Efficient Post-Quantum EPID Signatures From VOLE-in-the-Head

Zuming Liu[1] ⋆, Yanhong Xu[1,2] ⋆, Deng Tang[1] ✉, Neng Zeng[3]

[1] School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
[2] Key Laboratory of Intelligent Sensing System and Security (Ministry of Education), Hubei University, Wuhan, Hubei, China
[3] Department of Applied Mathematics, School of Mathematics and Physics, Xi'an Jiaotong-Liverpool University, Suzhou Industrial Park, Jiangsu, China

Abstract. Enhanced Privacy Identification (EPID) is a type of anonymous signature scheme. Due to its decentralized revocation procedure, EPID has been used in real-world systems for hardware enclave attestation and has been standardized in ISO/IEC. Recently, a prominent line of research has been contributing to the design of efficient EPID schemes based on post-quantum assumptions, including lattices and symmetric primitives. We observe that in this line of research, there is still room for improvement. In particular, Boneh et al. (CT-RSA 2019) proposed a new model for EPID signatures for attestation, which utilizes an accumulator to issue group credentials and enables group members to rejoin the system as needed. However, no formal security definitions were provided. In addition, all existing EPID schemes from symmetric primitives utilized the less efficient MPC-in-the-Head proof systems (Ishai et al., STOC 2007), resulting in large signature sizes.

In this work, we present an efficient hash-based EPID signature scheme derived from a recently developed VOLE-in-the-Head proof system (Baum et al., CRYPTO 2023). To this end, we first provide rigorous security requirements for EPID signatures that allow group members to rejoin the system. Next, we construct a hash-based accumulator accompanied by a zero-knowledge argument of knowledge of an accumulated value. Finally, we implement our EPID scheme and analyze its concrete efficiency. The results demonstrate the advantages of our scheme compared to other post-quantum constructions.

Keywords. Hash-based cryptography, EPID, VOLE-in-the-Head, Privacy-Preserving Primitives.

1 Introduction

EPID. The enhanced privacy Identification (EPID), initially introduced by Brickell and Li [11,13], is an anonymous signature scheme that allows a group member to sign messages in the name of the group without revealing its identity.

⋆ This denotes equal contribution.

Compared to other anonymous signatures such as group signature (GS) [15] and Direct anonymous attestation (DAA) [14], EPID enables signature-based revocation. Particularly, a signature can be directly placed on a signature revocation list (SRL), and an EPID signature includes an additional proof demonstrating that none of the signatures on the revocation list were generated using the current signing key. This proof of non-revocation is the core component of EPID systems and largely dominates the complexity.

EPID signatures have received considerable attention since their inception. Brickell and Li [12] proposed a more efficient EPID scheme from bilinear maps. The security of their scheme is based on strong Diffie-Hellman and the decisional Diffie-Hellman assumptions. This scheme [12] has been deployed in hardware enclave attestation (e.g., Intel SGX), and has also been included in ISO/IEC standards [1]. Later, Dall et al. [22] proposed cache attacks on the Intel SGX EPID implementation. Faonio et al. [25] introduced a subversion-resilient EPID scheme, which provides the same functionality and security as the original EPID against potentially subverted hardware. Faonio et al. [25] also designed a practical construction based on the external Diffie-Hellman assumptions in the random oracle model. All the aforementioned schemes assumed that SRL is managed by some trusted entity, such as the group manager or honest verifiers. Sanders and Traoré [42] revisited the security notion of EPID and proposed a scheme supporting malicious revocation, in which a malicious entity generates the revocation lists. They [42] then designed very efficient schemes secure against such powerful adversaries. Sanders [41] further improved the efficiency of EPID by using a proof of non-revocation that is not zero-knowledge but whose combination with other parts of EPID systems yields an anonymous scheme. This line of research has reached high success. Remarkably, the signature size of the scheme [41] achieving 80-bit security is around 1 KB and 50 KB when the size of the signature revocation list is 0 and 1000, respectively. It is worth mentioning that the above constructions are not post-quantum secure.

Post-Quantum EPID. Solutions based on lattices or hash functions have recently appeared in the literature. The first post-quantum EPID schemes were constructed by Boneh et al. [10], and relied only on symmetric primitives. The signature size is around 200 MB and can be reduced to a few megabytes if frequent interaction with the group manager is allowed. However, the impact of the signature revocation list is never made explicit, and no implementation was provided. Shortly after [10], Kassem et al. [33] proposed the first lattice-based EPID scheme from structured lattices. The signature sizes are around 850 MB, and hundreds of seconds are required for signing and verification. Chen et al. [19] improved upon [33] in terms of computation and communication efficiency. Nevertheless, the core component of EPID systems, i.e., the proof of non-revocation, is not described. Recently, Chen et al. [17] improved over [10] by proposing a new hash-based EPID scheme. Their scheme [17] can handle large group sizes (up to 2^{60} group members) by employing F-SPHINCS+ [16,18] to issue group credentials. F-SPHINCS+ [16,18] is modified from SPHINCS+ [9] to have fewer hash evaluations. In particular, F-SPHINCS+ no longer utilizes the

Winternitz one-time signature scheme and largely differs from the standardized SPHINCS+. More recently, Jeudy and Sanders [31] revisited the standard approach of non-revocation and proposed a novel technique to avoid heavy reliance on zero-knowledge proofs in proof of non-revocation. Their signature sizes [31] are the most compact to date. Unfortunately, they leave the implementation of their EPID scheme for future work. Nevertheless, we managed to provide an estimation of their runtimes. As commented by the authors [31], the signer is required to prove the knowledge of a group credential [3,38,30,29], and to generate one Falcon key and $|\mathsf{SRL}|$ Falcon signatures [40]. Here $|\mathsf{SRL}|$ is the cardinality of the revocation list SRL. We thus employ the numbers from [3,40] to estimate the runtime of [31]. However, since the experiments of [3,40] were conducted in different environments, the given estimations are for reference only. In Table 1, we provide a thorough comparison of existing post-quantum EPID constructions.

While existing schemes have made significant progress in terms of functionality, security, and efficiency, the state-of-the-art of EPID is still unsatisfactory. On the one hand, the most efficient lattice-based construction [31] lacks implementation, making its practical value unclear. On the other hand, the EPID constructions from symmetric primitives all utilize the Multi-Party-Computation-in-the-Head (MPCitH) proof systems [28,34], which are less efficient than the recently developed Vector-Oblivious-Linear-Evaluation-in-the-Head (VOLEitH) proof systems [7,6,5]. In this work, we aim to employ the most recent VOLEitH proof systems to construct more efficient EPID signatures from symmetric primitives.

Before introducing our contributions, let us first recall a generic approach adopted by previous EPID constructions. An EPID system consists of three types of entities, a group manager ($\mathcal{GM}$), group members, and verifiers. The $\mathcal{GM}$ is in charge of issuing group credentials/certificates to group members. This is typically accomplished by signing the public keys of group members, and the resulting signatures serve as the group credentials. In the process, group members also obtain secret keys that are only known to themselves. Later, a group member can prove membership to a verifier without revealing its identity by demonstrating the knowledge of the $\mathcal{GM}$'s signature on its public key, of which this member also knows the corresponding secret key. So far this is essentially GS [15,8]. The difference with GS lies in the mechanism for tracing and deterring malicious signing behaviors. Concretely, there is no opening procedure in EPID signatures. Instead, a group member signs a message with respect to a signature revocation list (SRL), and demonstrates that he did not produce any signature in SRL.

When it comes to concrete constructions, different techniques were proposed to achieve better efficiency. For instance, the proof of non-revocation proposed by Jeudy and Sanders [31] consists of generating one Falcon key and $|\mathsf{SRL}|$ Falcon signatures, which is supposed to be more efficient than the above generic method. Boneh et al. [10] proposed a new model that enables group members to rejoin the system and employs an accumulator to arrange group credentials. As a result, their second construction (secure in the new model) eliminates many expensive steps in the membership proof generation, leading to a significant reduction in

Table 1. Comparison of existing post-quantum EPID schemes. The maximum number of group members allowed in the EPID system is given in (). The second construction by Boneh et al. [10] and our scheme allow group members to rejoin the system and employ an accumulator to issue group certificates. The schemes proposed by Chen et al. [17] can handle group sizes up to 2^{60}, no other scheme can compare to it. Reported runtimes of [33,17] are taken from the original publications.

Schemes	Sign	Verify	Sig size	|SRL|	Security	Type
KFM+19 [33] (2^{32})	361 s	114 s	836 MB	0	128	lattice
	371 s	117 s	838 MB	100	128	lattice
	372 s	119 s	845 MB	500	128	lattice
	374 s	121 s	854 MB	1000	128	lattice
JS25 [31] (2^{32}) ⋆⋆	350 ms	150 ms	0.12 MB	0	128	lattice
	360 ms	151 ms	0.17 MB	10	128	lattice
	375 ms	154 ms	0.64 MB	100	128	lattice
	442 ms	168 ms	2.71 MB	500	128	lattice
	526 ms	186 ms	5.31 MB	1000	128	lattice
BEF19-I [10] (2^{30})	-	-	216.82 MB	?	128	symm.
BEF19-II [10] (2^{20})	-	-	3.45 MB	?	128	symm.
CDK+24 [17] (2^{20})	12.9 s	6.3 s	0.83 MB	0	129	symm.
	13.6 s	6.6 s	0.95 MB	10	129	symm.
	19.6 s	9.5 s	1.83 MB	100	129	symm.
	47.2 s	22.6 s	6.03 MB	500	129	symm.
	80.4 s	38.8 s	11.23 MB	1000	129	symm.
Ours (2^{20}) non-optimized implementation	0.7 s	0.5 s	0.22 MB	0	128	symm.
	0.9 s	0.7 s	0.29 MB	10	128	symm.
	3.1 s	2.3 s	0.95 MB	100	128	symm.
	13.7 s	10.4 s	3.88 MB	500	128	symm.
	26.6 s	20.1 s	7.54 MB	1000	128	symm.
Ours (2^{20}) optimized implementation	58 ms	65 ms	0.22 MB	0	128	symm.
	71 ms	75 ms	0.29 MB	10	128	symm.
	101 ms	114 ms	0.95 MB	100	128	symm.
	331 ms	326 ms	3.88 MB	500	128	symm.
	593 ms	587 ms	7.54 MB	1000	128	symm.

⋆⋆ The runtimes are estimated based on the experiments of [3,40].
- No implementation is performed.
? No description of the cardinality of the signature revocation list (SRL) is given.

signature size. However, security requirements are not formally defined for the new model.

Our Contributions. This work focuses on improving the state-of-the-art of EPID signatures. Our contributions are fourfold. First, we provide formal definitions for the new model proposed by Boneh et al. [10]. Particularly, definitions of unforgeability and anonymity of EPID signatures take into account the rejoining of group members and the impact of accumulator. Second, we construct a new Merkle tree accumulator based on the precursor of the standardized AES, namely, the Rijndael block cipher [21], rather than the less studied LowMC cipher [2]. In fact, many LowMC instances were broken (see, e.g., [36,37,43]). We then equip our accumulator with a companion zero-knowledge argument of knowledge of an accumulated value. Based on our accumulator, we present a new EPID signature scheme satisfying the stringent security requirements. Finally, we implement our EPID scheme and analyze its concrete efficiency. Table 1 shows that signature sizes of our scheme are comparable to those of the most efficient lattice-based construction [31], and are $1.5\times - 3.7\times$ more compact than the most efficient hash-based construction [17]. Signing times of our optimized implementation (see Section 4.3 for details) are $135\times - 222\times$ smaller than [17]. Since different configurations were employed, the experimental results might be biased. In particular, our optimized implementation is conducted on a workstation equipped with an AMD EPYC 9754 CPU @2.3 GHz and 512 GB RAM, leveraging AES-NI and OpenMP for parallelization. However, their programs [17] were executed on a laptop with Intel i7-8850H CPU @2.6 GHz and 32 GB RAM with the Ubuntu operating system. In addition, a single core was utilized. To make a (relatively) fair comparison, we provide a non-optimized implementation. This is conducted on a virtual machine configured with 16 GB RAM and an AMD Ryzen 7 4800H CPU running at 3 GHz. The results still demonstrate the advantages of our EPID scheme. Compared to the most efficient lattice-based construction [31], our scheme is advantageous when $|\mathsf{SRL}|$ is small while their scheme [31] scales better for larger $|\mathsf{SRL}|$.

We remark that the efficiency of our scheme stems from the employment of the more efficient VOLEitH proof system and the new EPID model. Specifically, the utilization of the accumulator shifts a substantial portion of tasks away from the zero-knowledge proof. However, this approach imposes a constraint on the group size, as it is impractical for the $\mathcal{GM}$ to manage a Merkle tree of 2^{30} leaves. Despite this limitation, the costs for signers and verifiers are logarithmic in the group size and hence manageable even for groups as large as 2^{30}, as also shown in Table 3 and Table 4. Nevertheless, we are investigating the construction of EPID signatures from the VOLEitH proof systems that can support an arbitrary number of group members.

Organizations. The rest of this paper is arranged as follows. Section 2 describes relevant preliminaries and our new security requirements of EPID signatures for attestation. Section 3 presents our hash-based accumulator and its companion zero-knowledge protocol. Section 4 provides the construction of our EPID signature

scheme, as well as its efficiency analysis. Due to space limit, details of security proofs are deferred to the Appendix.

2 Preliminaries

2.1 Accumulator

We first recall the syntax and security definitions of accumulator following [23].

$\mathsf{ASetup}(1^\lambda)$. Given a security parameter 1^λ, this algorithm outputs public parameters pp.

$\mathsf{AEval}(\mathsf{pp}, \mathcal{X} = \{x_0, \ldots, x_{N-1}\})$. Given the public parameters pp, and a set $\mathcal{X}$ of N values, this algorithm returns an accumulator value Λ.

$\mathsf{AWitCreate}(\mathsf{pp}, \mathcal{X}, \Lambda, x)$. Given the public parameters pp, a set $\mathcal{X}$, the corresponding accumulator value Λ, and a value x, this algorithm returns $\perp$ if $x \notin \mathcal{X}$. Else, this algorithm creates a witness wit_x for the fact that x is accumulated in Λ.

$\mathsf{AVerify}(\mathsf{pp}, \Lambda, x, \mathsf{wit}_x)$. Given pp, an accumulator value Λ, a value x and a witness wit_x, this algorithm outputs 1 if wit_x is a witness for x and outputs 0 otherwise.

An accumulator $\mathsf{Acc} = (\mathsf{ASetup}, \mathsf{AEval}, \mathsf{AWitCreate}, \mathsf{AVerify})$ is said to be correct if for all $\mathsf{pp} \leftarrow \mathsf{ASetup}(1^\lambda)$, for all $\Lambda \leftarrow \mathsf{AEval}(\mathsf{pp}, \mathcal{X})$, and for all $x \in \mathcal{X}$, we have $\mathsf{AVerify}(\mathsf{pp}, \Lambda, x, \mathsf{AWitCreate}(\mathsf{pp}, \mathcal{X}, \Lambda, x)) = 1$.We also require the following collision freeness property for accumulators. Informally, it demands the inability of any PPT adversary to output a witness for a non-accumulated value.

Definition 1 (Collision Freeness). *An accumulator is collision free if for all PPT adversaries,*

$$\Pr\left[\mathsf{pp} \leftarrow \mathsf{ASetup}(1^\lambda), (\mathcal{X}^*, x^*, \mathsf{wit}^*) \leftarrow \mathcal{A}(\mathsf{pp}) : x^* \notin \mathcal{X}^* \text{ and } \mathsf{AVerify}(\mathsf{pp}, \mathsf{AEval}(\mathsf{pp}, \mathcal{X}^*), x^*, \mathsf{wit}^*) = 1\right] \leq \mathsf{negl}(\lambda).$$

2.2 EPID Signatures for Attestation

We now present the syntax and security requirements of EPID signatures for attestation as defined by Boneh et al. [10]. The primary distinction between EPID signatures for attestation [10] and the original EPID signatures proposed by Brickel and Li [11] is the use of accumulators and the ability for group members to continually rejoin the system. When a group member $\mathcal{P}_i$ joins the group, the group manager $\mathcal{GM}$ updates the Merkle tree and provides $\mathcal{P}_i$ with a membership certificate cert_i that is connected to the root value Λ of the updated Merkle tree. In addition, $\mathcal{GM}$ generates a signature σ_Λ on the root Λ (by employing a standard signature scheme $\mathcal{SIG}$) and sends $(\Lambda, \sigma_\Lambda)$ to $\mathcal{P}_i$. The signature σ_Λ can be public as it leaks nothing about the identity of a group member. A signer only needs to prove their knowledge of the certificate, which is also connected to Λ. The signature will then include the proof as well as $(\Lambda, \sigma_\Lambda)$. The verification of

the signature checks the validity of the membership proof and the signature σ_Λ. Because the Mekle tree is updated each time a group member joins, this reduces the size of the anonymity set. In a worst case, $\mathcal{GM}$ may issue a sequence of Merkle tree roots such that each tree only contains a single valid certificate for one group member, thus completely breaking the anonymity guarantee. Fortunately, rejoining could mitigate the above concern, where group members could update the Merkle tree roots as well as the associated certificates, thereby enlarging the anonymity set. As pointed out in [10], rejoining is appropriate for attestation schemes such as those used in Intel SGX [20,32] that involve continuing contact with the group manager ($\mathcal{GM}$).

There are two methods to revoke a group member by either (i) adding the secret key of a revoked member into a key revocation list KRL or (ii) adding a signature of a revoked member to a signature revocation list SRL. In this work, we consider the scenario where SRL and KRL are managed by some trusted entities. When generating an EPID signature with respect to SRL, a group member is additionally required to produce a proof of non-revocation by demonstrating that his signing key did not generate any of the signatures in SRL. Validity of a signature is checked with respect to the lists SRL and KRL. An EPID signature Σ is valid if and only if (i) the proof of the group membership is valid, (ii) $(\Lambda, \sigma_\Lambda)$ is a valid message-signature pair for the scheme $\mathcal{SIG}$, (iii) the proof of non-revocation with respect to SRL is valid, (iv) and the signature Σ is not generated by any of the keys in KRL.

Syntax of EPID Signatures for Attestation. We assume the existence of an accumulator Acc and a standard signature scheme $\mathcal{SIG}$. In the following, $\mathcal{X}$ represents a set, Λ represents an accumulator value of $\mathcal{X}$, and σ_Λ is a standard signature on Λ. An EPID signature scheme for attestation contains polynomial time algorithms KeyGen, Join, Rejoin, Sign, Verify, $\mathsf{KeyRevoke}$, and $\mathsf{SigRevoke}$ defined below.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{gsk}, \mathsf{gpk})$**.** On input a security parameter 1^λ, output a key pair $(\mathsf{gsk}, \mathsf{gpk})$.

$\mathsf{Join}\langle \mathcal{GM}(\mathsf{gsk}, \mathcal{X}_{\mathsf{current}}), \mathcal{P}_i(\mathsf{gpk}) \rangle \to \big((\mathsf{cert}_i, \mathcal{X}_{\mathsf{new}}, \Lambda_{\mathsf{new}}, \sigma_{\Lambda_{\mathsf{new}}}) | (\mathsf{sk}_i, \mathsf{cert}_i, \Lambda_{\mathsf{new}}, \sigma_{\Lambda_{\mathsf{new}}})\big)$**.** This is an interactive protocol between the $\mathcal{GM}$ and a group member $\mathcal{P}_i$. Both parties have gpk as input, while the $\mathcal{GM}$ also has its secret key gsk and a set $\mathcal{X}_{\mathsf{current}}$ as inputs. If the interaction completes successfully, the $\mathcal{GM}$ computes a new set $\mathcal{X}_{\mathsf{new}}$, an accumulator value Λ_{new}, a signature $\sigma_{\Lambda_{\mathsf{new}}}$ on Λ_{new}, and a membership certificate cert_i for $\mathcal{P}_i$. The group member will obtain $(\Lambda_{\mathsf{new}}, \sigma_{\Lambda_{\mathsf{new}}})$, cert_i, and a secret key sk_i not known by the $\mathcal{GM}$.

$\mathsf{Rejoin}\langle \mathcal{GM}(\mathsf{gsk}, \mathcal{X}_{\mathsf{current}}, \Lambda_{\mathsf{current}}, \sigma_{\mathsf{current}}), \mathcal{P}_i(\mathsf{cert}_{i,\mathsf{prev}}, \Lambda_{\mathsf{prev}}, \sigma_{\Lambda_{\mathsf{prev}}}) \rangle \to (\mathsf{out}_1 | \mathsf{out}_2)$**.** This is an interactive protocol between $\mathcal{GM}$ and a group member $\mathcal{P}_i$ who wishes to update their group membership. The $\mathcal{GM}$ has its secret key gsk, a set $\mathcal{X}_{\mathsf{current}}$, the corresponding accumulator value $\Lambda_{\mathsf{current}}$, and a signature $\sigma_{\mathsf{current}}$ as inputs. The group member has its certificate $\mathsf{cert}_{i,\mathsf{prev}}$ and $(\Lambda_{\mathsf{prev}}, \sigma_{\Lambda_{\mathsf{prev}}})$ obtained from previous joining or rejoining protocols as inputs. If the interaction completes successfully, the $\mathcal{GM}$ computes an updated membership certifi-

cate $\mathsf{cert}_{i,\mathsf{current}}$ for $\mathcal{P}_i$, and gets $\mathsf{out}_1 = (\mathsf{cert}_{i,\mathsf{current}}, \mathcal{X}_{\mathsf{current}}, \Lambda_{\mathsf{current}}, \sigma_{\Lambda_{\mathsf{current}}})$ as output. The $\mathcal{P}_i$ obtains $\mathsf{out}_2 = (\mathsf{cert}_{i,\mathsf{current}}, \Lambda_{\mathsf{current}}, \sigma_{\Lambda_{\mathsf{current}}})$ as output.

$\mathsf{Sign}(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda, m, \mathsf{SRL}) \rightarrow \Sigma/\perp$**.** This algorithm is run by a group member $\mathcal{P}_i$. It takes as input the secret key sk_i, a membership certificate cert_i, an accumulator value Λ, a signature σ_Λ, and a signature revocation list SRL, and outputs a signature Σ or $\perp$.

$\mathsf{Verify}(\mathsf{gpk}, m, \Sigma, \mathsf{KRL}, \mathsf{SRL})$**.** This algorithm verifies the validity of the signature Σ on message m with respect to the key revocation list KRL and the signature revocation list SRL. It returns 1 to accept the signature and 0 to reject it.

$\mathsf{KeyRevoke}(\mathsf{gpk}, \mathsf{KRL}, \mathsf{sk}) \rightarrow \mathsf{KRL}$**.** This algorithm adds sk to KRL, so signatures created with this key will no longer be accepted.

$\mathsf{SigRevoke}(\mathsf{gpk}, \mathsf{KRL}, \mathsf{SRL}, (m, \Sigma)) \rightarrow \mathsf{SRL}$**.** This algorithm adds Σ to SRL, so signatures created with the secret key underlying Σ will no longer be accepted.

An EPID signature must satisfy correctness, anonymity and unforgeability.

Correctness. Informally, correctness demands that any signature generated by a group member who has successfully obtained a membership certificate from joining or rejoining procedures is accepted by the verification algorithm, as long as its secret key and signatures have not been revoked. Let S_i be the set of signatures generated by group member $\mathcal{P}_i$. An EPID signature is correct if for any $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, any $(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ obtained from joining and rejoining protocols, and any $m \in \{0,1\}^*$, if $(\mathsf{sk}_i \notin \mathsf{KRL}) \wedge (S_i \cap \mathsf{SRL} = \emptyset)$, the following holds:

$$\Pr[\mathsf{Verify}(\mathsf{gpk},\ m,\ \mathsf{Sign}(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda, m, \mathsf{SRL}),\ \mathsf{KRL},\ \mathsf{SRL}) = 1] = 1 - \mathsf{negl}(\lambda).$$

Anonymity. Anonymity states that a signature does not reveal the identity of its signer. This is modeled in the following experiments $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$ for $b \in \{0,1\}$ run between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.

Experiment: $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$

The challenger $\mathcal{C}$ maintains a list $\mathcal{L}_{\mathsf{cert}}$ that contains user secret key sk_i, group member certificate cert_i together with $(\Lambda_i, \sigma_{\Lambda_i})$, a signature list $\mathcal{L}_{\mathsf{sig}}$, and a corrupted user list $\mathcal{L}_{\mathsf{CU}}$.

Setup. The adversary $\mathcal{A}$ computes $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends gpk to $\mathcal{C}$.

Unrestricted queries. $\mathcal{A}$ has access to the following queries.

Join. $\mathcal{A}$ requests for creating a new group member $\mathcal{P}_i$. $\mathcal{C}$ ensures that $\mathcal{P}_i$ has not been requested before. Then $\mathcal{C}$ acts as $\mathcal{P}_i$ and runs the interactive Join protocol with $\mathcal{A}$, who plays the role of the $\mathcal{GM}$. In the end, $\mathcal{C}$ obtains $(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ and adds the tuple to $\mathcal{L}_{\mathsf{cert}}$.

Rejoin. $\mathcal{A}$ requests rejoining of a group member $\mathcal{P}_i$. $\mathcal{C}$ makes sure that $\mathcal{P}_i$ has joined or rejoined before. Then $\mathcal{C}$ and $\mathcal{A}$ run the interactive Rejoin protocol, in which $\mathcal{C}$ plays the role of user $\mathcal{P}_i$ and $\mathcal{A}$ plays the role of the $\mathcal{GM}$. In the end, $\mathcal{C}$ obtains updated $(\mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ and updates $\mathcal{L}_{\mathsf{cert}}$ accordingly.

Sign. When $\mathcal{A}$ requests a signature on a message m from $\mathcal{P}_i$ with respect to SRL, $\mathcal{C}$ ensures that SRL is a subset of $\mathcal{L}_{\mathsf{sig}}$, and that $\mathcal{P}_i$ has been created. Then $\mathcal{C}$ retrieves $(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ from $\mathcal{L}_{\mathsf{cert}}$, computes $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda, m, \mathsf{SRL})$, returns Σ to $\mathcal{A}$. $\mathcal{C}$ also adds Σ to $\mathcal{L}_{\mathsf{sig}}$.

Corrupt. When $\mathcal{A}$ requests the secret key of $\mathcal{P}_i$, $\mathcal{C}$ ensures that $\mathcal{P}_i$ has been created and $i \notin \mathcal{L}_{\mathsf{CU}}$. Then $\mathcal{C}$ retrieves sk_i from $\mathcal{L}_{\mathsf{cert}}$, returns sk_i to $\mathcal{A}$, and adds i to $\mathcal{L}_{\mathsf{CU}}$.

Challenge. In the challenge phase, $\mathcal{A}$ outputs a message m, a signature revocation list SRL, two indices i_0, i_1. $\mathcal{C}$ makes sure that $\mathcal{P}_{i_0}, \mathcal{P}_{i_1}$ have been created, $i_0 \notin \mathcal{L}_{\mathsf{CU}}$, $i_1 \notin \mathcal{L}_{\mathsf{CU}}$, and SRL does not contain signatures from either $\mathcal{P}_{i_0}$ or $\mathcal{P}_{i_1}$. Then, $\mathcal{C}$ retrieves $(\mathsf{sk}_{i_0}, \mathsf{cert}_{i_0}, \Lambda_{i_0}, \sigma_{\Lambda_{i_0}})$ and $(\mathsf{sk}_{i_1}, \mathsf{cert}_{i_1}, \Lambda_{i_1}, \sigma_{\Lambda_{i_1}})$ from $\mathcal{L}_{\mathsf{cert}}$. Particularly, $\mathcal{C}$ also makes sure $(\Lambda_{i_0}, \sigma_{\Lambda_{i_0}}) = (\Lambda_{i_1}, \sigma_{\Lambda_{i_1}})$. Finally, $\mathcal{C}$ computes $\Sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}_{i_b}, \mathsf{cert}_{i_b}, \Lambda_{i_b}, \sigma_{\Lambda_{i_b}}, m, \mathsf{SRL})$ and returns Σ^* to $\mathcal{A}$.

Restricted queries. $\mathcal{A}$ can make queries as before except that (a) $\mathcal{A}$ cannot make **Corrupt** queries on $\mathcal{P}_{i_0}$ or $\mathcal{P}_{i_1}$ and (b) $\mathcal{A}$ cannot make **Sign** queries with respect to SRL such that $\Sigma^* \in \mathsf{SRL}$.

Output. This experiment outputs b', which is returned by $\mathcal{A}$ at the end of the experiment.

Definition 2 (Anonymity). *An EPID signature scheme for attestation is anonymous if* $|\Pr[\mathsf{Exp}^{\mathsf{anony}-0}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{anony}-1}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]|$*, the advantage of any PPT adversary $\mathcal{A}$, is negligible in λ.*

Unforgeability. Unforgeability demands that a revoked user, either through keys or signatures, could not produce valid signatures. This requirement is captured in the following experiment $\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$.

Experiment: $\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$

The challenger $\mathcal{C}$ maintains $\mathcal{L}_{\mathsf{cert}}$, $\mathcal{L}_{\mathsf{sig}}$, $\mathcal{L}_{\mathsf{CU}}$ as in experiment $\mathsf{Exp}^{\mathsf{anony}-\mathsf{b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$.

Setup. $\mathcal{C}$ computes $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends gpk to $\mathcal{A}$.

Queries. $\mathcal{A}$ has access to the following queries.

Join. $\mathcal{A}$ requests creation of a new group member $\mathcal{P}_i$. $\mathcal{C}$ ensures that $\mathcal{P}_i$ has not been requested before. Then there are two cases, depending on the adversary's choice.

- $\mathcal{C}$ and $\mathcal{A}$ run the interactive Join protocol, with $\mathcal{C}$ playing the role of $\mathcal{GM}$ and $\mathcal{A}$ playing the role of $\mathcal{P}_i$. The challenger obtains $(\mathsf{cert}_i, \mathcal{X}, \Lambda, \sigma_\Lambda)$ and $\mathcal{A}$ obtains $(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda)$. $\mathcal{A}$ then sends sk_i to $\mathcal{C}$, who then adds i to $\mathcal{L}_{\mathsf{CU}}$ and adds $(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ to $\mathcal{L}_{\mathsf{cert}}$.
- $\mathcal{C}$ runs the Join protocol internally, and adds $(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ to $\mathcal{L}_{\mathsf{cert}}$. Also, $\mathcal{C}$ sends $(\mathsf{cert}_i, \Lambda, \sigma_\Lambda)$ to $\mathcal{A}$.

Rejoin. $\mathcal{A}$ requests rejoining of a group member $\mathcal{P}_i$. $\mathcal{C}$ ensures that $\mathcal{P}_i$ has joined or rejoined before. Then $\mathcal{C}$ and $\mathcal{A}$ run the interactive Rejoin with $\mathcal{C}$ acting as the $\mathcal{GM}$ and $\mathcal{A}$ acting as $\mathcal{P}_i$. Both parties obtain updated $(\mathsf{cert}_i, \Lambda, \sigma_\Lambda)$, and $\mathcal{C}$ updates $\mathcal{L}_{\mathsf{cert}}$ accordingly.

Sign. Same as in experiment $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$.
Corrupt. Same as in experiment $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$.

Forgery. Eventually, $\mathcal{A}$ outputs a message m^*, a signature revocation list SRL^*, a key revocation list KRL^*, a signature Σ^*.

Output. This experiment outputs 1 if (a) $\mathsf{Verify}(\mathsf{gpk}, m^*, \Sigma^*, \mathsf{KRL}^*, \mathsf{SRL}^*) = 1$, (b) for each $i \in \mathcal{L}_{\mathsf{CU}}$, either $\mathsf{sk}_i \in \mathsf{KRL}^*$ or one of the signatures created by $\mathcal{P}_i$ is in SRL^*, and (c) $\mathcal{A}$ did not obtain Σ^* by querying a signature on m^* with respect to SRL^*.

Definition 3 (Unforgeability). *An EPID signature scheme for attestation is unforgeable if the advantage* $\Pr[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]$ *of any PPT adversary* $\mathcal{A}$ *is negligible in* λ.

2.3 Zero-Knowledge from VOLE-in-the-head

We now provide an introduction to the Vector-Oblivious-Linear-Evaluation-in-the-Head (VOLEitH) proof system [7]. A VOLE correlation defined over $\mathbb{F}_{2^k}$ allows the sender (often the prover) to obtain $\mathbf{u} \in \mathbb{F}_2^l$, $\mathbf{v} \in \mathbb{F}_{2^k}^l$ and the receiver (often the verifier) to obtain $\mathbf{q} \in \mathbb{F}_2^l$ and $\Delta \in \mathbb{F}_{2^k}$ satisfying:

$$\mathbf{q} = \Delta \cdot \mathbf{u} + \mathbf{v}.$$

Denote this correlation by $[\![\mathbf{u}]\!]$. A VOLE correlation can be viewed as a commitment scheme for the sender to commit to $\mathbf{u}$. The commitment scheme is hiding since $\mathbf{v}$ masks $\mathbf{u}$ and is binding since a cheating sender would need to guess the value of Δ to open $(\mathbf{u}, \mathbf{v})$ to different values. In addition, VOLE correlations are linearly homomorphic. Specifically, given public coefficients $c_0, c_1, \ldots, c_l \in \mathbb{F}_{2^k}$ and $[\![u_i]\!]$ for $i \in [1, l]$, the parties can compute $[\![u]\!] = c_0 + \sum_{i=1}^{l} c_i [\![u_i]\!]$, where prover computes $u = c_0 + \sum_{i=1}^{l} c_i \cdot u_i$ and $v = \sum_{i=1}^{l} c_i \cdot v_i$, and verifier computes $q = c_0 \cdot \Delta + \sum_{i=1}^{l} c_i \cdot q_i$.

Given random VOLE correlations, Yang et al. [44] proposed an efficient proof system named QuickSilver for arbitrary low-degree polynomial constraints. In this work, we focus on degree-2 constraints. Let $f_i(X_1, \ldots, X_{\hat{l}}) = \sum_{j=0}^{2} g_{i,j}(X_1, \ldots, X_{\hat{l}})$ be a degree-2 polynomial over $\mathbb{F}_{2^k}$ in the degree-separated form, i.e., all items of $g_{i,j}$ has degree j. Suppose both parties obtain VOLE relations for w_i such that $q_i = \Delta \cdot w_i + v_i$ for $i \in [1, \hat{l}]$. Yang et al. [44] observe that

$$\begin{aligned} B_i &= \underbrace{\sum_{j=0}^{2} g_{i,j}(q_1, \ldots, q_{\hat{l}}) \cdot \Delta^{2-j}}_{\text{computable by } \mathcal{V}} \\ &= \underbrace{f_i(w_1, \ldots, w_{\hat{l}})}_{0 \text{ if } \mathcal{P} \text{ is honest}} \cdot \Delta^2 + \underbrace{A_{0,i}}_{\text{computable by } \mathcal{P}} + \underbrace{A_{1,i}}_{\text{computable by } \mathcal{P}} \cdot \Delta. \end{aligned} \tag{1}$$

Therefore, to prove $f_i(w_1, \ldots, w_{\hat{l}}) = 0$, it suffices to prove $B_i = A_{0,i} + A_{1,i} \cdot \Delta$. In addition, t constraints can be checked once via a random linear combination. The details of the QuickSilver protocol are recalled below.

Protocol 1: π_{QS}

INPUTS. Both parties know t degree-2 constraints $\{f_i(X_1, \ldots, X_{\hat{l}}) : i \in [1, t]\}$ over $\mathbb{F}_{2^k}$. The prover additionally knows $\mathbf{w} = (w_1, \ldots, w_{\hat{l}}) \in \mathbb{F}_2^{\hat{l}}$ satisfying $f_i(\mathbf{w}) = 0$ for all $i \in [1, t]$. The parties are given $\hat{l} + k$ random VOLE correlations $\{[\![u_i]\!]\}_{i \in [1, \hat{l}+k]}$ over $\mathbb{F}_{2^k}$, with $u_i \in \mathbb{F}_2$. Then both parties can convert $\{[\![u_i]\!]\}_{i \in [\hat{l}+1, \hat{l}+k]}$ into a single VOLE correlation $Q^* = \Delta \cdot U^* + V^*$ with $U^* \in \mathbb{F}_{2^k}$.

1. For $i \in [1, \hat{l}]$, the prover sends $d_i = w_i - u_i$ to the verifier. Due to the linearity of VOLE correlations, both parties obtain $[\![w_i]\!] = [\![u_i]\!] + d_i$ for $i \in [1, \hat{l}]$.
2. The verifier samples random values $\chi_1, \ldots, \chi_t \in \mathbb{F}_{2^k}$ and sends them to the prover. Upon receiving $\{\chi_i\}_{i \in [1,t]}$, the prover computes $A_{0,i}$ and $A_{1,i}$ for $i \in [1, t]$ according to (1), and $U = \sum_{i=1}^{t} A_{0,i} \cdot \chi_i + U^*$, $V = \sum_{i=1}^{t} A_{1,i} \cdot \chi_i + V^*$, and then sends (U, V) to the verifier.
3. The verifier computes B_i for $i \in [1, t]$ according to (1), and $Q = \sum_{i=1}^{\ell} B_i \cdot \chi_i + Q^*$.

OUTPUT. The verifier outputs 1 if $U \cdot \Delta + V = Q$.

From the description, we see that π_{QS} is designated verifier, i.e., the protocol is verified against the party who knows Δ. Fortunately, this can be made publicly verifiable via the VOLEitH techniques [7]. The main building block of VOLEitH is all-but-one vector commitment (VC), which can be used to realize VOLE correlations. In an all-but-one VC scheme, the prover first commits to $\mathbf{r}_0, \ldots, \mathbf{r}_{N-1} \in \mathbb{F}_2^l$. When the verifier sends a random index $\Delta \in [0, N-1]$, the prover opens all messages except $\mathbf{r}_\Delta$ to the verifier. Define

$$\mathbf{u} = \sum_{i=0}^{N-1} \mathbf{r}_i \in \mathbb{F}_2^l, \quad \mathbf{v} = -\sum_{i=0}^{N-1} i \cdot \mathbf{r}_i \in \mathbb{F}_{2^k}^l, \quad \mathbf{q} = \sum_{i=0}^{N-1} (\Delta - i) \cdot \mathbf{r}_i \in \mathbb{F}_{2^k}^l. \quad (2)$$

In (2), we assume $N = 2^k$ and there is a correspondence between $[0, N-1]$ and $\mathbb{F}_{2^k}$. It is then verifiable that $\mathbf{q} = \Delta \cdot \mathbf{u} + \mathbf{v}$.

We now show how the VOLEitH proof system works. Let C be a circuit over $\mathbb{F}_2$ with t multiplication gates and l input bits. This can be seen as t degree-2 constraints with witness size $l + t$ [1]. The prover and verifier proceed as follows.

1. The prover runs a GGM tree to generate N leaves $\mathbf{r}_0, \ldots, \mathbf{r}_{N-1} \in \mathbb{F}_2^{l+t+k}$, and then makes a commitment to these leaves. Next, the prover computes $\mathbf{u}, \mathbf{v}$ according to (2). The prover also computes $d_1, \ldots, d_l, d_{l+1}, \ldots, d_{l+t}$ as in π_{QS} and sends them to the verifier.

[1] Witness includes the l input bits and the t output bits of multiplication gates. The t constraints are to verify the t multiplication gates.

2. The verifier sends $\chi_1, \ldots, \chi_t$ to the prover.
3. The prover computes U, V as in π_{QS} and sends them to the verifier.
4. The verifier selects a random index $\Delta \in [0, N-1]$ and sends it to the prover.
5. The prover decommits all $\mathbf{r}_x$ except for $x = \Delta$ to the verifier. The verifier can now compute its $\mathbf{q}$ according to (2), and Q as in π_{QS}.

The above interactive protocol can be made non-interactive through the Fiat-Shamir transform [26] since it is public coin. As pointed out by Ouyang et al. [39], the resulting non-interactive protocol is also simulation-sound extractable [27].

For security level λ, however, the above protocol requires committing to $N = 2^\lambda$ values. In practice, the prover runs τ parallel GGM trees with $N = 2^k$ leaves such that $k\tau = \lambda$. Specifically, the prover and verifier obtain $\mathbf{q}_i = \Delta_i \cdot \mathbf{u}_i + \mathbf{v}_i$ for $i \in [0, \tau - 1]$ from the τ runs, where $\Delta_i, \mathbf{q}_i, \mathbf{v}_i$ are over $\mathbb{F}_{2^k}$ and $\mathbf{u}_i$ is over $\mathbb{F}_2$. Then the prover sends correction values $\mathbf{c}_i = \mathbf{u}_0 - \mathbf{u}_i$ for $i \in [1, \tau - 1]$ to the verifier. Due to the linearity of VOLE correlations, both parties obtain τ VOLE correlations (over $\mathbb{F}_{2^k}$) committing to $\mathbf{u}_0$ with respect to $\Delta_0, \ldots, \Delta_{\tau-1}$, respectively. These τ correlations are then lifted to a single VOLE correlation (over $\mathbb{F}_{2^{k\tau}}$) committing to $\mathbf{u}_0$. To ensure that correction values $\mathbf{c}_i$ are honestly generated, a consistency check is implemented with a linear hash [7].

To further improve the efficiency, Baum et al. [6] proposed batch all-but-one VC. In a nutshell, a single big tree is interpreted as embedding τ smaller trees. Then opening all but τ leaves of the big tree is more efficient than opening all but one leaf in each of the τ smaller trees, because with high probability some active paths in the big tree overlap. As a result, batch all-but-one opening causes variability in the number of internal nodes opened. Baum et al. [6] then fix a threshold number $\mathsf{T}_{\mathsf{open}}$ which cannot be exceeded during the opening phase.

The FAEST [5], a standard signature scheme constructed from applying VOLEitH protocols on AES one-way functions, also considers other optimizations. In this work, we will employ the optimized VOLEitH protocols as in [5] to implement our EPID signatures for attestation. In particular, the proof size is calculated as

$$\mathsf{size} = \tau \cdot (\hat{l} + 2 \cdot \lambda + 16) + \mathsf{T}_{\mathsf{open}} \cdot \lambda + 2 \cdot \lambda \cdot \tau + \lambda + 128 + 32, \tag{3}$$

where $\hat{l}$ is the size of witness in bits. We remark that to run the VOLEitH protocol [5] with small proof sizes, we need to convert the possibly complicate statements into degree-2 polynomial constraints with small witness size $\hat{l}$.

3 Hash-Based Merkle Tree Accumulator

In Section 3.1, we first present a Merkle tree accumulator from symmetric primitives. Next, Section 3.2 presents a zero-knowledge argument of knowledge (ZKAoK) of an accumulated value for the accumulator, which is crucial in designing our EPID signatures for attestation in Section 4.

3.1 Description of Our Accumulator

To build a Merkle tree accumulator [35,24,10] from symmetric primitives, we need a collision resistant hash function that compresses $2n$-bit strings into n-bit digests. While SHA families are efficient, they have high multiplicative complexity, making them unfriendly to privacy-preserving cryptographic protocols. Derler et al. [24] suggested using the LowMC round function in the sponge framework. Later, Boneh et al. [10] proposed applying the Davies-Meyer transformation to the LowMC cipher [2] as a collision resistant hash function on fixed-length inputs is required. The collision resistance of the Davies-Meyer transformation has been proven in the ideal cipher model. Note that LowMC cipher was initially proposed as an alternative to AES that has low multiplicative complexity. However, many instances of LowMC cipher have been broken (see, e.g., [36,37,43]). Thus, we propose applying the Davies-Meyer transformation to Rijndael [21], the precursor of the standardized AES. In particular, $H_{\mathsf{DM}}(m_1 \| m_2) = \mathsf{Rijn}(m_1, m_2) \oplus m_2$, where m_1, m_2 are of equal length. To achieve 128-bit security, the output length of $H_{\mathsf{DM}}(m_1 \| m_2)$ is required to be 256. Therefore, we resort to Rijndael-256. We note that achieving higher security requires a block cipher with a larger state. For instance, one can achieve 256-bit security by employing Vistrutah [4] with 512-bit keys and 512-bit block sizes. However, the security of such newly appeared ciphers requires further scrutiny from the community. For this reason, we focus on 128-bit security in this work.

Our hash-based Merkle tree accumulator $\mathsf{Acc}_{\mathsf{Rijn}}$ works as follows. The construction is adapted from the blueprint by previous works [35,24,10].

$\mathsf{ASetup}(1^\lambda)$**.** This algorithm chooses $n = 2\lambda$ and $H_{\mathsf{DM}} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ to be $H_{\mathsf{DM}}(m_1 \| m_2) = \mathsf{Rijn}(m_1, m_2) \oplus m_2$. Let $\mathsf{pp} = \{n, H_{\mathsf{DM}}\}$.

$\mathsf{AEval}(\mathsf{pp}, \mathcal{X} = \{x_0, x_1, ..., x_{N-1} \in \{0,1\}^n\})$**.** Given a set of $N = 2^a$ strings, this algorithm builds a depth-a binary tree and outputs an accumulator value Λ as follows.

- Let $u_{i,j}$ denote the j-th (the leftmost node is indexed by 0) node value in the i-th layer (the leaf nodes are indexed by a).
- Let the leaf nodes be $u_{a,j} = x_j$ for $j \in [0, N-1]$.
- For $i \in \{a-1, \ldots, 0\}$, $j \in [0, 2^i - 1]$, compute $u_{i,j} = H_{\mathsf{DM}}(u_{i+1,2j} \| u_{i+1,2j+1})$.
- Return the accumulator value $\Lambda = u_{0,0}$.

$\mathsf{AWitCreat}(\mathsf{pp}, \mathcal{X}, \Lambda, x)$**.** If $x \notin \mathcal{X}$, return $\perp$. Else, $x = x_k$ for some $k \in [0, N-1]$. Let $(k_1, \ldots, k_a)$ be the binary representation of k, with k_1 being the most significant bit. This algorithm returns $\mathsf{wit}_x = ((k_1, \ldots, k_a), w_a, \ldots, w_1) \in \{0,1\}^a \times (\{0,1\}^n)^a$, where

$$w_i = u_{i, \lfloor k/2^{a-i} \rfloor + \eta}, \text{ where } \eta = \begin{cases} 1 & \text{if } \ k_i = 0; \\ -1 & \text{if } \ k_i = 1. \end{cases} \tag{4}$$

In fact, $(w_a, \ldots, w_1)$ are the sibling nodes of the nodes on the path from x_k to the root.

$\mathsf{AVerify}(\mathsf{pp}, \Lambda, x, \mathsf{wit}_x)$. Parse $\mathsf{wit}_x = ((k_1, \ldots, k_a), w_a, \ldots, w_1) \in \{0,1\}^a \times (\{0,1\}^n)^a$. This algorithm computes $v_a, v_{a-1}, \ldots, v_1, v_0$ in the following manner. Set $v_a = x$, and

$$\text{for each } i \in \{a-1, \ldots, 1, 0\} : v_i = \begin{cases} H_{\mathsf{DM}}(v_{i+1} \| w_{i+1}) & \text{if } \ k_{i+1} = 0; \\ H_{\mathsf{DM}}(w_{i+1} \| v_{i+1}) & \text{if } \ k_{i+1} = 1. \end{cases} \tag{5}$$

This algorithm returns 1 if $v_0 = \Lambda$, and 0 otherwise.

The above accumulator is correct and collision free since H_{DM} is a collision resistant hash function, assuming that Rijndael is an ideal cipher.

3.2 ZKAoK of an Accumulated Value

We now present a zero-knowledge argument of knowledge that enables a prover $\mathcal{P}$ to demonstrate the knowledge of a value such that this value is correctly accumulated in the hash-based accumulator presented in Section 3.1.

Given the hash function H_{DM} and an accumulator value Λ, the prover is to prove in zero-knowledge the knowledge of $(x, \mathsf{wit}_x) \in \{0,1\}^n \times \big(\{0,1\}^a \times (\{0,1\}^n)^a\big)$ such that $\mathsf{AVerify}(H_{\mathsf{DM}}, \Lambda, x, \mathsf{wit}_x) = 1$.

To efficiently show that the above statement is true, we need to convert the recursive formulas (5) into degree-2 polynomial constraints with as small witness size as possible. To this end, we first employ the 2-to-1 multiplexer proposed by Derler et al. [24]. The multiplexer can be viewed as a function that uses a selection bit b to determine whether $v \in \{0,1\}^n$ or $w \in \{0,1\}^n$ is selected. Formally, $\mathsf{MP} : \{0,1\} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is defined as $\mathsf{MP}(b, v, w) = ((v \oplus w) \wedge b) \oplus v$, where $(v \oplus w) \wedge b$ is performed bit by bit. Therefore, formulas in (5) are equivalent to: for each $i \in \{a-1, \ldots, 1, 0\}$,

$$v_i = H_{\mathsf{DM}}\big(\ \mathsf{MP}(k_{i+1}, v_{i+1}, w_{i+1}) \ \|\ \mathsf{MP}(k_{i+1}, w_{i+1}, v_{i+1})\ \big). \tag{6}$$

Evaluation of $\mathsf{MP}(k_{i+1}, v_{i+1}, w_{i+1})$ and $\mathsf{MP}(k_{i+1}, w_{i+1}, v_{i+1})$ incurs non-linear operation $(v_{i+1} \oplus w_{i+1}) \wedge k_{i+1}$. Thus, we introduce extra witness g_{i+1}, and additionally prove that $(v_{i+1} \oplus w_{i+1}) \wedge k_{i+1} - g_{i+1} = 0$. This results in extra witness size $a \cdot n$.

Regarding evaluation of $H_{\mathsf{DM}}(m_1 \| m_2) = \mathsf{Rijn}(m_1, m_2) \oplus m_2$, let us first briefly recall how Rijndael works. Rijndael consists of an encryption schedule and a key expansion schedule. Both proceed in rounds, and each round contains field inversions over $\mathbb{F}_{2^8}$ (and other linear operations). To show the field inversions are performed correctly, we need to verify $s_{\mathsf{in}} \cdot s_{\mathsf{out}} = 1$, where $s_{\mathsf{in}}, s_{\mathsf{out}}$ are the byte strings preceding and following the field inversion. To further ensure that both are 0 if either s_{in} or s_{out} is, we instead verify $s_{\mathsf{in}}^2 \cdot s_{\mathsf{out}} - s_{\mathsf{in}} = 0$ and $s_{\mathsf{in}} \cdot s_{\mathsf{out}}^2 - s_{\mathsf{out}} = 0$. While the new equations have degree 3 over $\mathbb{F}_{2^8}$, it is shown that squaring can be performed linearly over bits, resulting in constraints of degree 2. Therefore, for each field inversion, only an extra 8-bit witness s_{out} is required.

Let R be the number of rounds in Rijndael with equal-sized key and message spaces. Then $R = 14$ for Rijndael-256. According to the mechanism of Rijndael-256, q_0 (7) extra s_{out} are required for each evaluation of $\mathsf{Rijn}(\cdot,\cdot)$ or H_{DM}. We evaluate H_{DM} for a times in total. This results in extra witness size $8 \cdot q_0 \cdot a$.

$$q_0 = \underbrace{(R-1)\cdot\frac{\lambda}{8}}_{\text{No. of field inver. in enc. sche.}} + \underbrace{R\cdot 8}_{\text{No. of field inver. in key expan.}} \tag{7}$$

Given the above preparation, we have transformed $\mathsf{AVerify}(H_{\mathsf{DM}}, \Lambda, x, \mathsf{wit}_x) = 1$ into a sequence of constraints with extended witnesses $\{k_i, v_i, w_i, g_i : i \in [1,a]\} \cup \{s_{\mathsf{out}:j\in[1,q_0 a]}\}$ of size $\hat{l} = a + 3na + 8 \cdot q_0 \cdot a$. The prover then runs the optimized VOLEitH protocol [5]. In Table 2, we present extended witness sizes and proof sizes under various parameters.

Table 2. Extended witness sizes $\hat{l}$ and membership proof sizes under various parameters. $\tau, \mathsf{T}_{\mathsf{open}}$ are the parameters for the VOLEitH proof systems following the specification in [5, Table 3.2].

Security	a	τ	$\mathsf{T}_{\mathsf{open}}$	extended witness size $\hat{l}$	proof size (KB)
$\lambda = 128$	5	11	103	24965	35.9
		16	112	24965	51.6
	10	11	103	49930	69.4
		16	112	49930	100.4
	15	11	103	74895	102.9
		16	112	74895	149.1
	20	11	103	99860	136.5
		16	112	99860	197.9

4 EPID Signatures from VOLE-in-the-head

This section presents our EPID signatures for attestation from symmetric primitives. In particular, we instantiate the generic construction by Boneh et al. [10, Section 4.2] with the following ingredients.

- A hash-based accumulator constructed from Section 3.1. Denote $\mathsf{Acc}_{\mathsf{Rijn}} = (\mathsf{ASetup}, \mathsf{AEval}, \mathsf{AWitCreat}, \mathsf{AVerify})$. This is used by the group manager to create certificates for group members.
- A pseudorandom function (PRF) family $\mathcal{F}_{\mathsf{Rijn}} = \{f_\lambda : \{0,1\}^{n(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}\}$, constructed as $f(m_1, m_2) = \mathsf{Rijn}(m_1, m_2) \oplus m_2$. As we have seen in Section 3.1, this family is also collision resistant over its key space in the ideal cipher model.

– A standard signature scheme FAEST [5] that is strongly unforgeable under chosen message attacks in the random oracle model. Denote $\mathsf{FAEST} = (\mathsf{FAEST.Setup}, \mathsf{FAEST.Sign}, \mathsf{FAEST.Verify})$. This is utilized by the group manager to sign accumulator values.
– The optimized VOLEitH proof system as presented in [5]. Denote $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$. This efficient and versatile simulation-sound extractable ZK protocol is employed by the signer when signing messages.

We remark that any strongly unforgeability signature scheme and any simulation-sound extractable NIZK proof system that proves the satisfiability of degree-2 constraints can be directly plugged into our construction.

4.1 Description of Our EPID Scheme for Attestation

Our scheme works as follows.

$\mathsf{KeyGen}(1^\lambda)$**.** Given a security parameter 1^λ, the algorithm generates group key pair $(\mathsf{gpk}, \mathsf{gsk})$ as follows.
– Let $n(\lambda) = 2\lambda$. Specify a PRF family $\mathcal{F}_{\mathsf{Rijn}} = \{f_\lambda : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n\}$, in which $f(m_1, m_2) = \mathsf{Rijn}(m_1, m_2) \oplus m_2$.
– Let $H_{\mathsf{DM}} : \{0,1\}^{2n} \to \{0,1\}^n$ be the collision resistant hash function, generated via $\mathsf{ASetup}(1^\lambda)$ from Section 3.1.
– Run the setup algorithm of the FAEST signature scheme, i.e., $(\mathsf{opk}, \mathsf{osk}) \leftarrow \mathsf{FAEST.Setup}(1^\lambda)$.
– Run the setup algorithm of the VOLEitH proof system [5], i.e., $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$.

Set group public key as $\mathsf{gpk} = \{\mathcal{F}_{\mathsf{Rijn}}, H, \mathsf{opk}, \mathsf{crs}\}$ and group secret key as $\mathsf{gsk} = \{\mathsf{osk}\}$.

$\mathsf{Join}\langle\mathcal{GM}(\mathsf{gpk}, \mathsf{gsk}, \mathcal{X}_{\mathsf{current}}), \mathcal{P}_i(\mathsf{gpk})\rangle$**.** This is an interactive protocol between the $\mathcal{GM}$ and a potential group member.
– Upon receiving a joining request from a group member $\mathcal{P}_i$, the $\mathcal{GM}$ samples a challenge $c_i \in \{0,1\}^n$ and sends it to $\mathcal{P}_i$.
– Next, $\mathcal{P}_i$ chooses a random secret key $\mathsf{sk}_i \leftarrow \{0,1\}^n$, and computes $t_i^{\mathsf{join}} = f(\mathsf{sk}_i, c_i) \in \{0,1\}^n$. Then $\mathcal{P}_i$ sends t_i^{join} to $\mathcal{GM}$.
– Upon receiving t_i^{join}, $\mathcal{GM}$ defines $x_i = (t_i^{\mathsf{join}}, c_i)$, and updates $\mathcal{X}_{\mathsf{current}}$ to $\mathcal{X}_{\mathsf{new}} = \mathcal{X}_{\mathsf{current}} \cup \{x_i\}$. Next, the $\mathcal{GM}$ runs $\Lambda_{\mathsf{new}} \leftarrow \mathsf{AEval}(H_{\mathsf{DM}}, \mathcal{X}_{\mathsf{new}})$ and $\mathsf{wit}_{x_i,\mathsf{new}} \leftarrow \mathsf{AWitCreat}(H_{\mathsf{DM}}, \Lambda_{\mathsf{new}}, \mathcal{X}_{\mathsf{new}}, x_i)$. Finally, the $\mathcal{GM}$ produces a signature $\sigma_{\Lambda_{\mathsf{new}}} \leftarrow \mathsf{FAEST.Sign}(\mathsf{gsk}, \Lambda_{\mathsf{new}})$ on the accumulator value Λ_{new}. Note that $x_i \in \{0,1\}^n \times \{0,1\}^n$ should be accumulated in the accumulator. However, our accumulator $\mathsf{Acc}_{\mathsf{Rijn}}$ from Section 3.1 only accepts n-bit leaf nodes. To this end, we propose applying H_{DM} on $(t_i^{\mathsf{join}}, c_i)$ to get an n-bit leaf value $H_{\mathsf{DM}}(t_i^{\mathsf{join}}, c_i) = \mathsf{Rijn}(t_i^{\mathsf{join}}, c_i) \oplus c_i$ and then running the accumulator $\mathsf{Acc}_{\mathsf{Rijn}}$.
– The $\mathcal{GM}$ sets $\mathsf{cert}_{i,\mathsf{new}} = (x_i, \mathsf{wit}_{x_i,\mathsf{new}})$ and sends a copy together with $(\Lambda_{\mathsf{new}}, \sigma_{\Lambda_{\mathsf{new}}})$ to $\mathcal{P}_i$.

$\mathsf{Rejoin}\langle \mathcal{GM}(\mathsf{gpk}, \mathsf{gsk}, \mathcal{X}_{\mathsf{current}}, \Lambda_{\mathsf{current}}, \sigma_{\Lambda_{\mathsf{current}}}), \mathcal{P}_i(\mathsf{sk}_i, \mathsf{cert}_{i,\mathsf{prev}}, \Lambda_{\mathsf{prev}}, \sigma_{\Lambda_{\mathsf{prev}}})\rangle$. When $\mathcal{P}_i$ with secret key sk_i, certificate $\mathsf{cert}_{i,\mathsf{prev}} = (x_i, \mathsf{wit}_{x_i,\mathsf{prev}})$, and $(\Lambda_{\mathsf{prev}}, \sigma_{\Lambda_{\mathsf{prev}}})$ requests to rejoin the group, if the $\mathcal{GM}$ accepts the request, the two parties proceed as follows.

- $\mathcal{P}_i$ sends $\mathsf{cert}_{i,\mathsf{prev}}$ and $(\Lambda_{\mathsf{prev}}, \sigma_{\Lambda_{\mathsf{prev}}})$ to the $\mathcal{GM}$.
- The $\mathcal{GM}$ runs $\mathsf{FAEST.Verify}(\mathsf{opk}, \Lambda_{\mathsf{prev}}, \sigma_{\Lambda_{\mathsf{prev}}})$ and $\mathsf{AVerify}(H, \Lambda_{\mathsf{prev}}, x_i, \mathsf{wit}_{x_i,\mathsf{prev}})$ to check the validity of $\mathcal{P}_i$'s membership. Abort if either check fails.
- Next, the $\mathcal{GM}$ checks if $x_i \in \mathcal{X}_{\mathsf{current}}$. Abort if this is not true.
- Else, the $\mathcal{GM}$ computes $\mathsf{wit}_{x_i,\mathsf{current}} \leftarrow \mathsf{AWitCreat}(H_{\mathsf{DM}}, \Lambda_{\mathsf{current}}, \mathcal{X}_{\mathsf{current}}, x_i)$ and then sends the updated certificate $\mathsf{cert}_{i,\mathsf{current}} = (x_i, \mathsf{wit}_{x_i,\mathsf{current}})$ and $(\Lambda_{\mathsf{current}}, \sigma_{\Lambda_{\mathsf{current}}})$ to $\mathcal{P}_i$.

$\mathsf{Sign}(\mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda, m, \mathsf{SRL})$. If $\mathsf{FAEST.Verify}(\mathsf{opk}, \Lambda, \sigma_\Lambda) = 0$ or there exists $\Sigma_j \in \mathsf{SRL}$ satisfying $t_{\Sigma_j} = f(\mathsf{sk}_i, r_{\Sigma_j})$, $\mathcal{P}_i$ aborts. Else, $\mathcal{P}_i$ performs the following steps. Let $\mathsf{cert}_i = ((t_i^{\mathsf{join}}, c_i), \mathsf{wit}_{x_i})$.

- Choose a random string $r_i \xleftarrow{\$} \{0,1\}^n \setminus \{c_i\}$ and compute tag $t_i = f(\mathsf{sk}_i, r_i) \in \{0,1\}^n$.
- Next, generate a VOLEitH proof of knowledge of $\eta = (\mathsf{sk}_i, t_i^{\mathsf{join}}, c_i, \mathsf{wit}_{x_i})$ such that the following conditions hold.
  (a) $t_i = f(\mathsf{sk}_i, r_i)$, i.e., the tag t_i is generated by the user's secret key sk_i and the randomnesses r_i.
  (b) $t_i^{\mathsf{join}} = f(\mathsf{sk}_i, c_i)$, i.e., t_i^{join} is generated by the user's secret key sk_i and the challenge c_i.
  (c) $r \neq c_i$.
  (d) $\mathsf{AVerify}(H_{\mathsf{DM}}, \Lambda, \mathsf{wit}_{x_i}, x_i) = 1$, i.e., $x_i = (t_i^{\mathsf{join}}, c_i)$ is honestly accumulated in Λ.
  (e) For each $\Sigma_j \in \mathsf{SRL}$, $t_{\Sigma_j} \neq f(\mathsf{sk}_i, r_{\Sigma_j})$, i.e., no signature generated by $\mathcal{P}_i$ has been placed into SRL.

  This is done by running $\pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{crs}, \xi, \eta)$ presented in Section 4.2, where $\xi = (\mathsf{gpk}, t_i, r_i, m, \mathsf{SRL}, \Lambda, \sigma_\Lambda)$ are public inputs.
- Return signature $\Sigma = (t_i, r_i, \pi, \Lambda, \sigma_\Lambda)$.

$\mathsf{Verify}(\mathsf{gpk}, m, \Sigma, \mathsf{KRL}, \mathsf{SRL})$. Parse $\Sigma = (t, r, \pi, \Lambda, \sigma_\Lambda)$. This algorithm verifies the validity of Σ on message m with respect to the two revocation lists KRL, SRL as follows.

- First, check the validity of $(\Lambda, \sigma_\Lambda)$. If $\mathsf{FAEST.Verify}(\mathsf{opk}, \Lambda, \sigma_\Lambda) = 0$, return 0.
- Next, check the validity of proof π. Define $\xi = (\mathsf{gpk}, t, r, m, \mathsf{SRL}, \Lambda, \sigma_\Lambda)$. If $\Pi.\mathsf{Verify}(\mathsf{crs}, \xi, \pi) = 0$, return 0.
- Check that the secret key underlying the signature Σ is not revoked. If there exists some $\mathsf{sk} \in \mathsf{KRL}$ satisfying $f(\mathsf{sk}, r) = t$, return 0.
- Return 1.

$\mathsf{KeyRevoke}(\mathsf{gpk}, \mathsf{KRL}, \mathsf{sk})$. Given the key revocation list KRL and a to-be-revoked secret key sk, this algorithm updates KRL to $\mathsf{KRL} \cup \{\mathsf{sk}\}$.

$\mathsf{SigRevoke}(\mathsf{gpk}, \mathsf{KRL}, \mathsf{SRL}, (m, \Sigma))$. This algorithm returns SRL if the algorithm $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma, \mathsf{KRL}, \mathsf{SRL})$ outputs 0. Else, return $\mathsf{SRL} \cup \{\Sigma\}$.

4.2 The Underlying Zero-Knowledge Protocol

Let us now present the ZKAoK invoked by the signer when generating EPID signatures. This protocol is an extension of the one for the Merkle Tree accumulator from section 3.2, for which the prover convinces the verifier of the following facts.

- The prover knows a secret key $\mathsf{sk}_i \in \{0,1\}^n$ corresponding to a tag $t_i \in \{0,1\}^n$ along with a randomness $r_i \in \{0,1\}^n$ in the signature, which satisfy $t_i = f(\mathsf{sk}_i, r_i) \in \{0,1\}^n$. Here, $f \in \mathcal{F}_{\mathsf{Rijn}}$. In other words, $t_i = \mathsf{Rijn}(\mathsf{sk}_i, r_i) \oplus r_i$.
- The prover knows a secret tag t_i^{join} and a secret challenge c_i that satisfy the same PRF constraint, that is $t_i^{\mathsf{join}} = f(\mathsf{sk}_i, c_i) = \mathsf{Rijn}(\mathsf{sk}_i, c_i) \oplus c_i$.
- $r \neq c_i$.
- The prover knows a witness wit_{x_i} satisfying $\mathsf{AVerify}(H_{\mathsf{DM}}, \Lambda, x_i, \mathsf{wit}_{x_i})$, where $x_i = (t_i^{\mathsf{join}}, c_i) \in \{0,1\}^n \times \{0,1\}^n$.
- All signatures on the signature revocation list are not generated by the signer's secret key sk_i. That is, for each $\Sigma_j \in \mathsf{SRL}$, $t_{\Sigma_j} \neq f(\mathsf{sk}_i, r_{\Sigma_j}) = \mathsf{Rijn}(\mathsf{sk}_i, r_{\Sigma_j}) \oplus r_{\Sigma_j}$. Note that a signature is of the form $\Sigma_j = (t_{\Sigma_j}, r_{\Sigma_j}, \cdots)$.

First, note that we applied H_{DM} on $(t_i^{\mathsf{join}}, c_i)$ to get the leaf value $H_{\mathsf{DM}}(t_i^{\mathsf{join}}, c_i) = \mathsf{Rijn}(t_i^{\mathsf{join}}, c_i) \oplus c_i$. Therefore, q_0 (7) auxiliary s_{out} are required for the evaluation of H_{DM}. Reformulating the remaining constraints for the Merkle tree accumulator follows from Section 3.2, which contains extended witnesses $\{k_i, v_i, w_i, g_i : i \in [1, a]\}$ and $q_0 \cdot a$ auxiliary s_{out}.

In the following, we only specify how to prove the newly appeared relations in the VOLEitH system. We start by introducing intermediate witness $t_{i,\Sigma_j} \in \{0,1\}^n$, which is the output of $\mathsf{Rijn}(\mathsf{sk}_i, r_{\Sigma_j}) \oplus r_{\Sigma_j}$, for each $\Sigma_j \in \mathsf{SRL}$. It is worth noting that t_{i,Σ_j} must be kept secret. Otherwise, signatures generated by the same secret key sk_i would share the same t_{i,Σ_j} for the same revoked signature Σ_j. In other words, signatures generated by the same sk_i are linkable.

We then observe that the relations to be proven fall into two categories. The first comprises equality constraints, which require proving the possession of $\mathsf{sk}_i, c_i, t_i^{\mathsf{join}}, \{t_{i,\Sigma_j}\}_j$ such that

$$\mathsf{Rijn}(\mathsf{sk}_i, r_i) \oplus r_i = t_i; \tag{8}$$

$$\mathsf{Rijn}(\mathsf{sk}_i, c_i) \oplus c_i = t_i^{\mathsf{join}}; \tag{9}$$

$$\mathsf{Rijn}(\mathsf{sk}_i, r_{\Sigma_j}) \oplus r_{\Sigma_j} = t_{i,\Sigma_j} \text{ for each } \Sigma_j \in \mathsf{SRL}. \tag{10}$$

Recall that in the protocol for the Merkle tree accumulator from Section 3.2, we have described how to prove the correct evaluation of $\mathsf{Rijn}(\cdot, \cdot)$. Therefore, we

need to introduce q_1 extra s_{out} for proving (8), (9), (10).

$$q_1 = \underbrace{(R-1)\cdot\frac{n}{8}\cdot 2}_{\text{for enc. sche. of (8),(9)}} + \underbrace{|\mathsf{SRL}|\cdot(R-1)\cdot\frac{n}{8}}_{\text{for enc. sche. of (10)}} + \underbrace{8\cdot R}_{\text{for } \mathsf{sk}_i \text{ expan.}}$$

The second category consists of inequality constraints. In particular, the prover needs to demonstrate the knowledge of $c_i, \{t_{i,\Sigma_j}\}_j$ such that

$$r_i \neq c_i; \tag{11}$$

$$t_{\Sigma_j} \neq t_{i,\Sigma_j} \quad \text{for each } \Sigma_j \in \mathsf{SRL}. \tag{12}$$

Let $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ and $b = (b_1, \ldots, b_n) \in \{0,1\}^n$. Then $a \neq b$ is equivalent to $\vee_{i=1}^{n}(a_i \oplus b_i) - 1 = 0$. Since $a_i \vee b_i = a_i \cdot b_i + a_i + b_i \bmod 2$, evaluation of $\vee_{i=1}^{n}(a_i \oplus b_i)$ involves $n-1$ multiplications. Let $u_1 = (a_1 \oplus b_1) \vee (a_2 \oplus b_2)$, and $u_j = u_{j-1} \vee (a_{j+1} \oplus b_{j+1})$ for $j \in [2, n-2]$. Note that the output of the last disjunction is 1. Therefore, we introduce extra $(n-2)$ bits for proving $a \neq b$. To show (11) and (12) are satisfied, $t_{\mathsf{neq}} = \big((n-2) + (n-2)\cdot|\mathsf{SRL}|\big)$-bit witness is needed.

To summarize, we have transformed all the constraints into a sequence of degree-2 polynomial constraints, with extended witnesses $\{s_{\mathsf{out},j} : j \in [1, q_0 + q_0 \cdot a + q_1]\} \cup \{k_i, v_i, w_i, g_i : i \in [1,a]\} \cup \{\mathsf{sk}_i, c_i, t_i^{\mathsf{join}}\} \cup \{t_{i,\Sigma_j} : \Sigma_j \in \mathsf{SRL}\} \cup \{u_j : j \in [1, t_{\mathsf{neq}}]\}$. The total size is $\hat{l} = 8(q_0 + q_0 a + q_1) + (a + 3na) + 3n + |\mathsf{SRL}| \cdot n + t_{\mathsf{neq}}$. The prover then runs the VOLEitH protocol [5].

4.3 Efficiency Analysis of Our EPID Signatures

In Table 3, we present proof sizes of our EPID signatures under various parameters. For the VOLEitH parameters, we follow those in [5, Table 3.2] to achieve 128-bit security level. Different values of $(\tau, \mathsf{T}_{\mathsf{open}})$ offer tradeoffs between communication and computation. Regarding the hash function H_{DM} and the PRF function f, we choose Rijndael-256 to instantiate them. Table 3 shows that proof sizes are less than 320 KB for group sizes up to 2^{30} and $|\mathsf{SRL}| = 0$. Nevertheless, proof sizes depend linearly on the size of signature revocation list, and increase to 7.6 MB when $|\mathsf{SRL}| = 1000$.

We then measure the performance of our EPID construction on a workstation equipped with an AMD EPYC 9754 processor (2.3 GHz) and 512 GB of memory. Our implementation is an architecture-specific C/C++ code optimized for x86-64 processors with AES-NI support, leveraging OpenMP for parallelization. Running times were averaged over 50 runs. Our implementation is publicly available [2] and open source. The benchmark results are shown in Table 4. When $(\tau, \mathsf{T}_{\mathsf{open}}) = (11, 103)$, the signing time and verification time are less than 280 ms for $|\mathsf{SRL}| = 0$, and increase to 3.4 seconds for $\mathsf{SRL} = 1000$. When $(\tau, \mathsf{T}_{\mathsf{open}}) = (16, 112)$, the times reduce to less than 100 ms for $|\mathsf{SRL}| = 0$ and less than 660 ms for $|\mathsf{SRL}| = 1000$. As we can see from Table 3 and Table 4,

[2] `https://github.com/ddlfi/EPID2025`

Table 3. Proof sizes of our EPID signatures for different values of $|\mathsf{SRL}|$ and group size $N = 2^a$.

λ	a	τ	$\mathsf{T}_{\mathsf{open}}$	proof sizes (KB)				
				$\|\mathsf{SRL}\| = 0$	$\|\mathsf{SRL}\| = 10$	$\|\mathsf{SRL}\| = 100$	$\|\mathsf{SRL}\| = 500$	$\|\mathsf{SRL}\| = 1000$
128	5	11	103	53.1	104.6	568.5	2629.9	5206.7
		16	112	76.6	151.6	826.2	3824.7	7572.7
	10	11	103	86.6	138.2	602.0	2663.4	5240.2
		16	112	125.4	200.3	875.0	3873.4	7621.5
	15	11	103	120.1	171.7	635.5	2696.9	5273.7
		16	112	174.1	249.1	923.7	3922.2	7670.2
	20	11	103	153.7	205.2	669.0	2730.5	5307.2
		16	112	222.9	297.9	972.5	3970.9	7719.0
	30	11	103	220.7	272.2	736.0	2797.5	5374.2
		16	112	320.4	395.4	1070.0	4068.5	7816.5

increasing the signature sizes by around 50% leads to $60\% - 80\%$ reductions in running times. This shows the tradeoff between proof sizes and running times. In addition, the proof of non-revocation significantly contributes to the overall complexity. This phenomenon is consistent with previous works, e.g., [31,17].

Table 4. Running times of our EPID signatures for different values of $|\mathsf{SRL}|$ and group size $N = 2^a$.

λ	a	τ	$\mathsf{T}_{\mathsf{open}}$	(signing time, verification time) ms				
				$\|\mathsf{SRL}\| = 0$	$\|\mathsf{SRL}\| = 10$	$\|\mathsf{SRL}\| = 100$	$\|\mathsf{SRL}\| = 500$	$\|\mathsf{SRL}\| = 1000$
128	5	11	103	$(143, 144)$	$(179, 175)$	$(419, 421)$	$(1554, 1549)$	$(2975, 2926)$
		16	112	$(53, 57)$	$(65, 69)$	$(94, 99)$	$(310, 311)$	$(569, 555)$
	10	11	103	$(164, 163)$	$(196, 197)$	$(452, 452)$	$(1583, 1580)$	$(2963, 2930)$
		16	112	$(55, 61)$	$(65, 70)$	$(95, 104)$	$(322, 312)$	$(573, 564)$
	15	11	103	$(192, 188)$	$(225, 223)$	$(484, 487)$	$(1645, 1644)$	$(3039, 3017)$
		16	112	$(57, 62)$	$(69, 73)$	$(97, 107)$	$(324, 322)$	$(585, 575)$
	20	11	103	$(207, 206)$	$(245, 251)$	$(511, 510)$	$(1666, 1660)$	$(3091, 3048)$
		16	112	$(58, 65)$	$(71, 75)$	$(101, 114)$	$(331, 326)$	$(593, 587)$
	30	11	103	$(275, 278)$	$(334, 332)$	$(584, 584)$	$(1752, 1724)$	$(3374, 3325)$
		16	112	$(97, 100)$	$(111, 116)$	$(154, 165)$	$(369, 372)$	$(655, 639)$

4.4 Security Analysis of Our EPID Signatures

In this section, we prove the correctness, anonymity, and unforgeability of our EPID construction from Section 4.1. In Theorem 1–3, we aim to make the statements as general as possible without specifying the concrete choices of the proof system Π and signature scheme $\mathcal{SIG}$. These two ingredients can be easily replaced by more efficient choices as long as they satisfy the required properties.

Theorem 1. *Let Π be any complete proof system that accepts degree-2 polynomial constraints, and $\mathcal{SIG}$ be any correct signature scheme. Assume that the pseudorandom function family $\mathcal{F}_{\mathsf{Rijn}}$ is cryptographically secure and collision resistant over its key space, and that the hash-based accumulator $\mathsf{Acc}_{\mathsf{Rijn}}$ is correct, then our EPID scheme satisfies correctness.*

Proof. First, note that the employed VOLEitH proof system is complete and FAEST signature scheme is correct. Let S_i be the set of signatures generated by group member $\mathcal{P}_i$ who has joined the system successfully via the interactive Join and Rejoin algorithms. Let sk_i and $\mathsf{cert}_i = (t_i^{\mathsf{join}}, c_i, \mathsf{wit}_{x_i})$ be the obtained secret key and certificate of $\mathcal{P}_i$, respectively. The constructed EPID scheme is correct if $\mathsf{sk}_i \notin \mathsf{KRL}$, $S_i \cap \mathsf{SRL} = \emptyset$, and $\Sigma_i \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}_i, \mathsf{cert}_i, \Lambda, \sigma_\Lambda, m, \mathsf{SRL})$, the following holds:

$$\Pr\left[\ \mathsf{Verify}(\ \mathsf{gpk},\ m, \Sigma_i,\ \mathsf{KRL},\ \mathsf{SRL}\ ) = 1\ \right] = 1 - \mathsf{negl}(\lambda).$$

Denote $\Sigma_i = (t, r, \pi, \Lambda, \sigma_\Lambda)$. Since $\mathcal{P}_i$ is an honest member, then the tuple $(\Lambda, \sigma_\Lambda)$ is obtained from the group manager, and $t = f(\mathsf{sk}_i, r)$ is honestly computed from sk_i. Due to the correctness of the FAEST signature scheme, $\mathsf{FAEST.Verify}(\mathsf{opk}, \Lambda, \sigma_\Lambda) = 1$.

Next, we show that $\mathcal{P}_i$ possesses a valid witness $\eta = (\mathsf{sk}_i, t_i^{\mathsf{join}}, c_i, \mathsf{wit}_{x_i})$ such that the following constraints (13) hold with all but negligible probability.

$$\begin{cases} t_i^{\mathsf{join}} = f(\mathsf{sk}_i, c_i), \\ t = f(\mathsf{sk}_i, r), \quad r \neq c_i, \\ \mathsf{AVerify}(H_{\mathsf{DM}}, \Lambda, (t_i^{\mathsf{join}}, c_i), \mathsf{wit}_{x_i}) = 1 \\ \text{for each } \Sigma_j \in \mathsf{SRL}, t_{\Sigma_j} \neq f(\mathsf{sk}_i, r_{\Sigma_j}) \end{cases} \tag{13}$$

The exception is the event E_1, that some revoked signature Σ_j satisfies $t_{\Sigma_j} = f(\mathsf{sk}_i, r_{\Sigma_j})$, occurs. Since $S_i \cap \mathsf{SRL} = \emptyset$, those revoked signatures are not generated via the secret key sk_i. Particularly, $t_{\Sigma_j} = f(\mathsf{sk}_{\Sigma_j}, r_{\Sigma_j})$ with $\mathsf{sk}_{\Sigma_j} \neq \mathsf{sk}_i$. Therefore, the event E_1 happens with negligible probability due to the collision resistance of $\mathcal{F}_{\mathsf{Rijn}}$ over its key space. As a result, the completeness of the proof system Π implies that $\Pi.\mathsf{Verify}(\mathsf{crs}, \xi, \pi) = 1$, where $\xi = (\mathsf{gpk}, t, r, m, \mathsf{SRL}, \Lambda, \sigma_\Lambda)$.

Now we argue that a revoked key sk_j will not satisfy the relation $t = f(\mathsf{sk}_j, r)$. Otherwise, we have $f(\mathsf{sk}_i, r) = f(\mathsf{sk}_j, r)$ with $\mathsf{sk}_i \neq \mathsf{sk}_j$. This will then violate the collision resistance of $\mathcal{F}_{\mathsf{Rijn}}$ over its key space.

To summarize, our EPID scheme satisfies correctness.

Theorem 2. *Let Π be any zero-knowledge proof system that accepts degree-2 polynomial constraints. Assume that the pseudorandom function family $\mathcal{F}_{\mathsf{Rijn}}$ is cryptographically secure, then our EPID scheme satisfies anonymity.*

The high-level intuition for anonymity is as follows. A signature Σ is of the form $(t_i, r_i, \pi, \Lambda, \sigma_\Lambda)$, where $t_i = f(\mathsf{sk}_i, r_i)$. Then the pseudorandomness of the family $\mathcal{F}_{\mathsf{Rijn}}$ ensures that (t_i, r_i) does not reveal any identity information about the signer. Next, the zero-knowledge property of the VOLEitH proof system guarantees that π reveals no information about the signer. Finally, $(\Lambda, \sigma_\Lambda)$ are public, and leak no information about the signer except that the signer is among the set accumulated in Λ. This is the reason that we require $(\Lambda_{i_0}, \sigma_{\Lambda_{i_0}}) = (\Lambda_{i_1}, \sigma_{\Lambda_{i_1}})$ to avoid trivial attacks in experiment $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$. The details of the proof are deferred to Appendix A.1.

Theorem 3. *Let Π be any simulation-sound extractable zero-knowledge proof system that accepts degree-2 polynomial constraints, and $\mathcal{SIG}$ be any strongly unforgeable siganture scheme. Assume that the pseudorandom function family $\mathcal{F}_{\mathsf{Rijn}}$ is cryptographically secure and collision resistant over its key space, the hash-based accumulator $\mathsf{Acc}_{\mathsf{Rijn}}$ is collision free, then our EPID scheme satisfies unforgeability.*

The high-level intuition for unforgeability is as follows. Assume that the adversary $\mathcal{A}$ has not obtained a certificate from the group manager for $x = (t^{\mathsf{join}}, c)$, implying that x is not accumulated in any Λ that is signed by $\mathcal{GM}$. Then $\mathcal{A}$ can output a forgery by either generating a valid witness wit_x for a signed Λ or forging a signature σ_Λ for a new Λ that accumulated x. This, however, would break the collision freeness of the accumulator or the unforgeability of $\mathcal{SIG}$. Let $\Sigma = (t, r, \pi, \Lambda, \sigma_\Lambda)$ be a valid signature with respect to m, SRL, and KRL, returned by a **Sign** query. A simpler strategy for $\mathcal{A}$ to output a forgery is to produce a new valid signature σ' for Λ and then to output m, $(t, r, \pi, \Lambda, \sigma')$, SRL and KRL. But this would break the strong unforgeability of $\mathcal{SIG}$. Next, if $\mathcal{A}$ does not know the secret key sk for some $t^{\mathsf{join}} = f(\mathsf{sk}, c)$, then $\mathcal{A}$ cannot produce $f(\mathsf{sk}, r)$ for a fresh r even if it sees many signatures signed by sk. This is guaranteed by the pseudorandomness of the family $\mathcal{F}_{\mathsf{Rijn}}$. Finally, collision resistance of $\mathcal{F}_{\mathsf{Rijn}}$ over its key space ensures that $\mathcal{A}$ cannot produce a different key sk' satisfying $f(\mathsf{sk}', c) = f(\mathsf{sk}, c)$ for a revoked key sk. Details are in Appendix A.2.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions. This work is partially supported by the National Cryptologic Science Fund of China under grant 2025NCSF02039, and by the National Science Foundation of China under grants 12401693, 62572305, and 12361141818. This work is also supported by the open project funding of the Key Laboratory of Intelligent Sensing System and Security (Ministry of Education), Hubei University.

References

1. ISO/IEC 20008-2:2013: Information technology - security techniques - anonymous digital signatures - part 2: Mechanisms using a group public key. *Standard, International Organization for Standardization, Geneva, CH*, 2013. `https://www.iso.org/standard/56916.html`.
2. M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 430–454. Springer, 2015.
3. S. Argo, T. Güneysu, C. Jeudy, G. Land, A. Roux-Langlois, and O. Sanders. Practical post-quantum signatures for privacy. In *CCS 2024*, pages 1523–1537. ACM, 2024.
4. R. Avanzi, B. Chakraborty, and E. List. The large block cipher family vistrutah. *IACR Cryptol. ePrint Arch.*, page 976, 2025. To appear in ToSC 2026.
5. C. Baum, W. Beullens, L. Braun, C. D. de Saint Guilhem, M. Klooß, C. Majenz, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl. Faest: algorithm specifications. Technical report, Technical report, National Institute of Standards and Technology, 2023. `https://faest.info/faest-spec-v2.0.pdf`.
6. C. Baum, W. Beullens, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl. One tree to rule them all: Optimizing GGM trees and owfs for post-quantum signatures. In *ASIACRYPT 2024*, volume 15484 of *LNCS*, pages 463–493. Springer, 2024.
7. C. Baum, L. Braun, C. D. de Saint Guilhem, M. Klooß, E. Orsini, L. Roy, and P. Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In *CRYPTO 2023*, volume 14085 of *LNCS*, pages 581–615. Springer, 2023.
8. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
9. D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe. The sphincs$^+$ signature framework. In *ACM CCS 2019,*, pages 2129–2146. ACM, 2019.
10. D. Boneh, S. Eskandarian, and B. Fisch. Post-quantum EPID signatures from symmetric primitives. In *CT-RSA 2019*, volume 11405 of *LNCS*, pages 251–271. Springer, 2019.
11. E. Brickell and J. Li. Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In *WPES 2007*, pages 21–30. ACM, 2007.
12. E. Brickell and J. Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. *Int. J. Inf. Priv. Secur. Integr.*, 1(1):3–33, 2011.
13. E. Brickell and J. Li. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *IEEE Trans. Dependable Secur. Comput.*, 9(3):345–360, 2012.
14. E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *CCS 2004*, pages 132–145. ACM, 2004.
15. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
16. L. Chen, C. Dong, N. E. Kassem, C. J. P. Newton, and Y. Wang. Hash-based direct anonymous attestation. In *PQCrypto 2023*, volume 14154 of *LNCS*, pages 565–600. Springer, 2023.

17. L. Chen, C. Dong, N. E. Kassem, C. J. P. Newton, and Y. Wang. A new hash-based enhanced privacy ID signature scheme. In *PQCrypto 2024*, volume 14771 of *LNCS*, pages 37–71. Springer, 2024.
18. L. Chen, C. Dong, C. J. P. Newton, and Y. Wang. Sphinx-in-the-head: Group signatures from symmetric primitives. *ACM Trans. Priv. Secur.*, 27(1):11:1–11:35, 2024.
19. L. Chen, Z. Xu, T. Tu, and Z. Wang. Lattice-based privacy enhanced identity protocol for sdo services. In *ICSIP 2023*, pages 609–613. IEEE, 2023.
20. V. Costan and S. Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, page 86, 2016.
21. J. Daemen and V. Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.
22. F. Dall, G. De Micheli, T. Eisenbarth, D. Genkin, N. Heninger, A. Moghimi, and Y. Yarom. Cachequote: Efficiently recovering long-term secrets of sgx epid via cache attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2), 2018.
23. D. Derler, C. Hanser, and D. Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *CT-RSA 2015*, volume 9048 of *LNCS*, pages 127–144. Springer, 2015.
24. D. Derler, S. Ramacher, and D. Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto 2018*, volume 10786 of *LNCS*, pages 419–440. Springer, 2018.
25. A. Faonio, D. Fiore, L. Nizzardo, and C. Soriente. Subversion-resilient enhanced privacy ID. In *CT-RSA 2022*, volume 13161 of *LNCS*, pages 562–588. Springer, 2022.
26. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
27. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.
28. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC 2007*, pages 21–30. ACM, 2007.
29. C. Jeudy and O. Sanders. Worst-case lattice sampler with truncated gadgets and applications. *IACR Cryptol. ePrint Arch.*, page 1952, 2024.
30. C. Jeudy and O. Sanders. Improved lattice blind signatures from recycled entropy. In *CRYPTO 2025*, volume 16000 of *LNCS*, pages 477–513. Springer, 2025.
31. C. Jeudy and O. Sanders. Lattice epid with efficient revocation. *IACR Cryptol. ePrint Arch.*, page 1225, 2025.
32. S. Johnson, V. Scarlata, C. Rozas, E. Brickell, F. Mckeen, et al. Intel software guard extensions: Epid provisioning and attestation services. *White Paper*, 1(1-10):119, 2016.
33. N. E. Kassem, L. Fiolhais, P. Martins, L. Chen, and L. Sousa. A lattice-based enhanced privacy ID. In *WISTP 2019*, volume 12024 of *LNCS*, pages 15–31. Springer, 2019.
34. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *CCS 2018*, pages 525–537. ACM, 2018.
35. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. *J. Cryptol.*, 36(3):23, 2023.

36. F. Liu, T. Isobe, and W. Meier. Cryptanalysis of full lowmc and lowmc-m with algebraic techniques. In *CRYPTO 2021*, volume 12827 of *LNCS*, pages 368–401. Springer, 2021.
37. F. Liu, S. Sarkar, G. Wang, W. Meier, and T. Isobe. Algebraic meet-in-the-middle attack on lowmc. In *ASIACRYPT 2022* -, volume 13791 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2022.
38. V. Lyubashevsky, G. Seiler, and P. Steuer. The lazer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In *CCS 2024*, pages 3125–3137. ACM, 2024.
39. Y. Ouyang, D. Tang, and Y. Xu. Code-based zero-knowledge from vole-in-the-head and their applications: Simpler, faster, and smaller. In *ASIACRYPT 2024*, volume 15488 of *LNCS*, pages 436–470. Springer, 2024.
40. T. Prest, F.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. Falcon. *Technical Report, NIST*, 2022.
41. O. Sanders. EPID with efficient proof of non-revocation. In *ASIA CCS 2022*, pages 1126–1138. ACM, 2022.
42. O. Sanders and J. Traoré. EPID with malicious revocation. In *CT-RSA 2021*, volume 12704 of *LNCS*, pages 177–200. Springer, 2021.
43. Y. Sun, J. Cui, and M. Wang. Improved attacks on lowmc with algebraic techniques. *IACR Trans. Symmetric Cryptol.*, 2023(4):143–165, 2023.
44. K. Yang, P. Sarkar, C. Weng, and X. Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *CCS 2021*, pages 2986–3001. ACM, 2021.

A Deferred Security Proofs

A.1 Proof of Theorem 2

Proof. First, note that the employed VOLEitH proof system is zero-knowledge. We prove the theorem via a sequence of games G_0, $G_0^{(1)}$, $G_0^{(2)}$, $G_0^{(3)}$, $G^{(4)}$, $G_1^{(3)}$, $G_1^{(2)}$, $G_1^{(1)}$, G_1. We define W_b or $W_b^{(i)}$ or $W^{(4)}$ to be the event that the corresponding game outputs 1. The first game G_0 is the anonymity experiment $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$ where the bit b chosen by the challenger is 0 while the last game G_1 is $\mathsf{Exp}^{\mathsf{anony-b}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda)$ where the bit b chosen by the challenger is 1. Therefore, $\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{anon-0}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]$ and $\Pr[W_1] = \Pr[\mathsf{Exp}^{\mathsf{anon-1}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]$. We now describe the remaining games and show that they are indistinguishable.

Game $G_b^{(1)}$. This game is similar to G_b, except that at the start of the game, challenger $\mathcal{C}$ chooses two distinct indices $a_0, a_1 \in [1, N]$ and the game returns $\perp$ if $\mathcal{A}$ does not choose a_0, a_1 in the challenge phase. Here N is an upper bound on the number of group members in the system. Since a_0, a_1 are chosen uniformly at random and independent of the choices of $\mathcal{A}$, $\Pr[W_b^{(1)}] = \frac{1}{N^2}\Pr[W_b]$.

Game $G_b^{(2)}$. This game is the same as the Game $G_b^{(1)}$, but the proofs in the signatures are produced via the simulator of the VOLEitH proof system Π. It is straightforward to see that Game $G_b^{(2)}$ and Game $G_b^{(1)}$ are indistinguishable

due to the zero-knowledge property of the proof system Π. We thus have $|\Pr[W_b^{(2)}] - \Pr[W_b^{(1)}]| \leq \mathsf{negl}(\lambda)$.

Game $G_b^{(3)}$. This game is the same as the Game $G_b^{(2)}$, but the value $f(\mathsf{sk}_{a_0}, \cdot)$ is replaced by a random string. In more detail, when $\mathcal{A}$ requests creation of a new group member $\mathcal{P}_{a_0}$, where $\mathcal{A}$ plays the role of the group manager, the challenger (on behalf of $\mathcal{P}_{a_0}$) sends a random string $t_{a_0}^{\mathsf{join}}$ to $\mathcal{A}$ upon receiving c_{a_0}. Also, when $\mathcal{A}$ requests a signature on a message from $\mathcal{P}_{a_0}$, the challenger samples a random t and generates a simulated proof as in Game $G_b^{(2)}$. Therefore, any PPT adversary is infeasible to distinguish Game $G_b^{(3)}$ from Game $G_b^{(2)}$ due to the pseudorandomness of the function family $\mathcal{F}_{\mathsf{Rijn}}$. Thus, $|\Pr[W_b^{(3)}] - \Pr[W_b^{(2)}]| \leq \mathsf{negl}(\lambda)$.

Game $G^{(4)}$. We modify Game $G_b^{(3)}$ by replacing all queries to $f(\mathsf{sk}_{a_1}, \cdot)$ with random strings. By the pseudorandomness of $\mathcal{F}_{\mathsf{Rijn}}$, $|\Pr[W^{(4)}] - \Pr[W_b^{(3)}]| \leq \mathsf{negl}(\lambda)$. Note that $G^{(4)}$ does not depend on the bit b anymore.

To summarize, we have established a chain of indistinguishable games

$$G_0^{(1)} \approx G_0^{(2)} \approx G_0^{(3)} \approx G^{(4)} \approx G_1^{(3)} \approx G_1^{(2)} \approx G_1^{(1)}.$$

In other words, $|\Pr[W_0^{(1)}] - \Pr[W_1^{(1)}]| \leq \mathsf{negl}(\lambda)$. Also, $\Pr[W_b^{(1)}] = \frac{1}{N^2}\Pr[W_b]$ for $b \in \{0,1\}$. As a result,

$$\begin{aligned} |\Pr[\mathsf{Exp}_{\mathsf{EPID},\mathcal{A}}^{\mathsf{anon}-0}(1^\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathsf{EPID},\mathcal{A}}^{\mathsf{anon}-1}(1^\lambda) = 1]| &= |\Pr[W_0] - \Pr[W_1]| \\ &= N^2 |\Pr[W_0^{(1)}] - \Pr[W_1^{(1)}]| \\ &\leq \mathsf{negl}(\lambda). \end{aligned}$$

This ends the proof.

A.2 Proof of Theorem 3

Proof. First, note that the employed VOLEitH proof system is simulation-sound extractable and zero-knowledge, and the FAEST signature scheme is strongly unforgeable. We prove the theorem via a series of hybrid games G_0, G_1, …, G_7. Define W_i to be the event that Game G_i outputs 1.

Game G_0. This is the experiment $\mathsf{Exp}_{\mathsf{EPID},\mathcal{A}}^{\mathsf{unfor}}(1^\lambda)$. Let $\mathcal{C}$ be the challenger and $\mathcal{A}$ be a PPT adversary. In this experiment, $\mathcal{A}$ receivers gpk from the challenger $\mathcal{C}$ and has access to **Join**, **Rejoin**, **Sign**, **Corrupt** queries. $\mathcal{C}$ also maintains $\mathcal{L}_{\mathsf{cert}}$, $\mathcal{L}_{\mathsf{sig}}$, and a set $\mathcal{L}_{\mathsf{CU}}$ of corrupted group members as defined in $\mathsf{Exp}_{\mathsf{EPID},\mathcal{A}}^{\mathsf{unfor}}(1^\lambda)$. At some point, $\mathcal{A}$ outputs a forgery $(m^*, t^*, r^*, \pi^*, \Lambda^*, \sigma_\Lambda^*)$, together with two revocation lists: $\mathsf{KRL}^*, \mathsf{SRL}^*$. This game outputs 1, i.e., $\mathcal{A}$ wins the unforgeability game, if

$$\mathsf{Verify}(\mathsf{gpk}, m^*, (t^*, r^*, \pi^*, \Lambda^*, \sigma_\Lambda^*), \mathsf{KRL}^*, \mathsf{SRL}^*) = 1; \tag{14}$$

for any $i \in \mathcal{L}_{\mathsf{CU}}$, either $\mathsf{sk}_i \in \mathsf{KRL}^*$ or $S_i \cap \mathsf{SRL}^* \neq \emptyset$, and $(t^*, r^*, \pi^*, \Lambda^*, \sigma^*_\Lambda) \notin \mathcal{L}_{\mathsf{sig}}$. Here, S_i is the set of signatures generated by corrupted group member $\mathcal{P}_i$. In this game, $\mathcal{C}$ additionally maintains a list $\mathcal{L}_{\mathsf{join}}$, which contains each accumulated set $\mathcal{X}$, the corresponding accumulated value Λ and its signature σ_Λ from **Join** and **Rejoin** queries. By definition, $\Pr[W_0] = \Pr[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]$.

Game G_1**.** Let E_1 be the event that $(\Lambda,\ \sigma_\Lambda)$ among valid group membership $(t^{\mathsf{join}},\ c,\ \mathsf{wit},\ \Lambda,\ \sigma_\Lambda)$ received from the **Rejoin** queries is not from $\mathcal{L}_{\mathsf{join}}$. This game follows from Game G_0 except that it outputs 0 if event E_1 happens. However, E_1 violates the strong unforgeability of the employed signature scheme FAEST. Thus, Game G_1 and Game G_0 are computationally indistinguishable, indicating $|\Pr[W_1] - \Pr[W_0]| \leq \mathsf{negl}(\lambda)$.

Game G_2**.** Let E_2 be the event that $(t^{\mathsf{join}},\ c)$ among valid group membership $(t^{\mathsf{join}},\ c,\ \mathsf{wit},\ \Lambda,\ \sigma_\Lambda)$ received from the **Rejoin** queries does not belong to the set $\mathcal{X}$ corresponding to $(\Lambda, \sigma_\Lambda)$. This game is identical to Game G_1 except that it outputs 0 if event E_2 occurs. It is not difficult to see that E_2 violates the collision freeness of the accumulator. Then Game G_2 and Game G_1 are computationally indistinguishable. Thus, $|\Pr[W_2] - \Pr[W_1]| \leq \mathsf{negl}(\lambda)$.

Game G_3**.** This is Game G_2 except that this game outputs 0 if event E_3, that $(\Lambda^*, \sigma^*_\Lambda)$ in the forgery is not from $\mathcal{L}_{\mathsf{join}}$, occurs. Following from the indistinguishability of Game G_1 and G_0, Game G_3 and Game G_2 are computationally indistinguishable. Thus, $|\Pr[W_3] - \Pr[W_2]| \leq \mathsf{negl}(\lambda)$.

Game G_4**.** This game follows Game G_3, but we run the simulated setup of the proof system Π. Also, for all signature queries, each proof π is produced via the simulator of the proof system Π. We then see that Game G_4 and Game G_3 are indistinguishable due to the zero-knowledge of the proof system Π. Therefore, $|\Pr[W_4] - \Pr[W_3]| \leq \mathsf{negl}(\lambda)$.

Game G_5**.** This game modifies Game G_4 as follows. Let $(m^*, t^*, r^*, \pi^*, \Lambda^*, \sigma^*_\Lambda)$ be a valid forgery output by the adversary $\mathcal{A}$, this game runs the extractor algorithm of the proof system Π and outputs 0 if extraction fails (E_5). Note that if $\mathcal{A}$ wins the unforgeability game, π^* is a valid proof for the statement $\xi^* = (\mathsf{gpk}, t^*, r^*, m^*, \mathsf{SRL}^*, \Lambda^*, \sigma^*_\Lambda)$. By the simulation-sound extractability of the proof system Π, extraction fails with negligible probability. We then obtain $|\Pr[W_5] - \Pr[W_4]| \leq \mathsf{negl}(\lambda)$. Let the extracted values be $(\mathsf{sk}^*, t^{\mathsf{join}*},\ c^*, \mathsf{wit}^*)$. By the simulation-sound extractability, the following constraints hold.

$$t^{\mathsf{join}*} = f(\mathsf{sk}^*, c^*), \tag{15}$$

$$t^* = f(\mathsf{sk}^*, r^*), \quad r^* \neq c^*, \tag{16}$$

$$\mathsf{AVerify}(H_{\mathsf{DM}}, \Lambda^*, \sigma^*_\Lambda, (t^{\mathsf{join}*}, c^*), \mathsf{wit}^*) = 1, \tag{17}$$

$$\text{for each } \Sigma_j \in \mathsf{SRL}^*, t_{\Sigma_j} \neq f(\mathsf{sk}^*, r_{\Sigma_j}). \tag{18}$$

Game G_6**.** This game is identical to Game G_5, except that this game returns 0 if event E_6, that the extracted $(t^{\mathsf{join}*}, c^*)$ does not belong to the set $\mathcal{X}^*$ corresponding to $(\Lambda^*, \sigma^*_\Lambda)$, happens. However, due to the collision freeness of the employed accumulator, this event occurs with negligible probability. Therefore, $|\Pr[W_6] - \Pr[W_5]| \leq \mathsf{negl}(\lambda)$.

Game G_7. This game is the same as the previous game, but this game outputs 0 if event E_7, that there exists a corrupted member $j \in \mathcal{L}_{\mathsf{CU}}$ satisfying $t^{\mathsf{join}*} = f(\mathsf{sk}_j, c^*)$, happens. In other words, $t^{\mathsf{join}*} = f(\mathsf{sk}^*, c^*) = f(\mathsf{sk}_j, c^*)$. However, the facts that (14) holds and event E_7 occurs imply $\mathsf{sk}^* \neq \mathsf{sk}_j$. This then violates the collision resistance of $\mathcal{F}_{\mathsf{Rijn}}$ over its key space. Therefore, $|\Pr[W_7] - \Pr[W_6]| \leq \mathsf{negl}(\lambda)$.

Therefore, we have showed that G_7 and G_0 are indistinguishable, and hence $|\Pr[W_7] - \Pr[W_0]| = |\Pr[W_7] - \Pr[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$. We now show that Game G_7 outputs 1 with negligible probability, assuming the pseudorandomness of the function family $\mathcal{F}_{\mathsf{Rijn}}$.

Let $\mathcal{A}$ be a PPT adversary in Game G_7. We now construct an adversary $\mathcal{B}$, which has internal access to $\mathcal{A}$, to break the pseudorandomness of the function family $\mathcal{F}_{\mathsf{Rijn}}$. To begin with, $\mathcal{B}$ is interacting with a PRF challenger who has either a pseudorandom function or a truly random function. $\mathcal{B}$ utilizes $\mathcal{A}$ as follows.

First, $\mathcal{B}$ selects a random index $n^* \in [1, N]$, where N is an upper on the number of group members in the system. Then $\mathcal{B}$ acts as the challenger in Game G_7 except that any queries to $f(\mathsf{sk}_{n^*}, \cdot)$ are replaced by the responses from the PRF challenger. In more detail, when $\mathcal{A}$ requests creation of a new group member $\mathcal{P}_{n^*}$, there are two cases to consider. If $\mathcal{B}$ runs the Join internally, $\mathcal{B}$ samples c_{n^*} as in Game G_7 and forwards c_{n^*} to its PRF challenger. Denote the response by $t^{\mathsf{join}}_{n^*}$. $\mathcal{B}$ then performs the remaining steps of the **Join** queries as in Game G_7. If $\mathcal{B}$ and $\mathcal{A}$ runs the Join together, $\mathcal{B}$ aborts and returns 0. $\mathcal{B}$ also aborts if $\mathcal{A}$ makes a corruption query of group member $\mathcal{P}_{n^*}$. Suppose $\mathcal{B}$ does not abort. When $\mathcal{A}$ further requests a signature on a message from $\mathcal{P}_{n^*}$, $\mathcal{B}$ samples $r \neq c_{n^*}$ and forwards r to its PRF challenger. Let the response be t. $\mathcal{B}$ then generates a simulated proof as in Game G_7. We remark that $\mathcal{A}$ may make **Rejoin** queries for $\mathcal{P}_{n^*}$. This will only update $(\mathsf{wit}_{x_{n^*}}, \Lambda, \sigma_\Lambda)$ while maintaining $(t^{\mathsf{join}}_{n^*}, c_{n^*})$ unchanged.

At some point, $\mathcal{A}$ outputs a forgery $(m^*, t^*, r^*, \pi^*, \Lambda^*, \sigma^*_\Lambda)$. If $\mathcal{A}$ wins the unforgeability game, $\mathcal{B}$ runs the extractor algorithm of the proof system Π and obtains $(\mathsf{sk}^*, t^{\mathsf{join}*}, c^*, \mathsf{wit}^*)$. If $((t^{\mathsf{join}*}, c^*), \mathsf{wit}^*, \Lambda^*, \sigma^*_\Lambda)$ is not equal to the output of $\mathcal{P}_{n^*}$ in Join or Rejoin algorithms, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ sends r^* to its PRF challenger and receives $\widetilde{t}$ back. If $\widetilde{t} = f(\mathsf{sk}^*, r^*) = t^*$, $\mathcal{B}$ returns 1 (interacting with a pseudorandom function) to its PRF challenger. Otherwise, $\mathcal{B}$ returns 0 (interacting with a truly random function). We remark that this is why we need to check $r \neq c$ when running the Sign algorithm. If this check is omitted, an adversary could set $r^* = c^*$ and render the query of r^* to the PRF challenger useless since $\mathcal{B}$ already learns the response.

Let us now calculate the advantage of $\mathcal{B}$ that attacks the pseudorandomness of the function family $\mathcal{F}_{\mathsf{Rijn}}$.

First, with probability at least $1/N$, $\mathcal{B}$ will not abort before $\mathcal{A}$ outputs a forgery. Note that n^* is chosen independently of $\mathcal{A}$'s view. Also, due to the indistinguishability of the games $G_0, \ldots, G_7$, we assume that events $E_1, E_2, E_3, E_5, E_6, E_7$ do not occur and constraints (15)-(18) are satisfied. This then implies that $((t^{\mathsf{join}*}, c^*), \mathsf{wit}^*, \Lambda^*, \sigma^*_\Lambda)$ is equal to the output of an honest member in

$mathsf{Join}$ or Rejoin algorithms. Such honest group member is $\mathcal{P}_{n^*}$ with probability at least $1/N$. To summarize, $\mathcal{B}$ will not abort with probability at least $1/N$.

Next, we argue that $\mathcal{B}$ successfully distinguishes a pseudorandom function from a truly random function, assuming that $\mathcal{B}$ does not abort.

If $\mathcal{B}$ is interacting with a pseudorandom function, then $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ as in Game G_7. We claim that key chosen by the PRF challenger, say sk, is exactly the extracted key sk^* with all but negligible probability. Suppose this is not true, then $f(\mathsf{sk}, c_{n^*}) = t^{\mathsf{join}}_{n^*}$ by the construction of $\mathcal{B}$. Also, $(t^{\mathsf{join}*}, c^*) = (t^{\mathsf{join}}_{n^*}, c_{n^*})$ with probability at least $1/N$. In addition, (15) is satisfied. As a result, $f(\mathsf{sk}, c_{n^*}) = f(\mathsf{sk}^*, c_{n^*})$, which then violates the collision resistance of $\mathcal{F}_{\mathsf{Rijn}}$ over its key space. Thus, $\mathsf{sk} = \mathsf{sk}^*$. This then implies that $\widetilde{t} = f(\mathsf{sk}, r^*) = f(\mathsf{sk}^*, r^*) = t^*$. $\mathcal{B}$ then always outputs 1 (interacting with a pseudorandom function).

If $\mathcal{B}$ is interacting with a truly random function, then $\widetilde{t}$ is a random string which collides with $t^* = f(\mathsf{sk}^*, r^*)$ with negligible probability. Then $\mathcal{B}$ returns 0 (interacting with a truly random function).

$$\begin{aligned}\mathsf{Adv}_{\mathsf{PRF},\mathcal{B}}(1^\lambda) &= |\Pr[\mathcal{B} \text{ returns } 1 \mid \text{PRF}] - \Pr[\mathcal{B} \text{ returns } 1 | \text{truely random}]| \\ &\geq |\frac{1}{N}\Pr[W_7] - \mathsf{negl}(\lambda)|.\end{aligned}$$

Due to the security of the pseudorandomness of $\mathcal{F}_{\mathsf{Rijn}}$, $\mathsf{Adv}_{\mathsf{PRF},\mathcal{B}}(1^\lambda)$ is negligible, so is $\Pr[W_7]$. Recall that we also have $|\Pr[W_7] - \Pr[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$. Hence, $\Pr[\mathsf{Exp}^{\mathsf{unfor}}_{\mathsf{EPID},\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$. This ends the proof.