Predicate Aggregate Signatures and Applications

Tian Qiu and Qiang Tang

The University of Sydney, Sydney, Australia
tqiu4893@uni.sydney.edu.au
qiang.tang@sydney.edu.au

Abstract. Motivated by applications in anonymous reputation systems and blockchain governance, we initiate the study of predicate aggregate signatures (PAS), which is a new primitive that enables users to sign multiple messages, and these individual signatures can be aggregated by a combiner, preserving the anonymity of the signers. The resulting PAS discloses only a brief description of signers for each message and provides assurance that both the signers and their description satisfy the specified public predicate.
We formally define PAS and give a construction framework to yield a logarithmic size signature, and further reduce the verification time also to logarithmic. We also give several instantiations for several concrete predicates that may be of independent interest.
To showcase its power, we also demonstrate its applications to multiple settings including multi-signatures, aggregate signatures, threshold signatures, (threshold) ring signatures, attribute-based signatures, etc, and advance the state of the art in all of them.

1 Introduction

Anonymous reputation systems are widely used in many applications. For example, on online platforms, for Internet peers to jointly establish accumulated ratings on the merchants/service providers or certain products, so that users that are not familiar with them, can have some context to make a better choice. Since the main necessary information is the accumulated rating, ensuring anonymity plays a crucial role to allow users to participate in the reputation systems. More specifically, in YouTube, each user registers at YouTube, and then gives his rating on each content as an "I like it" (like +1) or not, then there will be an accumulated content score shown in the platform. The accumulated score not only serves as a succinct representation/description, but also *hides the identities* of the voters. To reduce the reliance on fully trusting the platform, other important requirements are that the accumulated score should be *publicly verifiable*, so that users may have stronger confidence that the score is not manipulated by the platform; furthermore, in the anonymous setting, one potential threat arises when a malicious platform attempts to manipulate the ratings by repeatedly counting one user's vote for many times. Therefore, *additional measures* must

be taken to assure the verifier that each voter's contribution to the accumulated score is limited to a single vote.[1]

Naturally, individual votes can be realized via digital signatures from legitimate users (e.g., registered identities). To obtain a succinct accumulated score k, say up-votes, we would like to aggregate the corresponding identities and signatures to be a short "proof". The proof needs to ensure that indeed there are at least k signatures on "I like it" from k *distinct* identities.

The one-user-one-vote requirement above can be seen as a special policy that was put on the identities of those signatures. In broader applications, there could be more complex voting policies that could be expressed as a predicate on the voter identities. For example, in blockchain governance (e.g., Decentralized autonomous organizations (DAOs)[4]), decisions could be made by the whole community whose accounts hold sufficient amount of tokens. The final decision needs to be attested with a short proof that the voting result is indeed following the governing policy, and the proof would be stored onchain. Voting processes in DAOs offer a remarkable degree of flexibility and customization. These processes can be tailored and programmed to accommodate a wide range of requirements and preferences. For example, quadratic voting [1,2] allows the voter have budgets of credits which are converted to counted votes according to their square root. Delegated voting [5] allows users to delegate their voting power to trusted individuals or entities. Property-based voting [3] differentiates signers based on the properties of their non-fungible tokens used in voting.

Introducing predicate aggregate signatures. Motivated by above applications and many other relevant ones, in this paper, we are studying a general problem for aggregating signatures and keys on multiple messages, while ensuring that the signers satisfy some public predicate without disclosing their identities. We call such a cryptographic primitive *predicate aggregate signatures*, PAS for short.

More specifically, let us consider a set of users denoted as $\mathcal{U} = \{u_i\}_{i\in[n]}$ and a collection of messages $\mathcal{M} = \{m_j\}_{j\in[k]}$ drawn from a predefined message space. Users choose the messages to sign. There is also a combiner, who aggregates the corresponding signatures and signer identities/public keys into one succinct certificate/proof/signature and shows a description Δ of signers (like the number) on each message. The signature also confirms the legitimacy of both the signers and signatures, ensuring that the signers and the description adhere to a particular public predicate P, i.e., $P(S_1, \ldots, S_k, \Delta) = 1$, where each $S_i \subseteq \mathcal{U}$ is a subset of users.

For example, in the anonymous reputation system, the rate-once policy requires that each signer can only sign once at most. It means there is no duplicate signer in each subset, and all subsets are disjoint, i.e., $S_i \cap S_j = \emptyset$ for any $i \neq j$. Another example is the onchain voting system with special policy. Besides show-

[1] There are also other types of rating systems, such as Uber/Airbnb, that are based on accumulation on each transaction, so each user may rate on the same service provider more than once. We only consider the common version as a motivational example for our primitive.

ing the number of voters, the property policy [3] requires that some of the voters have special properties. By representing the property via index, the combiner can assure the policy is satisfied by proving that some voters' indices belong to a specific range, e.g., there is at least one voter in the subset who is a senior member with an index smaller than 50.

Inefficiencies of existing primitives. Despite that there are many relevant research on signature aggregation, anonymous authentication, and others, none of them gives a PAS in a satisfying way (as shown in Table 1). We first give a simple categorization of existing relevant primitives, and briefly describe insufficiency of each type, and defer a more detailed comparison to the full version. Besides that most of the primitives do not support a general policy validation on the signers, each of them lack some other critical properties. Jumping ahead, we will show that some of concrete instantiations of our PAS directly advance the state of the art of several of those well-studied primitives, see Table 2.

Table 1. Comparisons of relevant primitives.

Primitives	Trans. setup	Flexi. thld.	Agg across msgs[1]	Anony.	Signer Policy
Thld Sig. [37]	×	×	×	✓	×
Multi-Sig [13]	✓	✓	×	×	×
Agg-Sig. [14]	✓	✓	✓	×	×
Graded Sig. [28,10]	✓	✓	×	✓	×
Compact Cert. [33]	✓	✓	×	×	×
Thld-ring Sig. [18]	✓	×	×	✓	×
Attri-based Sig. [32]	×	×	×	✓	✓[2]
Our PAS.	✓	✓	✓	✓	✓

[1] Agg across msgs means signatures can be compressed among different messages.
[2] The predicate in this setting is applied to one single user's attribute set, while we consider predicate across multiple users.

Signature aggregations. Multi-signatures [13], aggregate signatures [14], threshold signatures [37] and several other relevant ones allow one to compress signatures from different users. Besides they usually have no anonymity guarantee, the former two have to explicitly provide the signer identities/public keys thus the total proof size and verification cost still remains at least linear to the threshold (which is usually linear to the total number of users); threshold signature, on the other hand, can have one single public key for verification, but via a trusted setup, when its threshold is fixed, and it does not support signature aggregation across multiple messages. Multi-key homomorphic signatures [29] evaluates the messages signed by different users but it does not protect the privacy of signers. All signers' identities are public which is not suitable for our anonymous setting.

Anonymous primitives. Anonymity oriented signatures such as ring signatures [35,17] and the linkable [31] and threshold versions [18], usually do not require

the identities of the ring being aggregated, and often with a fixed threshold. Scored anonymous credential [21] is used for privacy-preserving reputation enforcement. The user's reputation is decided by some service provider. While we are considering the reputation voting setting where a set of users rate a product by signing.

Attribute-based signatures (ABS) [32] allow a user to attest that his attributes satisfy certain predicate. Anonymity can be implicitly ensured if two users have the same attribute set. However, ABS requires a trusted key generation center, it does not consider signature aggregations, or policies across multiple users. In our context, each user independently generates their own keys, and our goal is to have the flexibility to aggregate signatures and apply predicates across multiple users.

Generic constructions. Generic zk-SNARKs could certainly provide a path for feasibility. By collecting numerous signatures from signers, the combiner can create a zk-SNARK proof that guarantees the existence of sufficient valid partial signatures satisfying the public predicate, while concealing the signers' identities and revealing only the counts. However, the generation of zk-SNARK proofs remains prohibitively expensive, and it relies on trusted setups and unfalsifiable assumptions. Some recent efforts have focused on constructing *dynamic threshold* signatures[2] [27,22] directly in the AGM model [26], whose actual security is not well-understood, and may have subtle vulnerabilities [38]. While we focus on building the PAS on classical and more standard assumptions.

1.1 Our Contributions

In this article, we formulate, construct and analyze the new primitive of predicate aggregate signatures to address remaining issues.

Formulating PAS. We give a formal definition and security models for predicate aggregate signatures.

As mentioned above, it allows registered users (public keys, identities known and made public) to sign on multiple messages and these signatures can be aggregated by a combiner who hides these signers' identities. The final signature only reveals a description of signers and guarantees that the signers and this description satisfy the public predicate. [3]

We formally define the security model of predicate aggregate signatures. It enjoys the following features simultaneously which advances existing primitives.

[2] They are a kind of special threshold signature that supports the *dynamic choice* of thresholds for each time of signature generation.

[3] In later, we would use the *dynamic threshold* as an example of the description. It reveals the number of users who have signed on the message. We choose it as the example for three reasons: (1) For a simpler presentation that shows how we can get our final construction step by step; (2) the dynamic threshold is a natural feature of our motivated anonymous reputation system; (3) the dynamic threshold aggregate signature itself might be of independent interests, and indeed it already advances the state of the art of several relevant signatures.

- *Transparent setup*: users generate pk, sk on their own, and a setup algorithm only publishes public parameters for the system.
- *Signer anonymity*: the adversary (not the combiner) cannot get any information about each individual signer identity/public key (e.g., whether he signed on a particular message) except the public description from the final signature, even if the adversary corrupts *all* users, including the target himself.
- *Unforgeability*: the adversary cannot convince the verifier, if it does not collect enough signatures, or the predicate is not satisfied. To facilitate such a notion, we generalize the classical proof of knowledge, and define a signer identity extractor.

Efficient constructions from standard assumptions. We proceed in several steps towards the full construction, with concrete efficient instantiations. Our starting point is the BLS aggregate signature [14]. It allows the combiner to aggregate a set of partial signatures on multiple messages.

Transparent setup. First of all, each user generates his secret-public key pair $(sk_i = x_i, pk_i = g^{x_i})$, and registers pk_i. To avoid the known rogue-key attacks [14], we first let each user run proof of knowledge of x_i during the registration. The system simply includes $pk_1, \ldots, pk_n$ and some common parameter $g_1, \ldots, g_n$ as public parameters and makes them available to everyone.

Succinct size solution. We start with the core building block of dynamic threshold aggregate signature. This can be considered as the special case where the predicate only requires the threshold counting is correct.

An intuitive idea is letting the combiner do more work: not only the partial signatures are aggregated, but their respective public keys are also compressed into a compact version. To protect signer anonymity, it also adds some blind factors to the compressed public keys and signatures. However, anonymity introduces a concern regarding the correctness of compressed public keys. Specifically, there is no guarantee that these compressed public keys are part of the legitimate/registered public key set.

Therefore, the combiner needs to produce an additional proof for the membership relation and duplication checks (that there are indeed t signatures from t *distinct* signing keys). A naïve attempt for the latter would be proving pairwise difference on all the compressed signing keys, which will yield a *quadratic* size proof. Some techniques in relevant primitives such as graded signatures [28] and signature of reputation [10] got around the challenge and sorted the public keys first, to do a sequential proof that $pk_i \neq pk_{i+1}$, which can push down the proof size to be *linear*. However, that is still quite cumbersome.

Alternatively, we observe that instead of proving relations among signer keys directly, we may leverage the published public keys in the public parameter. First, we can represent the included keys as a binary vector $\boldsymbol{b} = (b_1, ..., b_n)$, i.e., $b_i = 1$, if pk_i is in (has signed on the message), and 0 otherwise, and commit $\boldsymbol{b}$ in a succinct way (via vector commitment). Then we can prove an alternative statement that the committed vector is indeed *binary*. Now the Hamming weight of this vector will be corresponding to the threshold. Two remaining parts: (i)

each bit value is assigned correctly; (ii) Hamming weight is correctly computed. For (i), observe that when each pk_i is directly taken in as part of the system parameter, the "aggregated public keys" $\widehat{pk} = \Pi_i pk_i^{b_i}$ can be seemed as another "commitment" to the binary vector. We can establish the validity of the bit assignment by demonstrating that the previously committed binary vector is identical to the one contained in $\widehat{pk}$. While for (ii), Hamming weight can again be derived directly from inner product of the bit vector and all 1 vector, and proven using the efficient inner product argument from Bulletproofs [19].

Now we have a construction framework from the inner product argument and "binary" proof (that proves a committed vector is binary), which can be instantiated via Bulletproofs [19], yielding a signature of *logarithmic size* relative to the number of all users.

In the multiple (say k) messages setting, signers are divided into multiple sets depending on the message they have signed. A natural method is running the above proof generation for k times, so the total communication cost would have a multiplicative factor of k. Fortunately, by exploring the above technique further, we can generate a proof for k values on the knowledge of n-length binary vectors. In this way, these k proofs can be aggregated into one single proof for a $(k \cdot n)$-length binary vector. As a result, we achieve a communication cost of $O(k + \log n + \log k)$, comprising k aggregated public keys and additional proofs of size $O(\log n + \log k)$.

Reduce verification time. However, the above signature still requires a linear verification time (for example, even reading in all the public keys). To also reduce verification time, we propose a new proof system for the inner product and binary relations with structured parameters, that can reduce the verification time also to logarithmic.

There were previous efforts improving verification cost [23,30], in [23], the authors achieve logarithmic size and logarithmic verification time for inner product argument and range proof using *structured* reference string with highly correlated parameters in the form of $g, g^{x_1}, g^{x_2}, \ldots, g^{x_{\log n}}, g^{x_1 \cdot x_2}, g^{x_1 \cdot x_3}, g^{x_2 \cdot x_3}, g^{x_1 \cdot x_2 \cdot x_3} \ldots$, that separates the parameter into two parts: linear proving parameter and logarithmic verification parameter.

Unfortunately, as we would like a transparent setup, and public keys are generated by users themselves randomly, and then included as the public parameter, which clearly inconsistent with these structured parameters.

To work around this, we need to redesign the parameter generation and the statement for the proof. Besides the binary vector, we also commit the public keys via structure preserving commitment of [6] (also called AFGHO commitment). Introducing structured parameters into it is still compatible with the random generated public keys. Given these two commitments, we can prove another element is the inner pairing product of the two committed vectors. We observe that demonstrating the well-formedness of the aggregated public key is equivalent to proving that its bilinear map is equal to the inner pairing product between a binary vector and all public keys. Now we prove the validity of $\widehat{pk}$ by directly leveraging inner product argument between two committed vec-

tors. One of these vectors is a binary vector, whose correctness is guaranteed by a binary proof, while the other comprises all the public keys. By adjusting the AFGHO commitment with structured parameters, these proofs achieve efficiency with logarithmic communication costs and verification times. We refer detailed description in Sec. 4.1.

Achieve anonymity. In the anonymous setting with a blind factor r, where $\widehat{pk} = \Pi_i pk_i^{b_i} \cdot \tilde{g}^r$, several challenges arise when applying the previous method. These challenges include proving the last position of the binary vector is 1 and handling commitments of public keys together with the random factor $\tilde{g}^r$. These challenges are exacerbated by the anonymity requirements. See Sec. 4.1 for detailed discussion.

To mitigate these issues, a new approach is proposed. First, $\boldsymbol{b}$ and r are committed separately using distinct commitment keys, and proofs are generated for each. Then, by combining these two commitments, we can prove the presence of both a binary vector and a blind factor in specific positions. Subsequently, an inner pairing product argument is applied to these vectors, ensuring the well-formedness of the blinded aggregated public key.

Generic predicate. Then to lift the construction to support any arithmetic predicate on the signer identities, we observe that both techniques for the core building block is via Fiat-Shamir transformation on Σ-protocols. We can add the extra proof of predicate satisfaction similarly via Bulletproof with our optimized verification time, then use the classical And proof to bind them. The final proof is with logarithmic size and verification time, while its security can be based on the standard SXDH assumption.

Efficient instantiations for concrete predicates. As discussed above, such a special PAS with dynamic threshold already gives a better construction of multiple relevant signatures, as shown in Table 2.

We also give a concrete construction for the concrete predicate that all signer sets are also disjoint (that denotes the rate-once policy in the motivational application of anonymous reputation system).

It is a challenging task for the combiner to demonstrate the disjoint nature of all of these subsets of signers. In general, it would require comparing every pair of them and proving that they are indeed disjoint. However, this approach would necessitate a quadratic number of comparisons, leading to additional significant communication and computation cost.

It is worth noting that the binary feature can also be utilized in this case. Specifically, each public key subset can be represented as a binary vector. The addition of two binary vectors corresponds to the union of the corresponding subsets, including duplicate elements if any. In case the resulting sum vector remains binary, it implies that there are no duplicate elements in the union set, thereby indicating that the two sets are disjoint. By extending this approach to the k-subsets scenario, where we add all these binary vectors, we can demonstrate that all of the public key subsets are indeed disjoint.

Applications and extensions. Our PAS (including the building block alone) implies many interesting primitives such as threshold signature, aggregate signa-

ture, multi-signature, ring signature, threshold ring signature, etc; more importantly, our efficient construction with different instantiations of concrete predicates, can improve the state of the art of all those primitives. More specifically, when using our dynamic threshold aggregate signature, we can directly yield the first multi-signature, aggregate signature, graded signature and threshold signature with both $O(\log n)$ communication and verification cost, while the state-of-the-art construction of them (from standard assumptions except zk-SNARKs or AGM directly) are all having linear costs. See Table 2.

Table 2. Advancing relevant primitives.[1]

Primitives	Commun. Cost	Verify Cost.	Generation Cost.
Multi-Sig. [13]	$O(n)^2$	$O(n)$	$O(n)$
Agg-Sig. [14]	$O(n)^3$	$O(n)$	$O(n)$
Graded Sig. [28,10]	$O(n)$	$O(n)$	$O(n)$
Thld-ring Sig. [7]	$O(\log n)$	$O(n)$	$O(n)$
Using our $\mathsf{PAS}^\star$	$O(\log n)$	$O(\log n)$	$O(n)$

[1] In the comparison, we restrict only to the single message case in our PAS. If there are k messages to be signed, all others have a *multiplicative* factor k, while we only have an *additive* factor.

[2,3] Although their signatures can be aggregated, the signers' identities should also be transmitted which leads to linear communication cost, except recent ones [22,27] that rely on zk-SNARKs or AGMs directly.

* The last row means using our PAS with dynamic threshold as Δ, it implies the above primitives and advances their performance.

For multiple users and multiple messages, the combiner generates a PAS with threshold t_j for each m_j. It implies the aggregate signature and hides the signers' identities. For multiple users and one message, a PAS with threshold t implies that t different signers have signed on the message. It implies the threshold signature with transparent setup and dynamic threshold t and threshold ring signature with threshold t. It also naturally implies the multi-signature with t signers and the graded signature which indicates there are t different signers. When there is only one signer and one message, it implies the ring signature and attribute-based signature. The signer himself works as the combiner and shows the threshold is 1 with the proof of satisfying the predicate. It is equivalent to validating the signer's attribute. More details can be found in Sec. 6.

Anonymous reputation systems. Recent works on the anonymous reputation systems [12,11,24] achieve full anonymity at the cost of linear communication cost and quadratic verification complexity. We allow the combiner to know the signer identities and let the number of signers be the description. It generates a PAS which reveals the reputation states and its size is just logarithmic and can be verified in logarithmic time.

Onchain voting system. Certain voting policies necessitate that voters possess a particular property [3], and it can be denoted by their identity index. For instance, within an organization, senior members are associated with indices smaller than a threshold. The combiner's task is to demonstrate that among the signers, there is at least one whose index is lower than a specified threshold. In our design, relying on the binary vector, the combiner only needs to prove the existence of a single position in the vector where the value is 1 and the position is smaller than a specified threshold.

Extensions. We can also easily support further advanced properties such as *dynamic join, weighted, accountability* and more.

- New users can seamlessly join the system without causing any disruptions to existing users. The process of joining is transparent and does not have any adverse effects on other users. It just involves changing a single global public key in a publicly verifiable way.
- In the PAS scheme, each user can associate their public key with a weight and the Δ is defined as the total weight of the signers. As a result, when generating the PAS signature, it discloses the total weight of the signers involved. So our PAS also supports the weight aggregation feature like [33,20,22].
- Our PAS can be extended to support the accountability by adding an extra identities encryption layer. It is similar with the method of TAPS [15].

2 Preliminary

We use bold letter for vector, for example $\boldsymbol{a} = (a_1, ..., a_n) \in \mathbb{Z}_q^n$. We use $\circ$ to denote the Hadamard product: $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 \cdot b_1, ..., a_n \cdot b_n)$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^n$. We use $[k]$ to denote the integers in $\{1, 2, ..., k\}$. On input the security parameter 1^λ, a group generator $\mathsf{G.Gen}(1^\lambda)$ produces public parameters $\mathsf{G.pp} = (q, \mathbb{G}, g)$, where q is a prime of length λ, and $\mathbb{G}$ is a cyclic group of order q with generator g. Similarly, a bilinear group generator $\mathsf{BG.Gen}(1^\lambda)$ produces public parameters $\mathsf{BG.pp} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e)$ where $\mathbb{G}_1 = \langle g \rangle, \mathbb{G}_2 = \langle \tilde{g} \rangle, \mathbb{G}_T$ are groups of order q. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ defines $g_T = e(g, \tilde{g})$, the map is bilinear, (for all $a, b \in \mathbb{Z}_q$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$) and non-degenerate (for all generators g of $\mathbb{G}_1$, $\tilde{g}$ of $\mathbb{G}_2$, $\mathbb{G}_T = \langle e(g, \tilde{g}) \rangle$). We assume $\mathbb{G}_1 \neq \mathbb{G}_2$ and we are working on Type III groups [6] who do not have efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. We use $[a]_1, [b]_2, [c]_T$ denotes the element $g^a, \tilde{g}^b, g_T^c$ in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively. We use $[\boldsymbol{x}]_1$ denotes the vector $(g^{x_1}, ..., g^{x_n}) \in \mathbb{G}_1^n$ for $\boldsymbol{x} = (x_1, ..., x_n) \in \mathbb{Z}_q^n$. We write all groups additively, e.g., $[a]_1 + [b]_1 = [a + b]_1$ denotes $g^a \cdot g^b = g^{a+b}$, $b \cdot [a]_1 = [ab]_1$ denotes $(g^a)^b = g^{ab}$, $[\boldsymbol{x}]_1 + [\boldsymbol{y}]_1 = [\boldsymbol{x} + \boldsymbol{y}]_1$ denotes $(g^{x_1}, ..., g^{x_n}) \circ (g^{y_1}, ..., g^{y_n}) = (g^{x_1+y_1}, ..., g^{x_n+y_n})$, $[\boldsymbol{x}]_1 \circ \boldsymbol{y} = (g^{x_1 y_1}, ..., g^{x_n y_n})$, $[\boldsymbol{x}]_1^{\boldsymbol{y}} = \sum_{i=1}^n y_i \cdot [x_i]_1 = \Pi_{i=1}^n g^{x_i y_i}$. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_q^n$, let $\langle \boldsymbol{a}, \boldsymbol{b} \rangle := \sum_{i=1}^n a_i \cdot b_i$ denote the inner product between $\boldsymbol{a}, \boldsymbol{b}$. For $[\boldsymbol{a}]_1 \in \mathbb{G}_1^n$ and $[\boldsymbol{b}]_2 \in \mathbb{G}_2^n$, let $\langle [\boldsymbol{a}]_1, [\boldsymbol{b}]_2 \rangle := e([\boldsymbol{a}]_1, [\boldsymbol{b}]_2) = \sum_{i=1}^n e([a_i]_1, [b_i]_2)$ denote the inner pairing product between $[\boldsymbol{a}]_1, [\boldsymbol{b}]_2$. Given a vector $\boldsymbol{v} = (v_1, ..., v_n)$ with even n, we denote $\boldsymbol{v}_\ell = (v_1, ..., v_{n/2})$ and $\boldsymbol{v}_r = (v_{n/2+1}, ..., v_n)$. For $k \in \mathbb{Z}_q^*$ we use $\boldsymbol{k}^n$ to denote the vector containing the first n powers of k, i.e., $\boldsymbol{k}^n = (1, k, k^2, ..., k^{n-1})$.

2.1 Assumptions

Definition 1 (DDH assumption). *Let* $(q, \mathbb{G}, g) \leftarrow \mathsf{G.Gen}(1^\lambda)$ *be a group generator. The DDH assumption holds for* $\mathsf{G.Gen}$ *if the following distributions are indistinguishable:* $(g, g^a, g^b, g^{ab} : a, b \leftarrow_\$ \mathbb{Z}_q)$ *and* $(g, g^a, g^b, g^c : a, b, c \leftarrow_\$ \mathbb{Z}_q)$

Definition 2 (DLOG assumption). *Let* $(q, \mathbb{G}, g) \leftarrow \mathsf{G.Gen}(1^\lambda)$ *be a group generator. The DLOG assumption holds for* $\mathsf{G.Gen}$ *if for all PPT adversary* $\mathcal{A}$ *we have:* $\Pr[\mathcal{A}(q, \mathbb{G}, g, X) = x | (q, \mathbb{G}, g) \leftarrow \mathsf{G.Gen}(1^\lambda), x \leftarrow_\$ \mathbb{Z}_q, X = g^x] \leq negl(\lambda)$

Definition 3 (SXDH assumption [6]). *Let* $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$ *be a bilinear group generator. The SXDH assumption holds for* $\mathsf{BG.Gen}$ *if DDH assumption holds for* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*.*

Definition 4 (co-CDH assumption[14]). *Let* $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$ *be a bilinear group generator. The co-CDH assumption holds for* $\mathsf{BG.Gen}$ *if for all PPT* $\mathcal{A}$*, given* $[a]_1, [b]_2$ *where* $a, b \leftarrow_\$ \mathbb{Z}_q$*, the probability that* $\mathcal{A}$ *can produce* $[ab]_1$ *is negligible.*

Definition 5 (DPair assumption [6]). *Let* $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$ *be a bilinear group generator,* $n = poly(\lambda)$*. The double-pairing (DPair) assumption holds for* $\mathsf{BG.Gen}$ *if for all PPT adversary* $\mathcal{A}$*, given* $[r]_1 \leftarrow_\$ \mathbb{G}_1$*, the probability that* $\mathcal{A}$ *can produce* $[a]_2, [b]_2 \in \mathbb{G}_2$ *s.t.* $e([r]_1, [a]_2) + e(g, [b]_2) = [0]_T$ *and* $a, b \neq 0$ *is negligible.*

Definition 6 (ML-Find-Rep assumption [23]). *Let* $(q, \mathbb{G}, g) \leftarrow \mathsf{G.Gen}(1^\lambda)$ *be a group generator,* $n = poly(\lambda)$ *which a power of 2,* $\nu = \log n$*. The ML-Find-Rep assumption holds for* $\mathsf{G.Gen}$ *if for all PPT adversary* $\mathcal{A}$ *we have:* $\Pr[\mathcal{A}(q, \mathbb{G}, [\boldsymbol{r}], X) \rightarrow \boldsymbol{a} \in \mathbb{Z}_q^n \text{ s.t. } [\boldsymbol{r}]^{\boldsymbol{a}} = [0] \wedge \boldsymbol{a} \neq \boldsymbol{0} | (q, \mathbb{G}, g) \leftarrow \mathsf{G.Gen}(1^\lambda), (x_1, ..., x_\nu) \leftarrow_\$ \mathbb{Z}_q^\nu, \boldsymbol{r} = (1, x_1, x_2, x_1x_2, ..., x_1 \cdots x_\nu)] \leq negl(\lambda)$.

Definition 7 (DPair-ML assumption). *Let* $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathsf{BG.Gen}(1^\lambda)$ *be a bilinear group generator,* $n = poly(\lambda)$ *which a power of 2. The DPair-ML assumption holds for* $\mathsf{BG.Gen}$ *if for all PPT adversary* $\mathcal{A}$*, given* $[\boldsymbol{r}]_1 \in \mathbb{G}_1^n$*, where* $\boldsymbol{r} = (1, x_1, x_2, x_1x_2, ..., x_1 \cdots x_\nu)$ *for* $(x_1, ..., x_\nu) \leftarrow_\$ \mathbb{Z}_q^\nu$*, the probability that* $\mathcal{A}$ *can produce* $[\boldsymbol{s}]_2 \in \mathbb{G}_2^n$ *s.t.* $e([\boldsymbol{r}]_1, [\boldsymbol{s}]_2) = [0]_T$ *is negligible.*

The DPair-ML assumption is implied by the SXDH assumption and ML-Find-Rep assumption.

2.2 Cryptographic Primitives

Due to space constraints, we introduce some cryptographic primitives here briefly and defer the detailed preliminaries to the full version.

Commitment A commitment scheme allows one to commit to a chosen value secretly, with the ability to only open to the same committed value later. A commitment scheme Π_{cmt} consists of the following PPT algorithms:

$\mathsf{Setup}(1^\lambda) \to pp$: generates the public parameter pp.
$\mathsf{Com}(m; r) \to com$: generates the commitment for the message m using the randomness r.
Hiding. A commitment scheme is said to be hiding if the commitment does not reveal any information about the committed value.
Binding. A commitment scheme is said to be binding if a commitment can only be opened to one value.
Additively homomorphic. A commitment is additively homomorphic if for any values m_1, m_2 and randomness r_1, r_2: $\mathsf{Com}(m_1; r_1) + \mathsf{Com}(m_2; r_2) = \mathsf{Com}(m_1 + m_2; r_1 + r_2)$.

Pedersen Commitment For messages $\boldsymbol{m} \in \mathbb{Z}_q^n$ and any $i \in \{1, 2, T\}$, the Pedersen commitment is defined by:
$\mathsf{Setup}(1^\lambda) \to pp$: $\boldsymbol{g} \leftarrow_\$ \mathbb{G}_i^n, h \leftarrow_\$ \mathbb{G}_i$.
$\mathsf{Com}(\boldsymbol{m}; r) \to com$: $\mathsf{Com}(\boldsymbol{m}; r) = \boldsymbol{g}^{\boldsymbol{m}} \cdot h^r \in \mathbb{G}_i$ where $r \leftarrow_\$ \mathbb{Z}_p$.

The Pedersen commitment is additively homomorphic, perfectly hiding and computationally binding under the DLOG assumption.

AFGHO Commitment Abe et. al. [6] defined a structure preserving commitment to group elements. In this case we have the message space $\mathbb{G}_2^n$:
$\mathsf{Setup}(1^\lambda) \to pp$: Run $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow \mathcal{G}(1^\lambda)$, the commitment key $\mathsf{ck}_1 := \boldsymbol{g} \leftarrow_\$ \mathbb{G}_1^n$.
$\mathsf{Com}(\boldsymbol{m}; r) \to com$: for $[\boldsymbol{m}]_2 \in \mathbb{G}_2^n$, $\mathsf{Com}([\boldsymbol{m}]_2; [r]_2) = \langle \mathsf{ck}_1, [\boldsymbol{m}]_2 \rangle + e(g, [r]_2)$ where $[r]_2 \leftarrow_\$ \mathbb{G}_2$.

To commit to messages in $\mathbb{G}_1$, we can just interchange the role of $\mathbb{G}_1$ and $\mathbb{G}_2$ in the above construction with $\mathsf{ck}_2 \in \mathbb{G}_2^n$.

The AFGHO commitment is additively homomorphic, perfectly hiding and computationally binding under the SXDH assumption.

Structured AFGHO Based on the updatable common reference string technique of Daza et al. [23], we give the modified AFGHO commitment with structured commitment keys $\mathsf{ck}_1, \mathsf{ck}_2$ which are generated as below.

$$(pp, [r]_1 \in \mathbb{G}_1, \mathsf{ck}_1 = [\boldsymbol{r}]_1 \in \mathbb{G}_1^n, \mathsf{vk}_1 = [\boldsymbol{x}]_2 \in \mathbb{G}_2^\nu) \in \mathcal{L}_{\mathsf{Com}}^1 \Leftrightarrow$$
$$[r_1]_1 = [r]_1 \wedge \forall i \in [\nu], \forall j \in [2^{i-1}], [r_{2^{i-1}+j}]_1 = x_i [r_j]_1$$

$$(pp, [s]_2 \in \mathbb{G}_2, \mathsf{ck}_2 = [\boldsymbol{s}]_2 \in \mathbb{G}_2^n, \mathsf{vk}_2 = [\boldsymbol{y}]_1 \in \mathbb{G}_1^\nu) \in \mathcal{L}_{\mathsf{Com}}^2 \Leftrightarrow$$
$$[s_1]_2 = [s]_2 \wedge \forall i \in [\nu], \forall j \in [2^{i-1}], [s_{2^{i-1}+j}]_2 = y_i [s_j]_2$$

where $r, x_i \leftarrow_\$ \mathbb{Z}_q$ and $s, y_i \leftarrow_\$ \mathbb{Z}_q$ for all $i \in [\nu]$.

The structured AFGHO commitment is additively homomorphic, perfectly hiding and computationally binding under the SXDH and DPair-ML assumptions.

Zero-Knowledge Arguments of Knowledge (ZKAoK) A zero-knowledge argument of knowledge is a cryptographic protocol involving two parties: a prover and a verifier. In this protocol, the prover's objective is to provide convincing

proof to the verifier that a certain statement is true, without revealing any information about the underlying witness.

It consists of three PPT algorithms Setup, $\mathcal{P}$, and $\mathcal{V}$. The setup algorithm outputs a common reference string σ on inputting a security parameter λ. The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ are interactive algorithms. As the output of this protocol, we use the notation $\langle \mathcal{P}, \mathcal{V} \rangle = b$, where $b = 1$ if $\mathcal{V}$ accepts and $b = 0$ if $\mathcal{V}$ rejects. The proof is *public coin* if an honest verifier generates his responses to $\mathcal{P}$ uniformly.

Argument of knowledge. $(\mathbf{Setup}, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge for the relation $\mathcal{R}$ if it satisfies the following two definitions.

Perfect completeness. The prover can persuade the verifier if it possesses a witness that attests to the truth of the statement.

Computational witness-extended emulation. Whenever an adversary that produces an acceptable argument with some probability, there exist an emulator who can produce a similar argument with the same probability and provide a witness w simultaneously. It implies *soundness* which asserts that no PPT adversary can persuade the verifier when the statement is false. It also assures *knowledge soundness* which guarantees the existence of an extractor capable of producing a valid witness for the statement.

Honest-verifier special zero-knowledge (HVSZK). Given the verifier's challenge values, it is possible to simulate the entire argument without witness efficiently.

BLS aggregate signature We briefly review the BLS signature scheme and its signature aggregation mechanism [14]. Given an efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ in groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order q. Let g and $\tilde{g}$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, a hash function $H : \mathcal{M} \to \mathbb{G}_1$:

- **KeyGen**(): the user chooses $sk \leftarrow_{\$} \mathbb{Z}_q$, outputs (pk, sk) for $pk \leftarrow \tilde{g}^{sk} \in \mathbb{G}_2$.
- **Sign**(sk, m): output $\sigma \leftarrow H(m)^{sk} \in \mathbb{G}_1$.
- **Vrfy**(pk, m, σ): output 1 if $e(\sigma, \tilde{g}) = e(H(m), pk)$, otherwise, output 0.
- **Signature Aggregation**: Given triples (pk_i, m_i, σ_i) for $i \in [n]$, anyone can aggregate the signatures $\sigma_1, ..., \sigma_n$ into a single group element $\hat{\sigma} \leftarrow \Pi_{i \in [n]} \sigma_i \in \mathbb{G}_1$. Verification can be done by checking that if

$$e(\hat{\sigma}, \tilde{g}) = e(H(m_1), pk_1) \cdots e(H(m_n), pk_n).$$

  For all same messages it just needs to check if $e(\hat{\sigma}, \tilde{g}) = e(H(m_1), \Pi_{i=1}^{n} pk_i)$. It is *unforgeable* under the co-CDH assumption.

The rogue public-key attack and defense. Note that the aggregate public key $\Pi_{i=1}^{n} pk_i$ suffers the rogue public-key attack [9]. To prevent it, we use the Proof-of-Possession (PoP) mechanism [34] in the *registered key model*. In this approach, each party is required to provide a proof that they possess the private key corresponding to their public key. This proof can be included during the setup phase and ensures that only legitimate key owners can participate. In this paper, we implicitly assume the presence of PoP proofs for the public keys.

3 Predicate Aggregate Signatures

In this section, we formalize the predicate aggregate signatures and establish the security model for this concept. Predicate aggregate signatures enable users to sign multiple messages according to some predefined public policy, and these individual signatures can be aggregated by a combiner, preserving the anonymity of the signers. The resulting aggregate signature discloses only a brief description of the involved signers (e.g., count of signers for each message, total weight of signers, etc) and provides assurance that these signers and the description satisfy the specified policy denoted by a public predicate function.

This notion addresses the need for efficient and privacy-preserving signature schemes that allow for the signing of multiple messages while ensuring adherence to a given predicate. The security model encompasses the privacy of signers and the unforgeability of the predicate aggregate signature.

3.1 Syntax

In general, there are three parties in the system: signers who sign on the message; the combiner, who generates a predicate aggregate signature with a public description of the involved signers and proves that these signers and the description satisfy a public predicate; the verifier who verifies the correctness of the predicate aggregate signature.

- $\mathsf{Setup}(1^\lambda)$: On the security parameter λ, the system public parameters pp are generated. The message space is set as $M = \{m_j\}_{j\in[k]}$. There is a public policy Ω which decides the computation rule of the signers description Δ and the predicate function P_Ω.
  It also includes the key generation of users. Each user u_i generates his secret key sk_i and public key pk_i pair and broadcasts the public key. The combiner (or any other parties) collects the public keys and publishes the aggregation key ak and verification key vk which contains P_Ω.
- $\mathsf{ParSign}(sk_i, m_j)$: For a message m_j chosen from M, the user i signs on it using his secret key sk_i and sends (pk_i, m_j, σ_{ij}) to the combiner.
- $\mathsf{ParVrfy}(pk, m_j, \sigma)$: On receiving (pk_i, m_j, σ), anyone can verify it.
- $\mathsf{Combine}(ak, \{S_j\}_{j\in[k]}, \{\{\sigma_{ij}\}_{i\in S_j}\}_{j\in[k]}, \{m_j\}_{j\in[k]})$: When receiving sets of signatures $\{\{\sigma_{ij}\}_{i\in S_j}\}_{j\in[k]}$ on the message m_j from different signers w.r.t. index sets $\{S_j\}_{j\in[k]}$ (called signer sets) where each $S_j \subseteq [n]$, the combiner generates a signature Σ for the message set $M = \{m_j\}_{j\in[k]}$ with corresponding description Δ of these signer sets. It also proves that the signer sets and Δ satisfy the public predicate P_Ω decided by some policy Ω, i.e., $P_\Omega(S_1, ..., S_k, \Delta) = 1$.
- $\mathsf{Verify}(vk, M, \Delta, \Sigma)$: Given the verification key vk, messages $M = \{m_j\}_{j\in[k]}$, description Δ, the PAS signature Σ, anyone can check the validness by running $\mathsf{Verify}(vk, M, \Delta, \Sigma)$ and outputs one bit $b \in \{0, 1\}$ indicating if it is valid.

Remark 1. Δ is the description of signers in $S_1, ..., S_k$ whose partial signatures are used to generate the final PAS signature. It is computed via a *deterministic* function F specified in the policy Ω from the signer sets: $\Delta = F(S_1, ..., S_k)$. Note that it cannot display the signer identities plainly, since it ruins the anonymity directly. Although it is computed deterministically, in many cases, it leaks very little information about the signers. For example, it could be the size of each signer set, the combined weight of signers within each signer set, or simply demonstrating that the number exceeds a certain minimum threshold.

Remark 2. P_Ω is the predicate decided by the public policy Ω which takes the signer sets and a description as input. $P_\Omega(S_1, ..., S_k, \Delta) = 1$ indicates that $S_1, ..., S_k, \Delta$ satisfy the rule according to Ω.

3.2 Model

Correctness If enough valid signatures under different public keys are used to produce the signature Σ on the messages m_j for $j \in [k]$, the description Δ and the signer sets satisfy the public predicate $P_\Omega(S_1, ..., S_k, \Omega) = 1$, then the verification for (M, Δ, Σ) always outputs 1.

Anonymity The signer identities are hidden from the public. The PAS signature only discloses a description of the signer sets and whether they satisfy the predicate according to the public policy. Given any two valid signatures Σ_0, Σ_1 with the *same* descriptions and predicates from different signer sets on the same messages set M, they are indistinguishable even to some of these signers.

Unforgeability The adversary cannot generate a valid signature on multiple messages if it does not have enough signatures from different signers on each message, or the signer sets and the description do not satisfy the predicate.

Oracles We define the following oracles to model the adversary's ability. There is an honest user table HU, a corrupted user table CU and a queried message table QM which are initialized as empty.

- **add**(i): Add a new user u_i to the system. If i has not been queried before, run the key generation algorithm $(pk^*, sk^*) \leftarrow \mathsf{KeyGen}$, set $(pk_i, sk_i) = (pk^*, sk^*)$ and output pk_i. Add (u_i, pk_i, sk_i) to the honest user table HU.
- **corrupt**(pk_i): Corrupt an honest user in the system. If $pk_i \in \mathsf{HU}$, output sk_i, delete (u_i, pk_i, sk_i) from HU and add (u_i, pk_i, sk_i) to CU.
- **sign**(pk_i, m): If $pk_i \notin \mathsf{HU}$, ignore it. Otherwise, run $\sigma \leftarrow \mathsf{ParSign}(sk_i, m)$ and add (pk_i, m) to QM, output σ.

Anonymity This property ensures that signer identities are hidden from the public. Only the combiner knows the signer identities. Others just know the description of the signers and the signer sets with the description satisfy the public predicate. Formally speaking, given any two valid signatures Σ_0, Σ_1 from different signer sets on the same messages M with the same description Δ and both satisfy the predicate, nobody can distinguish them except the combiner.

In the anonymity definition, even the signers will not be able to distinguish two PAS signatures for the maliciously chosen messages with same thresholds. The anonymity experiment $\text{Exp}^{\mathsf{anony}}$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is formalized as follows.

- $\mathcal{A}$ receives the public parameters including the description function F and public predicate P_Ω specified by the policy Ω.
- $\mathcal{A}$ adds users to the system, it can query signatures of any signers on any messages and even corrupt all users. The global aggregation key ak and verification key vk are setup and given to $\mathcal{A}$.
- $\mathcal{A}$ chooses a set of messages $M = \{m_j\}_{j\in[k]}$, and two sets of index sets $S^0 = \{S_j^0\}_{j\in[k]}$ and $S^1 = \{S_j^1\}_{j\in[k]}$ where each $S_j^0, S_j^1 \subseteq [n]$. Then it generates partial signatures $\sigma_{u,j} \leftarrow \mathsf{ParSign}(sk_u, m_j)$, $\sigma_{v,j} \leftarrow \mathsf{ParSign}(sk_v, m_j)$ for all $j \in [k], u \in S^0$ and $v \in S^1$ and let $D_0 = \{\{\sigma_{u,j}\}_{u\in S^0}\}_{j\in[k]}, D_1 = \{\{\sigma_{v,j}\}_{v\in S^1}\}_{j\in[k]}$. It sends (M, S^0, S^1, D_0, D_1) to $\mathcal{C}$.
- $\mathcal{C}$ computes $\Delta_0 = F(S^0)$, $\Delta_1 = F(S^1)$ and checks these partial signatures. It aborts, if there exists any invalid partial signature or $S^0 = S^1$ or $\Delta_0 \neq \Delta_1$ or $P_\Omega(S^0, \Delta_0) \neq 1$ or $P_\Omega(S^1, \Delta_1) \neq 1$. Otherwise, continue.
- $\mathcal{C}$ randomly chooses one bit $b \leftarrow \{0, 1\}$ and sends $\mathcal{A}$ a predicate aggregate signature Σ with M generated from signatures of D_b by running $\Sigma \leftarrow \mathsf{Combine}(ak, S^b, D_b, M)$.
- $\mathcal{A}$ outputs a guess b'.
- Outputs 1 if $b' == b$, otherwise, outputs 0.

Definition 8. *A predicate aggregate signature is anonymous if any PPT adversary in the* $\text{Exp}^{\mathsf{anony}}$ *can only guess the bit correctly with probability negligibly close to* $\frac{1}{2}$*, i.e.,* $|\Pr[\text{Exp}^{\mathsf{anony}}(\mathcal{A}, \lambda) = 1] - \frac{1}{2}| \leq negl(\lambda)$.

Unforgeability This property ensures that given the public policy, any adversary cannot generate a valid signature on multiple messages if it does not have enough signatures on the messages or the signer sets and their description do not satisfy the predicate.

It contains two properties: (1). $\mathcal{A}$ can not produce a valid PAS signature which contains an honest signer who has never signed on that message. (2). All signer sets w.r.t. the messages must adhere to the specified predicate. It is infeasible for $\mathcal{A}$ to generate a valid PAS signature with a signer sets description but the signer sets and the description are unsatisfied for the predicate.

Note that due to the anonymity requirement, the signer identities are hidden and only their description is shown. So it is hard to decide whether $\mathcal{A}$ has broken the predicate satisfaction property. To address this dilemma, we introduce an extractor $\mathcal{E}$ that has the ability to reveal the signers' identities for each message from the predicate aggregate signature. It is inspired by the knowledge extractor for the knowledge soundness in zero-knowledge proof of knowledge.

Formally speaking, the unforgeability experiment $\text{Exp}^{\mathsf{unforge}}$ works as follows:

- $\mathcal{A}$ receives the public parameters and the public predicate.

- $\mathcal{A}$ can add users to the system then gets the aggregation key ak and verification key vk.
- $\mathcal{A}$ is an adaptive adversary, it can interact with $\mathcal{E}$ via querying oracles. It can then query signatures of any signers on any messages and corrupt users.
- $\mathcal{A}$ outputs a tuple (M, Δ, Σ).
- On a valid (M, Δ, Σ), $\mathcal{E}$ outputs the signers' identities $S_1, ..., S_k$.

$\mathcal{A}$ wins and the experiment outputs 1 only if (M, Δ, Σ) is valid w.r.t. vk and at least one of the following conditions is satisfied:

- $\exists\ i_j \in S_j$, such that $i_j \in \mathsf{HU}$ and $(pk_{i_j}, m_j) \notin \mathsf{QM}$;
- $P_\Omega(S_1, ..., S_k, \Delta) \neq 1$.

The first condition means $\mathcal{A}$ has never queried the **corrupt** oracle on pk_{i_j} or **sign** oracle on (pk_{i_j}, m_j). It models that $\mathcal{A}$ generates a valid PAS signature without enough valid partial signatures.

Definition 9. *A predicate aggregate signature is unforgeable if any PPT adversary in the above experiment can only win with negligible probability, i.e.,* $\Pr[\mathrm{Exp}^{\mathsf{unforge}}(\mathcal{A}, \lambda) = 1] \leq negl(\lambda)$

4 Constructions

In this section, we give the construction of the predicate aggregate signature. Formally speaking, the combiner aims to prove the knowledge of signatures and signers satisfying the following relation:

$$\begin{aligned} R_{\mathsf{PAS}} = \{&pk_1, \ldots, pk_n, m_1, \ldots, m_k, \Omega, \Delta; \{\sigma_{i1}\}_{i\in S_1}, \ldots, \{\sigma_{ik}\}_{i\in S_k} : \\ &\mathsf{ParVrfy}(pk_i, m_j, \sigma_{ij}) = 1\ \forall i \in S_j, j \in [k]; \\ &S_j \subseteq [n], \forall j \in [k]; P_\Omega(S_1, ..., S_k, \Delta) = 1\} \end{aligned}$$

where $pk_1, \ldots, pk_n$ are all the public keys, $m_1, \ldots, m_k$ are the candidate messages, S_j contains the indices of users who have signed on the message m_j, σ_{ij} is the signature from user i on message m_j. P_Ω is the predicate function decided by the public policy Ω.

The predicate function can be very simple that outputs 1 as long as the description Δ is correct and there is no other requirements on the policy. For any general policy that can be described by an arithmetic circuit, the predicate can be converted into a circuit and proved using the efficient zero-knowledge argument for arbitrary arithmetic circuits which have been studied in [16,19,23].

In this section, we mainly consider the anonymous reputation system with the rate-once policy as a non-trivial example. Here the description is the number of signers in each subset ($\Delta = \{t_j\}_{j\in[k]}$ where $t_j = |S_j|$) and all users can only sign once even for different messages. It means the signer sets for all messages are disjoint. We define the public predicate as follows:

$$P_\Omega(S_1, ..., S_k, \Delta) = \begin{cases} 1, \text{if } \Delta = \{t_j\}_{j\in[k]}, S_j \subseteq [n], |S_j| = t_j, \forall j \in [k]; \\ \quad \wedge S_{j_0} \cap S_{j_1} = \emptyset,\ \forall j_0, j_1 \in [k], j_0 \neq j_1 \\ 0, \text{otherwise.} \end{cases} \quad (1)$$

4.1 Construction Overview

Our starting point is the pairing-based BLS aggregate signature (see Sec. 2.2). Let n be the number of users in the system. On receiving a set of partial signatures σ_{ij} on each message $m_j \in M$ from signer $i \in [n]$, the combiner generates the aggregate signature $\hat{\sigma}$ and publishes it with the index set $S_j \in [n]$ of signers on message m_j for verification. The verifier computes the verification key for m_j w.r.t. S_j: $\widehat{pk}_j = \Pi_{i \in S_j} pk_i$ for $j \in [k]$. Subsequently, it verifies the validity of $\hat{\sigma}$.

An intuitive idea is letting the combiner also compress the public keys into a compact version which can be used for verifying the aggregated signature. To protect the privacy of signers, it also adds *blind factors* to the compressed public keys and respective aggregated signatures. The crux of our construction now revolves around proving the correctness of the compressed public keys without using linear descriptions while still enabling duplication checks.

We start from the single message case. Given all the public keys $\boldsymbol{pk} = (pk_1, ..., pk_n) \in \mathbb{G}_2^n$ and a set $S \subseteq [n]$, let $\boldsymbol{pk}_S = \{pk_i\}_{i \in S} \subseteq \{pk_i\}_{i \in [n]}$ to denote a subset of all public keys w.r.t. S. The blinded aggregation of public keys of $\boldsymbol{pk}_S$ is: $\widehat{pk}_S = \Pi_{i \in S} pk_i \cdot \tilde{g}^{r_S}$ where r_S is the blind factor. Note that S determines a binary vector $\boldsymbol{b}_S = (b_1, ..., b_n)$, $b_i = 1$ if $i \in S$, otherwise, $b_i = 0$.

The combiner (or prover $\mathcal{P}$) computes a commitment B on $\boldsymbol{b}_S$ and proves that the committed $\boldsymbol{b}_S$ is a binary vector whose Hamming weight is t. Then $\mathcal{P}$ proves that $\widehat{pk}_S$ can be expressed in the form of $\boldsymbol{pk}^{\boldsymbol{v}} \cdot \tilde{g}^r$ where $\boldsymbol{v}$ is same as the $\boldsymbol{b}_S$ in B and it knows the blind factor r. It means that $\widehat{pk}_S$ contains t different signers. Combined with an aggregate signature $\hat{\sigma}$ and a message m, the valid tuple $(\widehat{pk}_S, m, \hat{\sigma})$, the verifier can be convinced that the message has been signed by t different users.

For multiple messages $m_1, ..., m_k$ case, the index set of signers who have signed on m_j is S_j and their aggregated public keys are $\widehat{pk}_j$ for $j \in [k]$. Here $\mathcal{P}$ proves the knowledge of corresponding binary vectors $\boldsymbol{b}_j$ whose Hamming weight is t_j and the knowledge of blind factor r_j and the signer sets $S = \{S_1, ..., S_k\}$ and thresholds $T = \{t_1, ..., t_k\}$ satisfy the public predicate P_Ω.

Considering the rate-once policy in the anonymous reputation system as a concrete example, the predicate is denoted by $P_\Omega(S, T) := |S_j| = t_j, \forall j \in [k] \wedge S_{j_0} \cap S_{j_1} = \emptyset$, $\forall j_0, j_1 \in [k], j_0 \neq j_1$. Due to the binary feature, we observe that proving subsets disjoint can be achieved by demonstrating that the summation of all binary vectors is still binary with a Hamming weight $t = \sum_{j \in [k]} t_j$.

In summary, we formulate this process in the following relation:

$$
\begin{aligned}
R_1 = \{\{\widehat{pk}_j, m_j, t_j\}_{j=1}^k, \hat{\sigma}; \boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, r_1, ..., r_k : \\
e(\hat{\sigma}, \tilde{g}) = e(H(m_1), \widehat{pk}_1) \cdots e(H(m_k), \widehat{pk}_k) \\
\wedge \ \widehat{pk}_j = \boldsymbol{pk}^{\boldsymbol{b}_j} \cdot \tilde{g}^{r_j} \ \wedge \ \boldsymbol{b}_j \in \{0,1\}^n \ \wedge \ t_j = \langle \boldsymbol{1}^n, \boldsymbol{b}_j \rangle, \forall j \in [k] \\
\wedge \ \boldsymbol{b} = \Sigma_{j \in [k]} \boldsymbol{b}_j \in \{0,1\}^n\}
\end{aligned}
\tag{2}
$$

Strawman scheme from Bulletproofs In general, the public keys are group elements. It can be integrated with the public parameters of Bulletproofs which

are also group elements. Then we use Pederson commitment to commit the binary vector and prove the correctness of the aggregated public keys. The combiner has $\widehat{pk} = \Pi_{i=1}^{n} pk_i^{b_i} \cdot g^r$ and generates $B = \Pi_{i=1}^{n} g_i^{b_i} \cdot h^r$. It generates binary proof on B with parameter $(g_1, ..., g_n, h)$ and another binary proof on $B \cdot \widehat{pk}$ with parameter $(g_1 \cdot pk_1, ..., g_n \cdot pk_n, h \cdot g)$. It implies that $\widehat{pk}$ shares the same binary vector and randomness in B, so it is well-formed. Here the binary proof can be constructed from Bulletproofs [19] with logarithmic size. The final construction comes with almost the same cost.

For multiple k messages case, instead of generating k individual proofs, we generate a proof for k values on the knowledge of n-length binary vectors. It aggregates k proofs into one proof of a kn-length binary vector ($k \cdot n$ bits) via a random challenge value z and Schwartz-Zippel lemma. It is similar with the aggregated range proofs of Sec. 4.3 in [19]. As a result, we achieve a communication cost of $O(k+\log n+\log k)$ which is just logarithmic to the parameter n. However, the verification time scales linearly with the total bit length, denoted as $O(k \cdot n)$. This places a significant burden on the verifier, prompting us to explore more efficient construction methods that offer sublinear verification times.

Construction with Logarithmic Verifier Daza et al [23] improves Bulletproofs to achieve both logarithmic size and verification time. But it cannot be applied to our setting in the same way as above. Since it relies on structured parameters which are incompatible with the randomly chosen public keys. Integrating the public keys with these parameters would violate their structure and render the technique useless.

Plain case without anonymity. We start from the single message case without anonymity. We assume that the proof of possession has been done to prove the knowledge of secret key for each public key. We aim to prove that a given $\widehat{pk} \in \mathbb{G}_2$ can be expressed in the form of $\boldsymbol{pk}^{\boldsymbol{b}}$ where $\boldsymbol{b}$ is a binary vector. Note that the pairing $e(g, \boldsymbol{pk}^{\boldsymbol{b}}) = \sum_{i=1}^{n} e(g, pk_i^{b_i}) = \sum_{i=1}^{n} e([b_i]_1, pk_i) = \langle [\boldsymbol{b}]_1, \boldsymbol{pk} \rangle$. Instead of directly proving the form of $\widehat{pk}$, we compute the map $e(g, \widehat{pk})$ at first. By the bilinear property, it is sufficient to prove that the map result is also an inner pairing product between $[\boldsymbol{b}]_1$ and $\boldsymbol{pk}$. To this end, we leverage an inner pairing product (IPP) argument. It asserts that a given element in $\mathbb{G}_T$ is the inner pairing product between two vectors in $\mathbb{G}_1^n$ and $\mathbb{G}_2^n$ which are committed with AFGHO commitments. By imposing a specific structure on the commitment key (similar to [23], as we introduced in Sec. 2.2), the verification time is reduced to logarithmic in relation to n.

The remaining issue is proving the form of these two committed vectors. $\boldsymbol{pk}$ are public, so the commitment can be verified publicly. For $[\boldsymbol{b}]_1$, we develop a binary proof in which the verification time is also logarithmic to n. It proves the committed vector consists of elements which is either $[0]_1$ or $[1]_1$. Thus, $\boldsymbol{b} \in \{0, 1\}^n$.

Anonymous case. Now we consider the anonymous setting with a blind factor r: $\widehat{pk} = \boldsymbol{pk}^{\boldsymbol{b}} \cdot \tilde{g}^r$ and $e(g, \widehat{pk}) = \langle ([\boldsymbol{b}]_1, g), (\boldsymbol{pk}, \tilde{g}^r) \rangle$. When we attempt to follow the above method, some issues happen. The combiner needs to take additional work on proving: (i) the last position of the binary vector is 1; (ii) the public keys are

committed together with a random element $\tilde{g}^r$ and he knows r. The first task requires another invocation of binary proof and it is unclear how to prove the discrete logarithm of a committed group element without leaking r or $\tilde{g}^r$.

By the bilinear property, we also have $e(g, \widehat{pk}) = \langle([\boldsymbol{b}]_1, g^r), (\boldsymbol{pk}, \tilde{g})\rangle$. Even though the latter vector $(\boldsymbol{pk}, \tilde{g})$ is publicly known, the situation remains challenging because the binary proof mechanism does not inherently support proving the presence of a random value within the vector. Another observation is that $e(g, \widehat{pk}) = \langle[\boldsymbol{b}]_1, \boldsymbol{pk}\rangle + r \cdot e_T$. One may consider to extract $r \cdot e_T$ and prove the knowledge of r. But it would leak $\boldsymbol{b}$ and violate the anonymity.

To mitigate these issues, we commit $\boldsymbol{b}$ and r with different commitment keys and generate the proofs for them separately. Afterward, by combining these two commitments, we can ascertain the presence of both a binary vector and a blind factor in designated positions. Then we can apply the inner pairing product argument on these vectors and proves the well-formedness of the blinded aggregated public key.

For k messages setting with k aggregated public keys, the aggregation technology on kn-length binary vector can also be applied as we outlined in the strawman scheme.

In the next section, we introduce our inner pairing product argument and binary proof with logarithmic communication cost and logarithmic verification time. They can be rendered non-interactive by applying the Fiat-Shamir heuristic [25]. In Sec. 4.3, we present our PAS signature scheme with the rate-once policy in the anonymous reputation system.

4.2 Succinct Proofs with Logarithmic Verifier

Inner Pairing Product Argument We consider an argument for inner pairing product between two vectors $\boldsymbol{v}_i \in \mathbb{G}_i^n$ committed with structured AFGHO commitments with generators $(\mathsf{ck}_{3-i}, e_H) \in \mathbb{G}_i^n \times \mathbb{G}_T$ for $i \in \{1, 2\}$ where $e_H = e(g_H, \tilde{g}), g_H \leftarrow_{\$} \mathbb{G}_1$. In this section, we assume that the dimension n is a power of 2. If necessary, it is straightforward to add padding to the inputs to ensure this condition is met. Formally, we define a language:

$$
\begin{aligned}
(pp, P, C_1, C_2, [r]_1, \mathsf{ck}_1, \mathsf{vk}_1, [s]_2, \mathsf{ck}_2, \mathsf{vk}_2, e_H) \in \mathcal{L}_{\mathsf{IPP}} \Leftrightarrow \\
&([r]_1, \mathsf{ck}_1, \mathsf{vk}_1) \in \mathcal{L}^1_{\mathsf{Com}} \wedge ([s]_2, \mathsf{ck}_2, \mathsf{vk}_2) \in \mathcal{L}^2_{\mathsf{Com}} \wedge \\
&\exists \boldsymbol{v}_1 \in \mathbb{G}_1^n, \boldsymbol{v}_2 \in \mathbb{G}_2^n, r_{C_1}, r_{C_2}, r_P \in \mathbb{Z}_q : \\
&C_1 = \langle \boldsymbol{v}_1, \mathsf{ck}_2\rangle + r_{C_1} \cdot e_H \wedge C_2 = \langle \mathsf{ck}_1, \boldsymbol{v}_2\rangle + r_{C_2} \cdot e_H \wedge \\
&P = \langle \boldsymbol{v}_1, \boldsymbol{v}_2\rangle + r_P \cdot e_H
\end{aligned}
$$

Common input: $(pp, [r]_1, [s]_2, \mathsf{vk}_1, \mathsf{vk}_2, e_H), P, C_1, C_2 \in \mathbb{G}_T$
$\mathcal{P}$ input: $(\mathsf{ck}_1, \mathsf{ck}_2)$, $\boldsymbol{v}_1 \in \mathbb{G}_1^n, \boldsymbol{v}_2 \in \mathbb{G}_2^n, r_{C_1}, r_{C_2}, r_P \in \mathbb{Z}_q$
Statement: $(pp, P, C_1, C_2, [r]_1, \mathsf{ck}_1, \mathsf{vk}_1, [s]_2, \mathsf{ck}_2, \mathsf{vk}_2, e_H) \in \mathcal{L}_{\mathsf{IPP}}$
$\mathcal{P}$ and $\mathcal{V}$ proceed the protocol Π_{IPP} as follows:
If $n = 1$:

$C_1 = e(v_1, \mathsf{ck}_2) + r_{C_1} \cdot e_H, C_2 = e(\mathsf{ck}_1, v_2) + r_{C_2} \cdot e_H, P = e(v_1, v_2) + r_P \cdot e_H$

- $\mathcal{P}$ samples $s_1 \leftarrow_\$ \mathbb{G}_1, s_2 \leftarrow_\$ \mathbb{G}_2$, $r_{D_1}, r_{D_2}, r_{T_1}, r_{T_2} \leftarrow_\$ \mathbb{Z}_q$ and computes:
$D_1 = e(s_1, \mathsf{ck}_2) + r_{D_1} \cdot e_H, D_2 = e(\mathsf{ck}_1, s_2) + r_{D_2} \cdot e_H$
$T_1 = e(s_1, v_2) + e(v_1, s_2) + r_{T_1} \cdot e_H, T_2 = e(s_1, s_2) + r_{T_2} \cdot e_H$
- $\mathcal{P}$ sends D_1, D_2, T_1, T_2 to $\mathcal{V}$
- $\mathcal{V}$ replies with $c \leftarrow_\$ \mathbb{Z}_q^*$
- $\mathcal{P}$ computes and sends:
$u_1 = v_1 + c \cdot s_1$, $u_2 = v_2 + c \cdot s_2$,
$r_1 = r_{C_1} + c \cdot r_{D_1}$, $r_2 = r_{C_2} + c \cdot r_{D_2}$, $r_3 = r_P + c \cdot r_{T_1} + c^2 \cdot r_{T_2}$
- $\mathcal{V}$ accepts if:
$C_1 + c \cdot D_1 = e(u_1, \mathsf{ck}_2) + r_1 \cdot e_H \wedge$
$C_2 + c \cdot D_2 = e(\mathsf{ck}_1, u_2) + r_2 \cdot e_H \wedge$
$P + c \cdot T_1 + c^2 \cdot T_2 = e(u_1, u_2) + r_3 \cdot e_H$

Else $n > 1$, the reduce procedure:

- $\mathcal{P}$ samples $r_{1\ell}, r_{2\ell}, r_{P\ell}, r_{1r}, r_{2r}, r_{Pr} \leftarrow_\$ \mathbb{Z}_q$ and computes:
$C_{1\ell} \leftarrow \langle \boldsymbol{v}_{1\ell}, \mathsf{ck}_{2r} \rangle + r_{1\ell} \cdot e_H$, $C_{2\ell} \leftarrow \langle \mathsf{ck}_{1r}, \boldsymbol{v}_{2\ell} \rangle + r_{2\ell} \cdot e_H$,
$P_\ell \leftarrow \langle \boldsymbol{v}_{1r}, \boldsymbol{v}_{2\ell} \rangle + r_{P\ell} \cdot e_H$,
$C_{1r} \leftarrow \langle \boldsymbol{v}_{1r}, \mathsf{ck}_{2\ell} \rangle + r_{1r} \cdot e_H$, $C_{2r} \leftarrow \langle \mathsf{ck}_{1\ell}, \boldsymbol{v}_{2r} \rangle + r_{2r} \cdot e_H$,
$P_r \leftarrow \langle \boldsymbol{v}_{1\ell}, \boldsymbol{v}_{2r} \rangle + r_{Pr} \cdot e_H$
- $\mathcal{P}$ sends $C_{1\ell}, C_{2\ell}, P_\ell, C_{1r}, C_{2r}, P_r$
- $\mathcal{V}$ replies with $c \leftarrow_\$ \mathbb{Z}_q^*$
- $\mathcal{P}$ computes:
$\boldsymbol{v}_1' \leftarrow \boldsymbol{v}_{1\ell} c + \boldsymbol{v}_{1r} c^{-1} \in \mathbb{G}_1^{n'}$, $\boldsymbol{v}_2' \leftarrow \boldsymbol{v}_{2\ell} c^{-1} + \boldsymbol{v}_{2r} c \in \mathbb{G}_2^{n'}$,
$r'_{C_1} = r_{C_1} + r_{1\ell} \cdot c^2 + r_{1r} \cdot c^{-2}$, $r'_{C_2} = r_{C_2} + r_{2\ell} \cdot c^{-2} + r_{2r} \cdot c^2$,
$r'_P = r_P + r_{P\ell} \cdot c^{-2} + r_{Pr} \cdot c^2$
$\mathsf{ck}_1' = c\mathsf{ck}_{1\ell} + c^{-1}\mathsf{ck}_{1r}$, $\mathsf{ck}_2' = c^{-1}\mathsf{ck}_{2\ell} + c\mathsf{ck}_{2r}$,
$[r']_1 \leftarrow \{\mathsf{ck}_1'\}_1$, $[s']_2 \leftarrow \{\mathsf{ck}_2'\}_1$ (picks the first elements of $\mathsf{ck}_1', \mathsf{ck}_2'$)
- $\mathcal{P}$ sends $[r']_1$, $[s']_2$ to $\mathcal{V}$.
- $\mathcal{V}$ checks the following equations and aborts if any fails:

$$e([r']_1 - c[r]_1, [1]_2) = e(c^{-1}[r]_1, [x_\nu]_2),\ e([1]_1, [s']_2 - c^{-1}[s]_2) = e([y_\nu]_1, c[s]_2)$$

Update $\mathsf{vk}_1' = [\boldsymbol{x}']_2 \leftarrow ([x_i]_2)_{i \in [\nu-1]}$ and $\mathsf{vk}_2' = [\boldsymbol{y}']_1 \leftarrow ([y_i]_1)_{i \in [\nu-1]}$
- Both compute
$C_1' \leftarrow c^2 C_{1\ell} + C_1 + c^{-2} C_{1r}$, $C_2' \leftarrow c^{-2} C_{2\ell} + C_2 + c^2 C_{2r}$,
$P' = c^{-2} P_\ell + P + c^2 P_r$,
- The reduced statement is $(pp, P', C_1', C_2', [r']_1, \mathsf{ck}_1', \mathsf{vk}_1', [s']_2, \mathsf{ck}_2', \mathsf{vk}_2', g_H, e_H) \in \mathcal{L}_{\mathsf{IPP}}$ with the new witnesses $(\boldsymbol{v}_1', \boldsymbol{v}_2', r_1', r_2', r'_P)$.

Theorem 1. *The protocol presented is a Public Coin, HVSZK, interactive argument of knowledge for the relation $\mathcal{L}_{\mathsf{IPP}}$ with $O(\log n)$ round complexity, $O(n)$ prover complexity, and $O(\log n)$ communication and verification complexity under the SXDH and DPair-ML assumptions.*

The proof for Theorem 1 follows a similar structure to those found in [19,23,30]. Due to page constraints, we refer readers to the full version for detailed elaboration.

Binary Proofs with Logarithmic Verifier We consider an argument for a binary vector $\boldsymbol{b} = (b_1, \ldots, b_n) \in \{0,1\}^n$ with Hamming weight t. This binary vector is equivalent to a vector $\boldsymbol{v} \in \mathbb{G}_1^n$ where each element is either $[0]_1$ or $[1]_1$, and the number of $[1]_1$ is precisely t. Formally, we define a language:

$$\begin{aligned}(C, [r]_2, \mathsf{ck}_2, \mathsf{vk}_2, e_H, t) \in \mathcal{L}_{\mathsf{Bin}} \Leftrightarrow \\ ([r]_2, \mathsf{ck}_2, \mathsf{vk}_2) \in \mathcal{L}^1_{\mathsf{Com}} \wedge \\ \exists \boldsymbol{b} \in \{0,1\}^n, r_C \in \mathbb{Z}_q, s.t.: \\ C = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2 \rangle + r_C \cdot e_H \wedge \langle \mathbf{1}^n, \boldsymbol{b} \rangle = t_j,\end{aligned}$$

$\mathcal{P}$ proves that $\langle \mathbf{1}^n, \boldsymbol{b} \rangle = t \wedge \boldsymbol{b} \circ \boldsymbol{b'} = \mathbf{0}^n \wedge \boldsymbol{b'} = \boldsymbol{b} - \mathbf{1}^n$. Using random $y, \tau \in \mathbb{Z}_q^*$ from $\mathcal{V}$, these constraints can be re-written as:

$$\langle \boldsymbol{b} - \tau \cdot \mathbf{1}^n, \boldsymbol{y}^n \circ (\boldsymbol{b'} + \tau \cdot \mathbf{1}^n + \tau^2 \cdot \mathbf{1}^n) \rangle = \tau^2 \cdot t + \delta(y, \tau)$$

where $\delta(y, \tau) = (\tau - \tau^2) \cdot \langle \mathbf{1}^n, \boldsymbol{y}^n \rangle - \tau^3 \langle \mathbf{1}^n, \mathbf{1}^n \rangle \in \mathbb{Z}_q$. Thus the binary proof can be reduced to one inner pairing product argument. Concretely, $\mathcal{P}$ and $\mathcal{V}$ engage in the following protocol Π_{Bin}:

- On input $\boldsymbol{b} \in \{0,1\}^n$, $\mathcal{P}$ computes:
$\boldsymbol{b'} = \boldsymbol{b} - \mathbf{1}^n$, $[\boldsymbol{b}]_1 \in \mathbb{G}_1^n$ and $[\boldsymbol{b'}]_2 \in \mathbb{G}_2^n$, $r_{B_1}, r_{B_2} \leftarrow_{\$} \mathbb{Z}_q$,
commits to $[\boldsymbol{b}]_1$ and $[\boldsymbol{b'}]_2$:
$C = B_1 = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2 \rangle + r_{B_1} \cdot e_H$, $B_2 = \langle \mathsf{ck}_1, [\boldsymbol{b'}]_2 \rangle + r_{B_2} \cdot e_H$,
chooses blinding vectors and commits them:
$\boldsymbol{u}_1, \boldsymbol{u}_2 \leftarrow_{\$} \mathbb{Z}_q^n$, $r_{U_1}, r_{U_2} \leftarrow_{\$} \mathbb{Z}_q$,
$U_1 = \langle [\boldsymbol{u}_1]_1, \mathsf{ck}_2 \rangle + r_{U_1} \cdot e_H$, $U_2 = \langle \mathsf{ck}_1, [\boldsymbol{u}_2]_2 \rangle + r_{U_2} \cdot e_H$,
sends B_1, B_2, U_1, U_2 to $\mathcal{V}$
- $\mathcal{V}$ sends challenges $y, \tau \leftarrow_{\$} \mathbb{Z}_q^*$ to $\mathcal{P}$
- $\mathcal{P}$ computes $\mathsf{ck}'_1 \leftarrow \mathsf{ck}_1 \circ \boldsymbol{y}^{-n}$,
define the following polynomials:

$$l(X) = \boldsymbol{b} - \tau \cdot \mathbf{1}^n + \boldsymbol{u}_1 \cdot X \in \mathbb{Z}_q^n[X]$$

$$r(X) = \boldsymbol{y}^n \circ (\boldsymbol{b'} + \tau \cdot \mathbf{1}^n + \boldsymbol{u}_2 \cdot X) + \tau^2 \cdot \mathbf{1}^n \in \mathbb{Z}_q^n[X]$$

$$p(X) = \langle l(X), r(X) \rangle = p_0 + p_1 \cdot X + p_2 \cdot X^2 \in \mathbb{Z}_q^n[X]$$

Next, $\mathcal{P}$ needs to convince $\mathcal{V}$ that $p_0 = t \cdot \tau^2 + \delta(y, \tau)$.
- $\mathcal{P}$ chooses $\phi_1, \phi_2 \leftarrow_{\$} \mathbb{Z}_q^*$ and computes:
$P_1 = p_1 \cdot e(g, \tilde{g}) + \phi_1 \cdot e_H$, $P_2 = p_2 \cdot e(g, \tilde{g}) + \phi_2 \cdot e_H$
- $\mathcal{P}$ sends $P_1, P_2 \in \mathbb{G}_T$ to $\mathcal{V}$
- $\mathcal{V}$ sends $x \leftarrow_{\$} \mathbb{Z}_q^*$ to $\mathcal{P}$
- $\mathcal{P}$ computes:
$[\boldsymbol{l}]_1 = [l(x)]_1 = [\boldsymbol{b} - \tau \cdot \mathbf{1}^n + \boldsymbol{u}_1 \cdot x]_1 \in \mathbb{G}_1^n$,
$[\boldsymbol{r}]_2 = [r(x)]_2 = [\boldsymbol{y}^n \circ (\boldsymbol{b'} + \tau \cdot \mathbf{1}^n + \boldsymbol{u}_2 \cdot x) + \tau^2 \cdot \mathbf{1}^n]_2 \in \mathbb{G}_2^n$
$P = \langle [\boldsymbol{l}]_1, [\boldsymbol{r}]_2 \rangle \in \mathbb{G}_T$, $\phi_x = \phi_2 \cdot x^2 + \phi_1 \cdot x \in \mathbb{Z}_q$
$\mu_1 = r_{B_1} + r_{U_1} \cdot x, \mu_2 = r_{B_2} + r_{U_2} \cdot x \in \mathbb{Z}_q$
$\mathcal{P}$ sends $[\boldsymbol{l}]_1, [\boldsymbol{r}]_2, P, \phi_x, \mu_1, \mu_2$ to $\mathcal{V}$

$\mathcal{V}$ computes ck'_1 in the same way and computes:
$Q_1 = B_1 + x \cdot U_1 - \tau \cdot \langle [\mathbf{1}]_1, \mathsf{ck}_2 \rangle$, $Q_2 = B_2 + x \cdot U_2 + \tau \cdot \langle \mathsf{ck}'_1, [\boldsymbol{y}^n]_2 \rangle + \tau^2 \cdot \langle \mathsf{ck}'_1, [\mathbf{1}^n]_2 \rangle$
checks whether

$$P + \phi_x \cdot e_H \stackrel{?}{=} (t \cdot \tau^2 + \delta(y, \tau)) \cdot e(g, \tilde{g}) + x \cdot P_1 + x^2 \cdot P_2$$

$$Q_1 \stackrel{?}{=} \langle [\boldsymbol{l}]_1, \mathsf{ck}_2 \rangle + \mu_1 \cdot e_H, \; Q_2 \stackrel{?}{=} \langle \mathsf{ck}'_1, [\boldsymbol{r}]_2 \rangle + \mu_2 \cdot e_H$$

Note that the communication and verification cost are linear to n. To reduce them, $\mathcal{P}$ does not send $[\boldsymbol{l}]_1, [\boldsymbol{r}]_2$ directly, and $\mathcal{V}$ just computes $\mathsf{vk}'_1 \leftarrow \mathsf{vk}_1 \circ \bar{\boldsymbol{y}}$, where $\bar{\boldsymbol{y}} = (1, y^{-1}, \cdots, y^{-2^{\nu-1}})$, rather than ck'_1. To make sure $\mathcal{V}$ still can compute Q_2, $\mathcal{P}$ computes $Y = \langle \mathsf{ck}'_1, [\boldsymbol{y}^n]_2 \rangle = \langle \mathsf{ck}_1, [\mathbf{1}^n]_2 \rangle$ and $\Gamma = \langle \mathsf{ck}'_1, [\mathbf{1}^n]_2 \rangle$.

$\mathcal{P}$ sends Γ to $\mathcal{V}$ and proves its correctness as follows:
Let $\Gamma_\nu = \Gamma$, $\mathsf{ck}'_{1\nu} = \mathsf{ck}'_1$, for $i = \nu - 1$ to 1:

$\mathcal{P}$ sends $M_i = \mathsf{ck}'^{\mathbf{1}^{2^i}}_{1i} \in \mathbb{G}_1$, where $\mathsf{ck}'_{1i} \in \mathbb{G}_1^{2^i}$ is the left half of $\mathsf{ck}'_{1(i+1)}$;
$\mathcal{V}$ aborts if $e(M_i, ([1]_2 + \mathsf{vk}'_{1(i+1)})) \neq \Gamma_{i+1}$,
where $\mathsf{vk}'_{1(i+1)}$ is the $i+1$-th element of vk'_1,
otherwise let $\Gamma_i = e(M_i, [1]_2)$ and continue;

After $\nu - 1$ steps without abort, $\mathcal{V}$ can be convinced that Γ is correct:
$\Gamma = \langle \mathsf{ck}'_1, [\mathbf{1}^n]_2 \rangle$ and its computation time is $O(\nu) = O(\log n)$.
Thus, $\mathcal{V}$ can compute $Q_2 = B_2 + x \cdot U_2 + \tau \cdot Y + \tau^2 \cdot \Gamma$ in $O(\log n)$ time.
Thereafter, $\mathcal{P}$ runs the inner pairing product argument protocol Π_{IPP} with $\mathcal{V}$: $\mathcal{L}_{\mathsf{IPP}}(pp, P, Q_1, Q_2, [r]_1, \mathsf{ck}'_1, \mathsf{vk}'_1, [s]_2, \mathsf{ck}_2, \mathsf{vk}_2)$

Theorem 2. *The binary proof has perfect completeness, HVSZK and computational witness extended emulation under the SXDH and DPair-ML assumptions.*

Proof. The binary proof is a special case of the aggregated binary proof in Theorem 3 with $k = 1$. It can be regarded as a corollary of Theorem 3.

Aggregated Binary Proofs The prover is similar to the prover for a binary proof with $k \cdot n$ bits except the following modifications. Without loss of generality, we assume that $k \cdot n$ is still a power of 2. It proves that the committed values are $k \cdot n$ bits and they are the concatenation of k blocks. The number of 1's in the j-th block is t_j: $\boldsymbol{b} = (\boldsymbol{b}_1 || \cdots || \boldsymbol{b}_k)$ for all $j \in [k]$ where $\boldsymbol{b}_j \in \{0,1\}^n$ and $\langle \mathbf{1}^n, \boldsymbol{b}_j \rangle = t_j$. Formally, we define a language:

$$\begin{aligned}(C, [r]_2, \mathsf{ck}_2, \mathsf{vk}_2, e_H, \{t_j\}_{j \in [k]}) \in \mathcal{L}_{\mathsf{aBin}} \Leftrightarrow \\ ([r]_2, \mathsf{ck}_2, \mathsf{vk}_2) \in \mathcal{L}^1_{\mathsf{Com}} \wedge \\ \exists \boldsymbol{b} \in \{0,1\}^n, r_C \in \mathbb{Z}_q, s.t.: \\ C = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2 \rangle + r_C \cdot e_H \wedge \langle \mathbf{1}^n, \boldsymbol{b}_j \rangle = t_j, \forall j \in [k].\end{aligned}$$

We convert $\boldsymbol{b}$ to $[\boldsymbol{b}]_1 \in \mathbb{G}_1^{kn}$, and commit it into $B = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2 \rangle + r_B \cdot e_H$ where $r_B \leftarrow\!\!\$\, \mathbb{Z}_q$ and all t_j are public. The protocol in the former section is modified as follows:

$$l(X) = \boldsymbol{b} - \tau \cdot \mathbf{1}^{k \cdot n} + \boldsymbol{u}_1 \cdot X \in \mathbb{Z}_q^{k \cdot n}[X]$$

$$r(X) = \boldsymbol{y}^{k\cdot n} \circ (\boldsymbol{b'} + \tau \cdot \mathbf{1}^{k\cdot n} + \boldsymbol{u}_2 \cdot X) + \sum_{j=1}^{k} \tau^{j+1} \cdot (\mathbf{0}^{(j-1)\cdot n} || \mathbf{1}^n || \mathbf{0}^{(k-j)\cdot n}) \in \mathbb{Z}_q^{k\cdot n}[X]$$

$\delta(y,\tau) = (\tau - \tau^2) \cdot \langle \mathbf{1}^{k\cdot n}, \boldsymbol{y}^{k\cdot n} \rangle - \sum_{j=1}^{k} \tau^{j+2} \cdot \langle \mathbf{1}^n, \mathbf{1}^n \rangle$
The verification check needs to include each t_j:

$$P + \phi_x \cdot e_H = ((\sum_{j=1}^{k} (t_j \cdot \tau^{j+1}) + \delta(y,\tau)) \cdot e(g,\tilde{g}) + x \cdot P_1 + x^2 \cdot P_2$$

Q_2 needs to be updated to

$$Q_2 = B_2 + x \cdot U_2 + \tau \cdot Y + \sum_{j=1}^{k} \tau^{(j+1)} \cdot \langle \mathsf{ck}'_{1j}, [\mathbf{1}^n]_2 \rangle$$

where ck'_{1j} consists of the $((j-1)\cdot n+1)$-th element to the $(j\cdot n)$-th element of ck'_1.

Theorem 3. *The aggregated binary proof has perfect completeness, HVSZK and computational witness extended emulation under the SXDH and DPair-ML assumptions.*

The proof is analogous to that of the Range proof in [19], Appendix C. Due to page limitations, we defer the formal proof to the full version.

4.3 Efficient Construction from BLS Signatures

- **Setup**(1^λ) In this phase, the system parameters are generated, especially, the common reference string. On input the security parameter 1^λ, it produces the public parameters for the BLS scheme $pp_{bls} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot), H_1(\cdot)\}$. Here $\mathbb{G}_1, \mathbb{G}_2$ are asymmetric groups, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the Type III bilinear pairing operation, and $H_1 : \{0,1\}^* \to \mathbb{G}_1$ is the hash function. $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ is another hash function. Let k be the number of potential messages, n be the maximum number of users. W.l.o.g., we assume they are power of 2. The setup algorithm additionally outputs the structured common reference string $\mathsf{crs} = (pp_{\mathsf{com}}, [r]_1, \mathsf{ck}_1, \mathsf{vk}_1, [s]_2, \mathsf{ck}_2, \mathsf{vk}_2)$ as we described in Sec. 2.2. Note that $\mathsf{ck}_1 = (\mathsf{ck}_1^1 || \cdots || \mathsf{ck}_1^j || \cdots || \mathsf{ck}_1^k || \cdots || \mathsf{ck}_1^{2k}) \in \mathbb{G}_1^{2kn}$, ck_1^j denotes the $((j-1)n+1)$-th element to the jn-th element of ck_1. $\mathsf{ck}_2 = (\mathsf{ck}_2^1 || \cdots || \mathsf{ck}_2^k || \cdots || \mathsf{ck}_2^{2k}) \in \mathbb{G}_2^{2kn}$. Especially, let $\mathsf{ck}'_1 = (\mathsf{ck}_1^1 || \cdots || \mathsf{ck}_1^k) \in \mathbb{G}_1^{kn}$, $\mathsf{ck}'_2 = (\mathsf{ck}_2^1 || \cdots || \mathsf{ck}_2^k) \in \mathbb{G}_2^{kn}$ and we use $\mathsf{ck}_1^*, \mathsf{ck}_2^*$ to denote the first element of $\mathsf{ck}_1^{k+1}, \mathsf{ck}_2^{k+1}$ respectively. Output $pp = (pp_{bls}, \mathsf{crs}, H_2)$.
  Each user generates his secret key $sk_i \leftarrow\!\!\$\, \mathbb{Z}_q$ and public key $pk_i = [sk_i]_2 \in \mathbb{G}_2$ and broadcasts the public key with proof of knowledge of secret key.
  The combiner or any other parties collects the public keys $\boldsymbol{pk} = (pk_1, ..., pk_n)$ and publishes the commitments of them as $K_1 = \langle \mathsf{ck}_1^1, \boldsymbol{pk} \rangle, ..., K_k = \langle \mathsf{ck}_1^k, \boldsymbol{pk} \rangle$. It also publish the aggregation key $ak = (pp, \boldsymbol{pk})$ and the verification key $vk = (pp_{\mathsf{com}}, pp_{bls}, [r]_1, \mathsf{vk}_1, [s]_2, \mathsf{vk}_2, K_1, ..., K_k)$

- **ParSign**(m_j, sk_i) For a message m_j chosen from the message space $\mathcal{M}$, the user u_i signs on it using his secret key sk_i and sends (pk_i, m_j, σ_{ij}) to the combiner where $\sigma_{ij} = H(m_j)^{sk_i}$.
- **ParVrfy**(pk_i, m_j, σ_{ij}) On receiving (pk_i, m_j, σ_{ij}), the combiner verifies it. Output 1 if $e(H(m_j), pk_i) = e(\sigma_{ij}, \tilde{g})$, otherwise output 0.
- **Combine**$(ak, \{pk_i, m_j, \sigma_{ij}\}_{j\in[k]})$ When collecting a set of $\{pk_i, m_j, \sigma_i\}$, the combiner verifies them one by one. If all of them are valid, the combiner does as follows:
  - Let $S_j \subseteq [n]$ be the indices of signers who have signed on m_j, set $\boldsymbol{b}_j = (b_{1j}, ..., b_{nj}) \in \{0,1\}^n$, such that $b_{ij} = 1$ if $i \in S_j$, otherwise, $b_{ij} = 0$ and $t_j = \sum_{i=1}^{n} b_{ij}$;
  - Let $\boldsymbol{b} = (\boldsymbol{b}_1 || \cdots || \boldsymbol{b}_k)$, compute the commitment to $[\boldsymbol{b}]_1 \in \mathbb{G}_1^{kn}$: $B = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2' \rangle + r_B \cdot e_H$ where $r_B \leftarrow_\$ \mathbb{Z}_q$ and generate the binary proof π_{aBin} from Π_{aBin} w.r.t. the language $\mathcal{L}_{\mathsf{aBin}}(C, [r]_2, \mathsf{ck}_2', \mathsf{vk}_2', e_H, \{t_j\}_{j\in[k]})$;
  - For $j = 1$ to k, $m_j \in M$, choose $r_j \leftarrow_\$ \mathbb{Z}_q$, compute sub-aggregated public keys $\widehat{pk}_j = \Pi_{i\in S_j} pk_i \cdot \tilde{g}^{r_j} = \Pi_{i=1}^{n} pk_i^{b_{ij}} \cdot \tilde{g}^{r_j} = \boldsymbol{pk}^{\boldsymbol{b}_j} \cdot \tilde{g}^{r_j}$ and sub-aggregated signatures $\hat{\sigma}_j = \Pi_{i\in S_j} \sigma_i \cdot H(m_j)^{r_j}$;
  - For the k sub-aggregated public keys $\widehat{pk}_j$, compute $\widehat{PK} = \Pi_{j=1}^{k} \widehat{pk}_j^{z^{(j-1)}} = \boldsymbol{pk}^{\boldsymbol{b}_1} \cdot \boldsymbol{pk}^{z\boldsymbol{b}_2} \cdots \boldsymbol{pk}^{z^{k-1}\boldsymbol{b}_k} \cdot \tilde{g}^{r^*} = (\boldsymbol{pk} || \boldsymbol{pk}^z || \cdots || \boldsymbol{pk}^{z^{k-1}})^{(\boldsymbol{b}_1 || \cdots || \boldsymbol{b}_k)} \cdot \tilde{g}^{r^*}$, where $z = H_2(\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, B, \pi_{\mathsf{aBin}})$ and $r^* = \sum_{j=1}^{k} r_j \cdot z^{j-1}$.
  - Compute $X = r^* \cdot e(g, \mathsf{ck}_2^*) + r_X \cdot e_H$ and generate π_{pok} (via the Schnorr's protocol [36]) to prove the knowledge of r^* and r_X;
  - Compute $Q = B + X = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2' \rangle + e(g^{r^*}, \mathsf{ck}_2^*) + (r_B + r_X) \cdot e_H$, it means $Q = \langle ([\boldsymbol{b}]_1, g^{r^*}), (\mathsf{ck}_2', \mathsf{ck}_2^*) \rangle + (r_B + r_X) \cdot e_H \in \mathbb{G}_T$, so we know Q is the commitment of a binary vector and a randomness;
  - Compute $K = \langle (\mathsf{ck}_1', \mathsf{ck}_1^*), (\boldsymbol{pk}^{\boldsymbol{z}^k}, \tilde{g}) \rangle$, $E = \langle [\boldsymbol{b}]_1, \boldsymbol{pk}^{\boldsymbol{z}^k} \rangle + r^* \cdot e(g, \tilde{g}) \in \mathbb{G}_T$, where $\boldsymbol{pk}^{\boldsymbol{z}^k} = (\boldsymbol{pk} || \boldsymbol{pk}^z || ... || \boldsymbol{pk}^{z^{k-1}}) \in \mathbb{G}_2^{kn}$;
  - Based on E, Q, K, generates π_{IPP} from Π_{IPP} w.r.t. the language $\mathcal{L}_{\mathsf{IPP}}(E, Q, K, \mathsf{ck}_1, \mathsf{vk}_1, \mathsf{ck}_2, \mathsf{vk}_2, e_H)$ to prove that E is the *inner pairing product* of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ which are committed in Q and K: $E = \langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle + r_E \cdot e_H$, where $\boldsymbol{v}_1 = (g^{b_1}, ..., g^{b_{kn}}, g^{r^*}, [0]_1, ..., [0]_1) \in \mathbb{G}_1^{2kn}$ and $b_i \in \{0,1\}$, $r^* \in \mathbb{Z}_q$ which has been proved via π_{aBin} and π_{pok}, $\boldsymbol{v}_2 = (\boldsymbol{pk}^{\boldsymbol{z}^k}, \tilde{g}, [0]_2, ..., [0]_2) \in \mathbb{G}_2^{2kn}$ which is public.[4]
  - Generate the proof π_{disj} to prove all signer sets are disjoint: $S_{j_0} \cap S_{j_1} = \emptyset, \forall j_0, j_1 \in [k], j_0 \neq j_1$. It requires the combiner additionally prove that $\widehat{PK}' = \prod_{j=1}^{k} \widehat{pk}_j$ can be expressed in the form of $\boldsymbol{pk}^{\boldsymbol{b}'} \cdot \tilde{g}^{r'}$ using the same method as above, where $r' = \sum_{j=1}^{k} r_j$, $\boldsymbol{b}' = \sum_{j=1}^{k} \boldsymbol{b}_j$ is also a binary vector with $\sum_{j\in[k]} t_j$ ones.
  - Output the signature $\Sigma \leftarrow (\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, B, z, X, \pi_{\mathsf{aBin}}, \pi_{\mathsf{pok}}, \pi_{\mathsf{IPP}}, \pi_{\mathsf{disj}})$ with the thresholds $T = \{t_j\}_{j\in[k]}$ as Δ.

[4] We pad 'zeros' in $\boldsymbol{v}_1, \boldsymbol{v}_2$ since the dimension of commitment keys for $\mathcal{L}_{\mathsf{IPP}}$ is $2kn$, which is a power of 2.

- **Verify**(vk, M, T, Σ) Parse $\Sigma = (\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, B, z, X, \pi_{\mathsf{aBin}}, \pi_{\mathsf{pok}}, \pi_{\mathsf{IPP}}, \pi_{\mathsf{disj}})$, $T = \{t_j\}_{j\in[k]}$, compute:
  $z' = H_2(\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, B, \pi_{\mathsf{aBin}})$, $\widehat{\sigma} = \prod_{j\in[k]} \widehat{\sigma}_j$, $Q = B + X$,
  $K = K_1 + \cdots + K_j^{z'^{j-1}} + \cdots + K_k^{z'^{k-1}} + e(\mathsf{ck}_1^*, \tilde{g})$,
  $\widehat{PK} = \prod_{j=1}^{k} \widehat{pk}_j^{z'^{(j-1)}}$, $E = e(g, \widehat{PK})$, $\widehat{PK}' = \prod_{j=1}^{k} \widehat{pk}_j$, $\hat{t} = \sum_{j\in[k]} t_j$.
  Accept if all the following conditions are satisfied:
  - $z' = z$;
  - $\mathbf{Vrfy}(\{\widehat{pk}_j, m_j\}_{j\in[k]}, \hat{\sigma}) = 1$;
  - π_{aBin} is valid w.r.t. B, T;
  - π_{pok} is valid w.r.t. X;
  - π_{IPP} is valid w.r.t. Q, K, E;
  - π_{disj} are valid w.r.t. $\widehat{PK}', \hat{t}$.

General Predicate Satisfiability Proofs In our specific setting, where we aim to prove the predicate regarding the relations among signer sets, we focus on the committed binary vector. It serves as the representation of signers for different messages and plays a crucial role in our proof construction.

Recall that Daza et al. [23] proves that the committed vectors satisfy some circuits. By following the protocol of zero-knowledge SNARK for circuit satisfiability in [23] but with AFGHO commitment (which is also homomorphic same as the Pedersen commitment), we can obtain a circuit satisfiability proof that is compatible with our previous inner pairing product and binary proofs. Both communication cost and verification complexity are logarithmic to the size of the circuit.

5 Analysis

5.1 Performance Analysis

We first analyze the performance related to the inner pairing product and binary proofs. The inner pairing product and binary proof protocols require $\nu = \log(k{\cdot}n)$ rounds, where each round involves communication and computations. In each round, the size of the witness is halved. It leads to a communication complexity of $O(\nu)$ since the communication cost is constant in each round. The prover's computation complexity in round i is $O(2^{\nu-i+1})$. As a result, the overall prover complexity is $O(2^{\nu})$ since ν rounds are performed. On the other hand, the verifier's computation cost remains constant at $O(1)$ since it only needs to perform a fixed number of operations in each round. Consequently, the verifier's overall complexity is $O(\nu)$. For the predicate satisfaction proof with generic arithmetic circuit, the communication cost and verification complexity are logarithmic to the size of the circuit, which is $O(\log|\mathcal{C}|)$. The prover's computation complexity is $O(|\mathcal{C}|)$.

In the Combine phase, the combiner needs to verify the partial signatures, generate the sub-aggregated public keys and sub-aggregated signatures and the

computation cost is $O(n)$. The final signature contains additional aggregated public keys and signatures and thresholds w.r.t. each message. It leads to $O(k)$ communication cost. On the final signature, the verifier checks the sub-aggregated signatures with the respective message and sub-aggregated public key. It involves the $O(k)$ computation complexity. Then it verifies the correctness of these sub-aggregated public keys by checking these proofs.

In summary, the communication cost is $O(k + \nu + \log|\mathcal{C}|) = O(k + \log k + \log n + \log|\mathcal{C}|)$, the prover complexity is $O(2^{\nu} + |\mathcal{C}|) = O(k \cdot n + |\mathcal{C}|)$, and the verifier complexity is $O(\nu + \log|\mathcal{C}|) = O(\log k + \log n + \log|\mathcal{C}|)$. For the special case of rate-once policy, the communication cost is $O(k+\log k+\log n)$, the prover complexity is $O(k \cdot n)$, and the verifier complexity is $O(k + \log k + \log n)$.

5.2 Security Analysis

Theorem 4 (Anonymity). *The predicate aggregate signature is anonymous in the random oracle model under the SXDH assumption.*

Proof. Based on the rate-once policy in the anonymous reputation system, the predicate has been explained in Equation (1) and the description Δ is defined as $\Delta = T = \{t_j\}_{j\in[k]}$. In the anonymous experiment, $\Delta_0 = T_0, \Delta_1 = T_1$ and it is required that $T_0 = T_1$ for the challenge sets S^0, S^1. Given the signature $\Sigma = (\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, B, z, X, \pi_{\mathsf{aBin}}, \pi_{\mathsf{pok}}, \pi_{\mathsf{IPP}}, \pi_{\mathsf{disj}})$, and thresholds $T = \{t_j\}_{j\in[k]}$. Note that T, K leak nothing since they are the same in both challenges with identity sets S^0, S^1. We design the hybrid games as follows:

- $\mathsf{G}_{\mathsf{real}}$: This game is the same as the experiment, challenger chooses $b \leftarrow_{\$} \{0,1\}$ and generates the signature Σ_b honestly under the identities in S^b.
- G_1 : This game is similar with G_0 except that the proofs $\pi_{\mathsf{aBin}}, \pi_{\mathsf{pok}}, \pi_{\mathsf{IPP}}, \pi_{\mathsf{disj}}$ are simulated without witness.
- G_2 : This game is similar with G_1 except that the sub-aggregated public keys and signatures are generated randomly without using b as follows: for $j \in [k]$, choose $u_j \leftarrow_{\$} \mathbb{Z}_q$ and set $\widehat{pk}'_j = \tilde{g}^{u_j} \in \mathbb{G}_2$ and $\hat{\sigma}'_j = H(m_j)^{u_j} \in \mathbb{G}_1$ such that $e(H(m_j), \widehat{pk}'_j) = e(\hat{\sigma}'_j, \tilde{g})$.
- G_3: This game is similar with G_2 except that B, X are also chosen randomly independently: $B', X' \leftarrow_{\$} \mathbb{G}_T$. Note that the proofs $\pi_{\mathsf{aBin}}, \pi_{\mathsf{pok}}, \pi_{\mathsf{IPP}}, \pi_{\mathsf{disj}}$ are simulated without using witnesses. They can still be verified.

Compare G_1 with $\mathsf{G}_{\mathsf{real}}$, the only difference is that these proofs are simulated. Since these proofs are zero-knowledge under the SXDH assumptions in the random oracle model, the probability of distinguishing G_1 from $\mathsf{G}_{\mathsf{real}}$ is negligible. We have that $|\Pr[\mathsf{G}_{\mathrm{real}}(\mathcal{A}, \lambda) = 1] - \Pr[\mathsf{G}_1(\mathcal{A}, \lambda) = 1]| \leq negl(\lambda)$.

Compare G_2 with G_1, in G_1, $\widehat{pk}_j = \Pi_{i\in S_j^b} pk_i \cdot \tilde{g}^{r_j}$, $\widehat{\sigma}_j = \Pi_{i\in S^b} \sigma_i \cdot H(m_j)^{r_j}$ where $r_j \leftarrow_{\$} \mathbb{Z}_q$, so they are uniformly random and each pair satisfies $e(H(m_j), \widehat{pk}_j) = e(\hat{\sigma}_j, \tilde{g})$. In G_1, $\widehat{pk}'_j$ and $\hat{\sigma}'_j$ are also random elements in $\mathbb{G}_1, \mathbb{G}_2$ respectively, and satisfy the same kinds of relation. $\widehat{PK}'$ is generated from these random $\widehat{pk}'_j$ and

z, so is E'. Thus $(\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, z, \widehat{pk}, E)$ and $(\{\widehat{pk}'_j, \widehat{\sigma}'_j\}_{j\in[k]}, z', \widehat{PK}', E')$ have the same distribution. The probability that they can be distinguished is 0. We have that $|\Pr[\mathsf{G}_1(\mathcal{A},\lambda)=1] - \Pr[\mathsf{G}_2(\mathcal{A},\lambda)=1]| = 0$.

Compare G_3 with G_2, in G_2, B is the structured AFGHO commitment of $[\boldsymbol{b}]_1$, X is the Pedersen commitment of r^*. Since these commitments are perfect hiding, they are indistinguishable from the random B', X' in G_3. We have that $|\Pr[\mathsf{G}_2(\mathcal{A},\lambda)=1] - \Pr[\mathsf{G}_3(\mathcal{A},\lambda)=1]| = 0$.

In G_3, $\mathcal{A}$'s view is independent of b. Thus, $\mathcal{A}$ just outputs a random guess $\hat{b}$ in G_3, so its advantage is 0: $\Pr[\mathsf{G}_3(\mathcal{A},\lambda)=1] - 1/2 = 0$. In summary, we have

$$\begin{aligned}
&|\Pr[\mathrm{Exp}^{\mathsf{anony}}(\mathcal{A},\lambda)=1] - 1/2| = |\Pr[\mathsf{G}_{\mathrm{real}}(\mathcal{A},\lambda)=1] - 1/2| \\
\leq &|\Pr[\mathsf{G}_{\mathrm{real}}(\mathcal{A},\lambda)=1] - \Pr[\mathsf{G}_1(\mathcal{A},\lambda)=1]| + |\Pr[\mathsf{G}_1(\mathcal{A},\lambda)=1] - \Pr[\mathsf{G}_2(\mathcal{A},\lambda)=1]| \\
&+ |\Pr[\mathsf{G}_2(\mathcal{A},\lambda)=1] - \Pr[\mathsf{G}_3(\mathcal{A},\lambda)=1]| + |\Pr[\mathsf{G}_3(\mathcal{A},\lambda)=1] - 1/2| \leq negl(\lambda)
\end{aligned}$$

Theorem 5 (Unforgeability). *The predicate aggregate signature is unforgeable in the random oracle model under the co-CDH, SXDH and DPair-ML assumptions.*

Proof. First of all, we assume that the proof of possession has been done to prove the knowledge of secret key for each public key. Based on the rate-once policy in the anonymous reputation system, the predicate has been explained as in Equation (1). So in the unforgeability experiment, the adversary $\mathcal{A}$ wins if for the extracted identities sets $S_1, ..., S_k$, at least one of the following happens:

1. $\exists\ i_j \in S_j$, such that $\mathcal{A}$ has never queried the **corrupt** oracle on pk_{i_j} or **sign** oracle on (pk_{i_j}, m_j);
2. $\exists\ j \in [k]$, s.t. $t_j \neq |S_j|$;
3. $\exists\ i_j \in S_j$ and it appears more than once in S_j;
4. $\exists\ S_i$ and S_j which overlap: $S_i \cap S_j \neq \emptyset$.

We reduce the security of our scheme to the security of the underlying BLS signature which is unforgeable under co-CDH assumption, and non-interactive ZKAoK which is sound and knowledge sound under SXDH and DPair-ML assumptions. We elaborate it case by case.

Extract the identities and randomness: $\mathcal{A}$ outputs a non-trivial PAS forgery $\Sigma = (\{\widehat{pk}_j, \widehat{\sigma}_j\}_{j\in[k]}, B, z, X, \pi_{\mathsf{aBin}}, \pi_{\mathsf{pok}}, \pi_{\mathsf{IPP}}, \pi_{\mathsf{disj}})$ on message set $M = \{m_j\}_{j=1}^k$ with threshold $T = \{t_j\}_{j=1}^k$. Due to the witness extended emulation of $\Pi_{\mathsf{aBin}}, \Pi_{\mathsf{IPP}}$ and the knowledge soundness of Π_{pok}, there exists an extractor $\mathcal{E}$ who can extract the signer identities $\boldsymbol{b}_j$, r_j such that $\widehat{pk}_j = \boldsymbol{pk}^{\boldsymbol{b}_j} \cdot \tilde{g}^{r_j}$ as follows.

$\mathcal{E}$ can run the extractor χ_{aBin} for π_{aBin} to extract the committed elements $[\boldsymbol{b}]_1 \in \mathbb{G}_1^{kn}$ and randomness $r_B \in \mathbb{Z}_q$ s.t. $B = \langle [\boldsymbol{b}]_1, \mathsf{ck}_2 \rangle + r_B \cdot e_H$ and $\boldsymbol{b} \in \{0,1\}^{kn}$. $\mathcal{E}$ can rewind $\mathcal{A}$ on different z. For each z, $\mathcal{E}$ runs the extractor χ_{pok} for π_{pok} to extract the committed element $r^* \in \mathbb{Z}_q$ and randomness $r_X \in \mathbb{Z}_q$ s.t. $X = r^* \cdot e(g, \mathsf{ck}_2^*) + r_X \cdot e_H$ and runs the extractor χ_{IPP} for π_{IPP} to extract the committed elements $\boldsymbol{v}_1 \in \mathbb{G}_1^{2kn}, \boldsymbol{v}_2 \in \mathbb{G}_2^{2kn}$ and randomness $r_Q, r_K, r_E \in \mathbb{Z}_q$ s.t. $E = \langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle + r_E \cdot e_H$, $Q = \langle \boldsymbol{v}_1, (\mathsf{ck}_2, \mathsf{ck}_2^*) \rangle + r_Q \cdot e_H$, $K = \langle (\mathsf{ck}_1, \mathsf{ck}_1^*), \boldsymbol{v}_2 \rangle +$

$r_K \cdot e_H$. We know that $\boldsymbol{v}_1 = ([\boldsymbol{b}]_1, [r^*]_1, [0]_1, ..., [0]_1)$, $\boldsymbol{v}_2 = (\boldsymbol{pk}^{\boldsymbol{z}^k}, \tilde{g}, [0]_2, ..., [0]_2)$. Otherwise, it breaks the DPair-ML assumption. So $E = \langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle + r_E \cdot e_H = e(g, \widehat{G}) + e(g_H, \tilde{g}^{r_E})$, where $\widehat{G} = \boldsymbol{pk}^{\boldsymbol{z}^k \boldsymbol{b}} \cdot \tilde{g}^{r^*}$. We also have $\widehat{PK} = \prod_{j=1}^{k} \widehat{pk}_j^{z^{(j-1)}} \in \mathbb{G}_2$ s.t. $E = e(g, \widehat{PK})$. It means that $\widehat{PK} = \widehat{G}$ and $r_E = 0$, otherwise, it breaks the DPair assumption by finding a non-trivial pair $(N_1, N_2) = (\widehat{G}/\widehat{PK}, \tilde{g}^{r_e}) \in \mathbb{G}_2^2$ s.t. $e(g, N_1) + e(g_H, N_2) = [0]_T$. Thus $\widehat{PK}$ can be expressed in the form of $\boldsymbol{pk}^{\boldsymbol{z}^k \boldsymbol{b}} \cdot \tilde{g}^{r^*}$.

Repeating this for k different challenges z with the randomness r^*, we can compute r_j for $j \in [k]$ s.t. $r^* = \sum_{j=1}^{k} r_j \cdot z^{j-1}$ for each challenge and each sub-aggregated public key $\widehat{pk}_j$ is in the form of $\boldsymbol{pk}^{\boldsymbol{b}_j} \cdot \tilde{g}^{r_j}$ by Schwartz-Zippel lemma, where $\boldsymbol{b}_j$ is the j-th block in the extracted $\boldsymbol{b}$.

(1) Suppose that P_1=Pr[$\mathcal{A}$ wins and violates condition 1] is non-negligible. We prove that if $\mathcal{A}$ wins, we can use it in a black-box manner to construct an attacker $\mathcal{B}$ to break the unforgeability of the underlying BLS signature. $\mathcal{B}$ receives the BLS parameter pp_{bls} and the target BLS public key pk^*. It can also query the BLS signing oracle $\mathbf{Sign}_{\text{bls}}(\cdot)$ on pk^* and any message. $\mathcal{B}$ can emulate the experiment for $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ generates crs and sets $pp = (pp_{bls}, \mathsf{crs})$. $\mathcal{B}$ chooses an index $\hat{i} \leftarrow_{\$} [n]$ and sets $pk_{\hat{i}} = pk^*$. For other $i \in [n], i \neq \hat{i}$, it generates the secret keys $sk_i \leftarrow_{\$} \mathbb{Z}_q$ and sets public keys $pk_i = [sk_i]_2 \in \mathbb{G}_2$. $\mathcal{B}$ sends pp and all public keys to $\mathcal{A}$.

Emulate Corrupt and Sign oracles: For corruption oracle $\mathbf{corrupt}(\cdot)$: if $\mathcal{A}$ corrupts $pk_{\hat{i}}$, $\mathcal{B}$ aborts. Otherwise, for other identity corruption, $\mathcal{B}$ responds with the secret key sk_i. Note that $\mathcal{A}$is not allowed to corrupt all public keys.

For signing oracle $\mathbf{sign}(\cdot, \cdot)$: if $\mathcal{A}$ queries on $(pk_{\hat{i}}, m)$, $\mathcal{B}$ forwards m to its $\mathbf{Sign}_{bls}$ oracle and replies $\mathcal{A}$ with the signature it received. Otherwise, for other signer identities, $\mathcal{B}$ generates the signature on the message using secret key sk_i.

Breaking BLS Unforgeability: $\mathcal{A}$ outputs a non-trivial PAS forgery Σ on message set $M = \{m_j\}_{j=1}^{k}$ with threshold $T = \{t_j\}_{j=1}^{k}$. By the knowledge soundness, $\mathcal{B}$ can works like $\mathcal{E}$ as above to extract the signer identities $\boldsymbol{b}_j$, r_j such that $\widehat{pk}_j = \boldsymbol{pk}^{\boldsymbol{b}_j} \cdot \tilde{g}^{r_j}$. Based on each $\boldsymbol{b}_j$, we obtain the identity subset $S_j = \{i | b_{ji} = 1, b_{ji} \in \boldsymbol{b}_j\}$ for $j = 1$ to k and $S = \cup_{j=1}^{k} S_j$. Note that the non-triviality of the forgery implies that S includes at least one honest signer, who $\mathcal{A}$ did not corrupt. Otherwise, it proceeds as follows. $\mathcal{B}$ aborts if the target identity $\hat{i}$ is not included in S or $\hat{i} \in S_j$ w.r.t. a message m_j but $\mathcal{A}$ has queried **sign** oracle on $(pk_{\hat{i}}, m_j)$. Otherwise, $\mathcal{B}$ can locate the identity subset $S_{\hat{j}}$ w.r.t. the message $m_{\hat{j}}$ in which the target identity $\hat{i} \in S_{\hat{j}}$ and $\widehat{pk}_{\hat{j}} = \boldsymbol{pk}^{\boldsymbol{b}_{\hat{j}}} \cdot \tilde{g}^{r_{\hat{j}}}$. Given $\hat{\sigma}_j$, the randomness r_j and all other identities $i_j \in S_{\hat{j}}, i_j \neq \hat{i}$, $\mathcal{B}$ can computes their signatures σ_{i_j} on the message $m_{\hat{j}}$ and gets $\sigma_{i_j^*} = \hat{\sigma} / (\prod_{i \in S_j^* \setminus \{i_j^*\}} \sigma_i \cdot H(m_{\hat{j}})^{r_j})$ which is a valid signature of the target identity on $m_{\hat{j}}$ that $\mathcal{A}$ has never queried. So it is a successful BLS forgery and $\mathcal{B}$ wins.

Success Probability: Let $\epsilon_{\mathcal{A}}$ be the probability with which $\mathcal{A}$ outputs a valid forgery. It is easy to see that $\mathcal{B}$ breaks the unforgeability of BLS signature if it does not abort. We compute the lower bound of the probability with which $\mathcal{B}$ does not abort. Firstly, since $\hat{i}$ is chosen uniformly at random, $\mathcal{A}$ does not corrupt

$pk_{\hat{i}}$ with probability at least $1/n$. Let δ be the probability with which $\mathcal{B}$ extracts the witness successfully. Then the probability that $\hat{i} \in I$ is at least $1/n$. Let q_H be the number of queries on the random oracle, q_S be the number of queries on the signing oracle, the probability that $\mathcal{A}$ has never queried on $(pk_{\hat{i}}, m_{\hat{j}})$ is at least $(1 - \frac{1}{n \cdot q_H}) \cdot (1 - \frac{1}{n \cdot q_H - 1}) \cdots (1 - \frac{1}{n \cdot q_H - q_S}) = \frac{n \cdot q_H - q_S - 1}{n \cdot q_H}$. Finally, we obtain the success probability of $\mathcal{B}$ is $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}} \cdot \frac{n \cdot q_H - q_S - 1}{n^3 \cdot q_H}$. Due to the unforgeability of the BLS signature, $\epsilon_{\mathcal{B}}$ is negligible, so $\epsilon_{\mathcal{A}}$ is also negligible.
(2) Suppose that P_2=Pr[$\mathcal{A}$ wins and violates condition 2] is non-negligible. It implies that $\mathcal{A}$ generates a valid binary proof for t_j with an incorrect witness whose Hamming weight is not t_j. It contradicts with the statement of π_{aBin} which breaks the soundness of the underlying ZKAoK.
(3) Suppose that P_3=Pr[$\mathcal{A}$ wins and violates condition 3] is non-negligible. It implies that $\mathcal{A}$ generates a valid binary proof with an incorrect witness which contains a number larger than 1. It also contradicts to the statement of π_{aBin}, so the soundness is broken.
(4) Suppose that P_4=Pr[$\mathcal{A}$ wins and violates condition 4] is non-negligible. It implies that $\mathcal{A}$ generates a valid binary proof for the sum of commitments with an incorrect witness which contains a number larger than 1. It contradicts to the statement of π_{disj}.

In summary, $\Pr[\mathrm{Exp}^{\mathsf{unforge}}(\mathcal{A}, \lambda) = 1] = P_1 + P_2 + P_3 + P_4 \leq negl(\lambda)$.

6 Applications and Extensions

Our PAS can be used to construct many other types of signatures by invoking the Combine algorithm as a blackbox to compress the final signature. Letting dynamic threshold be the specific description and predicate function only requires the correctness of the threshold, our efficient scheme for single message also improves their state-of-the-art works in terms of trust model (relies on trusted party or non-standard assumptions) and efficiency as shown in Table 2. We explain them as follows.

(1) Our PAS implies threshold signatures with transparent setup [5] Each signer generates their public key and secret key by themselves. Some of them sign on the same message and send to the combiner. Taking the valid partial signatures as input, the combiner runs the Combine algorithm to generate the final PAS signature with a threshold t. The verifier can be convinced that there are t different signers sign on this message.

(2) We get a multi-signature by letting everyone sign on the same message, aggregation is done via the Combine algorithm and the predicate only requires that the threshold number is correct.

[5] Our dynamic threshold aggregate signature with transparent setup also offers a solution for multiverse threshold signature (MTS) [8]. For any subset of users interested in forming a universe with a specific threshold, the aggregation and verification keys can be computed from their public keys. Then run the Combine algorithm to get a PAS signature with the number of signers.

(3) We get an aggregate signature which hides the signer identities from PAS, where aggregation is done via Combine algorithm and the predicate function is specified according to the concrete rule.

(4) We get a graded signature by letting everyone sign on the same message, aggregation is done via the Combine algorithm and the predicate only requires that the number of signers is correct. It ensures each of them can sign only once without leaking their identities.

(5) We get a threshold ring signature with prefixed threshold t by setting there is a single message and the predicate function always outputs 1 and modifying the verification algorithm a bit. Via the Combine algorithm, a PAS signature is produced. Now besides verifying whether it is valid, the verifier also checks whether the number of signers in PAS is larger than t. If yes, it is a valid threshold ring signature, otherwise not. When t is 1, it is a ring signature.

Anonymous reputation system An anonymous reputation system enables users to rate products they have purchased. The primary security guarantee offered by such systems is privacy, allowing users to write reviews anonymously for any purchased products. However, to prevent abuse or misuse, a *rate-once policy* is implemented. This means that if a user attempts to write multiple reviews for the same product, their reviews will become publicly traceable or linked. It requires the final signature is linear to the number of signer and the verification time is quadratic. Recent works on the anonymous reputation systems [12,11,24] achieve full anonymity at the cost of linear communication cost and quadratic verification complexity.

We consider a relaxed but reasonable setting where a combiner is allowed to know the signers' identities. It can be the shopping website who knows the user accounts when they login. But it cannot manipulate the final result even colludes with some of users. It cannot violate the rate-once policy and cannot generate review on behalf of other honest users. Malicious users and combiner cannot rate more than once or forge any other honest user's signature. The combiner produces a PAS in which the thresholds disclose the reputation states, and both the size and the verification time are logarithmic in the number of all users.

Onchain voting system In more extensive scenarios, there might exist more intricate voting policies that can be defined as conditions based on the identities of the voters. For instance, in blockchain governance, such as Decentralized Autonomous Organizations (DAOs), determinations might be reached by the entire community, provided their accounts possess a significant number of tokens. Certain policies necessitate that voters possess a particular property [3], and it can be denoted by their identity index. For instance, within an organization, senior members are associated with indices smaller than a threshold. The combiner's task is to demonstrate that among the signers, there is at least one whose index is lower than a specified threshold. In our design, relying on the binary vector, the combiner only needs to establish the existence of a single position in the vector where the value is 1 and the position is smaller than a specified threshold.

6.1 Extensions

Dynamic join. New users can seamlessly join the system without causing any disruptions to existing users. The process of joining is transparent and does not have any adverse effects on other users. A new user broadcasts his public key with PoP for registration. On verifying its validness, the combiner updates its aggregation key by adding this public key. Other honest verifier can also update the verification key by including the new public key in the commitment of all public keys. These updates are publicly verifiable and incur only a constant cost.

Weight aggregation. In the PAS scheme, each user can associate themselves with an additional weight. This weight represents their significance or influence within the system. The weight vector $\boldsymbol{w}$ is public where each weight value w_i binds with a public key pk_i. In the Combine algorithm, the combiner additionally computes the sum of weights $W = \langle \boldsymbol{b}, \boldsymbol{w} \rangle$ as the description Δ. As a result, when the PAS generates a signature, it discloses the total weight of the signers involved. So our PAS also supports the weight aggregation like [33,20,22].

Accountability. Our PAS focuses on the privacy of signers, and it can be extended to support the accountability by adding an extra identities encryption layer. This approach bears similarity to the method employed in TAPS [15]. In this extended system, the description pertains to the minimum threshold required for the number of signers. The combiner also encrypts both the count of signers and their identities. The predicate function ensures that they are correctly encrypted under the specified public key, and that the size of the signer set surpasses the minimum threshold.

6.2 Open problems

More efficient scheme. Although our construction achieves logarithmic verification time, the verifier needs to do the pairing operation in each round which is expensive. Is it possible to design a more efficient PAS scheme in which the verifier only needs to preform constant number of pairing operations and logarithmic group operations? One possible direction is studying the technology in Dory [30]. We leave it to the future work.

Multiple layers combination. In the current setting, our primary objectives include ensuring signer anonymity, improving efficiency, and accommodating diverse predicate requirements for the signers of different messages, which are orthogonal to aggregate multiple (already aggregated) signatures and proofs on different messages. Considering the "more layers combination" feature together with our goals presents challenges, and at this stage, it is unclear how to achieve it, which would be a very interesting open question for future study.

Acknowledgments We would like to thank anonymous reviewers of ASIACRYPT 2023 for their insightful feedbacks. We thank Dr. Hanwen Feng for valuable suggestions. This work was supported in part by research awards from Stellar Development Foundation, Ethereum Foundation, Protocol Labs, SOAR Prize and Digital Science Initiative Pilot Project from USYD.

References

1. Quadratic moloch. https://github.com/DemocracyEarth/dao (2019)
2. Quadratic voting in colorado: 2020. https://www.radicalxchange.org/media/blog/quadratic-voting-in-colorado-2020/ (January 2021)
3. Erc721 voting-power based on some property. https://forum.openzeppelin.com/t/erc721-voting-power-based-on-some-property/24550 (February 2022)
4. Daos. https://ethereum.org/en/dao/ (2023)
5. How to delegate votes in the unlock dao. https://unlock-protocol.com/guides/delegation/ (May 2023)
6. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. Journal of Cryptology **29**, 363–421 (2016)
7. Attema, T., Cramer, R., Rambaud, M.: Compressed σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In: Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV. pp. 526–556. Springer (2021)
8. Baird, L., Garg, S., Jain, A., Mukherjee, P., Sinha, R., Wang, M., Zhang, Y.: Threshold signatures in the multiverse. Cryptology ePrint Archive (2023)
9. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 390–399 (2006)
10. Bethencourt, J., Shi, E., Song, D.: Signatures of reputation. In: Financial Cryptography and Data Security: 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers 14. pp. 400–407. Springer (2010)
11. Blömer, J., Bobolz, J., Porzenheim, L.: A generic construction of an anonymous reputation system and instantiations from lattices. Cryptology ePrint Archive (2023)
12. Blömer, J., Juhnke, J., Kolb, C.: Anonymous and publicly linkable reputation systems. In: Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers. pp. 478–488. Springer (2015)
13. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II. pp. 435–464. Springer (2018)
14. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22. pp. 416–432. Springer (2003)
15. Boneh, D., Komlo, C.: Threshold signatures with private accountability. In: Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV. pp. 551–581. Springer (2022)
16. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Advances in

Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 327–357. Springer (2016)
17. Bootle, J., Elkhiyaoui, K., Hesse, J., Manevich, Y.: Dualdory: Logarithmic-verifier linkable ring signatures through preprocessing. In: Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part II. pp. 427–446. Springer (2022)
18. Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings. pp. 465–480. Springer (2002)
19. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short Proofs for Confidential Transactions and More. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 315–334. IEEE, San Francisco, CA (May 2018). https://doi.org/10.1109/SP.2018.00020, `https://ieeexplore.ieee.org/document/8418611/`
20. Chaidos, P., Kiayias, A.: Mithril: Stake-based threshold multisignatures. Cryptology ePrint Archive (2021)
21. Chow, S.S., Ma, J.P., Yuen, T.H.: Scored anonymous credentials. In: International Conference on Applied Cryptography and Network Security. pp. 484–515. Springer (2023)
22. Das, S., Camacho, P., Xiang, Z., Nieto, J., Bunz, B., Ren, L.: Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. Cryptology ePrint Archive (2023)
23. Daza, V., Ràfols, C., Zacharakis, A.: Updateable inner product argument with logarithmic verifier and applications. In: Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part I 23. pp. 527–557. Springer (2020)
24. El Kaafarani, A., Katsumata, S., Solomon, R.: Anonymous reputation systems achieving full dynamicity from lattices. In: Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22. pp. 388–406. Springer (2018)
25. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
26. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38. pp. 33–62. Springer (2018)
27. Garg, S., Jain, A., Mukherjee, P., Sinha, R., Wang, M., Zhang, Y.: hints: Threshold signatures with silent setup. Cryptology ePrint Archive (2023)
28. Kiayias, A., Osmanoglu, M., Tang, Q.: Graded signatures. In: Information Security: 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings 18. pp. 61–80. Springer (2015)
29. Lai, R.W., Tai, R.K., Wong, H.W., Chow, S.S.: Multi-key homomorphic signatures unforgeable under insider corruption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 465–492. Springer (2018)

30. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II. pp. 1–34. Springer (2021)
31. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings 9. pp. 325–335. Springer (2004)
32. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings. pp. 376–392. Springer (2011)
33. Micali, S., Reyzin, L., Vlachos, G., Wahby, R.S., Zeldovich, N.: Compact certificates of collective knowledge. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 626–641. IEEE (2021)
34. Ristenpart, T., Yilek, S.: The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In: Advances in Cryptology-EUROCRYPT 2007: 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007. Proceedings 26. pp. 228–245. Springer (2007)
35. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret: Theory and applications of ring signatures. Essays in memory of Shimon Even **3895**, 164–186 (2006)
36. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Advances in Cryptology—CRYPTO'89 Proceedings 9. pp. 239–252. Springer (1990)
37. Shoup, V.: Practical threshold signatures. In: Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 207–220. Springer (2000)
38. Zhandry, M.: To label, or not to label (in generic groups). In: Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part III. pp. 66–96. Springer (2022)