mathcal{S}^*$'s previous message to Auth. If the client does not abort, this event occurs with overwhelming probability due to the EUF-CMA security of Sig.

Now, consider any item, message pair (x, m) encrypted by the client. Suppose first that $x \in D$. In this case, one of $(g, A, H(x), \widetilde{T}_{h_0(x)})$ or $(g, A, H(x), \widetilde{T}_{h_1(x)})$ is a DH tuple and other is not. Indeed, we know that one of the tuples is a DH tuple by the conditioning on E_{same} and the soundness of the ZK-NIAoK proof system for $\mathcal{R}_{\mathsf{DH}}$, and the other is not since $h_0(x) \neq h_1(x)$ and there are no duplicate entries in $\widetilde{T}$. Thus, by the two properties of the Diffie-Hellman self-reduction given in Section 2.1, and the semantic security of $(\mathsf{RobEnc}, \mathsf{RobDec})$, the simulator's encryption described above is computationally indistinguishable from the real encryption of (x, m).

Next, suppose that $x \notin D$. In this case, it holds with overwhelming probability that neither of $(g, A, H(x), \widetilde{T}_{h_0(x)})$ or $(g, A, H(x), \widetilde{T}_{h_1(x)})$ is a DH tuple. Indeed, if $x = T_i$ for some i, where T was the table included in $\mathcal{S}^*$'s message to Auth, then we know that neither is a DH tuple since $h_0(x), h_1(x) \neq i$, and there are no duplicate entries in $\widetilde{T}$. Otherwise, say that x was not in the table T but $(g, A, H(x), \widetilde{T}_{h_b(x)})$ is a DH tuple for some $b \in \{0,1\}$. In this case, by running the extractors for the two ZK-NIAoKs, we can extract α such that $A = g^\alpha$ and r such that $\widetilde{T}_{h_b(x)} = g^r$ from $\mathcal{S}^*$. As in the proof of Theorem 2, these can be used to obtain the discrete logarithm of $H(x)$, breaking the hardness of the discrete logarithm problem. Thus, by the second property of the Diffie-Hellman self-reduction, and the semantic security of $(\mathsf{RobEnc}, \mathsf{RobDec})$, the simulator's encryption described above is computationally indistinguishable from the real encryption of (x, m).

Security against a malicious client. This follows from the same argument as that given in the proof of Theorem 2.

Security against a malicious Auth. This follows immediately from the security against outsiders with element-hiding (Definition 6) of $\Pi_{\mathsf{SPCE}}^{\mathsf{Basic-EH}}[\mathcal{U}, \mathcal{M}, n, \epsilon]$ and the zero-knowledge of the ZK-NIAoK.

$\square$

Theorem 4. $\Pi_{\mathsf{SPCGS}}[\mathcal{M}, \mathcal{I}, n, \epsilon]$ *satisfies correctness, ϵ-traceability, unframeability, client-client anonymity, and client-client unlinkability. Moreover, if Π_{SPCE} satisfies bounded-set security, then Π_{SPCGS} securely emulates $\mathcal{F}_{\mathsf{anon}}^{\mathcal{P}[\mathsf{BS}]}$, and if Π_{SPCE} satisfies authenticated-set security, then Π_{SPCGS} securely emulates $\mathcal{F}_{\mathsf{anon}}^{\mathcal{P}[\mathsf{AS}]}$.*

Proof. First, it is easy to see that the bounded-set and authenticated-set anonymity properties follow directly from the corresponding security properties of SPCE, plus zero-knowledge of the ZK-NIAoK. Next, we show each additional property separately.

Traceability. Consider a hybrid experiment where the random oracle H is lazily sampled as follows. The oracle $\mathcal{O}_{\mathsf{Sign}}$ will respond with a signature $(\mathsf{ct}, \mathsf{crs}, \pi)$ that is computed by running the zero-knowledge simulator on instance $(\mathsf{pp}, \mathsf{pk}_{\mathsf{SPCE}}, \mathsf{vk}_{\mathsf{Sig}}, m, \mathsf{ct})$ 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 $\mathcal{O}_{\mathsf{Sign}}$ is a fresh query. However, if a ct produced by $\mathcal{O}_{\mathsf{Sign}}$ has been previously queried by the adversary, then this would contradict the semantic security (security against an outsider) of SPCE, since it would imply the ability to predict a ciphertext output by the encryption 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 $(\mathsf{pp}, \mathsf{pk}^*)$ from the one-way relation challenger, picks a uniformly random $\mathcal{O}_{\mathsf{HKG}}$ query to answer with pk^*, and then if the adversary outputs a winning $(m, \sigma) := (m, (\mathsf{ct}, \mathsf{crs}, \pi))$ according to Definition 8, runs the knowledge extractor on $(\tau, (\mathsf{pp}, \mathsf{pk}_{\mathsf{SPCE}}, \mathsf{vk}_{\mathsf{Sig}}, m, \mathsf{ct}), \pi)$, where τ is the trapdoor recorded along with crs, to produce a witness $(\mathsf{pk}, s, w, \sigma_{\mathsf{id}}, r)$, and returns w to the challenger. We claim that, conditioned on $(m, \sigma, \mathsf{pk} := \mathsf{Open}(\mathsf{msk}, \sigma))$ satisfying the conditions of Definition 8, it holds that (i) $(\mathsf{pp}, \mathsf{pk}, w) \in \mathcal{R}$ 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 $\mathcal{O}_{\mathsf{Sign}}$ oracle. Indeed, note that $\mathsf{ct} \notin \mathsf{SPCE.Enc}(\mathsf{pk}_{\mathsf{SPCE}}, m, \mathsf{pk}')$ for any $\mathsf{pk}' \neq \mathsf{pk}$, by the perfect ϵ-correctness of SPCE, and $\mathsf{pk} = \mathsf{SPCE.Dec}(\mathsf{sk}_{\mathsf{SPCE}}, \mathsf{ct})$. But we know that any query (m', pk') to $\mathcal{O}_{\mathsf{Sign}}$ must be such that $(m', \mathsf{pk}') \neq (m, \mathsf{pk})$, since $(m, \mathsf{pk}) \notin \mathcal{J}$. This establishes point (i). Next, we can claim that $\mathsf{pk} \in \mathcal{I}_{\mathsf{Hon}}$. This follows because $\mathsf{pk} \notin \mathcal{I}_{\mathsf{Adv}}$, and by 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 $\mathcal{O}_{\mathsf{AKG}}$ or $\mathcal{O}_{\mathsf{HKG}}$.

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 $\mathcal{O}_{\mathsf{Sign}}$, 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 $(\mathsf{pp}, \mathsf{pk}^*)$, picks a uniformly random $\mathcal{O}_{\mathsf{HKG}}$ 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 $(m, \sigma, \mathsf{pk} := \mathsf{Open}(\mathsf{msk}, \sigma))$ satisfying the conditions of Definition 9, it holds that $(\mathsf{pp}, \mathsf{pk}, w) \in \mathcal{R}$ 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 9. This first condition again holds due to the perfect ϵ-correctness of SPCE, which implies that (m, ct) could not have been queried to H by $\mathcal{O}_{\mathsf{Sign}}$, and the knowledge extraction of the ZK-NIAoK.

Client-client anonymity and unlinkability. 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 $(\mathsf{pp}, \mathsf{pk}_{\mathsf{SPCE}}, \mathsf{vk}_{\mathsf{Sig}}, m, \mathsf{ct})$ 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 SPCE. Moreover, in this hybrid, client-client anonymity (Definition 10) follows directly from the security against outsiders of SPCE, and client-client unlinkability (Definition 11) follows directly from the security against outsiders with element-hiding of SPCE.

$\square$