Blockchain Governance via Sharp Anonymous Multisignatures*

Wonseok Choi[1], Xiangyu Liu[2], and Vassilis Zikas[3]

[1] DGIST
[2] CISPA
[3] Georgia Tech
wonseok@dgist.ac.kr, xiangyu.liu@cispa.de, vzikas@gatech.edu

Abstract. Electronic voting has occupied a large part of the cryptographic protocols literature. The recent reality of blockchains—in particular, their need for online governance mechanisms—has brought new parameters and requirements to the problem. We identify the key requirements of a blockchain governance mechanism, namely correctness (including eliminative double votes), voter anonymity, and traceability, and investigate mechanisms that can achieve them with minimal interaction and under assumptions that fit the blockchain setting.

First, we define a signature-like primitive, which we term *sharp anonymous multisignatures* (in short, ♯AMS) that tightly meets the needs of blockchain governance. In a nutshell, ♯AMSs allow any set of parties to generate a signature, e.g., on a proposal to be voted upon, which, if posted on the blockchain, hides the identities of the signers/voters but reveals their number. This can be seen as a (strict) generalization of threshold ring signatures (TRS).

We next turn to constructing such ♯AMSs and using them in various governance scenarios—e.g., single vote vs. multiple votes per voter. In this direction, although the definition of TRS does not imply ♯AMS, one can compile some existing TRS constructions into ♯AMS. This raises the question: What is the TRS structure that allows such a compilation? To answer the above, we devise templates for TRSs. Our templates encapsulate and abstract the structure that allows for the above compilation—most of the TRS schemes that can be compiled into ♯AMS are, in fact, instantiations of our template. This abstraction makes our template generic for instantiating TRSs and ♯AMSs from different cryptographic assumptions (e.g., DDH, LWE, etc.). One of our templates is based on chameleon hashes, and we explore a framework of lossy chameleon hashes to understand their nature fully.

Finally, we turn to how ♯AMS schemes can be used in our applications. We provide fast (in some cases non-interactive) ♯AMS-based blockchain governance mechanisms for a wide spectrum of assumptions on the honesty (semi-honest vs malicious) and availability of voters and proposers.

1 Introduction

Since the emergence of Bitcoin [65] in 2009, the world of cryptocurrencies and blockchain platforms has witnessed a surge in popularity. One of the distinguishing features of these blockchain platforms is their decentralized nature, wherein decision-making authority is distributed among various actors within the ecosystem.

There are two basic governance mechanisms: off-chain governance, as seen in Bitcoin and Ethereum, and on-chain governance, exemplified by projects like Algorand [22], Tezos [39], and EOS. While off-chain governance allows core contributors to work more seamlessly, it contradicts the philosophy of decentralization. Conversely, on-chain governance faces technical challenges, and this area of research is still relatively new and in its early stages.

Regardless of the governance mechanism used, blockchain platforms face a common vulnerability: community divisions, often resulting in hard forks. A hard fork arises when stakeholders disagree on a critical change, leading to some sticking with the current chain while others adopt the new one. Alternatively, competing updates may further fragment the community. These divisions can erode cohesion, devalue the platform, and jeopardize security. Security concerns are paramount, as a reduced number of resources supporting a fork can make it susceptible to attacks like 51% attacks.

* This work was done when authors were at Purdue University.

On-Chain Governance and Voting Systems for Improvement Proposals. On-chain governance has emerged as a solution to centralization issues associated with blockchain technology, involving all network nodes in decision-making. However, ensuring decentralization and preventing domination by a limited faction of developers and miners is crucial. The potential for conflicts and hard forks within the blockchain community drives this vigilance. At the core of on-chain governance is the voting mechanism, where stakeholders decide on improvement proposals for the ecosystem, potentially involving significant changes.

Generally, there are three periods for on-chain governance: the posting, voting, and announcement periods. The developers submit their improvement proposals to the blockchain during the posting period. Then, in the voting period, eligible voters participate in the voting protocol to vote for their preferred proposals. Finally, the voting result is announced in the announcement period, and the most voted proposal is elected.

There are several foundational requirements for a robust voting system.

Correctness. The accuracy of the voting result is perhaps the most important property. A key goal here is to prevent double-voting, wherein a voter casts more than one vote on the same or different proposals. This act of multiple-voting contradicts the standard single-vote setting, wherein each voter is restricted to voting only once, irrespective of the proposal chosen. We emphasize that the permissibility of multiple-voting is contingent upon the specific application. In this context, double-voting includes instances where a malicious voter attempts to cast more than one vote for a single proposal, and such duplications are promptly nullified.

(Unconditional) Anonymity of Voters. A crucial factor to consider in blockchain systems is their immutability. Once signatures are uploaded, they remain permanently etched in the system. This implies that over time, the authorship of a linkable or traceable ring signature [58, 33] could potentially be unveiled due to inadequacies in the underlying computational assumptions. Therefore, for optimal applicability in blockchain governance, the assurance of unconditional anonymity is one of the imperative features of a voting system in blockchain.

Traceability. If a malicious voter attempts to vote twice, we need to have an efficient mechanism to trace its identity. Note that this traceability property is a relaxation/fallback of the above strong correctness to allow for more practical constructions. Indeed, traceability is irrelevant if double-voting is infeasible.

Receipt-freeness. It should be impossible to generate a proof that a voter indeed cast an intended vote for others to see. This property has been widely studied [3, 4, 13, 27, 28, 42, 44, 46, 47, 53, 54, 55, 63, 66, 73].

Cryptographic Mechanisms for Blockchain Voting/Governance Systems. In pursuing our objective to design a voting system that ensures traceability and unconditional anonymity, we encounter inherent challenges when leveraging traditional cryptographic tools such as signature schemes or Multi-Party Computations (MPC).

Linkable Ring Signatures (LRS). Linkable ring signatures (LRS) [58] are a prevalent tool in constructing e-voting systems. In LRS, a user can sign a message on behalf of a group/ring while maintaining anonymity, as their identity remains concealed within the signature. Moreover, if a user signs twice for the same message and on behalf of the same group, these signatures are "linked", making it evident that they originate from a single actual signer. This inherent linkability property of LRS plays a vital role in preventing double-voting. However, it's important to note that the link algorithm in LRS does not disclose the specific identity of the signer when two signatures are linked. Consequently, LRS does not fulfill the traceability property, implying that a malicious voter engaging in double-voting may go unpunished. This lack of traceability can undermine the system's balance and fairness, necessitating additional measures to ensure accountability and uphold the integrity of the voting process.

Traceable Ring Signature. An alternative approach is to employ traceable ring signatures [33], where the link algorithm establishes the link between signatures and discloses the signer's identity. However, unconditional anonymity is compromised in traceable ring signatures, as proved in [23]. Striking a balance between traceability and unconditional anonymity in ring signatures appears challenging, presenting a fundamental trade-off within this cryptographic context.

Multi-Party Computation. Multi-party computation (MPC) [80, 38, 11, 20] appears to be the ultimate solution to the above voting problem. MPC allows n parties to compute any given function on their

inputs securely so that no malicious party (or coalition) can learn the inputs of other parties (privacy), and no party can affect the output any more than choosing their own input as such MPC can directly be used to realize our voting functionality by having each voter submit their votes and output the appropriate tally. MPC-privacy (which can be information-theoretic [11, 20, 70]) will ensure unconditional anonymity; MPC-correctness (for the appropriate function) can ensure our above voting correctness property. Traceability is a more elusive goal, but it can also be achieved by so-called identifiable MPC [45] (which ensures that upon aborting the identity of a cheater is revealed).[4]
Unfortunately, despite its very general functionality, MPC is also not the right solution to our problem: For starters, information-theoretic MPC needs an honest majority of the parties [38], an assumption which is unrealistic in our setting.[5] Even if one is willing to resort to security with (identifiable) abort and no fairness,[6] we still need to implement Byzantine broadcast—which requires in the worst case a polylogarithmic in the party-set size—or use again several on-chain rounds. Even given broadcast, turning an MPC protocol identifiable above into one with guaranteed output delivery (which is needed in our application) would require restarting the computation whenever it aborts (and potentially removing the identified cheater); this would again yield a larger number of rounds which is undesirable.

From Signature Schemes to an Interactive Structure. The above discussion highlights the inherent challenge of satisfying all requirements simultaneously. However, despite this seemingly conflicting nature, there is a way to address it. In this work, we provide a positive answer by proposing a structured approach embedded in the signature generation process. We carefully define an interactive structure——a simple protocol——within the signature generation. This structured approach is a key innovation that enables us to design a signature scheme that simultaneously conceals individual identities while revealing the number of signers involved. This dual property is a critical advancement for secure and reliable e-voting systems. We call this new signature protocol Sharp Anonymous Multisignatures ($\sharp$AMS). The term "sharp" is used in analogy with complexity theory, as in $\sharp P$, indicating that our signatures output the number of valid signers rather than a mere validity bit. Meanwhile, "anonymous" denotes that the signers' identities remain hidden. Equipped with these properties, $\sharp$AMS is a perfect fit for e-voting systems, especially in the context of blockchain governance. We discuss the details of our contributions below.

1.1 Our Contributions

$\sharp$ Anonymous MultiSignatures. Our first and major contribution is proposing and formalizing a new concept of signing *protocol*, dubbed $\sharp$AMS that tightly meets the needs of blockchain governance. The protocol allows any set of parties to collaborate jointly and outputs unconditionally anonymous signatures. To compare with threshold ring signatures (TRS), $\sharp$AMS does not need the threshold and always generates a valid signature regardless of the number of parties, and the verification algorithm reveals the number of parties. Regarding the number of parties as a threshold, which varies every signing, this can be seen as a strict generalization of TRS.

Generic Compiler from TRS with A Flexible Threshold. Despite the above separation of TRS and $\sharp$AMS, it turns out that several instantiations of TRS actually possess a *flexible threshold* property, i.e., these TRSs can change the threshold depending on the actual number of signers.

To characterize the class of TRSs that admit such a lifting to $\sharp$AMSs, we provide a generic template for TRSs that (1) abstracts many such "liftable" schemes and (2) admits a generic compiler to transform to $\sharp$AMS. Several existing TRS constructions can be seen as instantiations of our template, which implies that those TRS schemes are more versatile than previously known.

[4] In our blockchain governance application, we need a property which is stronger than identifiability, namely *public verifiability*, which informally ensures that an abort provides a cheating certificate that can be verified even by a non-MPC party later on, e.g., [5].

[5] Although there are solutions which replace the honest majority of parties assumption with an assumption on the resource distribution, e.g., honest majority of hashing-power or stake [34], they come at a high cost in terms of blockchain utilization—multiple on-chain rounds—which renders them mainly of theoretical interest.

[6] This is already a discount in security that should be avoided.

♯AMS Constructions from Chameleon Hashes. Using the above, we provide concrete ♯AMS schemes from (lossy) chameleon hashes in a black-box model—which covers the post-quantum case. In a nutshell, we propose the following types of constructions:

C1. A basic construction with three communication rounds. This construction achieves unconditional anonymity.
C2. A fault-tolerant variant of C1—for an arbitrary number of corruptions— without any overhead. This construction achieves unconditional anonymity and public verifiability.

Voting Systems from ♯AMS Constructions. Since anyone can verify the number of signers from a ♯AMS signature, we can immediately turn the constructions into voting systems. Furthermore, we develop a *conditioned key generation paradigm* to enable the single-vote setting.

V1. A basic system implemented by C2. This allows multiple voting. The system can tolerate malicious voters (those who claimed to vote but later quit). Including the posting period and the announcement period, voting can be completed within two on-chain blocks.
V2. A round-optimal system implemented by C1 in the multiple-vote setting by leveraging one-time key generation. This system satisfies the "vote-and-go" property.
V3. A variant of V2 for the single-vote setting, based on the conditioned key generation paradigm. V3 requires one additional on-chain round when there are malicious users. However, if the maximum number of proposals is known in advance, the voting process still requires only two on-chain blocks, as in V1 and V2.

Notably, our voting systems possess several desirable properties, which are discussed in depth in Section K. We briefly introduce these properties here:

On-Chain. The vote will be uploaded on the blockchain and cannot be altered further. All our systems achieve this property.
Vote-and-go. Voters can leave immediately after casting their vote (by sending their cryptographic information on the vote) without any interaction. V2 and V3 achieve this property.
Vote-Count Concealment. Voters remain unaware of the current vote count (and therefore of the votes of others) until the results are announced. V2 and V3 achieve this property.

Remark 1 (On receipt-freeness). We would like to mention that our protocols do not satisfy receipt-freeness properties due to their simple structure. Any voting scheme that requires voters to choose their own randomness, like ours, is vulnerable to a well-known generic receipt-freeness attack [44]. In such an attack, a malicious voter can generate randomness using a one-way function and later provide the input to the one-way function to prove the source of the randomness. To resolve this problem, one would need to rely on a much more complex system (e.g., [44] assumes homomorphic encryption, an honest assumption on the authorities, and many other assumptions). While this is an intriguing and interesting open problem, we are focusing on proposing a new signature protocol, ♯AMS, and thus leave this problem for future research.

A Framework of Lossy Chameleon Hashes. As a by-product, we explore a framework of lossy chameleon hashes, including its relationship with existing cryptographic primitives (e.g., lossy identification and lossy encryption) and more concrete constructions from various assumptions. See Fig. 1 for details.

1.2 Related Works

We refer to Appendix J for more related works on multisignatures, threshold signatures, and ring signatures.
Threshold Ring Signatures. ♯AMS in this paper are highly related to threshold ring signatures (TRS) [16]. An (n, t)-TRS scheme allows t or more members to generate a ring signature together, and the actual signers remain anonymous. The verification algorithm in TRS will output either 0 or 1, indicating the validity w.r.t. the threshold t. Differently, by defining ♯AMS, we emphasize the property that the verification outputs exactly the credibility of the signature, i.e., how many users have participated in the generation. This holds

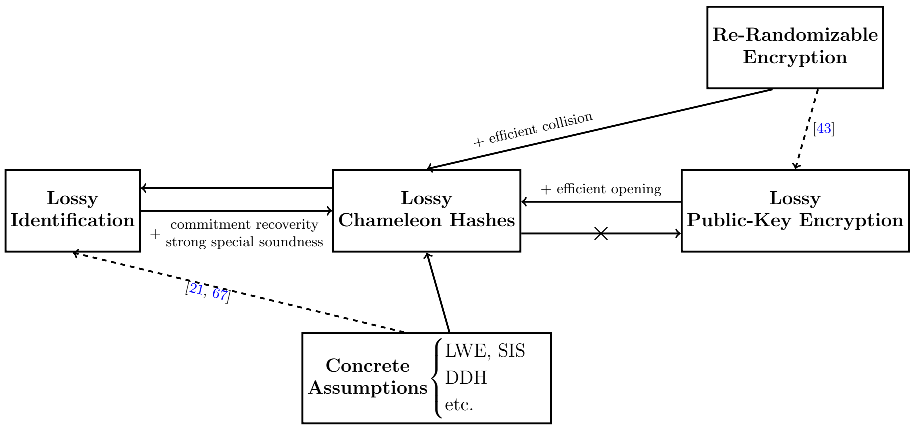

Fig. 1. A framework for lossy chameleon hashes (solid arrows: shown in this work; dashed line: shown in previous works).

even against malicious users who behave wantonly when signing. For example, a malicious user may quit at the middle or contribute senseless results in the signing process. Therefore, ♯AMS is strictly stronger than TRS.

There are many threshold ring signature schemes [57, 19, 78, 24] whose threshold t is changeable every signing. By adding t into the message to be signed, a TRS scheme with a flexible threshold can be tuned to a ♯AMS scheme.

Graded Signatures. Kiayias, Osmanoglu, and Tang [50] proposed the concept of graded signatures that allow a combiner/user to combine a set of different signatures w.r.t. the same message to a "consolidated" signature. Meanwhile, the consolidated signature leaks the actual number of signatures used to consolidate the graded signature and nothing else, which is the same as a ♯AMS scheme. However, graded signatures [50] have two phases in signing, which means that every participant has to sign by themselves and then pass their signature share to someone for consolidation. Due to such a definition, graded signatures cannot achieve unconditional anonymity (as we already discussed above, unconditional anonymity of voters is more desirable in blockchain applications). Therefore, ♯AMS provides more robust security.

Meanwhile, a trusted authority is required in graded signatures for generating global key pairs. The concrete instantiation in [50] relies on building blocks like structure-preserving signatures and Groth-Sahai proofs [40] and consequently cannot achieve post-quantum security.

Blockchain Governance and E-voting. Beck et al. [6], Pelt et al. [69], and Kiayias and Lazos [49] discussed core properties for blockchain governance in multiple disciplines. Khan et al. [48], Venugopalan and Homoliak et al. [76], and Gersbach et al. [37] also focused on blockchain governance, especially based on decision-making processes and voting in terms of game-theoretical analyses.

Many papers in various fields have studied e-voting systems that combine blockchain and cryptographic primitives to realize distributed and decentralized voting systems. However, the cryptography community, which is the one that can technically contribute the most, has less interest in the topic. A few papers introduce e-voting systems derived from advanced cryptographic primitives such as linkable ring signatures. In terms of ring signatures, Lyu et al. [59] and Russo et al. [72] present systems that use blockchain, smart contracts, and ring signatures. Most of the other blockchains' e-voting systems rely not only on simple cryptographic primitives but also heavily on complex functionalities, e.g., smart contracts and zero-knowledge proofs. Note that our proposed solutions can simplify their approaches much since ♯AMS provides not only privacy and authenticity but also signatures with the corresponding number of signers in the primitive level.

1.3 Technical Overview

This subsection briefly overviews the techniques and concepts used in this paper.

Formalization of ♯AMS. We start from formalizing ♯Anonymous Multisignature (♯AMS) and its security definitions. Suppose a group of t users, $\{U_i\}_{i \in G}$, wants to sign a message msg together, and the resulting signature σ reveals only the number of participants t and nothing else. We refer to t as the credibility of the signature.

The first issue is the anonymity property, which means that the identities of the t actual signers are hidden among the total of n users ($n \geq t$). Besides, as a (group) signature scheme, ♯AMS should ensure unforgeability; namely, any adversary controlling fewer than t users cannot forge a valid signature on a new message that shows a credibility of t.

As we discussed above, if every signer U_i contributes their share using their own secret key and the signature is simply a concatenation of different shares (as in linkable/traceable ring signatures), it seems impossible to achieve both unconditional anonymity and traceability [23].[7] Therefore, we focus on interactive signing processes and introduce a moderator P in the signing protocol. To generate a ♯AMS signature, each signer communicates only with P and not with other signers. It is P's responsibility to count the number of participants and finally produce a ♯AMS signature. In this case, even a signer cannot de-anonymize a ♯AMS signature (i.e., they remain unaware of the other participants). P can be a member of G or not, and is assumed to be honest. We believe this is a simple yet reasonable assumption; see more discussion in Sec. 3. Even if a malicious P leaks the quorum of signers later, their own identity would be revealed at the same time, and consequently, P would face punishment from the system.

Generic Compiler from Threshold Ring Signatures. A ♯AMS scheme directly implies a threshold ring signature (TRS) scheme since the verification algorithm Ver in ♯AMS returns the real number of participants in the signing process, while Ver in TRS returns only a single bit. Therefore, the definition of ♯AMS is stronger than that of TRS.

Nevertheless, we surprisingly notice that many TRS schemes (e.g., [57, 19, 18, 77, 41]) possess a *flexible threshold* property. Namely, the threshold t does not need to be fixed when generating the public/secret key pairs; it can be changed in every signing session. Motivated by this, we design a generic compiler that tunes any TRS scheme with a flexible threshold into a ♯AMS scheme:

- A random participant P among the signers is selected as the moderator in ♯AMS.
- P knows the quorum of signers, hence the number t. To generate a ♯AMS signature on message msg, it starts the TRS signing protocol on the message $(\mathsf{msg}||t)$ and outputs the TRS signature $\tilde{\sigma}$ and t as the final ♯AMS signature.
- In the verification of ♯AMS, if $\tilde{\sigma}$ is valid in TRS, then the threshold t is returned.

Thanks to this compiler and the rich literature on TRS, we immediately obtain many ♯AMS schemes from well-studied TRS schemes, based on the DL assumption [19], the RSA assumption [57], the SIS assumption [18], the code assumption [26], and others.

C1: Construction from (Lossy) Chameleon Hashes. A chameleon hash (CH) function [52] is a special hash function indexed by a hash key hk, which is associated with a trapdoor td. It takes as input two parts: a message m[8] and randomness r. On the one hand, given only the hash key, it is hard to find a collision. On the other hand, with the help of the trapdoor, finding collisions becomes easy.

Chameleon hashes can be converted into signature schemes via the well-known Fiat-Shamir paradigm [31], where hk and td serve as the public key and the signing key, respectively. To sign a message msg, the signer first randomly samples dummy values $\bar{m}$ and $\bar{r}$, and then uses its trapdoor to find a randomness r for m so

[7] In the application of voting systems, if double voting is feasible, then traceability is necessary.

[8] Note that the message m in chameleon hashes is different from the message msg to be signed in signature schemes. Here, we slightly abuse the notation to keep consistent with previous works.

that it collides with $(\bar{m}, \bar{r})$ on h, where $m = H(\mathsf{msg}, h)$ with $H(\cdot)$ a random oracle and h the chameleon hash value of $(\bar{m}, \bar{r})$.

Now, we extend this idea to the $\sharp$AMS setting. Suppose there are n users, and each user U_i has its own hash key hk_i and trapdoor td_i. We borrow the idea of the t-out-of-n zero-knowledge proof from [25] by Cramer, Damgård, and Schoenmakers. That is, given n random values m_i, the group of signers can find t random values r_i that create collisions. For the message msg, we design a $\sharp$AMS signature in the form of $\sigma = (t, m_1, \ldots, m_n, r_1, \ldots, r_n)$, where $m_1, \ldots, m_n$ are required to satisfy the following linear equations.

$$\begin{cases} a_{1,1}m_1 + a_{1,2}m_2 + \ldots + a_{1,n}m_n = u_1, \\ a_{2,1}m_1 + a_{2,2}m_2 + \ldots + a_{2,n}m_n = u_2, \\ \ldots \\ a_{t,1}m_1 + a_{t,2}m_2 + \ldots + a_{t,n}m_n = u_t. \end{cases} \tag{1}$$

Here, the coefficients $(a_{i,j})$ are public parameters, and $(u_1, \ldots, u_t)$ are the random outputs of the hash function $H(h_1, \ldots, h_n, \mathsf{msg}, t)$, with $(h_1, \ldots, h_n)$ being the corresponding chameleon hash (CH) values. To satisfy Eq. (1), at least t trapdoors are necessary to determine the corresponding randomness r_i, thereby proving that at least t users have participated in the signing process.

Instead of using the above linear equations, we can also set a polynomial g of degree at least $n - t$ and require that

$$g(0) = u, \quad g(i) = m_i \text{ for all } i \in [n], \text{ where } u = H(h_1, \ldots, h_n, \mathsf{msg}, t). \tag{2}$$

The security of the CH-based signature scheme, as well as our $\sharp$AMS scheme, is based on the collision-resistance property of the underlying chameleon hash scheme. However, the reduction in the security proof is not black-box since it relies on the forking lemma [8] to find a collision. Inspired by the ideas of lossy trapdoor functions [68] and lossy encryption [7, 43], we define lossy chameleon hashes (LCH) in this work to enable a black-box reduction in the security proof. An LCH scheme operates in two modes: the collision mode, which behaves as a standard CH, and the lossy mode, where a lossy hash key is used instead of a normal hash key, and the adversary (even if computationally unbounded) cannot find randomness for a random message that hashes to a previously chosen hash value.

We present an efficient instantiation of LCH from the learning with errors (LWE) assumption and the short integer solution (SIS) assumption. We also explore the relationship between LCH and other primitives, including lossy identification schemes [1], re-randomizable encryption, and lossy encryption [7, 43]; see Fig. 1 and Appendix B.

C2: Fault-Tolerant Variant. For better application in blockchain governance, we consider a variant of the above construction in the faulty signer setting. A faulty signer may quit in the middle of the protocol or return malicious results to the moderator. To address this, we develop a fault-tolerant variant of our $\sharp$AMS scheme. This is possible because our (L)CH-based $\sharp$AMS scheme has "backward compatibility": namely, even if a voter quits in the middle, by exposing the identity of the faulty voter, the moderator can still output a signature with credibility $(t - 1)$. More precisely, let F be the set of faulty signers. In the fault-tolerant $\sharp$AMS scheme, we tailor the signature into the form

$$\sigma' := \left(t, F, \{m_i, r_i\}_{i \in [n] \setminus F}, \{h_i\}_{i \in F}\right).$$

If there exists a solution $\{m_i\}_{i \in F}$ for the linear equation (1) or (2), then the verification algorithm will output $(t - |F|)$ instead of t. Consequently, the voters in F will lose their anonymity as a cost of their malfeasance.

V1: Blockchain Governance via $\sharp$AMS Schemes. $\sharp$AMS directly implies a voting protocol since a $\sharp$AMS signature leaks the number of signers. The protocol consists of the following four periods using scheme C2:

1. The posting period, in which each developer publishes their improvement proposal on the blockchain.
2. The declaration period, in which each voter (a signer in $\sharp$AMS) signals their willingness to support a proposal by sending a hash value of a dummy message and randomness to the developer.

3. The signing period, in which the developer performs the signature algorithm based on the number of supporters. Specifically, developers compute all m_i values according to the received h_i and Eq. (1) or (2). They then return the m_i values to the supporters, who use their trapdoors to find collisions r_i and send them back to the developer. This helps the developer complete the generation of the $\sharp$AMS signature.
4. The announcement period, in which each developer uploads their $\sharp$AMS signature to the blockchain. The voting result is then included in the block and published across the network.

Our protocol enables multiple improvement proposals to compete for votes, and the proposal with the most votes is elected.

To generate a $\sharp$AMS signature, the developer needs three rounds of interaction with its supporters during the signing period. This opens the door to faulty attacks by malicious voters: for instance, a voter may send h_i and claim to participate, but then abort or send incorrect randomness r' after receiving m from the developer. Thanks to our fault-tolerant $\sharp$AMS scheme, such attacks do not compromise the entire voting system, since a (fault-tolerant) $\sharp$AMS signature always reveals the true number of (honest) participants.

V2: Round Optimization. Notice that in the protocol above, voters must wait for the message m_i from the developer after declaring their support. To achieve the "vote-and-go" property, we further optimize the protocol to a single round. Our idea is to use $\sharp$AMS in a one-time paradigm. That is, voters U_i, regardless of whether they intend to vote for a proposal, generate a new hash key and trapdoor pair $(hk_i^{(j)}, td_i^{(j)})$ for some proposal by developer P_j. Then, supporters of P_j secretly send their trapdoors to P_j using a standard public-key encryption scheme. With knowledge of all secret keys, P_j can generate a $\sharp$AMS signature without further interaction with the supporters. To prevent a malicious developer from generating one-time hash keys on their own, we additionally require every hash key to be accompanied by a signature proving the authority of its owner.

V3: Single-Vote Setting via the Conditioned Key Generation Paradigm. We further extend the above protocol to the single-vote setting, where each user in the voting system can vote for at most one proposal. Since there are multiple developers P_j, each with their own proposal, every voter U_i now needs to generate multiple key pairs $\{(hk_i^{(j)}, td_i^{(j)})\}$, where the superscript (j) indicates the key pair is for the j-th proposal.

The protocol as previously described does not work in the single-vote setting because a malicious voter could vote on different proposals using different trapdoors. We employ the same method used in constructing $\sharp$AMS to prevent multiple votes. Recall that to vote on a proposal $IP^{(j)}$ (i.e., participate in the signing process for that proposal), user U_i must possess the trapdoor $td_i^{(j)}$ corresponding to the hash key $hk_i^{(j)}$ specifically designed for $IP^{(j)}$. Suppose there are p proposals $(IP^{(j_1)}, ..., IP^{(j_p)})$, then U_i must generate p hash keys $(hk_i^{(j_1)}, ..., hk_i^{(j_p)})$. We apply a so-called *conditioned key generation paradigm*, which sets a restriction among the p hash keys so that the voter can know at most one trapdoor.

Let $\mathcal{HK}$ be the space of hash keys for (lossy) chameleon hashes, and let $\mathbf{B} = (b_{i,j}) \in \mathcal{HK}^{(p-1)\times p}$ be a matrix of full rank. Define $\mathbf{hk}_i^\top = (hk_i^{(j_1)}, ..., hk_i^{(j_p)})^\top$. We require $\mathbf{B} \cdot \mathbf{hk}_i = \hat{\mathbf{hk}}_i$, where $\hat{\mathbf{hk}}_i \in \mathcal{HK}^{p-1}$ is the output of some hash function $H(IP^{(j_1)}, ..., IP^{(j_p)}, i)$.

If $H(\cdot)$ behaves as a random oracle, then to satisfy the above equation, every user can know at most one trapdoor among the total p hash keys. Consequently, the single-voting property is achieved.

1.4 Roadmap

This paper is organized as follows. In Section 2, we introduce the definitions of (lossy) chameleon hashes. The definition and security properties of $\sharp$AMS are formally described in Section 3. We provide a generic transformation for most threshold ring signature (TRS) schemes in Section 4. In Section 5, we propose an efficient construction of $\sharp$AMS from (lossy) chameleon hashes ((L)CH). Section 6 presents three voting systems and their applications in blockchain governance.

Due to the page limit, most technical details are provided in the appendices. In Appendices A and C, we introduce additional cryptographic preliminaries and security notions for ♯AMS, respectively. Appendix B explores lossy chameleon hashes and another new primitive. In Appendix D, we further discuss the relationship between TRS and ♯AMS.

For formal security proofs, we analyze strong unforgeability of ♯AMS under different adversarial models in Appendices E and F, and analyze strong unforgeability based on standard chameleon hashes via the forking lemma in Appendix H. Instantiations of constraint functions are shown in Appendix G. For the fault-tolerant (C2 variant) scheme, we provide an additional security proof and discussion in Appendix I. We conclude with further discussion in Appendix K.

2 Preliminaries

We refer to Appendix A for basic notations and cryptographic primitives like signatures.

Definition 1 (Chameleon Hashes [52]). *A chameleon hash (CH) scheme consists of the following three algorithms. Namely* $\mathsf{CH} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{TdColl})$.

- $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$*: The key generation algorithm takes as input the security parameter* 1^λ*, and outputs a hash key* hk *and a trapdoor* td*. W.l.o.g., we assume* hk *implicitly determines the message space* $\mathcal{M}$*, the randomness space* $\mathcal{R}$*, and the hash space* $\mathcal{H}$.
- $h \leftarrow \mathsf{Hash}(hk, m, r)$*: The hash algorithm takes as input* hk*, a message* $m \in \mathcal{M}$ *and a randomness* $r \in \mathcal{R}$*, and outputs the hash value* $h := \mathsf{Hash}(hk, m, r)$.
- $r' \leftarrow \mathsf{TdColl}(td, m, r, m')$*: The trapdoor collision algorithm takes as input the trapdoor* td*, a message-randomness pair* (m, r) *and another message* m'*, and outputs* r' *such that* $\mathsf{Hash}(hk, m, r) = \mathsf{Hash}(hk, m', r')$.

Definition 2 (Lossy Chameleon Hashes). *A lossy chameleon hash (LCH) scheme consists of four algorithms,* $\mathsf{LCH} = (\mathsf{Gen}, \mathsf{LGen}, \mathsf{Hash}, \mathsf{TdColl})$*, where* $\mathsf{Gen}, \mathsf{Hash}, \mathsf{TdColl}$ *are defined as in Definition 1, and* LGen *is defined as follows.*

- $hk \leftarrow \mathsf{LGen}(1^\lambda)$*: The lossy key generation algorithm takes as input the security parameter* 1^λ*, and outputs a lossy hash key* hk.

We refer to Appendix B for the security notations and a detailed framework of LCH.

3 Sharp Anonymous Multisignatures

In this section we formally define sharp anonymous multisignatures and their security notions. Let n be the total number of signers, and $G \subseteq [n]$ be a group of signers that want to generate a ♯AMS signature together. Let $t = |G|$.

Definition 3 (Sharp Anonymous Multisignatures (♯AMS)). *A sharp anonymous multisignature (♯AMS) scheme* $\sharp\mathsf{AMS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *consists of the following three algorithms/protocols:*

- $(vk, sk_1, ..., sk_n) \leftarrow \mathsf{Gen}(1^\lambda, n)$*. The key generation algorithm* Gen *takes as input the security parameter* λ *and the number of signers* n*, and outputs a verification key* vk*, and signing keys* $(sk_1, ..., sk_n)$ *for different signers. W.l.o.g., we assume that* vk *is implicitly contained in every* sk_i.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, P, G \subseteq [n], \{sk_i\}_{i \in G})$*. The signing protocol takes place between a moderator* P *and a group of signers* $G \subseteq [n]$*, where* P *takes as input* vk *and the message* msg*, and each signer* U_i *takes* msg *and its own secret key* sk_i *as input. The moderator* P *can be a member of* $[n]$ *or not. Finally,* P *outputs a signature* σ.
  If we focus solely on the algorithmic properties of the signing process, we will ignore the moderator P *by denoting it as* $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, G \subseteq [n], \{sk_i\}_{i \in G})$.

– $t \leftarrow \mathsf{Ver}(vk, \mathsf{msg}, \sigma)$. *The verification algorithm* Ver *takes as input the verification key* vk*, a message* msg *and a signature* σ*, and outputs a number* $t \in [n] \cup \{0\}$*, indicating the number of signers for this signature.*[9]
We say a signature σ *(w.r.t. a message* msg*) is* t*-valid (resp., invalid), if* $\mathsf{Ver}(vk, \mathsf{msg}, \sigma) = t$ *(resp.,* $\mathsf{Ver}(vk, \mathsf{msg}, \sigma) = 0$*).*

Correctness. *For any* $(vk, sk_1, ..., sk_n) \leftarrow \mathsf{Gen}(1^\lambda, n)$*, any message* msg*, any group* $G \subseteq [n]$ *of honest signers, any honest moderator* P*, and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, P, G, \{sk_i\}_{i \in G})$*, it holds that* $\mathsf{Ver}(vk, \mathsf{msg}, \sigma) = |G|$.

In the above definition, we assume all users' keys are generated via a centralized algorithm Gen, which is more generalized and covers the case where there is a trusted setup. In this work, we focus on the non-trusted setup case, and each user samples its own key pair and then publishes the public key. Moreover, we focus on cryptographic property of $\sharp$AMS and do not consider the details of the signing protocol, e.g., adaptive corruptions during the running, protocol rounds, and just provide a proof-of-concept use of $\sharp$AMS.

We require the unforgeability and anonymity for the security of $\sharp$AMS.

– **Unforgeability**. The adversary that controls less than t participants cannot forge a signature that is t-valid.
– **Anonymity**. From a signature the adversary learns nothing about the quorum of the signers G that contributed to the signature, except the size $|G|$.

We formalize the security definitions via the following security experiments.

Definition 4 (Unforgeability of $\sharp$AMS). *Consider the following unforgeability experiment* $\mathsf{Exp}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *between the challenger* $\mathcal{C}$ *and the adversary* $\mathcal{A}$*.*

1. $\mathcal{A}$ *sets the maximum number of signers* n*.*
2. $\mathcal{C}$ *generates* $(vk, sk_1, ..., sk_n) \leftarrow \mathsf{Gen}(1^\lambda, n)$ *and passes* vk *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *has access to two oracles* $\mathcal{O}(\cdot,\cdot,\cdot)$ *and* $\mathcal{O}_{corr}(\cdot)$*. Here the signing oracle* $\mathcal{O}(\mathsf{msg}, P, G)$ *returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, P, G, \{sk_i\}_{i \in G})$ *(with* P *the moderator) and adds* (msg, σ) *into the set* $\mathcal{S}$*. The corruption oracle* $\mathcal{O}_{corr}(i)$ *returns* sk_i*.*
4. *Finally* $\mathcal{A}$ *outputs* $(\mathsf{msg}^*, \sigma^*)$*.*

Let $t^* \leftarrow \mathsf{Ver}(vk, \mathsf{msg}^*, \sigma^*)$*.* $\mathsf{Exp}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs* 1 *if*

(1) $t^* > t'$*, where* t' *is the total number of queries to* $\mathcal{O}_{corr}(\cdot)$*; and*
(2) $\mathcal{A}$ *never asks* $\mathcal{O}(\mathsf{msg}^*, P, G)$ *such that* $|G| = t^*$*.*

Define by $\mathsf{Adv}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *the probability that* $\mathsf{Exp}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs 1. We say that* $\sharp$AMS *is unforgeable, if for all PPT adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

Remark 2 (On the Formalization of Unforgeability). One might wonder why we require "$\mathcal{A}$ never asks $\mathcal{O}(\mathsf{msg}^*, P, G)$ s.t. $|G| = t^*$" at the end of the experiment. Intuitively, a more "reasonable" definition should allow $\mathcal{A}$ to win, if it asks $\mathcal{O}(\mathsf{msg}^*, P, G)$ with $|G| = t^*$, and later forges a t^*-valid signature σ^* from a different set $G' \neq G$. However, since the identities are hidden from the signature, the challenger cannot detect whether σ^* comes from a group G' that is different from G (i.e., whether $\mathcal{A}$ wins in a non-trivial way). Therefore, to prevent trivial attacks, in Definition 4, $\mathcal{A}$ is forbidden to query $\mathcal{O}(\mathsf{msg}^*, P, G)$ with $|G| = t^*$.

Definition 5 (Strong Unforgeability). *Consider the strong unforgeability experiment* $\mathsf{Exp}^{s\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$*, which is defined as* $\mathsf{Exp}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *in Definition 4, except that condition* (2) *is replaced with*

(2') $(\mathsf{msg}^*, \sigma^*) \notin \mathcal{S}$*.*

[9] Note that vk contains information of $[n]$ so we omit $[n]$ from the input of Ver, which will be explicitly denoted for TRS schemes because of consistency from previous works.

Define by $\mathsf{Adv}^{s\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *the probability that* $\mathsf{Exp}^{s\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs 1. We say that* $\sharp\mathsf{AMS}$ *is strongly unforgeable, if for all PPT adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{s\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *is negligible in* λ.

We also define the weak unforgeability and (strong/weak) unforgeability under static corruptions, see Appendix C.

Now we formally define the strong anonymity of $\sharp\mathsf{AMS}$.

Definition 6 ((Unconditionally) Strong Anonymity of $\sharp$AMS). *Consider the following strong anonymity experiment* $\mathsf{Exp}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *between* $\mathcal{C}$ *and* $\mathcal{A}$.

1. *$\mathcal{A}$ sets the maximum number of signers n.*
2. *$\mathcal{C}$ generates $(vk, sk_1, ..., sk_n) \leftarrow \mathsf{Gen}(1^\lambda, n)$ and passes vk to $\mathcal{A}$. Meanwhile, $\mathcal{C}$ randomly samples a bit $b \xleftarrow{\$} \{0,1\}$.*
3. *$\mathcal{A}$ has access to two oracles $\mathcal{O}(\cdot,\cdot,\cdot,\cdot,\cdot)$ and $\mathcal{O}_{corr}(\cdot)$, where the signing oracle $\mathcal{O}(\mathsf{msg}, P_0, G_0, P_1, G_1)$ returns $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, P_b, G_b, \{sk_i\}_{i\in G_b})$ if $|G_0| = |G_1|$ and $\perp$ otherwise, and the corruption oracle $\mathcal{O}_{corr}(i)$ returns sk_i.*
4. *Finally $\mathcal{A}$ outputs b'.*

$\mathsf{Exp}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs 1 if $b' = b$. Let* $\mathsf{Adv}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \Rightarrow 1] - 1/2|$*. We say that* $\sharp\mathsf{AMS}$ *has strong anonymity (resp., unconditional and strong anonymity) if for all PPT (resp., computationally unbounded) adversary $\mathcal{A}$, the advantage* $\mathsf{Adv}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *is negligible in* λ.

The aforementioned definition requires that anonymity holds even if all secret keys are leaked (that is why we formalize it as "strong" anonymity). In the above definition, if we restrict that $\mathcal{A}$ cannot ask $\mathcal{O}_{corr}(i)$ with $i \in (G_0 - C_1) \cup (G_1 - G_0)$ for all (G_0, G_1) involved in $\mathcal{O}(\cdot,\cdot,\cdot,\cdot,\cdot)$, then we define the anonymity where it might be easy to decide whether a signer i has participated in the generation of a signature with the knowledge of sk_i. This is somewhat similar to the so-called *culpability* property in [58]. In other words, one signer is able to claim the authorship of some (ring) signature by revealing its secret key (and some other private information, if necessary) to the public.

4 Generic Compiler of $\sharp$AMS from TRS with A Flexible Threshold

In this section, we provide a generic transformation for threshold ring signature (TRS) schemes with flexible threshold, which implies that such TRS schemes are more versatile than their original definitions. We refer to Appendix D for the definition and security of TRS, and a detailed discussion for the relationship between TRS and $\sharp$AMS.

Generic compiler from TRS with flexible threshold to $\sharp$AMS. Here, we show the most generalized version of compilers without considering optimization. We will take a deeper look at specific cases in the next section. Let $\mathsf{TRS} = (\mathsf{Gen}, \mathsf{TSign}, \mathsf{Ver})$ be a TRS scheme with a flexible threshold. We design $\sharp$AMS scheme $\sharp\mathsf{AMS}$ as follows.

- $\mathsf{Gen}(1^\lambda, n)$. For $i = 1, ..., n$, invoke $(pk_i, sk_i) \leftarrow \mathsf{TRS.Gen}(1^\lambda)$. Return $vk := (pk_1, ..., pk_n)$ and $\{sk_i\}_{i\in[n]}$.
- $\mathsf{Sign}(\mathsf{msg}, P, G \subseteq [n], \{sk_i\}_{i\in G})$. Every signer U_i where $i \in G$ first sends a HELLO message to the moderator P so that P will know the number of signers $t := |G|$. Then, the t signers run the threshold signing protocol $\mathsf{TRS.TSign}(\mathsf{msg}||t, [n], vk, G, \{sk_i\}_{i\in G})$ with P works as a moderator. Let $\tilde{\sigma}$ be the output of $\mathsf{TRS.TSign}$. Finally, P outputs $\sigma := (t, \tilde{\sigma})$ as a $\sharp$AMS signature.[10]
- $\mathsf{Ver}(vk, \mathsf{msg}, \sigma)$. Parse $\sigma = (t, \tilde{\sigma})$. If $\mathsf{TRS.Ver}(t, [n], vk, \mathsf{msg}, \tilde{\sigma}) = 1$ then output t. Otherwise, output 0.

The security of $\sharp$AMS constructed above inherits from the security of the underlying TRS scheme, and we omit the detailed proof here.

[10] If $\tilde{\sigma}$ itself already contains a threshold t then P just outputs $\sigma := \tilde{\sigma}$.

5 Constructions of ♯AMS from Lossy Chameleon Hashes

In this section we propose efficient constructions of ♯AMS from lossy chameleon hashes (LCH) in a black-box mode.

5.1 Formalization of Constraint Functions

First we introduce the definition of constraint functions, an important tool in constructing the t-out-of-n proofs [25].

Definition 7 (Constraint Functions). *Let n, t be positive integers, $n \geq t \geq 1$, and $\mathcal{M}$, $\mathcal{U}$ be two finite sets. We call $\mathbf{F}_\theta : (\mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathcal{M}^n \times \mathcal{U}) \to \{0,1\}$ a series of constraint functions indexed by θ, if it has the following properties.*

- *$\mathbf{F}_\theta(n, t, m_1, ..., m_n, u)$ is efficiently evaluated.*
- *There exists an efficient and deterministic algorithm $f(\cdot)$, such that $f(n, t, m_1, ..., m_n)$ outputs the unique u (if it exists) satisfying $\mathbf{F}_\theta(n, t, m_1, ..., m_n, u) = 1$.*
- *For any $G \subseteq [n]$ and $|G| = t$, there exists two efficient sample algorithms s_{fwd} and s_{back} that both output $(m_1, ..., m_n, u)$, and the two distributions are identical.*
  - *Forward sample algorithm $s_{fwd}(n, t, G)$ first randomly chooses $\{m_i\}_{i\in[n]}$, and then computes/samples $\{m_i\}_{i\in G}$ and u according to $\mathbf{F}_\theta$.*
  - *Backward sample algorithm $s_{back}(n, t, G)$ first randomly chooses $\{m_i\}_{i\in[n]\setminus G}$ and u, and then computes $\{m_i\}_{i\in G}$ according to $\mathbf{F}_\theta$.*
- *Interdependency.*
  - *If $\mathbf{F}_\theta(n, t, m_1, ..., m_n, u) = \mathbf{F}_\theta(n, t, m'_1, ..., m'_n, u) = 1$, then either $(m_1, ..., m_n) = (m'_1, ..., m')$, or there are at least t different $i \in [n]$ such that $m_i \neq m'_i$.*
  - *For randomly sampled u and u', if $\mathbf{F}_\theta(n, t, m_1, ..., m_n, u) = \mathbf{F}_\theta(n, t, m'_1, ..., m'_n, u') = 1$, then with overwhelming probability there are at least t different $i \in [n]$ such that $m_i \neq m'_i$.*
- *Randomness. Conditioned on $\mathbf{F}_\theta(n, t, m_1, ..., m_n, u) = 1$, if u distributes uniformly, then*
  - *either there exist at least t different $i \in [n]$ such that m_i distribute uniformly, or*
  - *for any $i \in [n]$, m_i distributes uniformly.*

We refer to two instantiations of $\mathbf{F}_\theta$ from linear equations and polynomial interpolation in Appendix G.

5.2 C1: Interactive ♯AMS

Now, we describe our generic construction of ♯AMS from (lossy) chameleon hashes ((L)CH) as follows. We note that the underlying building blocks (LCH or CH) affect only the way of security proofs, not the construction itself.

Construction. Let $\mathsf{LCH} = (\mathsf{Gen}, \mathsf{LGen}, \mathsf{Hash}, \mathsf{TdColl})$ be a lossy chameleon hash scheme with message space $\mathcal{M}$ a field. Let $\mathbf{F}_\theta : (\mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathcal{M}^n \times \mathcal{U}) \to \{0,1\}$ be a constraint function indexed by θ. $H(\cdot) : \{0,1\}^* \to \mathcal{U}$ is a hash function that is modeled as a random oracle.

- $(vk, sk_1, ..., sk_n) \leftarrow \mathsf{Gen}(1^\lambda, n)$. For $i \in [n]$, invoke $(hk_i, td_i) \leftarrow \mathsf{LCH.Gen}(1^\lambda)$. Return $vk := (hk_1, ..., hk_n)$ and $\{sk_i\}_{i\in[n]} := \{td_i\}_{i\in[n]}$.
- $\mathsf{Sign}(\mathsf{msg}, P, G, \{sk_i\}_{i\in G})$.
  - For every signer $i \in G$, it samples $\bar{m}_i \xleftarrow{\$} \mathcal{M}, \bar{r}_i \xleftarrow{\$} \mathcal{R}$, and sends $h_i \leftarrow \mathsf{Hash}(hk_i, \bar{m}_i, \bar{r}_i)$ to the moderator P.
  - The moderator counts $t := |G|$ from the received messages.
  - For every user $i \in [n] \setminus G$, P samples $m_i \xleftarrow{\$} \mathcal{M}, r_i \xleftarrow{\$} \mathcal{R}$ and computes $h_i \leftarrow \mathsf{Hash}(hk_i, m_i, r_i)$.

  - P invokes $u \leftarrow H(vk, h_1, ..., h_n, \mathsf{msg}||t)$.[11] Then it computes $\{m_i\}_{i \in G}$ according to the backward sample algorithm $s_{back}(\cdot)$ of $\mathbf{F}_\theta$ such that

$$\mathbf{F}_\theta(n, t, m_1, ..., m_n, u) = 1.$$

    Then for $i \in G$, P sends m_i to signer i.
  - For every $i \in G$, signer i computes $r_i \leftarrow \mathsf{LCH.TdColl}(td_i, \bar{m}_i, \bar{r}_i, m_i)$ and sends r_i to P.
  - Finally P outputs the signature $\sigma := (t, \{m_i\}_{i\in[n]}, \{r_i\}_{i\in[n]})$.
- $\mathsf{Ver}(vk, \mathsf{msg}, \sigma)$. Parse $\sigma = (t, \{m_i\}_{i\in[n]}, \{r_i\}_{i\in[n]})$. Compute $h_i \leftarrow \mathsf{Hash}(hk_i,\ m_i, r_i)$ for all $i \in [n]$. Let $u \leftarrow H(vk, h_1, ..., h_n, \mathsf{msg}||t)$. Return t if $\mathbf{F}_\theta(n, t, m_1,\ ..., m_n, u) = 1$, and 0 otherwise.

Generality of the Construction. Our construction above exhibits strong generality. In Section 4, we have already shown a generic compiler from TRS with a flexible threshold to ♯AMS. In fact, many existing TRS constructions can be categorized within the above framework due to the equivalence between chameleon hashes and Sigma protocols (identification schemes) [9] and the result in this work (Appendix B). For example, by instantiating with $\mathbf{F_A}$ and the DL-based CH [52], we get the TRS scheme in [19], and by instantiating with $\mathbf{F}_p$ and the DL-based CH, we get the TRS scheme in [57]. Thanks to this generic construction, we immediately get more schemes of ♯AMS (also TRS) from lattices [36, 17], isogenies [30].

Theorem 1. *If* LCH *is strongly secure (i.e., it has κ-uniformity, γ-random trapdoor collision, strong collision resistance, indistinguishability, and ϵ-lossiness) and unique, and $\mathbf{F}_\theta$ is a constraint function, then* ♯AMS *constructed above has strong unforgeability and unconditionally strong anonymity under static corruptions. More precisely, for any PPT adversary $\mathcal{A}$, there exist PPT algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$, such that* $\max(Time(\mathcal{B}_1), Time(\mathcal{B}_2)) \approx Time(A)$, *and*

$$\mathsf{Adv}^{s\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \le n(\mathsf{Adv}^{s\text{-}cr}_{\mathsf{LCH},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{B}_2}(\lambda)) + n \cdot \epsilon + \frac{1}{|\mathcal{M}|} + \frac{Q_{sign} + Q_H}{2^{n\kappa}} + Q_{sign} n \cdot \gamma,$$

$$\mathsf{Adv}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \le \frac{Q_{sign} n}{2} \cdot \gamma,$$

where Q_{sign} and Q_H are the numbers of signing queries (in the strong unforgeability experiment or the strong anonymity experiment) and hash queries, respectively.

We refer to Appendix E for the proof due to the page limitation.

Similarly, we get the following theorem for (regular) unforgeability.

Theorem 2. *If* LCH *is secure (i.e., it has κ-uniformity, γ-random trapdoor collision, collision resistance, indistinguishability, and ϵ-lossiness) and $\mathbf{F}_\theta$ is a constraint function, then* ♯AMS *constructed above has unforgeability and unconditional strong anonymity under static corruptions. More precisely, for any PPT adversary $\mathcal{A}$, there exist PPT algorithm $\mathcal{B}$ such that $Time(\mathcal{B}) \approx Time(A)$, and*

$$\mathsf{Adv}^{unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \le n\mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{B}}(\lambda) + n \cdot \epsilon + \frac{1}{|\mathcal{M}|} + \frac{Q_{sign} + Q_H}{2^{n\kappa}} + Q_{sign} n \cdot \gamma,$$

where Q_{sign} and Q_H are the numbers of signing queries and hash queries, respectively.

Security under Adaptive Corruptions and the Tightness of Unforgeability. We prove the security of unforgeability in the static corruption model (Definition 29 in Appendix C). The proof can also be extended to the adaptive model (Def. 4 and 5), but it suffers from a large loss factor 2^n. We present the security bound under adaptive corruptions in Appendix F. At a high level, to make use of the lossiness property of LCH, the challenger has to make sure that all hash keys of non-corrupted users are generated in the lossy mode. However, there is no corresponding trapdoor for a lossy hash key, which means that to deal with $\mathcal{A}$'s adaptive

[11] By including both msg and the threshold t into the hash input, the non-malleability is achieved in our signature scheme.

corruptions, the challenger has to decide the way of key generation very carefully so that it can offer the trapdoor of a user when a corruption happens on it.

We also give another proof based on normal chameleon hashes (Definition 1) in Appendix H, which has a quadratic loss but relies on the forking lemma (Def. 1 in Appendix A). The reduction is tight in the algebraic group model (AGM) [32], if we instantiate ♯AMS with DL-based chameleon hashes [52].

5.3 C2: The Fault-Tolerating Variant

In this subsection, we consider faulty signers, i.e., signers who quit in the middle of signing or return faulty results to the moderator P. We propose a fault-tolerant variant of the (L)CH-based ♯AMS construction. In this variant, even if there exist faulty signers who behave maliciously, the moderator P can still output a ♯AMS that is t-valid, where t is the number of honest signers in the generation of that signature. Besides, the identities of those faulty signers are exposed, and anonymity holds only for the honest signers.

Faulty Attacks. We first introduce possible faulty attacks and then introduce a variant of (L)CH-based ♯AMS for tolerating faults while generating signatures in the following. Note that our constructions need communication with a developer within a fixed period to ensure the number of total signers. This leads to faulty attacks in the presence of a faulty node that declares to join the signing process but fails to generate a partial signature. In other words, the signing process suffers from the following attack even if only one faulty node exists: In the declaration period, the faulty node U_k faithfully follows the first two steps of the interaction. Then after receiving m_k from P, it aborts or sends a faulty r_k.

Since P cannot compute r_k without td_k, it cannot produce a t-valid ♯AMS signature. Furthermore, since $\{m_j\}_{j\in G}$ are determined by $H(vk, h_1, ..., h_n, \mathsf{msg}||t)$ from $\{h_j\}_{j\in[n]}$ and $t = |G|$, P cannot generate a signature of $G \setminus \{k\}$ by simply discarding U_k.

Fault-Tolerant ♯AMS. We follow the same notion as the previous section. The Gen algorithm and the Sign protocol are also essentially the same as the previous one, but only the following treatment is included:

– In the signing protocol, if a signer U_i does not response for P's message m_i, or the signer U_i returns a wrong r_i such that $\mathsf{Hash}(hk_i, m_i, r_i) \neq h_i$, then P include i into the set F. Here h_i is the first message U_i sens to P (if U_i does not send h_i, then P would not include U_i into the signer group G).
  Finally, P outputs the fault-tolerated signature
$$\sigma := \big(t, F, \{m_i, r_i\}_{i\in[n]\setminus F}, \{h_i\}_{i\in F}\big)\,.$$
– $t \leftarrow \mathsf{Ver}(vk, \mathsf{msg}, \sigma)$. Pause $\sigma := \big(t, F, \{m_i, r_i\}_{i\in[n]\setminus F}, \{h_i\}_{i\in F}\big)$. Compute $h_i \leftarrow \mathsf{Hash}(hk_i, m_i, r_i)$ for all $i \in [n] \setminus F$. Let $u \leftarrow H(vk, h_1, ..., h_n, \mathsf{msg}||t)$. Return $(t - |F|)$ if there exist a solution $\{m_i\}_{i\in F}$ such that $\mathbf{F}_\theta(n, t, m_1, ..., m_n, u) = 1$, and 0 otherwise.

We show that the fault-tolerant ♯AMS scheme has weak unforgeability (against malicious signers) and unconditional strong anonymity (for honest signers).

Theorem 3. *If* LCH *is secure (i.e., it has* κ*-uniformity,* γ*-random trapdoor collision, strong collision resistance, indistinguishability, and* ϵ*-lossiness) and* $\mathbf{F}_\theta$ *is a constraint function, then the fault-tolerant ♯AMS scheme above has weak unforgeability and strong anonymity under static corruptions. More precisely, for any PPT adversary* $\mathcal{A}$*, there exist PPT algorithms* $\mathcal{B}_1$ *and* $\mathcal{B}_2$*, such that* $\max(Time(\mathcal{B}_1), Time(\mathcal{B}_2)) \approx Time(A)$,

$$\mathsf{Adv}^{w\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \le n(\mathsf{Adv}^{s\text{-}cr}_{\mathsf{LCH},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{B}_2}(\lambda)) + n\cdot\epsilon + \frac{1}{|\mathcal{M}|} + \frac{Q_{sign}+Q_H}{2^{n\kappa}} + Q_{sign}n\cdot\gamma,$$

$$\mathsf{Adv}^{s\text{-}anony}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \le \frac{Q_{sign}n}{2}\cdot\gamma$$

where Q_{sign} *and* Q_H *are the numbers of signing queries (in the strong unforgeability experiment or the strong anonymity experiment) and hash queries, respectively.*

We refer to Appendix I for the proof sketch.

6 Applications: Blockchain Governance and the Beyond

$\sharp$AMS can be massively applied in the scenario of blockchain and privacy-preserving, where authenticity and privacy are required simultaneously. The first and the most significant application of $\sharp$AMS is blockchain governance, especially about ranking improvement proposals, which is one of the most uprising topics in the blockchain era. Our main goal is to implement a ranking – in other words, a voting – system on formal on-chain blockchain governance. (E-)voting systems need a signature scheme to prevent double-voting or any other possible exploitations related to confidentiality.

In this section, we give a generic treatment to achieve blockchain governance by developing a voting system via $\sharp$AMS. Note that our applications are simplified to give a showcase of using our $\sharp$AMS. Applying our scheme to more complex voting designs could be an interesting direction. However, our simple solutions have already achieved many attractive properties, e.g., lightweight and publicly verifiable—thus, there is no need for tallying authorities.

Note that in our schemes, anonymity does not hold for the moderator, meaning that the moderator (whether honest or malicious) knows the identities of all signers. However, unforgeability still holds even against a malicious moderator. Recall that in blockchain governance, a developer P who intends to propose an improvement proposal will start the $\sharp$AMS protocol as the moderator. P's goal is to gather as many supporters as possible, and then generate a $\sharp$AMS signature to demonstrate the credibility of the proposal and thereby win the RFP. A malicious moderator P can do nothing except reduce t—which trivially harms P's utility. For this reason, we assume the moderator is honest in the following analysis.

6.1 V1: Blockchain Governance via $\sharp$AMS

As introduced in Section 3, the verification function of $\sharp$AMS returns the number of the signers who participated in the signature generation, which can turn into a voting protocol. In our concrete scheme of $\sharp$AMS, each participant generates their own keys and publishes them, which is a straightforward pre-processing step in voting via blockchain. The voting protocol enables multiple improvement proposals to compete for votes, and the one with the most votes is elected. Our on-chain governance voting protocol processes as follows:

1. A *voting session* implies the whole process of this protocol. For each RFP, the participants run a voting session to evaluate proposals. Each session has four time periods: *posting period*, *declaration period*, *signing period*, and *announcement period*. We denote $[n]$ as the index set of voters while U_i implies an i-th voter for $i \in [n]$. Similarly, for $p < n$, $[p]$ is the index set of developers who propose an improvement proposal while P_j stands for j-th developer and $IP^{(j)}$ stands for the proposal made by P_j. Note that $P_j = U_j$ for $j \in [p]$, which means P_j is also eligible to vote and will lead the quorum of $IP^{(j)}$. P_j serves a dual role as both the initiator of $IP^{(j)}$ and the representative of voters of that proposal, ensuring the concealment of their identities during signature generation.
2. In the posting period, for all $j \in [p]$, P_j submits $IP^{(j)}$. This can be done via the blockchain network using a smart contract.
3. In the declaration period, for each $i \in [n]$, U_i expresses their will to vote on $IP^{(j)}$ by generating a random string pair $(\bar{m}_i, \bar{r}_i)$ using their own randomness, which should be kept in secret, and sending $h_i = \mathsf{Hash}(hk_i, \bar{m}_i, \bar{r}_i)$ to P_j.
4. In the signing period, P_j sends m_i to U_i and U_i sends r_i to P_j, where m_i and r_i are defined in the description of $\sharp$AMS in Section 5.
5. In the announcement period, P_j generates a signature $\sigma^{(j)}$ from (m_i, r_i) pairs for each voter and uploads it to the blockchain system. In the end, the most voted proposal is elected.

See Fig. 2 for the pictorial explanation of our voting system.

6.2 V2: Round Optimization

Recall that in the voting scheme above, after publishing the improvement proposal, each participant has to execute three rounds of interaction before uploading the $\sharp$AMS signature on the blockchain.

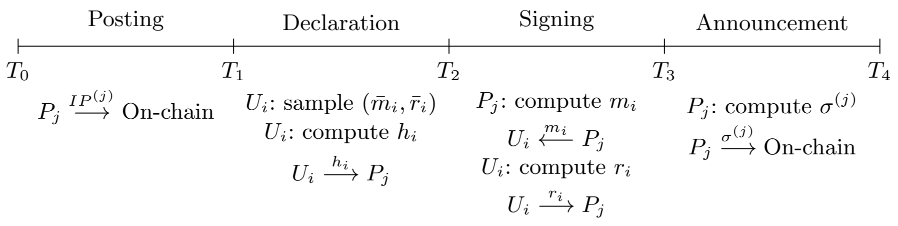
Posting
Declaration
Signing
Announcement
T_0
T_1
T_2
T_3
T_4
$P_j \xrightarrow{IP^{(j)}}$ On-chain
U_i: sample $(\bar{m}_i, \bar{r}_i)$
U_i: compute h_i
$U_i \xrightarrow{h_i} P_j$
P_j: compute m_i
$U_i \xleftarrow{m_i} P_j$
U_i: compute r_i
$U_i \xrightarrow{r_i} P_j$
P_j: compute $\sigma^{(j)}$
$P_j \xrightarrow{\sigma^{(j)}}$ On-chain

Fig. 2. Timeline of V1. Here U_i votes to $IP^{(j)}$.

– Round 1 (in the declaration period): P_j receives h_i as a voting-claim from its supporter U_i.
– Round 2 (in the signing period): P_j computes m_i according to the signing algorithm of $\sharp$AMS and sends m_i to U_i.
– Round 3 (in the signing period): U_i returns r_i such that $\mathsf{Hash}(hk_i, m_i, r_i) = h_i$.

After Round 3, P_j can finish signing and upload the signature σ on the blockchain. Now we come up with the following question:

Can we reduce the round complexity? Namely, can we optimize the protocol such that the developer P_j can generate the $\sharp$AMS signature immediately after receiving the voting claims from its supporters?

The natural idea is to let the supporter, say U_i, send its secret key (the trapdoor of chameleon hash schemes) directly to the developer P_j. With the knowledge of all secret keys, P_j can now generate a $\sharp$AMS signature without interaction with its supporters. However, this will totally expose users' secret keys to the developers in a single vote event, which is not the case users want.

Our idea is to use $\sharp$AMS in a one-time paradigm. That is, for each proposal $IP^{(j)}$ proposed by the developer P_j, the supporter U_i generates a new hash key and trapdoor pair $(hk_i^{(j)}, td_i^{(j)})$, and then sends the key pair to P_j. For a user U_k who does not want to support $IP^{(j)}$, it also generates a key pair $(hk_k^{(j)}, td_k^{(j)})$, but then sends only the hash key to P_j. This hash key is used for P_j to add U_k into the anonymous group to hide the identities of the real voters. Moreover, we have the following two modifications.

1. To make sure that a hash key $hk_i^{(j)}$ is generated from voter U_i but not from developer P_j (otherwise it can always make an n-valid $\sharp$AMS signature with n the total number of voters), U_i will sign a signature on $hk_i^{(j)}$ to show its authority. (See Appendix A for the syntax of signatures.)
2. The message from U_i to P_j is encrypted using P_j's public key, so that no eavesdropper except P_j will know the corresponding trapdoor $td_i^{(j)}$. Meanwhile, the message, either $hk_i^{(j)}$ or $(hk_i^{(j)}, td_i^{(j)})$, is padded to the same length, preventing the side-leakage of anonymity.

Formally, the round-optimal protocol is described as follows. Let $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a (regular) signature scheme with unforgeability, and PKE be a public key encryption scheme with CPA security. At the beginning, we assume that every user U_i for $i \in [n]$ has its own key pairs $(\widetilde{vk_i}, \widetilde{sk_i})$ of Sig and (pk_i, sk_i) of PKE.

1. In the posting period, developers $P_1, ..., P_p$ submit their improvement proposals $IP^{(1)}, ..., IP^{(p)}$ to the blockchain network using a smart contract.
2. In the declaration & signing period, for all $i \in [n]$ and $j \in [p]$,
   (a) U_i invokes $(hk_i^{(j)}, td_i^{(j)}) \leftarrow \mathsf{LCH.Gen}(1^\lambda)$ and $cert_i^{(j)} \leftarrow \mathsf{Sign}(\widetilde{sk_i}, hk_i^{(j)})$;
   (b) If U_i wants to vote $IP^{(j)}$, U_i computes $ct_{i\to j} \leftarrow \mathsf{Enc}(pk_j, hk_i^{(j)} || cert_i^{(j)} || td_i^{(j)})$;
   (c) If U_i does not want to vote $IP^{(j)}$, U_i computes $ct_{i\to j} \leftarrow \mathsf{Enc}(pk_j, hk_i^{(j)} \; || cert_i^{(j)} || \mathbf{0})$, where $\mathbf{0}$ is a zero-string of length $|td_i^{(j)}|$;
   (d) U_i sends $ct_{i\to j}$ to P_j.

3. In the announcement period, for each $j \in [p]$, after decrypting $ct_{i\to j}$ for all $i \in [n]$, the developer P_j obtains a series of hash keys $\{hk_i^{(j)}\}_{i\in[n]}$ as well as their corresponding certificates $\{cert_i^{(j)}\}_{i\in[n]}$. Meanwhile, P_j also gets a group of trapdoors $\{td_i^{(j)}\}$ from users U_i who are willing to support $IP^{(j)}$. P_j can check the validity of certificates using long-term verification keys. If the verification fails with respect to user U_i, then P_j discards U_i from the anonymous group. Finally, P_j generates a $\sharp$AMS signature $\sigma^{(j)}$ for its proposal $IP^{(j)}$ and uploads the $\sharp$AMS signature, all hash keys, and certificates concerning the proposal $IP^{(j)}$ to the blockchain system.

Note that if a user U_i behaves in a Byzantine manner, for example, by failing to send $ct_{i\to j}$ to P_j, this misbehavior will be easily detected after the announcement period ends. In such cases, all developers can simply re-run the announcement period with the participant set updated to $[n] \setminus i$. This approach naturally generalizes to tolerate any number of faulty users, with the only cost being an additional announcement period for each round of fault recovery.

6.3 V3: Single-Vote Setting via the Conditioned Key Generation Paradigm

A voting system in the single-vote setting is desirable in many applications. For example, if two proposals conflict, an unruly user voting on both will undoubtedly disrupt the normal voting process and outcomes. However, in the protocols described above, users can vote on several proposals. More precisely, suppose there are two distinct proposals, $IP^{(j)}$ by P_j and $IP^{(k)}$ by P_k with $j \neq k \in [p]$. User U_i can generate two key pairs $(hk_i^{(j)}, td_i^{(j)})$ and $(hk_i^{(k)}, td_i^{(k)})$, then vote on both proposals by sending the corresponding key pairs to P_j and P_k, respectively. Due to the anonymity property of $\sharp$AMS, the verifier cannot determine which users have voted multiple times.

To prevent double voting, we introduce a new paradigm, dubbed the conditioned key generation paradigm. Recall that to vote on a proposal $IP^{(j)}$ (i.e., to participate in the signing process for the proposal), user U_i must have the trapdoor $td_i^{(j)}$ corresponding to the hash key $hk_i^{(j)}$ specifically designed for $IP^{(j)}$. User U_i needs to generate p hash keys $(hk_i^{(1)}, ..., hk_i^{(p)})$ initially. Our idea is to enforce a restriction such that among all p hash keys, the user can know at most one trapdoor.

The conditioned key generation paradigm is applying t-out-of-n proof strategy during the subkey generation. Assume the hash key space $\mathcal{HK}$ is a field. Let $\mathbf{B} = (b_{i,j}) \in \mathcal{HK}^{(p-1)\times p}$ be a public matrix of full rank. Let $\mathbf{hk}_i^\top = ((hk_i^{(1)}, ..., hk_i^{(p)})^\top$. Now we require

$$\mathbf{B} \cdot \mathbf{hk}_i = \hat{\mathbf{hk}}_i,$$

where $\hat{\mathbf{hk}}_i \in \mathcal{HK}^{p-1}$ is the output of some hash function $H(IP^{(j_1)}, ..., IP^{(j_p)}, i)$.

If the hash key generated via $\mathsf{Gen}(1^\lambda)$ is computationally indistinguishable from a uniform hash key in $\mathcal{HK}$, and $H(\cdot)$ is a random oracle, then to satisfy the aforementioned equation, each user has at most one trapdoor of the p hash keys. Single-voting property is achieved as a result. See Fig. 3 for the pictorial explanation of our round-optimal voting system in multiple and single-vote settings.

References

[1] Abdalla, M., Fouque, P., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012, Cambridge, UK, April 15-19, 2012. Proceedings. vol. 7237, pp. 572–590. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_34, https://doi.org/10.1007/978-3-642-29011-4_34

[2] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. pp. 99–108. ACM (1996). https://doi.org/10.1145/237814.237838, https://doi.org/10.1145/237814.237838

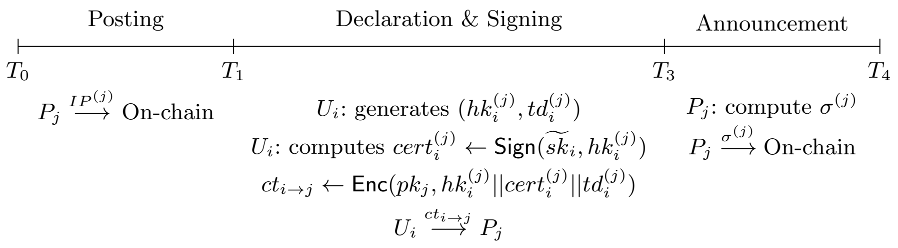
Posting
Declaration & Signing
Announcement
T_0
T_1
T_3
T_4
$P_j \xrightarrow{IP^{(j)}}$ On-chain
U_i: generates $(hk_i^{(j)}, td_i^{(j)})$
U_i: computes $cert_i^{(j)} \leftarrow \mathsf{Sign}(\widetilde{sk}_i, hk_i^{(j)})$
$ct_{i\to j} \leftarrow \mathsf{Enc}(pk_j, hk_i^{(j)}||cert_i^{(j)}||td_i^{(j)})$
$U_i \xrightarrow{ct_{i\to j}} P_j$
P_j: compute $\sigma^{(j)}$
$P_j \xrightarrow{\sigma^{(j)}}$ On-chain

Fig. 3. Timeline of V2 and V3. Here U_i votes to $IP^{(j)}$. In V2, U_i also needs to compute and send $ct_{i\to k} = \mathsf{Enc}(pk_k, hk_i^{(k)}||cert_i^{(k)}||td_i^{(k)})$ or $ct_{i\to k} = \mathsf{Enc}(pk_k, hk_i^{(k)}||cert_i^{(k)}||\mathbf{0})$ to P_k for all $k \in [p]$. In V3, there should be a unique $j \in [p]$ for each $i \in [n]$ such that U_i uniquely upvotes to $IP^{(j)}$. For all $k \in [p]$ such that $k \neq j$, U_i should generate $hk_i^{(k)}$ by following the rule described in the context.

[3] Alwen, J., Ostrovsky, R., Zhou, H.S., Zikas, V.: Incoercible multi-party computation and universally composable receipt-free voting. In: Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II 35. pp. 763–780. Springer (2015)

[4] Backes, M., Hritcu, C., Maffei, M.: Automated verification of remote electronic voting protocols in the applied pi-calculus. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium, CSF 2008, Pittsburgh, Pennsylvania, USA, 23-25 June 2008. pp. 195–209. IEEE Computer Society (2008). https://doi.org/10.1109/CSF.2008.26, https://doi.org/10.1109/CSF.2008.26

[5] Baum, C., Orsini, E., Scholl, P., Soria-Vazquez, E.: Efficient constant-round MPC with identifiable abort and public verifiability. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II. vol. 12171, pp. 562–592. Springer (2020). https://doi.org/10.1007/978-3-030-56880-1_20, https://doi.org/10.1007/978-3-030-56880-1_20

[6] Beck, R., Müller-Bloch, C., King, J.L.: Governance in the blockchain economy: A framework and research agenda. J. Assoc. Inf. Syst. **19**(10), 1 (2018), https://aisel.aisnet.org/jais/vol19/iss10/1

[7] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, Cologne, Germany, April 26-30, 2009. Proceedings. vol. 5479, pp. 1–35. Springer (2009). https://doi.org/10.1007/978-3-642-01001-9_1, https://doi.org/10.1007/978-3-642-01001-9_1

[8] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) CCS 2006. pp. 390–399. ACM (2006). https://doi.org/10.1145/1180405.1180453, https://doi.org/10.1145/1180405.1180453

[9] Bellare, M., Ristov, T.: Hash functions from sigma protocols and improvements to VSH. In: Pieprzyk, J. (ed.) Advances in Cryptology - ASIACRYPT 2008, Melbourne, Australia, December 7-11, 2008. Proceedings. vol. 5350, pp. 125–142. Springer (2008). https://doi.org/10.1007/978-3-540-89255-7_9, https://doi.org/10.1007/978-3-540-89255-7_9

[10] Bellare, M., Ristov, T.: A characterization of chameleon hash functions and new, efficient designs. J. Cryptol. **27**(4), 799–823 (2014). https://doi.org/10.1007/s00145-013-9155-8, https://doi.org/10.1007/s00145-013-9155-8

[11] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 1–10. ACM (1988). https://doi.org/10.1145/62212.62213, https://doi.org/10.1145/62212.62213

[12] Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In: Advances in Cryptology - EUROCRYPT 1993. vol. 765, pp. 274–285. Springer (1993). https://doi.org/10.1007/3-540-48285-7_24, https://doi.org/10.1007/3-540-48285-7_24

[13] Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: Leighton, F.T., Goodrich, M.T. (eds.) Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada. pp. 544–553. ACM (1994). https://doi.org/10.1145/195058.195407, https://doi.org/10.1145/195058.195407

[14] Bettaieb, S., Schrek, J.: Improved lattice-based threshold ring signature scheme. In: Gaborit, P. (ed.) PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings. vol. 7932, pp. 34–51. Springer (2013). https://doi.org/10.1007/978-3-642-38616-9_3, https://doi.org/10.1007/978-3-642-38616-9_3

[15] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y. (ed.) Public Key Cryptography - PKC 2003. vol. 2567, pp. 31–46. Springer (2003). https://doi.org/10.1007/3-540-36288-6_3, https://doi.org/10.1007/3-540-36288-6_3

[16] Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) Advances in Cryptology - CRYPTO 2002, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. vol. 2442, pp. 465–480. Springer (2002). https://doi.org/10.1007/3-540-45708-9_30, https://doi.org/10.1007/3-540-45708-9_30

[17] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: EUROCRYPT 2010, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. vol. 6110, pp. 523–552. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_27, https://doi.org/10.1007/978-3-642-13190-5_27

[18] Cayrel, P., Lindner, R., Rückert, M., Silva, R.: A lattice-based threshold ring signature scheme. In: Abdalla, M., Barreto, P.S.L.M. (eds.) Progress in Cryptology - LATINCRYPT 2010, Puebla, Mexico, August 8-11, 2010, Proceedings. vol. 6212, pp. 255–272. Springer (2010). https://doi.org/10.1007/978-3-642-14712-8_16, https://doi.org/10.1007/978-3-642-14712-8_16

[19] Chan, T.K., Fung, K., Liu, J.K., Wei, V.K.: Blind spontaneous anonymous group signatures for ad hoc groups. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) Security in Ad-hoc and Sensor Networks, First European Workshop, ESAS 2004, Heidelberg, Germany, August 6, 2004, Revised Selected Papers. vol. 3313, pp. 82–94. Springer (2004). https://doi.org/10.1007/978-3-540-30496-8_8, https://doi.org/10.1007/978-3-540-30496-8_8

[20] Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 11–19. ACM (1988). https://doi.org/10.1145/62212.62214, https://doi.org/10.1145/62212.62214

[21] Chaum, D., Evertse, J., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Chaum, D., Price, W.L. (eds.) Advances in Cryptology - EUROCRYPT '87, Amsterdam, The Netherlands, April 13-15, 1987, Proceedings. vol. 304, pp. 127–141. Springer (1987). https://doi.org/10.1007/3-540-39118-5_13, https://doi.org/10.1007/3-540-39118-5_13

[22] Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theor. Comput. Sci. **777**, 155–183 (2019). https://doi.org/10.1016/j.tcs.2019.02.001, https://doi.org/10.1016/j.tcs.2019.02.001

[23] Choi, W., Liu, X., Xia, L., Zikas, V.: K-linkable ring signatures and applications in generalized voting. IACR Cryptol. ePrint Arch. p. 243 (2025), https://eprint.iacr.org/2025/243

[24] Chow, S.S.M., Hui, L.C.K., Yiu, S.: Identity based threshold ring signature. In: Park, C., Chee, S. (eds.) ICISC 2004, Seoul, Korea, December 2-3, 2004, Revised Selected Papers. vol. 3506, pp. 218–232. Springer (2004). https://doi.org/10.1007/11496618_17, https://doi.org/10.1007/11496618_17

[25] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. vol. 839, pp. 174–187. Springer (1994). https://doi.org/10.1007/3-540-48658-5_19, https://doi.org/10.1007/3-540-48658-5_19

[26] Dallot, L., Vergnaud, D.: Provably secure code-based threshold ring signatures. In: Parker, M.G. (ed.) Cryptography and Coding 2009, Cirencester, UK, December 15-17, 2009. Proceedings. vol. 5921, pp. 222–235. Springer (2009). https://doi.org/10.1007/978-3-642-10868-6_13, https://doi.org/10.1007/978-3-642-10868-6_13

[27] Delaune, S., Kremer, S., Ryan, M.: Coercion-resistance and receipt-freeness in electronic voting. In: 19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy. pp. 28–42. IEEE Computer Society (2006). https://doi.org/10.1109/CSFW.2006.8, https://doi.org/10.1109/CSFW.2006.8

[28] Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols: A taster. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections, New Directions in Electronic Voting. Lecture Notes in Computer Science, vol. 6000, pp. 289–309. Springer (2010). https://doi.org/10.1007/978-3-642-12980-3_18, https://doi.org/10.1007/978-3-642-12980-3_18

[29] Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Garay, J.A. (ed.) Public-Key Cryptography - PKC 2021, Virtual Event, May 10-13, 2021, Proceedings, Part II. vol. 12711, pp. 1–31. Springer (2021). https://doi.org/10.1007/978-3-030-75248-4_1, https://doi.org/10.1007/978-3-030-75248-4_1

[30] El Kaafarani, A., Katsumata, S., Pintore, F.: Lossy csi-fish: Efficient signature scheme with tight reduction to decisional CSIDH-512. In: Public-Key Cryptography - PKC 2020. vol. 12111, pp. 157–186. Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_6, https://doi.org/10.1007/978-3-030-45388-6_6

[31] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. vol. 263, pp. 186–194. Springer (1986). https://doi.org/10.1007/3-540-47721-7_12, https://doi.org/10.1007/3-540-47721-7_12

[32] Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II. vol. 10992, pp. 33–62. Springer (2018). https://doi.org/10.1007/978-3-319-96881-0_2, https://doi.org/10.1007/978-3-319-96881-0_2

[33] Fujisaki, E., Suzuki, K.: Traceable ring signature. In: Okamoto, T., Wang, X. (eds.) Public Key Cryptography - PKC 2007, Beijing, China, April 16-20, 2007, Proceedings. vol. 4450, pp. 181–200. Springer (2007). https://doi.org/10.1007/978-3-540-71677-8_13, https://doi.org/10.1007/978-3-540-71677-8_13

[34] Garay, J.A., Kiayias, A., Ostrovsky, R.M., Panagiotakos, G., Zikas, V.: Resource-restricted cryptography: Revisiting MPC bounds in the proof-of-work era. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. vol. 12106, pp. 129–158. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_5, https://doi.org/10.1007/978-3-030-45724-2_5

[35] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. IACR Cryptol. ePrint Arch. p. 432 (2007), http://eprint.iacr.org/2007/432

[36] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, 2008. pp. 197–206. ACM (2008). https://doi.org/10.1145/1374376.1374407, https://doi.org/10.1145/1374376.1374407

[37] Gersbach, H., Mamageishvili, A., Schneider, M.: Vote delegation and misbehavior. In: Caragiannis, I., Hansen, K.A. (eds.) SAGT 2021, Aarhus, Denmark, September 21-24, 2021, Proceedings. vol. 12885, p. 411. Springer (2021), https://link.springer.com/content/pdf/bbm%3A978-3-030-85947-3%2F1.pdf

[38] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987). https://doi.org/10.1145/28395.28420, https://doi.org/10.1145/28395.28420

[39] Goodman, L.: Tezos—a self-amending crypto-ledger white paper. URL: https://www. tezos. com/static/papers/white paper. pdf **4**, 1432–1465 (2014)

[40] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27. pp. 415–432. Springer (2008)

[41] Haque, A., Scafuro, A.: Threshold ring signatures: New definitions and post-quantum security. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II. vol. 12111, pp. 423–452. Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_15, https://doi.org/10.1007/978-3-030-45388-6_15

[42] Heather, J., Schneider, S.A.: A formal framework for modelling coercion resistance and receipt freeness. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7436, pp. 217–231. Springer (2012). https://doi.org/10.1007/978-3-642-32759-9_19, https://doi.org/10.1007/978-3-642-32759-9_19

[43] Hemenway, B., Libert, B., Ostrovsky, R., Vergnaud, D.: Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011, Seoul, South Korea, December 4-8, 2011. Proceedings. vol. 7073, pp. 70–88. Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_4, https://doi.org/10.1007/978-3-642-25385-0_4

[44] Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 539–556. Springer (2000)

[45] Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II. vol. 8617, pp. 369–386. Springer (2014). https://doi.org/10.1007/978-3-662-44381-1_21, https://doi.org/10.1007/978-3-662-44381-1_21

[46] Jonker, H.L., de Vink, E.P.: Formalising receipt-freeness. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4176, pp. 476–488. Springer (2006). https://doi.org/10.1007/11836810_34, https://doi.org/10.1007/11836810_34

[47] Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Chaum, D., Jakobsson, M., Rivest, R.L., Ryan, P.Y.A., Benaloh, J., Kutylowski, M., Adida, B. (eds.) Towards Trustworthy Elections, New

Directions in Electronic Voting. Lecture Notes in Computer Science, vol. 6000, pp. 37–63. Springer (2010). https://doi.org/10.1007/978-3-642-12980-3_2, https://doi.org/10.1007/978-3-642-12980-3_2

[48] Khan, N., Ahmad, T., Patel, A., State, R.: Blockchain governance: An overview and prediction of optimal strategies using nash equilibrium. CoRR **abs/2003.09241** (2020), https://arxiv.org/abs/2003.09241

[49] Kiayias, A., Lazos, P.: Sok: Blockchain governance. In: Herlihy, M., Narula, N. (eds.) AFT 2022, Cambridge, MA, USA, September 19-21, 2022. pp. 61–73. ACM (2022). https://doi.org/10.1145/3558535.3559794, https://doi.org/10.1145/3558535.3559794

[50] Kiayias, A., Osmanoglu, M., Tang, Q.: Graded signatures. In: Information Security - ISC 2015. vol. 9290, pp. 61–80. Springer (2015). https://doi.org/10.1007/978-3-319-23318-5_4, https://doi.org/10.1007/978-3-319-23318-5_4

[51] Komlo, C., Goldberg, I.: FROST: flexible round-optimized schnorr threshold signatures. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers. vol. 12804, pp. 34–65. Springer (2020). https://doi.org/10.1007/978-3-030-81652-0_2, https://doi.org/10.1007/978-3-030-81652-0_2

[52] Krawczyk, H., Rabin, T.: Chameleon hashing and signatures. IACR Cryptol. ePrint Arch. p. 10 (1998), http://eprint.iacr.org/1998/010

[53] Küsters, R., Truderung, T.: An epistemic approach to coercion-resistance for electronic voting protocols. In: 30th IEEE Symposium on Security and Privacy (SP 2009), 17-20 May 2009, Oakland, California, USA. pp. 251–266. IEEE Computer Society (2009). https://doi.org/10.1109/SP.2009.13, https://doi.org/10.1109/SP.2009.13

[54] Küsters, R., Truderung, T., Vogt, A.: Verifiability, privacy, and coercion-resistance: New insights from a case study. In: 32nd IEEE Symposium on Security and Privacy, SP 2011, 22-25 May 2011, Berkeley, California, USA. pp. 538–553. IEEE Computer Society (2011). https://doi.org/10.1109/SP.2011.21, https://doi.org/10.1109/SP.2011.21

[55] Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion resistance and its applications. Journal of Computer Security **20**(6), 709–764 (2012)

[56] Li, X., Zheng, D., Chen, K.: Efficient linkable ring signatures and threshold signatures from linear feedback shift register. In: Jin, H., Rana, O.F., Pan, Y., Prasanna, V.K. (eds.) ICA3PP 2007, Hangzhou, China, June 11-14, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4494, pp. 95–106. Springer (2007). https://doi.org/10.1007/978-3-540-72905-1_9, https://doi.org/10.1007/978-3-540-72905-1_9

[57] Liu, J.K., Wei, V.K., Wong, D.S.: A separable threshold ring signature scheme. In: Lim, J.I., Lee, D.H. (eds.) Information Security and Cryptology - ICISC 2003, Seoul, Korea, November 27-28, 2003, Revised Papers. vol. 2971, pp. 12–26. Springer (2003). https://doi.org/10.1007/978-3-540-24691-6_2, https://doi.org/10.1007/978-3-540-24691-6_2

[58] Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings. vol. 3108, pp. 325–335. Springer (2004). https://doi.org/10.1007/978-3-540-27800-9_28, https://doi.org/10.1007/978-3-540-27800-9_28

[59] Lyu, J., Jiang, Z.L., Wang, X., Nong, Z., Au, M.H., Fang, J.: A secure decentralized trustless e-voting system based on smart contract. In: 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 13th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE 2019, Rotorua, New Zealand, August 5-8, 2019. pp. 570–577. IEEE (2019). https://doi.org/10.1109/TRUSTCOM/BIGDATASE.2019.00082, https://doi.org/10.1109/TrustCom/BigDataSE.2019.00082

[60] Melchor, C.A., Cayrel, P., Gaborit, P.: A new efficient threshold ring signature scheme based on coding theory. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, Proceedings. vol. 5299, pp. 1–16. Springer (2008). https://doi.org/10.1007/978-3-540-88403-3_1, https://doi.org/10.1007/978-3-540-88403-3_1

[61] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012, Cambridge, UK, April 15-19, 2012. Proceedings. vol. 7237, pp. 700–718. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_41, https://doi.org/10.1007/978-3-642-29011-4_41

[62] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. In: 45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings. pp. 372–381. IEEE Computer Society (2004). https://doi.org/10.1109/FOCS.2004.72, https://doi.org/10.1109/FOCS.2004.72

[63] Michels, M., Horster, P.: Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 125–132. Springer (1996)

[64] Munch-Hansen, A., Orlandi, C., Yakoubov, S.: Stronger notions and a more efficient construction of threshold ring signatures. In: Progress in Cryptology - LATINCRYPT 2021. vol. 12912, pp. 363–381. Springer (2021). https://doi.org/10.1007/978-3-030-88238-9_18, https://doi.org/10.1007/978-3-030-88238-9_18

[65] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (May 2009), http://www.bitcoin.org/bitcoin.pdf

[66] Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1361, pp. 25–35. Springer (1997). https://doi.org/10.1007/BFB0028157, https://doi.org/10.1007/BFb0028157

[67] Pan, J., Wagner, B.: Lattice-based signatures with tight adaptive corruptions and more. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022, Virtual Event, March 8-11, 2022, Proceedings, Part II. vol. 13178, pp. 347–378. Springer (2022). https://doi.org/10.1007/978-3-030-97131-1_12, https://doi.org/10.1007/978-3-030-97131-1_12

[68] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 187–196. ACM (2008). https://doi.org/10.1145/1374376.1374406, https://doi.org/10.1145/1374376.1374406

[69] van Pelt, R., Jansen, S., Baars, D., Overbeek, S.: Defining blockchain governance: A framework for analysis and comparison. Inf. Syst. Manag. **38**(1), 21–41 (2021). https://doi.org/10.1080/10580530.2020.1720046, https://doi.org/10.1080/10580530.2020.1720046

[70] Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: Johnson, D.S. (ed.) Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA. pp. 73–85. ACM (1989). https://doi.org/10.1145/73007.73014, https://doi.org/10.1145/73007.73014

[71] Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) Advances in Cryptology - ASIACRYPT 2001, Gold Coast, Australia, December 9-13, 2001, Proceedings. vol. 2248, pp. 552–565. Springer (2001). https://doi.org/10.1007/3-540-45682-1_32, https://doi.org/10.1007/3-540-45682-1_32

[72] Russo, A., Anta, A.F., Vasco, M.I.G., Romano, S.P.: Chirotonia: A scalable and secure e-voting framework based on blockchains and linkable ring signatures. In: Xiang, Y., Wang, Z., Wang, H., Niemi, V. (eds.) 2021 IEEE International Conference on Blockchain, Blockchain 2021, Melbourne, Australia, December 6-8, 2021. pp. 417–424. IEEE (2021). https://doi.org/10.1109/BLOCKCHAIN53845.2021.00065, https://doi.org/10.1109/Blockchain53845.2021.00065

[73] Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J. (eds.) Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding. Lecture Notes in Computer Science, vol. 921, pp. 393–403. Springer (1995). https://doi.org/10.1007/3-540-49264-X_32, https://doi.org/10.1007/3-540-49264-X_32

[74] Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000, Bruges, Belgium, May 14-18, 2000, Proceeding. vol. 1807, pp. 207–220. Springer (2000). https://doi.org/10.1007/3-540-45539-6_15, https://doi.org/10.1007/3-540-45539-6_15

[75] Stinson, D.R., Strobl, R.: Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001, Sydney, Australia, July 11-13, 2001, Proceedings. vol. 2119, pp. 417–434. Springer (2001). https://doi.org/10.1007/3-540-47719-5_33, https://doi.org/10.1007/3-540-47719-5_33

[76] Venugopalan, S., Homoliak, I.: Always on voting: A framework for repetitive voting on the blockchain. CoRR **abs/2107.10571** (2021), https://arxiv.org/abs/2107.10571

[77] Wei, B., Du, Y., Zhang, H., Zhang, F., Tian, H., Gao, C.: Identity based threshold ring signature from lattices. In: Au, M.H., Carminati, B., Kuo, C.J. (eds.) NSS 2014, Xi'an, China, October 15-17, 2014, Proceedings. vol. 8792, pp. 233–245. Springer (2014). https://doi.org/10.1007/978-3-319-11698-3_18, https://doi.org/10.1007/978-3-319-11698-3_18

[78] Wong, D.S., Fung, K., Liu, J.K., Wei, V.K.: On the rs-code construction of ring signature schemes and a threshold setting of RST. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003, Huhehaote, China, October 10-13, 2003, Proceedings. vol. 2836, pp. 34–46. Springer (2003). https://doi.org/10.1007/978-3-540-39927-8_4, https://doi.org/10.1007/978-3-540-39927-8_4

[79] Xu, F., Lv, X.: A new identity-based threshold ring signature scheme. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Anchorage, Alaska, USA, October 9-12, 2011. pp. 2646–2651. IEEE (2011). https://doi.org/10.1109/ICSMC.2011.6083996, https://doi.org/10.1109/ICSMC.2011.6083996

[80] Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. pp. 160–164. IEEE Computer Society (1982). https://doi.org/10.1109/SFCS.1982.38, https://doi.org/10.1109/SFCS.1982.38

A Basic Notations and Additional Preliminaries

Let $\lambda \in \mathbb{N}$ denote the security parameter. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, ..., \mu\}$. Denote by $x := y$ the operation of assigning y to x. Denote by $x \overset{\$}{\leftarrow} \mathcal{S}$ the operation of sampling x uniformly at random from a set $\mathcal{S}$. For a distribution $\mathcal{D}$, denote by $x \leftarrow \mathcal{D}$ the operation of sampling x according to $\mathcal{D}$. For an algorithm $\mathcal{A}$, denote by $y \leftarrow \mathcal{A}(x; r)$, or simply $y \leftarrow \mathcal{A}(x)$, the operation of running $\mathcal{A}$ with input x and randomness r and assigning the output to y. For deterministic algorithms $\mathcal{A}$, we also write as $y := \mathcal{A}(x)$ or $y := \mathcal{A}(x; r)$. "PPT" is short for probabilistic polynomial-time.

Statistical distances and min-entropy. Let X and Y be two random variables (distributions) defined over $\mathcal{S}$. The min-entropy of X is defined as $\mathbf{H}_\infty(X) := -\log(\max_{s\in\mathcal{S}} \Pr[X = s])$. The statistical distance between X and Y is defined as $\mathit{\Delta}_{X,Y} := 1/2 \sum_{s\in\mathcal{S}} |\Pr[X = s] - \Pr[Y = s]|$. If $\mathit{\Delta}_{X,Y} = \epsilon$, we also say X and Y are ϵ-close.

A.1 Public key encryption

Definition 8 (Public-Key Encryption). *A public key encryption (PKE) scheme consists of the following three algorithms. Namely,* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. *The key generation algorithm takes as input the security parameter* 1^λ*, and outputs an encryption public key* pk *and a decryption secret key* sk.
- $ct \leftarrow \mathsf{Enc}(pk, \mu)$. *The encryption algorithm takes as input* pk *and a message* μ*, and outputs a ciphertext* ct*. If the randomness* r *is specific, then we also denote it as* $ct \leftarrow \mathsf{Enc}(pk, \mu; r)$ *or* $ct := \mathsf{Enc}(pk, \mu; r)$.
- $\mu' \leftarrow \mathsf{Dec}(sk, ct)$. *The decryption algorithm takes as input* sk *and a ciphertext* ct*, and outputs the decryption result* μ'.

Correctness. *For every* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ *and every message* μ*, it holds that* $\mathsf{Dec}(sk, \mathsf{Enc}(pk, \mu)) = \mu$.

Definition 9 (CPA Security of PKE). *A public key encryption scheme* PKE *is secure, if for every PPT adversary* $\mathcal{A}$*, its advantage*

$$\mathsf{Adv}^{cpa}_{\mathsf{PKE},\mathcal{A}}(\lambda) := |\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{A}^{\mathcal{O}_0(pk,\cdot,\cdot)} = 1] - \Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{A}^{\mathcal{O}_1(pk,\cdot,\cdot)} = 1]|$$

is negligible in λ*, where oracle* $\mathcal{O}_b(pk, \mu_0, \mu_1)$ *returns* $ct \leftarrow \mathsf{Enc}(pk, \mu_b)$.

Definition 10 (Min-Entropy of PKE). *A PKE scheme* PKE *has* κ*-min-entropy, if for every* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, every* μ*, it holds that* $\mathbf{H}_\infty(\mathsf{Enc}(pk, \mu; r)) \geq \kappa$*, where* $r \overset{\$}{\leftarrow} \mathcal{R}$ *and* $\mathcal{R}$ *is the randomness space of* PKE.

A.2 Signatures

Definition 11 (Signatures). *A signature (SIG) scheme consists of the following three algorithms. Namely,* $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$.

- $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*: The key generation algorithm takes as input the security parameter* 1^λ*, and outputs a verification key* vk *and a secret key* sk.
- $\sigma \leftarrow \mathsf{Sign}(sk, \mathsf{msg})$*: The signing algorithm takes as input* sk *and a message* msg*, and outputs a signature* σ.
- $1/0 \leftarrow \mathsf{Ver}(vk, \mathsf{msg}, \sigma)$*: The verification algorithm* Ver *takes as input* pk*, a message* msg *and a signature* σ*, and outputs a bit indicating the validity of* σ.

Correctness. *For any* $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, any* msg *and* $\sigma \leftarrow \mathsf{Sign}(sk, \mathsf{msg})$*, it holds that* $\mathsf{Ver}(vk, \mathsf{msg}, \sigma) = 1$.

Definition 12 (Unforgeability of Signatures). *Let* Sig *be a signature scheme. Consider the following unforgeability experiment* $\mathsf{Exp}^{unforg}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ *between the challenger* $\mathcal{C}$ *and the adversary* $\mathcal{A}$.

1. $\mathcal{C}$ generates $(vk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and passes vk to $\mathcal{A}$.
2. $\mathcal{A}$ has access to the signing oracle $\mathcal{O}(\mathsf{msg})$ that returns $\sigma \leftarrow \mathsf{Sign}(sk, \mathsf{msg})$.
3. Finally $\mathcal{A}$ outputs a forgery $(\mathsf{msg}^*, \sigma^*)$.

$\mathsf{Exp}^{unforg}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ *outputs 1 if* $\mathsf{Ver}(vk, \mathsf{msg}^*, \sigma^*) = 1$ *and $\mathcal{A}$ never asks $\mathcal{O}(\mathsf{msg}^*)$.*

Define by $\mathsf{Adv}^{unforg}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ the probability that $\mathsf{Exp}^{unforg}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ outputs 1. We say that Sig is unforgeable, if for all PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{unforg}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ is negligible in λ.

A.3 Forking Lemma

In this subsection, we review the forking lemma proposed by Bellare and Never [8], which is an important tool for the security proof of the CH-based construction.

Lemma 1 (Forking Lemma [8]). *Fix an integer Q and a set $\mathcal{M}$. Let $\mathcal{B}$ be a randomized algorithm that on input $x, u^{(1)}, ..., u^{(Q)}$ returns a pair (j, σ), where the first element is an integer in $[Q]$ and the second element is referred as a side output. Let $\mathcal{X}$ be a distribution. We define acc, the accepting probability of $\mathcal{B}$ in the experiment as follows.*

$$acc := \Pr\left[\begin{matrix} x \xleftarrow{\$} \mathcal{X}, u^{(1)}, ..., u^{(Q)} \xleftarrow{\$} \mathcal{M} \\ (j, \sigma) \leftarrow \mathcal{B}(x, u^{(1)}, ..., u^{(Q)}) \end{matrix} : 1 \le j \le Q\right].$$

The forking algorithm $\mathcal{F}_\mathcal{B}$ associated with $\mathcal{B}$ is a randomized algorithm that takes input x and proceeds as follows.

```
Set random coins $\rho$ for $\mathcal{B}$
$u^{(1)}, ..., u^{(Q)} \xleftarrow{\$} \mathcal{M}$
$(j, \sigma) \leftarrow \mathcal{B}(x, u^{(1)}, ..., u^{(Q)}; \rho)$
If $j = 0$ then return $(0, \perp, \perp)$
$u^{(j)'}, ..., u^{(Q)'} \xleftarrow{\$} \mathcal{M}$
$(j', \sigma') \leftarrow \mathcal{B}(x, u^{(1)}, ..., u^{(j-1)}, u^{(j)'}, ..., u^{(Q)'}; \rho)$
If $(j = j' \wedge u^{(j)} \neq u^{(j)'})$ then return $(j, \sigma, \sigma')$
Otherwise return $(0, \perp, \perp)$
```

Define

$$frk := \Pr[x \leftarrow \mathcal{X}, (b, \cdot, \cdot) \leftarrow \mathcal{F}_\mathcal{B}(x) : b \neq 0].$$

Then

$$frk \ge acc \cdot (acc/Q - 1/|\mathcal{M}|).$$

B A Framework for Lossy Chameleon Hashes

In this section, we show a framework of lossy chameleon hashes defined in Definition 2, including the equivalence with lossy identification schemes, some generic constructions, and some concrete constructions from the DDH assumption and the LWE & SIS assumptions, see Fig. 1.

B.1 Security of (Lossy) Chameleon Hashes

Definition 13 (Security of CH). *A chameleon hash scheme CH is secure (resp., strongly secure) if it has uniformity, random trapdoor collision, and collision resistance (resp., strong collision resistance).*

κ-uniformity. *For any $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$, if (m, r) is distributed uniformly over $\mathcal{M} \times \mathcal{R}$, then $\mathbf{H}_\infty(\mathsf{Hash}(hk, m, r)) \geq \kappa$. Specifically, if $\mathbf{H}_\infty(\mathsf{Hash}(hk, m, r)) = \log(|\mathcal{H}|)$, we say that* CH *has perfect uniformity.*[12]

γ-random trapdoor collision (RTC). *For any $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$ and any m, m', if r is uniformly distributed over $\mathcal{R}$, then $r' \leftarrow \mathsf{TdColl}(td, m, r, m')$ has a statistical distance γ to the uniform distribution over $\mathcal{R}$. If $\gamma = 0$, we say that* CH *has perfect RTC.*

Collision resistance (CR). *For any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{cr}_{\mathsf{CH},\mathcal{A}}(\lambda) := \Pr\left[\begin{matrix}(hk, td) \leftarrow \mathsf{Gen}(1^\lambda); \\ (m, r, m', r') \leftarrow \mathcal{A}(hk)\end{matrix} : \begin{matrix}\mathsf{Hash}(hk, m, r) = \mathsf{Hash}(hk, m', r') \\ \wedge\ m \neq m'\end{matrix}\right] \leq negl(\lambda).$$

Strong collision resistance (S-CR). *For any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{s\text{-}cr}_{\mathsf{CH},\mathcal{A}}(\lambda) := \Pr\left[\begin{matrix}(hk, td) \leftarrow \mathsf{Gen}(1^\lambda); \\ (m, r, m', r') \leftarrow \mathcal{A}(hk)\end{matrix} : \begin{matrix}\mathsf{Hash}(hk, m, r) = \mathsf{Hash}(hk, m', r') \\ \wedge\ (m, r) \neq (m', r')\end{matrix}\right] \leq negl(\lambda).$$

Remark 3 (Random Trapdoor Collision in Lattice-based Constructions). For lattice-based CH schemes (e.g., [17]), the randomness r is sampled according to a (non-uniform) distribution $\tilde{\mathcal{R}}$ over $\mathcal{R}$. In this case, we modify the definition to the case over non-uniform distributions. Namely, for any m, m', if r is sampled according to $\tilde{\mathcal{R}}$, then the output $r' \leftarrow \mathsf{TdColl}(td, m, r, m')$ enjoys a distribution which is γ-close to $\tilde{\mathcal{R}}$.

Definition 14 (Security of LCH). *A lossy chameleon hash* LCH *is secure (resp. strongly secure), if it has uniformity, random trapdoor collision (RTC), collision resistance (resp., strong collision resistance), indistinguishability, and lossiness. The first three properties are defined as in Definition 13 (note that uniformity and RTC also hold for all $hk \leftarrow \mathsf{LGen}(1^\lambda)$), and the last two are defined as follows.*

Indistinguishability. *For any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{A}} := \left|\Pr[(hk, td) \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{A}(hk) = 1] - \Pr[hk \leftarrow \mathsf{LGen}(1^\lambda) : \mathcal{A}(hk) = 1]\right| \leq negl(\lambda).$$

ϵ-Lossiness. *For any (even information-theoretic) adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that*

$$\Pr\left[\begin{matrix}hk \leftarrow \mathsf{LGen}(1^\lambda); (h, st) \leftarrow \mathcal{A}_1(hk); \\ m \xleftarrow{\$} \mathcal{M}; r \leftarrow \mathcal{A}_2(m, st)\end{matrix} : \mathsf{Hash}(hk, m, r) = h\right] \leq \epsilon.$$

Definition 15 (Uniqueness of (L)CH). *We say* CH *or* LCH *is perfectly unique, if for every hk generated from $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$ and $hk \leftarrow \mathsf{LGen}(1^\lambda)$, every m, there do not exist two district $r \neq r'$ such that $\mathsf{Hash}(hk, m, r) = \mathsf{Hash}(hk, m, r')$.*

B.2 Equivalence with Lossy Identification Schemes

Bellare and Ristov [9, 10] proved that chameleon hashes and 3-move Sigma protocols are equivalent, where the hash value, the message, and the randomness of a chameleon hash corresponds to the commitment, the challenge, and the response of a Sigma protocol, respectively. In this subsection, we extend the equivalence to the lossy mode, by showing that lossy chameleon hashes and lossy identification schemes [1] are equivalent.

We first recall the definition and security properties of lossy identification schemes. By default, the lossy identification scheme we consider here is a 3-move protocol (as that in Sigma protocols) which consists of a commitment, a challenge, and a response as its transcript.

Definition 16 (Lossy Identification [1]). *A lossy identification scheme consists of the following four algorithms. Namely,* $\mathsf{LID} = (\mathsf{Gen}, \mathsf{LGen}, \mathsf{Prove}, \mathsf{Ver})$.

[12] We relax the definition of uniformity by taking m's randomness into the probability space. A stronger definition guarantees the min-entropy for any (hk, td) and any m. As we will see, the relaxed definition here is sufficient for the security proof in Section 5.

– $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. *The normal key generation algorithm takes as input the security parameter* λ *and outputs a verifier's public key* pk *and a prover's secret key* sk.
– $pk \leftarrow \mathsf{LGen}(1^\lambda)$. *The lossy key generation algorithm takes as input the security parameter* λ *and outputs a lossy public key* pk.
– Prove. *The prover algorithm takes as input the current conversation transcript and outputs the next message to be sent to the verifier.*
– Ver. *The (deterministic) verification algorithm* Ver *takes the transcript as input and output a bit, where 1 indicates acceptance and 0 otherwise.*

We assume by default, the commitment space, the challenge space $\mathcal{CH}$, *and the response space* $\mathcal{RP}$ *are defined by the public key* pk *(either generated via normal or lossy way). Moreover, for every* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, *there is a transcript oracle that works as follows.*

$$\begin{array}{l}
\mathcal{O}^{trans}_{pk,sk}() : \\
(cmt, st) \leftarrow \mathsf{Prove}(sk) \\
ch \xleftarrow{\$} \mathcal{CH} \\
resp \leftarrow \mathsf{Prove}(sk, st, ch) \\
output\ (cmt, ch, resp)
\end{array}$$

Definition 17. *Let* LID *be an identification scheme. We say* LID *is secure if it satisfies the follow properties.*

ρ-completeness for normal keys. *For every* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{Ver}(pk, cmt, ch, resp) = 1$ *holds with probability at least* ρ, *where* $(cmt, ch, resp) \leftarrow \mathcal{O}^{trans}_{pk,sk}()$.

Simulatability of transcripts (honest-verifier zero-knowledge). *For every* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, *there exists a PPT simulator* Sim *that takes only* pk *as input and outputs a simulated transcript* $(cmt, ch, resp) \leftarrow \mathsf{Sim}^{trans}_{pk}()$, *which distributes* η*-close to the output of* $\mathcal{O}^{trans}_{pk,sk}()$.

κ-min-entropy. *For any* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, *let* $(cmt, st) \leftarrow \mathsf{Prove}(sk)$ *and* $[\cdot]_1$ *be a function that maps* (cmt, st) *to* cmt, *then*

$$\mathbf{H}_\infty([\mathsf{Prove}(sk)]_1) \geq \kappa,$$

where the probability is taken over the randomness coin used in Prove.

Indistinguishability. *For any PPT adversary* $\mathcal{A}$, *the distinguish advantage*

$$\mathsf{Adv}^{ind}_{\mathsf{LID},\mathcal{A}}(\lambda) := |\Pr[(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda) : \mathcal{A}(pk) = 1] - \Pr[pk \leftarrow \mathsf{LGen}(1^\lambda) : \mathcal{A}(pk) = 1]|$$

is negligible in λ.

ϵ-lossiness. *For any (even all-powerful) adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *it holds that*

$$\Pr\left[\begin{array}{c} pk \leftarrow \mathsf{LGen}(1^\lambda); (cmt, st) \leftarrow \mathcal{A}_1^{\mathsf{Sim}^{trans}_{pk}()}(pk); \\ ch \xleftarrow{\$} \mathcal{CH}; r \leftarrow \mathcal{A}_2(ch, st) \end{array} : \mathsf{Ver}(pk, cmt, ch, resp) = 1\right] \leq \epsilon,$$

where $\mathsf{Sim}^{trans}_{pk}()$ *is the simulator described above.*

Besides, we also need the following two properties for identification schemes when constructing lossy chameleon hashes from them.

Definition 18 (Commitment Recoverability of LID). LCH *is commitment recoverable if there exists a deterministic algorithm* Com *that takes as inputs* $(pk, ch, resp)$ *and outputs a recovered commitment* cmt', *and* $\mathsf{Ver}(pk, cmt, ch, resp) = 1$ *if and only if* $cmt = \mathsf{Com}(pk, ch, resp)$.

Moreover, there exists a special simulator Sim, *which outputs an* η*-close simulated transcript by randomly sampling* $(ch, resp)$ *and then outputting* $(\mathsf{Com}(pk, ch, resp), ch, resp)$, *and* $(ch, resp)$ *can be served as the inner state* st *for* $cmt = \mathsf{Com}(pk, ch, resp)$ *in the* Prove *algorithm.*

Definition 19 (Strong Special Soundness of LID [10]). *LCH has strong special soundness, if for any PPT adversary, its advantage*

$$\mathsf{Adv}^{sss}_{\mathsf{LID},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c}(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)\\ (cmt, ch, resp, ch', resp') \leftarrow \mathcal{A}(pk)\end{array} : \begin{array}{c}(ch, resp) \neq (ch', resp')\\ \wedge\ \mathsf{Ver}(pk, cmt, ch, resp) = 1\\ \wedge\ \mathsf{Ver}(pk, cmt, ch', resp') = 1\end{array}\right]$$

is negligible in λ.

From LCH to LID. Let $\mathsf{LCH} = (\mathsf{Gen}, \mathsf{LGen}, \mathsf{Hash}, \mathsf{TdColl})$ be a secure lossy identification scheme. We construct a lossy identification scheme LID from LCH as follows.

	Prove and Ver:		
	Prover		Verifier
	(hk, td)		hk
$\underline{\mathsf{Gen}(1^\lambda)}$:			
$(hk, td) \leftarrow \mathsf{LCH.Gen}(1^\lambda)$			
Output $(pk, sk) := (hk, td)$	$\bar{m} \xleftarrow{\$} \mathcal{M}$; $\bar{r} \xleftarrow{\$} \mathcal{R}$		
	$h := \mathsf{Hash}(hk, \bar{m}, \bar{r})$	$\xrightarrow{cmt:=h}$	
$\underline{\mathsf{LGen}(1^\lambda)}$:		$\xleftarrow{ch:=m}$	$m \xleftarrow{\$} \mathcal{M}$
$hk \leftarrow \mathsf{LCH.LGen}(1^\lambda)$			
Output $pk := hk$	$r \leftarrow \mathsf{TdColl}(td, \bar{m}, \bar{r}, m)$	$\xrightarrow{resp:=r}$	If $h = \mathsf{Hash}(hk, m, r)$: output 1
			Otherwise: output 0

Fig. 4. Construction of lossy identification schemes from lossy chameleon hashes.

Theorem 4. *Let LCH be a strongly secure lossy chameleon hash scheme (i.e., it has κ-uniformity, γ-random trapdoor collision, strong collision, indistinguishability and ϵ-lossiness), then LID constructed in Fig. 4 is a secure lossy identification scheme with commitment recoverability.*

Proof. We show that ID has completeness, simulatability of transcripts, min-entropy, indistinguishability, lossiness, and commitment recoverability.

1-completeness. This is directly implied by the correctness of $\mathsf{LCH.TdColl}$.
γ-simulatability of transcripts. We construct the PPT simulator Sim as follows.

$$\begin{array}{l|l}
\mathcal{O}^{trans}_{hk,td}() : & \mathsf{Sim}_{pk}() : \\
\bar{m}, m \xleftarrow{\$} \mathcal{M}; \bar{r} \xleftarrow{\$} \mathcal{R} & m \xleftarrow{\$} \mathcal{M}; r \xleftarrow{\$} \mathcal{R} \\
h := \mathsf{Hash}(hk, m, r) & h := \mathsf{Hash}(hk, m, r) \\
r \leftarrow \mathsf{TdColl}(td, \bar{m}, \bar{r}, m) & \text{output } (h, m, r) \\
\text{output } (h, m, r) &
\end{array}$$

Since h is totally determined by pk, m, and r, the only difference between the above two distribution is the generation of r. Due to the γ-random trapdoor collision property of LCH, we know that $\mathcal{O}^{trans}_{hk,td}()$ and $\mathsf{Sim}_{pk}()$ have a statistical distance γ.

κ-min-entropy. Since the commitment cmt for Prove algorithm is just the hash value for random $\bar{m}$ and $\bar{r}$, the κ-min-entropy directly follows from the κ-uniformity of LCH.
Indistinguishability. This is directly implied by the indistinguishability of LCH.
ϵ-lossiness. This is directly implied by the ϵ-lossiness of LCH.
Commitment recoverability. This is straightforward since the commitment is the hash value of $m = ch$ and $r = resp$.

From LID to LCH. Let $\mathsf{LID} = (\mathsf{LID.Gen}, \mathsf{LID.LGen}, \mathsf{Prove}, \mathsf{Ver})$ be a secure lossy identification scheme with commitment recoverability. We construct a lossy chameleon hash scheme LCH from LID as follows.

```
$\underline{\mathsf{Gen}(1^\lambda):}$
$(pk, sk) \leftarrow \mathsf{LID.Gen}(1^\lambda)$
Output $(hk, td) := (pk, sk)$

$\underline{\mathsf{LGen}(1^\lambda):}$
$pk \leftarrow \mathsf{LID.Gen}(1^\lambda)$
Output $hk := pk$
```

```
$\underline{\mathsf{Hash}(pk, ch, resp):}$
$cmt := \mathsf{Com}(pk, ch, resp)$
// Com is the deterministic algorithm defined in
// commitment recoverability
Output $h := cmt$

$\underline{\mathsf{TdColl}(sk, ch, resp, ch'):}$
$cmt := \mathsf{Com}(pk, ch, resp)$
$st := (ch, resp)$
$resp' \leftarrow \mathsf{Prove}(sk, st, ch')$
```

Fig. 5. Construction of lossy chameleon hashes schemes from lossy identification schemes.

Theorem 5. *Let* LID *be a secure lossy identification scheme (i.e., it has ρ-completeness, η-simulatability, κ-min-entropy, indistinguishability, ϵ-lossiness) with commitment recoverability and strong special soundness, then* LCH *constructed in Fig. 5 is a secure lossy chameleon hash scheme.*

Proof. We show that LCH constructed in Figure 5 has uniformity, random trapdoor collision, strong collision resistance, indistinguishability, and lossiness.

Correctness of trapdoor collision. This is guaranteed by the η-simulatability and ρ-completeness of LID. We notice that the correctness of LCH might be imperfect.

$(\kappa + \log \eta)$-uniformity. Due to the commitment recoverability and simulatability of LID, the simulated distribution $(cmt = \mathsf{Com}(pk, ch, resp), ch, resp)$ (for random ch and $resp$) is η-close to the normal protocol transcript distribution $(cmt', ch', resp')$. Moreover, we know cmt' has a min-entropy of κ. Therefore, $cmt := \mathsf{Com}(pk, ch, resp)$ has a min-entropy at least $(\kappa + \log \eta)$.

η-random trapdoor collision. According to the η-simulatability of LID, the simulated transcript $(cmt, ch, resp)$ (for random ch and $resp$) has a statistical distance η with the real transcript, which means that the distributions of $resp$ and $resp'$ are η-close. The η-random trapdoor collision holds as a result.

Strong collision resistance. This is implied by the commitment recoverability and the strong special soundness of LID. Recall that $\mathsf{Ver}(pk, cmt, ch, resp)$ returns 1 if and only if $cmt = \mathsf{Com}(pk, ch, resp)$, where Com is the deterministic algorithm defined in Definition 18. Therefore, a tuple $(cmt, ch, resp, ch', resp)$ that breaks the strong special soundness directly leads to a collision of the constructed LCH.

Indistinguishability. This is directly implied by the indistinguishability of LID.

ϵ-lossiness. This is directly implied by the ϵ-lossiness of LID.

B.3 Construction from Re-Randomizable Encryption

We first recall the definition of re-randomizable encryption.

Definition 20. *A re-randomizable encryption (RPKE) scheme consists of four algorithms* $\mathsf{R\text{-}PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$, *where the first three algorithms are defined as normal PKE (Definition 8), and*

– *the re-randomize algorithm* ReRand *takes as input the public key pk, a ciphertext ct and a randomness r', and outputs a re-randomized ciphertext $ct' \leftarrow \mathsf{ReRand}(pk, ct; r)$.*

We require that for every $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, every message μ and $\bar{r}$, the following two distributions are identical,

$$\{r \xleftarrow{\$} \mathcal{R} : \mathsf{Enc}(pk, \mu; r)\} \text{ and } \{r \xleftarrow{\$} \mathcal{R} : \mathsf{ReRand}(pk, \mathsf{Enc}(pk, \mu; \bar{r}); r)\},$$

where $\mathcal{R}$ is the randomness space of R-PKE.

Besides the re-randomize algorithm ReRand, we additionally require an efficient collision algorithm to find our a randomness with which ReRand will re-randomize a ciphertext to a specific one.

Definition 21 (Efficient Collision of R-PKE). *Let* $\mathsf{R\text{-}PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$ *be an R-PKE scheme. We say* $\mathsf{R\text{-}PKE}$ *has efficient collision, if there exists an efficient collision algorithm* Coll *that, for any* m, $r_0, r_1, r_0' \xleftarrow{\$} \mathcal{R}$, $\mathsf{Coll}(pk, m, r_0, r_1, r_0')$ *outputs* r_1' *such that*

$$\mathsf{ReRand}(pk, \mathsf{Enc}(pk, m; r_0); r_0') = \mathsf{ReRand}(pk, \mathsf{Enc}(pk, m; r_1); r_1').$$

Meanwhile, if r_0' *is sampled according to some distribution* $\mathcal{R}$*, then* $r_1' \leftarrow \mathsf{Coll}(pk, m, r_0, r_1, r_0')$ *also satisfies the distribution* $\mathcal{R}$.

Some examples of R-PKE include ElGamal, Paillier, Regev, etc.

Now we describe our construction of LCH from R-PKE. The idea mainly follows the construction in [43], but we additionally require an efficient collision algorithm so that the constructed LCH has correctness (i.e., there exists a trapdoor to find collisions efficiently in the collision mode).

Construction of LCH from R-PKE. Let $\mathsf{R\text{-}PKE}$ be a secure R-PKE scheme with efficient collision. Let $\mathcal{R}$ be the randomness space of $\mathsf{R\text{-}PKE}$. The construction of LCH is shown as follows, where the message space is $\mathcal{M} = \{0,1\}^\ell$, and the randomness space is $\mathcal{R}^\ell$.

– $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$. $(pk, sk) \leftarrow \mathsf{R\text{-}PKE.Gen}(1^\lambda)$.
  For $i \in [\ell]$: $\bar{r}_{i,0}, \bar{r}_{i,1} \xleftarrow{\$} \mathcal{R}$, $c_{i,0} := \mathsf{Enc}(pk, 0; \bar{r}_{i,0})$, $c_{i,1} := \mathsf{Enc}(pk, 0; \bar{r}_{i,1})$.
  Return $hk := (pk, c_{1,0}, c_{1,1}, ..., c_{\ell,0}, c_{\ell,1})$ and $td := (\bar{r}_{1,0}, \bar{r}_{1,1}, ..., \bar{r}_{\ell,0}, \bar{r}_{\ell,1})$.
– $hk \leftarrow \mathsf{LGen}(1^\lambda)$. $(pk, sk) \leftarrow \mathsf{R\text{-}PKE.Gen}(1^\lambda)$.
  For $i \in [\ell]$: $\bar{r}_{i,0}, \bar{r}_{i,1} \xleftarrow{\$} \mathcal{R}$, $c_{i,0} := \mathsf{Enc}(pk, 0; \bar{r}_{i,0})$, $c_{i,1} := \mathsf{Enc}(pk, 1; \bar{r}_{i,1})$.
  Return $hk := (pk, c_{1,0}, c_{1,1}, ..., c_{\ell,0}, c_{\ell,1})$.
– $h \leftarrow \mathsf{Hash}(hk, m, r)$. Let $m = (m_1, ..., m_\ell) \in \{0,1\}^\ell$ and $r = (r_1, ..., r_\ell) \in \mathcal{R}^\ell$.
  For $i \in [\ell]$: $h_i := \mathsf{ReRand}(pk, c_{i,m_i}; r_i)$.
  Return $h := (h_1, ..., h_\ell)$.
– $r' \leftarrow \mathsf{TdColl}(td, m, r, m')$. Let $m = (m_1, ..., m_\ell) \in \{0,1\}^\ell$, $m' = (m_1', ..., m_\ell') \in \{0,1\}^\ell$, and $r = (r_1, ..., r_\ell) \in \mathcal{R}^\ell$.
  For $i \in [\ell]$:
  - If $m_i = m_i'$, then $r_i' := r_i$.
  - If $m_i \neq m_i'$, then $r_i' \leftarrow \mathsf{Coll}(pk, 0, \bar{r}_{i,0}, \bar{r}_{i,1}, r_i)$.

  Return $r' := (r_1', ..., r_\ell')$.

Theorem 6. *If* $\mathsf{R\text{-}PKE}$ *is a secure R-PKE scheme with efficient collision and* κ*-min-entropy, then* LCH *constructed above is a secure lossy chameleon hash scheme.*

Proof. We prove the properties of LCH as follows.

Correctness and (perfect) random trapdoor collision. These two properties follow directly from the efficient collision of $\mathsf{R\text{-}PKE}$.

$(\ell\kappa)$-uniformity. For every $i \in [\ell]$, any $\bar{r}_i$, $\mathsf{ReRand}(pk, \mathsf{Enc}(pk, 0; \bar{r}_i); r_i)$ has a min-entropy κ if r_i is uniformly distributed. The $(\ell\kappa)$-uniformity holds as a result.

Indistinguishability. This follows from the CPA security of $\mathsf{R\text{-}PKE}$. Namely, there exists a PPT algorithm $\mathcal{B}$ such that

$$\mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{cpa}_{\mathsf{R\text{-}PKE},\mathcal{B}}(\lambda).$$

Collision resistance. We first change the generation of hk from $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$ to $hk \leftarrow \mathsf{LGen}(1^\lambda)$, due to the indistinguishability property.

If the adversary $\mathcal{A}$ finds a collision (m, r, m', r') under the lossy key hk, then there exists at least one i s.t. $m_i \neq m_i'$ (w.l.o.g. we assume $m_i = 0$), and

$$\mathsf{ReRand}(pk, \mathsf{Enc}(pk, 0; \bar{r}_{i,0}); r_i) = h_i = \mathsf{ReRand}(pk, \mathsf{Enc}(pk, 1; \bar{r}_{i,1}); r_i').$$

Due to the correctness of $\mathsf{R\text{-}PKE}$, h_i can be decrypted to exactly either 0 or 1. Therefore collision resistance holds.

$2^{-\ell}$-lossiness. The analysis is the same as that in collision resistance. For any fixed $h = (h_1, ..., h_\ell)$, the message $\tilde{m}$ encrypted in h is uniquely determined, due to the correctness of $\mathsf{R\text{-}PKE}$. Therefore, for a random m, as long as $m \neq \tilde{m}$ (which happens with probability $(1-2^{-\ell})$), there does not exist randomness r such that $\mathsf{Dec}(sk, \mathsf{ReRand}(pk, c_{i,m_i}; r_i)) = \tilde{m}_i$ for all i.

B.4 Construction from Lossy PKE with Efficient Opening

We first recall the lossy encryption (L-PKE) as follows.

Definition 22. *A lossy encryption (L-PKE) scheme consists of four algorithms* $\mathsf{L\text{-}PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{LGen})$*, where the first three algorithms are defined as normal PKE (Definition 8), and*

– *the lossy key generation algorithm* LGen *takes as input the security parameter λ and outputs a lossy public key $pk \leftarrow \mathsf{LGen}(1^\lambda)$.*

We require that for every $pk \leftarrow \mathsf{LGen}(1^\lambda)$, every μ_0, μ_i, the following two distribution are γ-close,

$$\{r \leftarrow \mathcal{R} : \mathsf{Enc}(pk, \mu_0, r)\} \text{ and } \{r \leftarrow \mathcal{R} : \mathsf{Enc}(pk, \mu_1, r)\},$$

where $\mathcal{R}$ is the randomness space of $\mathsf{L\text{-}PKE}$.

Definition 23 (Efficient Openability of L-PKE). *An L-PKE scheme* $\mathsf{L\text{-}PKE}$ *has efficient openability, if for every $pk \leftarrow \mathsf{LGen}(1^\lambda)$, there exists a trapdoor td generated along with pk in* LGen *and an efficient algorithm* Open*, such that given any μ, μ', r, $\mathsf{Open}(td, \mu, r, \mu')$, with overwhelming probability it outputs a randomness r' such that $\mathsf{Enc}(pk, \mu, r) = \mathsf{Enc}(pk, \mu', r')$. Namely, a lossy ciphertext can be opened to any plaintext μ'.*

Moreover, we require that the output $r \leftarrow \mathsf{Open}(td, \mu, r, \mu')$ satisfies the same distribution as r except a negligible probability.

Now we construct a lossy chameleon hash scheme from any lossy PKE scheme.

Construction of LCH from L-PKE. Let $\mathsf{L\text{-}PKE}$ be a secure L-PKE scheme with efficient openability, and $\mathcal{M}$ and $\mathcal{R}$ be the message space and randomness space of $\mathsf{R\text{-}PKE}$, respectively. At a high lever, the injective/lossy mode in L-PKE corresponds to the lossy/collision mode in LCH.

– $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$. $pk \leftarrow \mathsf{L\text{-}PKE.LGen}(1^\lambda)$. Let td be the trapdoor used for efficient openability (Definition 23). Return $hk := pk$ and td.
– $hk \leftarrow \mathsf{LGen}(1^\lambda)$. $(pk, sk) \leftarrow \mathsf{L\text{-}PKE.Gen}(1^\lambda)$. Return $hk := pk$.
– $h \leftarrow \mathsf{Hash}(hk, m, r)$. Let $hk = pk$. $ct \leftarrow \mathsf{Enc}(pk, m; r)$. Return $h := ct$.
– $r' \leftarrow \mathsf{TdColl}(td, m, r, m')$. Let $ct \leftarrow \mathsf{Enc}(pk, m; r)$. Return $r' \leftarrow \mathsf{Open}(td, m, r, m')$.

Theorem 7. *If* $\mathsf{L\text{-}PKE}$ *is a secure L-PKE scheme with efficient openability and κ-min-entropy, then* LCH *constructed above is a secure lossy chameleon hash scheme.*

Proof. We prove the properties of LCH as follows.

Correctness. Recall that the collision hash key is actually a lossy public key of $\mathsf{L\text{-}PKE}$. Due to the efficient openability of $\mathsf{L\text{-}PKE}$, the correctness of LCH holds with overwhelming probability.
κ-uniformity. This is directly implied by the κ-min-entropy of $\mathsf{L\text{-}PKE}$.
Random trapdoor collision. This follows from the efficient openability of $\mathsf{L\text{-}PKE}$.
Indistinguishability. This follows directly from the indistinguishability of $\mathsf{L\text{-}PKE}$.
Collision resistance. To prove the collision resistance of LCH, we first change the generation of hk from $(hk, td) \leftarrow \mathsf{Gen}(1^\lambda)$ to $hk \leftarrow \mathsf{LGen}(1^\lambda)$, due to the indistinguishability property.
If the adversary $\mathcal{A}$ find a collision (m, r, m', r') under this lossy key hk (i.e., under the injective public key pk), then $m \neq m'$ and

$$\mathsf{Enc}(pk, m; r) = \mathsf{Enc}(pk, m'; r').$$

Obviously this cannot happen due to the correctness of $\mathsf{L\text{-}PKE}$.

$1/|\mathcal{M}|$-lossiness. In the lossy mode of LCH (i.e., the injective mode of $\mathsf{L\text{-}PKE}$), for any fixed h, the message $\tilde{m}$ encrypted in h is uniquely determined due to the correctness of $\mathsf{L\text{-}PKE}$. Therefore, for a random m, as long as $m \neq \tilde{m}$ (which happens with probability $(1 - 1/|\mathcal{M}|)$), there does not exist randomness r such that $\mathsf{Enc}(pk, m; r) = h$.

B.5 Construction from LWE

In this subsection we show the LWE-based construction of LWE, which was presented as a dual-mode commitment scheme by Pan and Wagner [67]. Here we provide the proof of collision resistance based on the SIS assumption.

Let n, m, q be positive integers. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times m} (m > n)$ and vector $\mathbf{u} \in \mathbb{Z}_q^n$, define the n-dimensional lattice $\Lambda(\mathbf{A}) := \{\mathbf{y} \in \mathbb{R}^n | \mathbf{y} = \mathbf{A}\mathbf{x}, \mathbf{x} \in \mathbb{Z}^m\}$, the orthogonal lattice $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m | \mathbf{A}\mathbf{x} = 0^n \mod q\}$ and shifted lattice $\Lambda_{\mathbf{u}}^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m | \mathbf{A}\mathbf{x} = \mathbf{u} \mod q\}$.

The Gaussian function with parameter s and center $\mathbf{c}$ is defined as $\rho_{s,\mathbf{c}} : \mathbb{R}^n \to \mathbb{R}$, $\rho_{s,\mathbf{c}}(\mathbf{x}) := \exp(-\pi||\mathbf{x} - \mathbf{c}||^2/s^2)$. For countable set $\mathcal{S} \in \mathbb{R}^n$, the discrete Gaussian distribution $D_{\mathcal{S},s,\mathbf{c}}(\mathbf{x})$ parameterized with s and $\mathbf{c}$ is defined as $D_{\mathcal{S},s,\mathbf{c}}(\mathbf{x}) := \rho_{s,\mathbf{c}}(\mathbf{x}) / \sum_{\mathbf{x}\in\mathcal{S}} \rho_{s,\mathbf{c}}(\mathbf{x})$ for $\mathbf{x} \in \mathcal{S}$ and $D_{\mathcal{S},s,\mathbf{c}}(\mathbf{x}) := 0$ for $\mathbf{x} \notin \mathcal{S}$. Usually we omit $\mathbf{c}$ if $\mathbf{c} = \mathbf{0}$.

We recall the following lemmas from [67], which are originally presented in [2, 62, 35, 36].

Lemma 2. *Let n, m, q be positive integers and $m \geq 2n \log q$. Consider any $\omega(\sqrt{\log m})$ function and $s \geq \omega(\sqrt{\log m})$. Then for all but negligible (in n) fraction of all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ the distribution $\{\mathbf{A}\mathbf{e} | \mathbf{e} \leftarrow D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{A}),s}\}$ is statisticalluy close to uniform distribution over $\mathbb{Z}_q^n$. Moreover, the conditional distribution of $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,s}$ given $\mathbf{u} = \mathbf{A}\mathbf{e} \mod q$ is exactly $D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{A}),s}$.*

Lemma 3. *Let n, m, q be positive integers and $m \geq 2n \log q$. Consider any $\omega(\sqrt{\log m})$ function and $s \geq \omega(\sqrt{\log m})$. Then for all but at most q^{-n} fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and any vector $\mathbf{u} \in \mathbb{Z}_q^n$, it holds that $\Pr[||\mathbf{x}|| > s\sqrt{m} | \mathbf{x} \leftarrow D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{A}),s}] \leq 2^{-m+1}$.*

Let $\mathbf{G}$ be the gadget matrix defined in [61]. Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, n, m, q be positive integers and $m \geq n\lceil \log q \rceil$. A matrix $\mathbf{R} \in \mathbb{Z}^{(m-n\lceil \log q\rceil)\times n\lceil \log q\rceil}$ is a trapdoor for $\mathbf{A}$ is $\mathbf{A}[-\mathbf{R}^\top | \mathbf{I}_{n\lceil \log q\rceil}]^\top = \mathbf{G}$.

Lemma 4. *There exist PPT algorithms* $\mathsf{GenTrap}$ *and* $\mathsf{SampleD}$ *and constants $C_0 > 0$, $C_1 \leq 3$ such that for positive integers n, m, q, $q \geq 2$, $m \geq 3n \log q$, $w := n\lceil \log q \rceil$, and any $\omega(\sqrt{\log n})$ function the following holds with overwhelming probability over all random choices:*

1. *For any $s \geq \omega(\log n)$, the algorithm* $\mathsf{GenTrap}(n, m, s, q)$ *outputs matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} \in \mathbb{Z}^{(m-w)\times w}$ such that $\mathbf{A}$ is statistically close to uniform matrix over $\mathbb{Z}_q^{n\times m}$, $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with entriex sampled from $D_{\mathbb{Z},s}$ and $s_1(\mathbf{R}) \leq s \cdot C_0 \cdot (\sqrt{m-w} + \sqrt{w})$.*
2. *For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ with trapdoor $\mathbf{R}$, for any $\mathbf{u} \in \mathbb{Z}_q^n$ and $s \geq C_1 \cdot \sqrt{s_1(\mathbf{R})^2 + 1}\omega(\sqrt{\log n})$, the distribution $\{\mathbf{z} | \mathbf{z} \leftarrow \mathsf{SampleD}(\mathbf{A}, \mathbf{R}, \mathbf{u}, s)\}$ is statistically close to $D_{\lambda_{\mathbf{u}}^\perp(\mathbf{A}),s}$.*

Next we present two hardness assumptions on lattices.

Definition 24 (The LWE assumption). *Let $n = n(\lambda), m = poly(n), q$ be positive integers and $\mathbf{x}$ be an error distribution over $\mathbb{Z}$. The learning with errors (LWE) assumption states that for any PPT adversary $\mathcal{A}$, its advantage*

$$\begin{aligned}\mathsf{Adv}^{lwe}_{[n,q,\chi],\mathcal{A}}(\lambda) := \Big|&\Pr[\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}; \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n; \mathbf{e} \leftarrow \chi^m : \mathcal{A}(\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top) = 1] \\ &- \Pr[\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}; \mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m : \mathcal{A}(\mathbf{A}, \mathbf{b}^\top) = 1]\Big|\end{aligned}$$

is negligible in λ.

Definition 25 (The SIS assumption). *Let $n = n(\lambda), m, q$ be positive integers and β be a positive real. The short integer solution (SIS) assumption states that for any PPT adversary $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{sis}_{[n,q,\beta,m],\mathcal{A}}(\lambda) := \Pr[\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}; \mathbf{x} \leftarrow \mathcal{A}(\mathbf{A}) : \mathbf{A}\mathbf{x} = 0^n \wedge \mathbf{x} \neq 0^m \wedge ||\mathbf{x}|| \leq \beta]$$

is negligible in λ.

```
$\underline{\mathsf{Gen}(1^\lambda)}$
$(\mathbf{A}, \mathbf{T_A}) \leftarrow TrapdoorGen()$
Return $(hk, td) := (\mathbf{A} \in \mathbb{Z}_q^{(n+\ell)\times m)}, \mathbf{T_A})$

$\underline{\mathsf{LGen}(1^\lambda)}$
$\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$
$\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n\times \ell}$; $\mathbf{E} \leftarrow D^{m\times \ell}_{\mathbb{Z},\alpha q}$
$\underline{\mathbf{A}} := \mathbf{S}^\top \mathbf{A} + \mathbf{E}^\top$
Return $hk := \mathbf{A} := \begin{bmatrix} \bar{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \in \mathbb{Z}_q^{(n+\ell)\times m}$
```

```
$\underline{\mathsf{Hash}(hk, \mathbf{m} \in \{0,1\}^\ell, \mathbf{r})}$
$\mathbf{z} := \mathbf{A}\mathbf{r} + \begin{bmatrix} \mathbf{0} \\ \lfloor q/2 \rceil \cdot \mathbf{m} \end{bmatrix}$
Return $\mathbf{z}$
$\underline{\mathsf{TdColl}(td, \mathbf{m}, \mathbf{r}, \mathbf{m}')}$
$\mathbf{z}' := \mathbf{A}\mathbf{r} + \begin{bmatrix} \mathbf{0} \\ \lfloor q/2 \rceil \cdot (\mathbf{m} - \mathbf{m}') \end{bmatrix}$
$\mathbf{r}' \leftarrow Resample(\mathbf{A}, \mathbf{T_A}, \mathbf{z}', s)$
If $||\mathbf{r}'|| > s\sqrt{m}$: return $\perp$
Return $\mathbf{r}'$
```

Fig. 6. LCH construction from the LWE and SIS assumptions.

Theorem 8 (Security of the Lattice-based Construction [67]). *Under the LWE assumption, the construction in Fig. 6 has completeness, simulatability, min-entropy, indistinguishability, lossiness, and commitment recoverability.*

Theorem 9. *Under the LWE and SIS assumptions,* LCH *in Fig. 6 is a secure lossy chameleon hash scheme.*

Proof. Thanks to Theorem 5 and Theorem 8, it is sufficient to prove the collision resistance.

Recall that in the generation of hash key, $\mathbf{A}$ is statistically close to a random matrix over $\mathbb{Z}_q^{(n+\ell)\times m}$. Under a random matrix $\mathbf{A} = \begin{bmatrix} \bar{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix}$, a collision $(\mathbf{m}, \mathbf{r}, \mathbf{m}', \mathbf{r}')$ such that

$$\mathbf{A}\mathbf{r} + \begin{bmatrix} \mathbf{0} \\ \lfloor q/2 \rceil \cdot \mathbf{m} \end{bmatrix} = \mathbf{A}\mathbf{r}' + \begin{bmatrix} \mathbf{0} \\ \lfloor q/2 \rceil \cdot \mathbf{m}' \end{bmatrix}$$

directly implies a SIS solution $(\mathbf{r} - \mathbf{r}')$ to $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times m}$.

B.6 Construction from DDH

In this subsection we present a construction of LCH from the DDH assumption, which is the well-known DDH-based lossy identification scheme by Chaum et al. [21].

We first recall the background on the discrete logarithm assumption and the DDH assumption.

Let GGen be a group generation algorithm that outputs $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a cyclic group of prime order q with generator g.

Definition 26 (The DL assumption). *The discrete logarithm (DL) assumption states that, for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{dl}_{\mathbb{G},\mathcal{A}}(\lambda) := \Pr[x \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(\mathbb{G}, q, g, g^x) = x]$$

is negligible over λ.

Definition 27 (The DDH assumption). *The decisional Diffie-Hellman (DDH) assumption states that, for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{ddh}_{\mathbb{G},\mathcal{A}}(\lambda) := \left|\Pr[x, y \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1]\right|$$

is negligible over λ.

Note that the DL assumption is implied by the DDH assumption.

Construction. Let $(\mathbb{G}, q, g) \leftarrow \mathbb{G}(1^\lambda)$, and $\tilde{g}$ is another random generator of $\mathbb{G}$. The DDH-based construction of LCH [21] is shown in Fig. .

```
Gen(1^λ)
    $\alpha \leftarrow \mathbb{Z}_q$
    Return $td := x$, $hk := (X, \tilde{X}) := (g^x, \tilde{g}^x)$

LGen(1^λ)
    $x, x' \leftarrow \mathbb{Z}_q$ s.t. $x \neq x'$
    Return $hk := (X, \tilde{X}) := (g^x, \tilde{g}^{x'})$

Hash(hk, m, r)
    Return $h := (X^m g^r || \tilde{X}^m \tilde{g}^r)$

TdColl(td, m, r, m')
    Return $r' := x(m - m') + r$
```

Fig. 7. LCH construction from the LWE and SIS assumptions.

Theorem 10 ([21, 29]). *Under the DDH assumption,* LCH *above is a strongly secure lossy chameleon hash scheme.*

C More Security Notions for ♯AMS

Definition 28 (Weak Unforgeability). *Let* $\sharp\mathsf{AMS}$ *be an ♯AMS scheme. Consider the weak unforgeability experiment* $\mathsf{Exp}^{w\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$*, which is defined as* $\mathsf{Exp}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *in Definition 4, except that condition* (2) *is replaced with*

(2") $\mathcal{A}$ never asks $\mathcal{O}(\mathsf{msg}^*, P, G)$ such that $|G| \geq t^*$.

Define by $\mathsf{Adv}^{w\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *the probability that* $\mathsf{Exp}^{w\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs 1. We say that* $\sharp\mathsf{AMS}$ *is weakly unforgeable, if for all PPT adversary $\mathcal{A}$, the advantage* $\mathsf{Adv}^{w\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *is negligible in λ.*

Remark 4 (On the meaning of weak unforgeability). Recall that the output t of the verification algorithm is a metric to measure how "reliable" a signature is, i.e., how many different signers agree on the message and participate in the signing process. Therefore, in many applications (e.g., the blockchain governance discussed in Section 6), the adversary's goal is to forge a t-valid signature with t as high as possible. To formalize this kind of security, we define the above-mentioned weak unforgeability by strengthening the restriction of the adversary.

Adaptive corruption endows a powerful attack capability for the adversary, which might be too strong to realize. The static corruption model is enough for many applications such as applications: V2 and V3. We formally define the (strong/weak) unforgeability under static corruptions.

Definition 29 ((Strong/Weak) Unforgeability under Static Corruptions). *Let* $\sharp\mathsf{AMS}$ *be an ♯AMS scheme. Consider the following unforgeability experiment* $\mathsf{Exp}^{unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.*

1. *$\mathcal{A}$ sets the maximum number of signers n and claims a group of signers $G' \subseteq [n]$ to be corrupted.*

2. $\mathcal{C}$ *generates* $(vk, sk_1, ..., sk_n) \leftarrow \mathsf{Gen}(1^\lambda, n)$ *and passes* $(vk, \{sk_i\}_{i\in G'})$ *to* $\mathcal{A}$.
3. $\mathcal{A}$ *has access to the signing oracle* $\mathcal{O}(\cdot,\cdot,\cdot)$*, which inputs* (msg, P, G) *and returns* $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, G, \{sk_i\}_{i\in G})$ *and adds* (msg, σ) *into the set* $\mathcal{S}$.
4. *Finally* $\mathcal{A}$ *outputs* $(\mathsf{msg}^*, \sigma^*)$.

Let $t^* \leftarrow \mathsf{Ver}(vk, \mathsf{msg}^*, \sigma^*)$. $\mathsf{Exp}^{unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs* 1 *if*

(1) $t^* > t'$*, where* t' *is the total number of queries to* $\mathcal{O}_{corr}(\cdot)$*; and*
(2) $\mathcal{A}$ *never asks* $\mathcal{O}(\mathsf{msg}^*, P, G)$ *such that* $|G| = t^*$.

Define by $\mathsf{Adv}^{unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *the probability that* $\mathsf{Exp}^{unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *outputs 1. We say that* $\sharp\mathsf{AMS}$ *is unforgeable under static corruptions, if for all PPT adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *is negligible in* λ.

If condition (2) *is replaced with*

(2') $(\mathsf{msg}^*, \sigma^*) \notin \mathcal{S}$.

then we define the strong unforgeability of $\sharp\mathsf{AMS}$*, the corresponding experiment and advantage are denoted by* $\mathsf{Adv}^{s\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *and* $\mathsf{Adv}^{s\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$.

If condition (2) *is replaced with*

(2") $\mathcal{A}$ *never asks* $\mathcal{O}(\mathsf{msg}^*, P, G)$ *such that* $|G| \geq t^*$.

then we define the weak unforgeability of $\sharp\mathsf{AMS}$*, the corresponding experiment and advantage are denoted by* $\mathsf{Adv}^{w\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$ *and* $\mathsf{Adv}^{w\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda)$.

D Discussion on the Relationship Between TRS and $\sharp$AMS

D.1 Issues of Threshold Ring Signatures

We first recall the definition of threshold ring signatures (TRS) from [16] and [57].

Definition 30 (TRS). *A threshold ring signature (TRS) scheme consists of three algorithms:* $\mathsf{TRS} = (\mathsf{Gen}, \mathsf{TSign}, \mathsf{Ver})$.

- $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda, t)$*: The key generation algorithm takes as input the security parameter* 1^λ *and a threshold* t*, and outputs a public key* pk *and a secret key* sk.
- $\sigma \leftarrow \mathsf{TSign}(\mathsf{msg}, R, \{pk_i\}_{i\in R}, T, \{sk_i\}_{i\in T})$*: The threshold signing algorithm/protocol takes as input a message* msg*, a group of users* R *and their public keys* $\{pk_i\}_{i\in R}$*, a group of real signers* T *and their the secret key* $\{sk_i\}_{i\in T}$*, and outputs a signature* σ.
- $1/0 \leftarrow \mathsf{Ver}(t, R, \{pk_i\}_{i\in R}, \mathsf{msg}, \sigma)$*: The verification algorithm* Ver *takes as input* t*, a group of users* R *and their public keys* $\{pk_i\}_{i\in R}$*, a message* msg *and a signature* σ*, and outputs a bit indicating the validity of* σ.

Correctness. *For any* $T \subseteq R$ *such that* $|T| \geq t$*, any* msg *and* $\sigma \leftarrow \mathsf{TSign}(\mathsf{msg}, R, \{pk_i\}_{i\in R}, T, \{sk_i\}_{i\in T})$*, it holds that* $\mathsf{Ver}(t, R, \{pk_i\}_{i\in R}, \mathsf{msg}, \sigma) = 1$.

In many previous works [16, 57, 19, 24, 26], the threshold signing process TSign is just regarded as an algorithm, but rather than an interactive protocol among at least t signers. Under such definitions, the behavior of TRS becomes ambiguous: Does the signer know the threshold t? Does the anonymity hold among the signer group T, i.e., whether a signer in T knows its cooperators' identity?[13]

To address this issue, when defining $\sharp$AMS we introduce a moderator in the signing process. We stress that the moderator perceives the quorum of the signers (hence also t) during the signing process, and every signer only needs to communicate with the moderator. Moreover, even if the moderator gets corrupted later and is willing to reveal the quorum of real signers, it either cannot convince others about the disclosure in our construction C1, or will leak its identity as well and consequently be penalized from the system in our construction C2.

[13] There are some works [64] that consider the anonymity even against the inner signers, as the *strong* anonymity defined in Definition 6.

D.2 Relationship between TRS and $\sharp$AMS

Our newly defined $\sharp$AMS closely relates to TRS, though $\sharp$AMS is a stronger primitive than TRS.

1. By the definition of TRS, a group of t signers can get together to sign a signature, and the threshold t might be fixed (e.g., [56, 79]) or not (e.g., [57, 19, 14]) when generating the public/secret key pairs. But in $\sharp$AMS, any number of signers can cooperate and finally output a signature. That is, the creditability t is flexible in every signing.
2. The output of the verification algorithm in TRS is a single bit indicating whether a signature w.r.t. a message and a threshold is valid or not. While in $\sharp$AMS, the output t is an integer indicating the *real* number of signers who have participated in the signing process, which offers more information than that in TRS.

Owing to the security concerns in ad-hoc networks, TRS has attracted increasingly widespread attention for the past decades. In fact, for most existing TRS schemes [16, 57, 19, 60, 18, 14, 77, 41], the threshold t is changeable every signing, though how to share information of t is ambiguous from the definition. We call this kind of scheme *TRS with flexible threshold*.

Remark 5 (Why Sign the Threshold t Together with msg in the Generic Compiler?). A secure signature should include all public information, e.g., a receiver's address, side-channel information, and the threshold t in our case, into the message to be signed. We show an unsafe counterexample of $\sharp$AMS from linkable ring signatures (LRS) [58] here. As introduced in the introduction, LRS allows a user to sign a message on behalf of a ring, and its two signatures on the same message will be linked. A natural idea for constructing $\sharp$AMS from LRS is to let every signer in the group contribute its own linkable ring signature, and the verification algorithm just returns the count of unlinked and valid signatures. However, if the creditability t is not included in the message to be signed, then unforgeability does not hold anymore. To see this, suppose the forger now corrupts a signer U_i and gets its secret key. Then after seeing a t-valid $\sharp$AMS signature σ generated by a group excluding U_i, the forger can easily output a $(t+1)$-valid $\sharp$AMS signature $(\sigma||\sigma_i)$, where σ_i is a linkable ring signature by U_i. Another example of such insecurity is shown in [78].

Remark 6 (Why Sign the Threshold t Together with msg in the Generic Compiler?). A secure signature should include all public information, e.g., a receiver's address, side-channel information, and the threshold t in our case, into the message to be signed. We show an unsafe counterexample of $\sharp$AMS from linkable ring signatures (LRS) [58] here. As introduced in the introduction, LRS allows a user to sign a message on behalf of a ring, and its two signatures on the same message will be linked. A natural idea for constructing $\sharp$AMS from LRS is to let every signer in the group contribute its own linkable ring signature, and the verification algorithm just returns the count of unlinked and valid signatures. However, if the creditability t is not included in the message to be signed, then unforgeability does not hold anymore. To see this, suppose the forger now corrupted a signer U_i and gets its secret key. Then after seeing a t-valid $\sharp$AMS signature σ generated by a group excluding U_i, the forger can easily output a $(t+1)$-valid $\sharp$AMS signature $(\sigma||\sigma_i)$, where σ_i is a linkable ring signature by U_i. Another example of insecurity is shown in [78].

E Proof of Theorem 1

In this section we prove Theorem 1 (the strong unforgeability of $\sharp\mathsf{AMS}$).

Proof. For simplicity, we will model the signing process of $\sharp$AMS as a algorithm and ignore the moderator in the following proof.

First, we prove the strong unforgeability under static corruptions via hybrid games $\mathsf{G}_0-\mathsf{G}_4$. Before describing the hybrid games, we specify some notations used in the proof. Let $(\mathsf{msg}^*, \sigma^* = (t', m_1^*, ..., m_n^*, r_1^*, ..., r_n^*))$ be $\mathcal{A}$'s final forgery and $t^* \leftarrow \mathsf{Ver}(vk, \mathsf{msg}^*, \sigma^*)$. For $\mathcal{A}$ to win, it must hold that $t' = t^*$. Therefore, in the following proof we implicitly assume that $t' = t^*$. Meanwhile, for $i \in [n]$, let $h_i^* \leftarrow \mathsf{Hash}(hk_i, m_i^*, r_i^*)$, and let $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*) = u^*$.

Game G_0. This is just the original strong unforgeability experiment.

$$\Pr[\mathsf{G}_0 \Rightarrow 1] = \mathsf{Adv}^{s\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda).$$

Game G_1. If $\mathcal{A}$ never asks $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ before outputting the final forgery, then G_1 outputs 0 directly.

Since H works as a random oracle, if $\mathcal{A}$ never asks $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ before, then u^* is randomly distributed over $\mathcal{U}$. Recall that for every (n, t) and $(m_1, ..., m_n)$, there exists only one u such that $\mathbf{F}(n, t, m_1, ..., m_n, u) = 1$. Therefore, the probability that $\mathcal{A}$ wins is at most $1/|\mathcal{U}|$, and we have

$$|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]| \le 1/|\mathcal{U}| \le 1/|\mathcal{M}|.$$

Game G_2. In this game $\mathcal{C}$ changes the simulation of the signing oracle as follows. Upon receiving a signing query $\mathcal{O}(\mathsf{msg}, G)$ ($|G| = t$), $\mathcal{C}$ randomly samples m_i and r_i for all $i \in [n]$, computes $h_i \leftarrow \mathsf{Hash}(hk_i, m_i, r_i)$, and reprograms the random oracle such that $H(vk, h_1, ..., h_n, \mathsf{msg}||t) = u$, where u is computed according to the forward sample algorithm $s_{fwd}(n, t, G)$ of $\mathbf{F}_\theta$. If $\mathcal{C}$ fails to reprogram, i.e., $H(vk, h_1, ..., h_n, \mathsf{msg}||t)$ has already been defined before, then G_2 outputs $\perp$ and aborts. At last $\mathcal{C}$ returns the signature $\sigma = (t, \{m_i\}_{i\in[n]}, \{r_i\}_{i\in[n]})$ to $\mathcal{A}$.

First, we argue that G_2 and G_1 are statistically close if it does not abort. Notice that the outputs $(m_1, ..., m_n, u)$ of $s_{fwd}(n, t, G)$ and $s_{back}(n, t, G)$ satisfy the identical distribution, where u outputted by $s_{back}(n, t, G)$ enjoys a random distribution. Meanwhile, the distribution of r_i ($i \in G$) is γ-close to the original signing algorithm in G_1, due to the γ-random trapdoor collision property of LCH.

Then, we bound the abort probability in G_2. Recall that LCH has κ-uniformity, i.e., for all $i \in [n]$, $\mathbf{H}_\infty(\mathsf{Hash}(hk_i, m_i, r_i)) \ge \kappa$ for randomly sampled m_i and r_i. Suppose $\mathcal{A}$ asks at most Q_{sign} signing queries and Q_H hash queries. By the union bound, we have

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \le (Q_{sign} + Q_H)/2^{n\kappa} + Q_{sign} n \cdot \gamma.$$

Game G_3. We add an extra abort rule in this game. Define by reuse the event that, $\mathcal{A}$ has ever asked $\mathcal{O}(\mathsf{msg}^*, G)$ with some $|G| = t^*$ and gets $\sigma = (t^*, m_1, ..., m_n,\ r_1, ..., r_n)$ back, and

1. $\mathsf{Hash}(hk_i, m_i, r_i) = \mathsf{Hash}(hk_i, m_i^*, r_i^*)$ for all $i \in [n]$; and
2. There exists i such that $(m_i^*, r_i^*) \ne (m_i, r_i)$, and $\mathcal{A}$ never asks $\mathcal{O}_{corr}(i)$.

If reuse happens, then G_3 outputs $\perp$ and aborts.

We can easily construct a reduction algorithm $\mathcal{B}_1$ that breaks the strong collision resistance of the underlying uncorrupted hk_i if reuse happens. Via a standard hybrid argument, we know that

$$|\Pr[\mathsf{G}_2 \Rightarrow 1] - \Pr[\mathsf{G}_3 \Rightarrow 1]| \le \Pr[\mathsf{reuse}] \le n \cdot \mathsf{Adv}^{s\text{-}cr}_{\mathsf{LCH},\mathcal{B}_1}(\lambda).$$

Game G_4. Let $G' \subseteq [n]$ be the corruption group by $\mathcal{A}$ and $\widetilde{G} := [n] \setminus G'$. In this game, $\mathcal{C}$ generates $hk_i \leftarrow \mathsf{LGen}(1^\lambda)$ instead of $(hk_i, td_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for all $i \in \widetilde{G}$. The simulations of $H(\cdot)$ and $\mathcal{O}(\mathsf{msg}, G)$ are the same as G_3. Note that $\mathcal{C}$ can simulate a $\sharp$AMS signature for group G that contains $i \in \widetilde{G}$ without knowing a trapdoor td_i.

We can easily construct a reduction algorithm $\mathcal{B}_2$ to reduce the indistinguishability between G_3 and G_4 into the indistinguishability of LCH. Therefore,

$$|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \le n \cdot \mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{B}_2}(\lambda).$$

Now we argue that in G_4, even an all-powerful adversary $\mathcal{A}$ cannot win with a non-negligible probability due to the ϵ-lossiness of LCH.

Let $(\mathsf{msg}^*, \sigma^* = (t', m_1^*, ..., m_n^*, r_1^*, ..., r_n^*))$ be $\mathcal{A}$'s forgery and $t^* \leftarrow \mathsf{Ver}(vk,\ \mathsf{msg}^*, \sigma^*)$. Let $|G'| = t'$. Obviously we have $t' < t^* \le n$. Define by W_4 the event that G_4 outputs 1. We divide W_4 into the following three sub-events.

1. Event W_4^1: $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ was defined when $\mathcal{A}$ asking a query $\mathcal{O}(\mathsf{msg}^*, G)$ with $|G| = t^*$ and the response is $\sigma = (t^*, m_1, ..., m_n, r_1, ..., r_n)$, and $m_i = m_i^*$ for all $i \in [n]$.
2. Event W_4^2: $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ was defined when $\mathcal{A}$ asking a query $\mathcal{O}(\mathsf{msg}^*, G)$ with $|G| = t^*$ and the response is $\sigma = (t^*, m_1, ..., m_n, r_1, ..., r_n)$, and there exists at least one i s.t. $m_i \neq m_i^*$.
3. Event W_4^3: $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ was defined upon some hash query by $\mathcal{A}$.

We analyze W_4^1 first. Recall that we require $\mathcal{A}$'s final forgery to be different from all signatures obtained from the signing oracle. That is, there exists some i such that $m_i = m_i^*$, $r_i \neq r_i^*$, and $\mathsf{Hash}(hk_i, m_i, r_i) = \mathsf{Hash}(hk_i, m_i^*, r_i^*)$. Due to the uniqueness of LCH, it is impossible for $i \in G'$ (the corrupted group). Moreover, W_4^1 cannot happen for $i \in \widetilde{G}$ (the uncorrupted/lossy group) due to the extra abort rule reuse in G_3. Therefore, we have $\Pr[W_4^1] = 0$.

Then we analyze W_4^2. According to the interdependency of constraint function $\mathbf{F}_\theta$, if W_4^2 happens, then there must exist at least one $i \in \widetilde{G}$ such that $m_i \neq m_i^*$. Together with the extra abort rule reuse in G_3, we get that $\Pr[W_4^2] = 0$.

Then we bound $\Pr[W_4^3]$ according to the ϵ-lossiness of LCH. Recall that if G_5 does not abort, then all hash keys $\{hk_i\}_{i \in \widetilde{G} = ([n] - G')}$ are generated in the lossy mode. Under the hash query $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}||t^*)$, a random u^* is returned. According to the randomness property of $\mathbf{F}_\theta$, either there exist at least t different $i \in [n]$ such that m_i distribute uniformly, or for any $i \in [n]$, m_i distributes uniformly. From the analysis above we know, there exists at least one i s.t. hk_i is a lossy hash key. Then according to the ϵ-lossiness, given h_i, for a random m_i, the probability that $\mathcal{A}$ can find a randomness r_i with $\mathsf{Hash}(hk_i, m_i, r_i) = h_i$ is at most ϵ. By the union bound, we have

$$\Pr[\mathsf{G}_4 \Rightarrow 1] \leq (n - t^* + 1)\epsilon \leq n\epsilon,$$

which finishes the proof of strong unforgeability.

It is still left to prove the unconditional strong anonymity. Recall that in the signing, m_i for $i \notin G$ is randomly distributed, and m_i for $i \in G$ is computed from $\{m_i\}_{i \in [n] \setminus G}$ and randomly sampled u according to $\mathbf{F}_\theta$. Since the two sample algorithms s_{fwd} and s_{back} have identical distributions, m_i for $i \in G$ distributes the same as m_i for $i \notin G$. Therefore, to prove the strong anonymity, it is sufficient to prove that for any $m, m' \in \mathcal{M}$, the following two distributions are indistinguishable:

$$\left\{(m, r) \middle| r \xleftarrow{\$} \mathcal{R}\right\} \text{ and } \left\{(m, r) \middle| \begin{array}{c}(hk, td) \leftarrow \mathsf{LCH.Gen}(1^\lambda), \bar{r} \xleftarrow{\$} \mathcal{R}, \\ r \leftarrow \mathsf{LCH.TdColl}(td, \bar{m}, \bar{r}, m)\end{array}\right\}.$$

Thanks to the γ-random trapdoor collision property, the above two distributions are γ-close, and unconditional and strong anonymity holds as a result.

Taking the above all together, we obtain the desired result.

F Unforgeability under Adaptive Corruptions

In this section, we analyze the (strong) unforgeability of LCH-based $\sharp$AMS schemes in the adaptive corruption model.

Theorem 11. *If* LCH *is strongly secure (i.e., it has κ-uniformity, γ-random trapdoor collision, strong collision resistance, indistinguishability, and ϵ-lossiness) and unique, and* $\mathbf{F}_\theta$ *is a constraint function, then the $\sharp$AMS scheme* $\sharp\mathsf{AMS}$ *constructed in Section 5 has strong unforgeability and strong anonymity under adaptive corruptions. More precisely, for any PPT adversary $\mathcal{A}$, we have another PPT algorithm $\mathcal{B}$ such that $Time(\mathcal{B}) \approx Time(A)$, and*

$$\mathsf{Adv}_{\sharp\mathsf{AMS},\mathcal{A}}^{s\text{-}unforg}(\lambda) \leq n2^n \mathsf{Adv}_{\mathsf{LCH}}^{s\text{-}cr}(\lambda) + n2^n \mathsf{Adv}_{\mathsf{LCH},\mathcal{B}}^{ind}(\lambda) + n2^n \cdot \epsilon + \frac{1}{|\mathcal{M}|} + \frac{Q_{sign} + Q_H}{2^{n\kappa}} + Q_{sign} n \cdot \gamma,$$

where Q_{sign} and Q_H are the numbers of signing queries (in the strong unforgeability experiment or the strong anonymity experiment) and hash queries, respectively.

Proof. Similar to that in the proof of Theorem 1, we prove the strong unforgeability via hybrid games $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3, \widetilde{\mathsf{G}}_3, \mathsf{G}_4$. All hybrid games except $\widetilde{\mathsf{G}}_3$ are defined the same, and we safely omit the details here.

Game $\widetilde{\mathsf{G}}_3$. In this game $\mathcal{C}$ randomly samples a subgroup $\widetilde{G} \subseteq [n]$ at the beginning, and whenever $\mathcal{A}$ terminates and outputs its forgery, $\mathcal{C}$ checks whether $\mathcal{A}$ asks $\mathcal{O}_{corr}(i)$ for all $i \in ([n] - \widetilde{G})$ exactly, i.e., whether $\widetilde{G}$ is exactly the subgroup of users that $\mathcal{A}$ does not corrupt. If not, $\widetilde{\mathsf{G}}_4$ outputs $\perp$ and aborts.

There are at most 2^n different subgroups for $[n]$. Via a standard complexity argument, we have that

$$\Pr[\mathsf{G}_3 \Rightarrow 1] \leq 2^n \cdot \Pr[\widetilde{\mathsf{G}}_3 \Rightarrow 1].$$

Game G_4. In this game, $\mathcal{C}$ generates $hk_i \leftarrow \mathsf{LGen}(1^\lambda)$ instead of $(hk_i, td_i) \leftarrow \mathsf{Gen}(1^\lambda)$ for all $i \in \widetilde{G}$. We have

$$|\Pr[\widetilde{\mathsf{G}}_3 \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \leq n \cdot \mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{B}'}(\lambda).$$

Combined with hybrid games $\mathsf{G}_1, ..., \mathsf{G}_4$, we finish the proof.

Following the same proof steps in Theorem 1, we have the following theorem.

Theorem 12. *If* LCH *is secure (i.e., it has* κ*-uniformity,* γ*-random trapdoor collision, collision resistance, indistinguishability, and* ϵ*-lossiness) and* $\mathbf{F}_\theta$ *is a constraint function, then* $\sharp\mathsf{AMS}$ *constructed in Section 5 has unforgeability under adaptive corruptions. More precisely, for any PPT adversary* $\mathcal{A}$*, we have another PPT algorithm* $\mathcal{B}$ *such that* $Time(\mathcal{B}) \approx Time(A)$*, and*

$$\mathsf{Adv}^{unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \leq n2^n \mathsf{Adv}^{ind}_{\mathsf{LCH},\mathcal{B}}(\lambda) + n2^n \cdot \epsilon + \frac{1}{|\mathcal{M}|} + \frac{Q_{sign} + Q_H}{2^{n\kappa}} + Q_{sign} n \cdot \gamma,$$

where Q_{sign} *and* Q_H *are the numbers of signing queries and hash queries, respectively.*

G Instantiations of Constraint Functions

Assume $\mathcal{M}$ be a finite field. We show two classical instantiations of constraint functions.

Constraint Function $\mathbf{F}_\mathbf{A}$ by Linear Equation Systems.

Let $\mathcal{U} := \mathcal{M}^t$. Let $\mathbf{A} = (a_{i,j})_{(i,j\in[n])}$ be an invertible matrix such that, for every $t \in [n]$ and subset $G \subseteq [n]$ with $|G| = t$, elements $(a_{i,j})_{(i\in[t],j\in G)}$ form an invertible submatrix. For example, we can set $\mathbf{A}$ as a Vandermonde matrix.

Define $\mathbf{F}_\mathbf{A}(n, t, m_1, ..., m_n, u = (u_1, ..., u_t)) = 1$, if and only if the following linear equation system holds.

$$\begin{cases} a_{1,1}m_1 + a_{1,2}m_2 + ... + a_{1,n}m_n = u_1, \\ a_{2,1}m_1 + a_{2,2}m_2 + ... + a_{2,n}m_n = u_2, \\ ... \\ a_{t,1}m_1 + a_{t,2}m_2 + ... + a_{t,n}m_n = u_t. \end{cases} \tag{3}$$

– The forward sample algorithm $s_{fwd}(n, t, G)$ first randomly samples m_i for all $i \in [n]$, then computes $(u_1, ..., u_t)$ according to (3).
– The backward sample algorithm $s_{bck}(n, t, G)$ first randomly samples m_i for all $i \in [n] \setminus G$, and u_i for all $i \in [t]$, then computes $\{m_i\}_{i\in G}$ according to (3).

Proof. Now we prove that $\mathbf{F}_\mathbf{A}$ defined above is a family of constraint functions.

– Given $n, t, m_1, ..., m_n$ and $u = (u_1, ..., u_t)$, it is easy to evaluate $\mathbf{F}_\mathbf{A}(n, t, m_1, ..., m_n, u)$ by checking the above linear equation system.
– Given $n, t, m_1, ..., m_n$, one can compute the unique u that satisfies the above linear equation system.
– Since $\mathbf{A}$ is a full-rank matrix, the forward sample algorithm and the backward sample algorithm have the identical distribution $(m_1, ..., m_n, u)$.

– Interdependency.
  - Assume $\mathbf{F}_\mathbf{A}(n,t,m_1,...,m_n,u) = \mathbf{F}_\mathbf{A}(n,t,m'_1,...,m'_n,u) = 1$. Define $\Delta m = (m_1 - m'_1, ..., m_n - m'_n)^\top \in \mathcal{M}^n$, and we have $\mathbf{A}\Delta m = \mathbf{0}$. Suppose Δm has less than t non-zero elements. Then we can separate a vector $\Delta\bar{m} \in \mathcal{M}^t$ and a full-rank submatrix $\bar{\mathbf{A}} \in \mathcal{M}^{t\times t}$ such that $\bar{\mathbf{A}}\Delta\bar{m} = \mathbf{A}\Delta m = \mathbf{0}$. Thus, $\Delta\bar{m} = \bar{\mathbf{A}}^{-1}\mathbf{0} = \mathbf{0}$, which means that $(m_1, ..., m_n) = (m'_1, ..., m'_n)$.
  - Define $\Delta u := u - u'$ and we have $\mathbf{A}\Delta m = \Delta u$. Divide $\mathbf{A}$ into two submatrices $\bar{\mathbf{A}} \in \mathcal{M}^{t\times t}$ of full rank and $\tilde{\mathbf{A}} \in \mathcal{M}^{t\times(n-t)}$. Similarly, divide Δm into two vectors $\Delta\bar{m} \in \mathcal{M}^t$ and $\Delta\tilde{m} \in \mathcal{M}^{n-t}$. Thus we have $\bar{\mathbf{A}}\Delta\bar{m} + \tilde{\mathbf{A}}\Delta\tilde{m} = \Delta u$. Since $\bar{\mathbf{A}}$ has full rank, $\Delta\bar{m} = \bar{\mathbf{A}}^{-1}(\Delta u - \tilde{\mathbf{A}}\Delta\tilde{m})$ is randomly distributed over $\mathcal{M}^t$ if Δu is randomly sampled. Therefore, with overwhelming probability, $\Delta\bar{m}$ contains no zero-elements, which means that there are at least t different $i \in [n]$ such that $m_i \neq m'_i$.

– Randomness. Following the same argument above, we rewrite $\mathbf{F}_\mathbf{A}(n,t,m_1, ..., m_n) = 1$ as $\bar{\mathbf{A}}\bar{m} + \tilde{\mathbf{A}}\tilde{m} = u$, where $\bar{m} \in \mathcal{M}^t$, $\tilde{m} \in \mathcal{M}^{n-t}$, and $u \in \mathcal{M}^t$. Then $\bar{m} = \bar{\mathbf{A}}^{-1}(u - \tilde{\mathbf{A}}\tilde{m})$ is randomly distributed over $\mathcal{M}^t$ if u is randomly sampled.

Constraint Function $\mathbf{F}_p$ by Polynomial Interpolation.
Let $\mathcal{H} := \mathcal{M}$. Let $(P_0, ..., P_n) := ((0,u),(1,m_1),...,(n,m_n))$ be $n+1$ points of a polynomial.
Define $\mathbf{F}_p(n,t,m_1,...,m_n,u) = 1$, if and only if $P_0, ..., P_n$ form a polynomial of degree at least $(n-t)$ by polynomial interpolation.

– The forward sample algorithm $s_{fwd}(n,t,G)$ first randomly samples a polynomial of degree $n-t$, then it computes $u := g(0)$ and $m_i := g(i)$ for $i \in [n]$.
– The backward sample algorithm $s_{bck}(n,t,G)$ first randomly samples m_i for $i \in [n] \setminus G$ and u, then forms a polynomial g from the $(n-t+1)$ points $((0,u), \{(i,m_i)\}_{i\in[n]\setminus G})$ by polynomial interpolation. For $i \in G$, it computes $m_i := g(i)$.

Proof. Now we prove that $\mathbf{F}_p$ defined above is a family of constraint functions.

– Given $n,t,m_1,...,m_n$ and u, we form an $(n-t)$-degree polynomial from $P_0, ..., P_{n-t}$, and $\mathbf{F}_p(n,t,m_1,...,m_t,u) = 1$ if and only if $P_{n-t+1}, ..., P_n$ are points on the polynomial.
– Given $n,t,m_1,...,m_n$, if there exists u satisfying $\mathbf{F}_p(n,t,m_1,...,m_n,u) = 1$, then from $(n-t+1)$ points $P_1, ..., P_{n-t+1}$, we can construct the polynomial via interpolation and then obtain the constant coefficient u.
– The $(n-t)$-degree polynomial from the backward sample algorithm is randomly distributed as that in the forward sample algorithm.
– Interdependency.
  - Suppose $\mathbf{F}_p(n,t,m_1,...,m_n,u) = \mathbf{F}_p(n,t,m'_1,...,m'_n,u) = 1$. If there are at least t different i such that $m_i \neq m'_i$ (i.e., there are more than $(n-t)$ positions i such that $m_i = m'_i$), then more than $(n-t)$ points of the two polynomials g and g' are the same. Along with $P_0 = (0,u)$, we know $g = g'$, and consequently $m_i = m'_i$ for all $i \in [n]$.
  - Let g and g' be two $(n-t)$-degree polynomials w.r.t. $(m_1,...,m_n,u)$ and $(m'_1,...,m'_n,u')$. If there are more than $(n-t)$ different i such that $m_i = m'_i$, then with these $(n-t+1)$ points one can construct the same polynomial g and g', and consequently $u = u'$. Since u, u' are sampled randomly, this happens with negligible probability.

– Randomness. The equation $\mathbf{F}_p(n,t,m_1,...,m_n,u) = 1$ indicates an $(n-t)$-degree polynomial $g(X) = u + a_1X + a_2X^2 + ... + a_{n-t}X^{n-t}$ for some coefficients $a_1, ..., a_{n-t}$. If u distributes uniformly over $\mathcal{M}$, then for every $i \in [n]$, $m_i = g(i) = u + \sum_{j\in[n-t]} a_j i^{n-t}$ is a uniform distribution over $\mathcal{M}$.

H Security Proof for Standard Chameleon Hash Variants

Theorem 13. *If* CH *is strongly secure (i.e., it has κ-uniformity, random trapdoor collision, and strong collision resistance) and unique, and $\mathbf{F}_\theta$ is a constraint function, then the $\sharp$AMS scheme* $\sharp\mathsf{AMS}$ *constructed* *in Section 5 has strong unforgeability and strong anonymity under static corruptions. More precisely, for any PPT adversary $\mathcal{A}$, there exist PPT algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that $Time(\mathcal{B}_1) \approx Time(\mathcal{B}_2) \approx Time(\mathcal{A})$, and*

$$\mathsf{Adv}^{s\text{-}unforg\text{-}sta\text{-}corr}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda) \leq \sqrt{n(Q_{sign}+Q_H)\mathsf{Adv}^{s\text{-}cr}_{\mathsf{CH},\mathcal{B}_1}(\lambda)+\epsilon_1} + n\mathsf{Adv}^{s\text{-}cr}_{\mathsf{CH},\mathcal{B}_2}(\lambda)+\epsilon_2,$$

where ϵ_1, ϵ_2 are some negligible functions in λ, and Q_{sign} and Q_H are the numbers of signing queries and hash queries, respectively.

Proof. We mainly focus on the proof of strong unforgeability since proof of strong anonymity is the same as that in Section 5.

The theorem is proved via four hybrid games $\mathsf{G}_0 - \mathsf{G}_3$, which are the same as those in the proof of Theorem 1. We will use the forking lemma to bound the probability that $\mathcal{A}$ wins in G_3.

Let $(\mathsf{msg}^*, \sigma^* = (t', m_1^*, ..., m_n^*, r_1^*, ..., r_n^*))$ be $\mathcal{A}$'s final forgery and $t^* \leftarrow \mathsf{Ver}(vk, \mathsf{msg}^*, \sigma^*)$. Obviously we have $t' = t^*$ if $\mathcal{A}$ wins. For $i \in [n]$, let $h_i^* \leftarrow \mathsf{Hash}(hk_i, m_i^*, r_i^*)$, and let $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*) = u^*$.

Game G_0. This is just the original unforgeability experiment. We have

$$\Pr[\mathsf{G}_0 \Rightarrow 1] = \mathsf{Adv}^{s\text{-}unforg}_{\sharp\mathsf{AMS},\mathcal{A}}(\lambda).$$

Game G_1. If $\mathcal{A}$ never asks $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ before outputting the final forgery, then G_1 outputs 0 directly. We have

$$|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]| \leq 1/|\mathcal{M}|.$$

Game G_2. In this game $\mathcal{C}$ changes the way of signing oracle's simulation as follows. Upon receiving a signing query $\mathcal{O}(\mathsf{msg}, G)$ ($|G| = t$), $\mathcal{C}$ randomly samples m_i and r_i for all $i \in [n]$, computes $h_i \leftarrow \mathsf{Hash}(hk_i, m_i, r_i)$, and reprograms the random oracle such that $H(vk, h_1, ..., h_n, \mathsf{msg}||t) = u^{(j)}$, where $u^{(j)}$ is computed according to the constraint function $\mathbf{F}$. If $\mathcal{C}$ fails to reprogram, i.e., $H(vk, h_1, ..., h_n, \mathsf{msg}||t)$ has already been defined before, then G_2 outputs $\perp$ and aborts. At last $\mathcal{C}$ returns the signature $\sigma := (t, \{m_i\}_{i\in[n]}, \{r_i\}_{i\in[n]})$ to $\mathcal{A}$.

We have

$$|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq (Q_{sign}+Q_H)/2^{n\kappa}.$$

Game G_3. We add an extra abort rule in this game. Define by reuse the event that, $\mathcal{A}$ has ever asked $\mathcal{O}(\mathsf{msg}^*, G)$ with some $|G| = t^*$ and gets $\sigma = (t^*, m_1, ..., m_n, r_1, ..., r_n)$ back, and

1. $\mathsf{Hash}(hk_i, m_i, r_i) = \mathsf{Hash}(hk_i, m_i^*, r_i^*)$ for all $i \in [n]$;
2. There exists i such that $(m_i^*, r_i^*) \neq (m_i, r_i)$, and $\mathcal{A}$ never asks $\mathcal{O}_{corr}(i)$.

If reuse happens, then G_3 outputs $\perp$ and aborts.

We have

$$|\Pr[\mathsf{G}_2 \Rightarrow 1] - \Pr[\mathsf{G}_3 \Rightarrow 1]| \leq \Pr[\mathsf{reuse}] \leq n \cdot \mathsf{Adv}^{s\text{-}cr}_{\mathsf{LCH},\mathcal{B}_1}(\lambda).$$

Next, we use the forking lemma to bound the probability that $\mathcal{A}$ wins in G_3. The high level idea is to construct an algorithm $\mathcal{B}$ that simulates G_3 for the forger $\mathcal{A}$, and outputs $(j^*, (\mathsf{msg}^*, \sigma^*))$ as long as $\mathcal{A}$ successfully outputs a forgery $(\mathsf{msg}^*, \sigma^*)$, where j denotes that for the j-th query to the random oracle $\mathcal{A}$ asks $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$. Then, we show the forking algorithm $\mathcal{F}_\mathcal{B}$ associated to $\mathcal{B}$ that finds a collision under some specific chameleon hash key, which completes the proof.

Following the same argument of W_5^1 and W_5^2 in G_5 in the proof of Theorem 1, we know that if G_3 does not abort, then $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ is defined upon some hash query by $\mathcal{A}$. We construct algorithm $\mathcal{B}$ as follows. Let $(hk', td') \leftarrow \mathsf{CH.Gen}(1^\lambda)$ and $Q := Q_{sign} + Q_H$, where Q_{sign} and Q_H are the numbers of signing queries and hash queries, respective. Given the initial input $(hk', u^{(1)}, ..., u^{(Q)})$ and the randomness coin ρ, $\mathcal{B}$ randomly samples $i^* \xleftarrow{\$} [n]$, and let $hk_{i^*} := hk'$. For other $i \in [n] \setminus \{i^*\}$, $\mathcal{B}$ invokes $(hk_i, td_i) \leftarrow \mathsf{CH.Gen}(1^\lambda)$. Then it sets $vk := (hk_1, ..., hk_n)$ and sends vk to $\mathcal{A}$.

$\mathcal{B}$ simulates $\mathcal{O}_{corr}(\cdot)$ and $\mathcal{O}(\cdot,\cdot)$ for $\mathcal{A}$ as follows.

– Simulation of corruption queries $\mathcal{O}_{corr}(i)$. If $i = i^*$ then $\mathcal{B}$ aborts, otherwise it returns td_i.
– Simulation of the j-th query $H(\cdot)$. Return $u^{(j)}$ directly.
– Simulation of the j-th query $\mathcal{O}(\mathsf{msg}, G)$. Let $t := |G|$. $\mathcal{B}$ randomly samples m_i for $i \in [n] \setminus G$, and computes $\{m_i\}_{i \in G}$ from $\{m_i\}_{i \in [n] \setminus G}$ and $u^{(j)}$ according to the forward sample algorithm of the constraint function $\mathbf{F}$. Obviously the distribution of $\{m_i\}_{i \in [n]}$ is the same as that in G_2. Then, it randomly samples r_i for all $i \in [n]$, computes $h_i \leftarrow \mathsf{Hash}(hk_i, m_i, r_i)$, and reprograms the random oracle such that $H(vk, h_1, ..., h_n, \mathsf{msg}||t) = u^{(j)}$. Finally $\mathcal{B}$ returns the signature $\sigma := (t, \{m_i\}_{i \in [n]}, \{r_i\}_{i \in [n]})$.

Finally, $\mathcal{A}$ outputs its forgery $(\mathsf{msg}^*, \sigma^*)$. If $\mathcal{B}$ does not abort in the simulation, and $\mathcal{A}$'s forgery is valid and related to the j-th query, then $\mathcal{B}$ outputs $(j, (\mathsf{msg}^*, \sigma^*))$. In any other case $\mathcal{B}$ outputs $(0, \perp)$.

For $\mathcal{A}$ to win in G_3, at least one signer is not corrupted. Since i^* is randomly chosen from $[n]$ and totally hidden from the adversary, $\mathcal{B}$ does not abort with probability at least $1/n$. Therefore, the probability that $\mathcal{B}$ outputs $(j, (\mathsf{msg}^*, \sigma^*))$ is at least $acc_{\mathcal{B}} \geq \Pr[\mathsf{G}_3 \Rightarrow 1]/n$.

According to the forking lemma, there exists a forking algorithm $\mathcal{F}_{\mathcal{B}}$ associated to $\mathcal{B}$ that takes input hk', and with probability $frk \geq acc_{\mathcal{B}} \cdot (acc_{\mathcal{B}}/Q - 1/|\mathcal{M}|)$ it outputs $(j, (\mathsf{msg}^*, \sigma^*), (\mathsf{msg}^{*\prime}, \sigma^{*\prime}))$ (recall that j indexes the forking point, i.e., $u^{(j)} \neq u^{(j)'}$). Let $\sigma^* = (t^*, m_1^*, ..., m_n^*, r_1^*, ..., r_n^*)$ and $\sigma^{*\prime} = (t^{*\prime}, m_1^{*\prime}, ..., m_n^{*\prime}, r_1^{*\prime}, ..., r_n^{*\prime})$. Upon to the point of the j-th hash query, the environment of $\mathcal{A}$ provided by $\mathcal{B}$ in the first and the second run are identical, since $\mathcal{B}$ uses the same inputs, random tape and values $u^{(1)}, ..., u^{(j-1)}$ to generate $\mathcal{A}$'s inputs and oracle responses. Therefore, the two executions of $\mathcal{A}$ are identical up to this point, and the arguments of both hash queries must be the same, implying that $\mathsf{msg}^*||t^* = \mathsf{msg}^{*\prime}||t^{*\prime}$, and $(h_1^*, ..., h_n^*) = (h_1^{*\prime}, ..., h_n^{*\prime})$, where $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*)$ and $H(vk, h_1^{*\prime}, ..., h_n^{*\prime}, \mathsf{msg}^{*\prime}||t^{*\prime})$ are the j-th queries in the first and the second run, respectively.

For the first running of $\mathcal{F}_{\mathcal{B}}$, we have $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*) = u^{(j)}$. For the second running of $\mathcal{F}_{\mathcal{B}}$, we have $H(vk, h_1^*, ..., h_n^*, \mathsf{msg}^*||t^*) = u^{(j)'}$. According to the interdependency of $\mathbf{F}$, there exist at least t^* pairs $(m_i^*, m_i^{*\prime})$ such that $m_i^* \neq m_i^{*\prime}$, $h_i^* \leftarrow \mathsf{Hash}(hk_i, m_i^*, r_i^*)$, and $h_i^* \leftarrow \mathsf{Hash}(hk_i, m_i^{*\prime}, r_i^{*\prime})$. Since $\mathcal{A}$ can corrupt up to $t^* - 1$ signers, with probability at least $1/(n - t^* + 1)$, we successfully find a collision under the specific chameleon hash key hk'.

Taking all together, we obtain the desired result.

I Proof Sketch of Theorem 3 (Fault-Tolerant ♯AMS)

Proof Sketch. In the proof of weak unforgeability, the hybrid games $\mathsf{G}_0 - \mathsf{G}_4$ are defined similarly, except that G_3 and events W_4^1, W_4^2 in G_4 are skipped since we do not consider strong unforgeability. Let $(\mathsf{msg}^*, \sigma^* = (t^*, F, \{m_i^*, r_i^*\}_{[n] \setminus F}, \{h_i^*\}_{i \in F}))$ be $\mathcal{A}$'s final forgery. Let G' be the corruption group and $|G'| = t'$. For $\mathcal{A}$ to win in G_4, it must hold that $t' < (t^* - |F|)$. Recall that in G_4, all $(n - t')$ non-corrupted hash keys are generated in the lossy mode. Since $t' < (t^* - |F|)$ and $t^* > |F|$, according to the randomness property of $\mathbf{F}_\theta$, we know among G there exists at least one i s.t. hk_i is a lossy hash key. Then according to the ϵ-lossiness, given h_i, for a random m_i, the probability that $\mathcal{A}$ can find a randomness r_i with $\mathsf{Hash}(hk_i, m_i, r_i) = h_i$ is at most ϵ, and weak unforgeability holds as a result.

The proof of unconditional strong anonymity (for honest signers) follows directly from the random trapdoor collision property of LCH, i.e., from the statistical indistinguishability of the following distributions for all $m, m' \in \mathcal{M}$:

$$\left\{(m, r) \middle| r \xleftarrow{\$} \mathcal{R}\right\} \text{ and } \left\{(m, r) \middle| \begin{array}{c} (hk, td) \leftarrow \mathsf{LCH.Gen}(1^\lambda), \bar{r} \xleftarrow{\$} \mathcal{R}, \\ r \leftarrow \mathsf{LCH.TdColl}(td, \bar{m}, \bar{r}, m) \end{array}\right\},$$

where the left distribution corresponds to users $i \in [n] \setminus G$, and the right distribution corresponds to honest users $i \in G \setminus F$.

Remark 7 (Why Does Fault-tolerant ♯AMS Schemes Achieve Weak Unforgeability Only?). Let $F \subset F'$. It is easy to see that if $\sigma = (t, F, \{m_i, r_i\}_{i \in [n] \setminus F}, \{h_i\}_{i \in F})$ is a $(t - |F|)$-valid ♯AMS signature, then

$$\sigma' := (t, F', \{m_i, r_i\}_{i \in [n] \setminus F'}, \{h_i\}_{i \in F'})$$

is a $(t - |F'|)$-valid $\sharp$AMS signature. That says, a malicious leader can always decrease the "credibility" of a valid $\sharp$AMS signature. Therefore, the fault-tolerant $\sharp$AMS scheme above can achieve weak unforgeability only. However, we emphasize that in the application of blockchain governance, a malicious developer/moderator P earns nothing meaningful from this kind of attacks. Furthermore, P will get punished due to a malicious disclosure in the voting systems presented in the next section, since all transcript messages between P and the signer are signed using their (regular) digital signatures.

J More Related Works of $\sharp$AMS

Multisignatures. Multisignatures (MS) [15, 8] allow n different parties generate a single signature on a common message, and the size of the signature is short. However, privacy is not considered in multisignatures, and the identities of signers/authors are totally exposed in the signature.

Threshold Signatures. A (t, n)-threshold signature scheme enables a group of n parties to sign on a message if more than t parties participate in the signing. Besides, the threshold t and the quorum of t participants are hidden from the signature. Most threshold signatures [15, 74, 75, 51] are based on the secret sharing schemes, and they all face a shortcoming that the threshold t needs to be fixed before generating the key pairs.

Ring Signatures. In ring signatures (RS) [71] (a.k.a. spontaneous anonymous group signatures [19]), a user can sign a message on behalf of a group without revealing its identity. Usually, ring signatures have no group manager, and anonymity holds unconditionally. Linkable ring signatures (LRS) [58], which are tailored to the voting systems, prevent a user votes/signs twice via introducing a Link algorithm to detect double-voting. An LRS scheme naturally implies a $\sharp$AMS scheme (like the construction by Munch-Hansen et al. [64]), where the signature is just a union of different LRS shares. However, such a construction can only achieve a "weaker" version of anonymity (cf. Def. 28 in Appendix C), and the signature size is quadratic in the ring size n, if we pursue unconditional anonymity and do not involve an accumulator [12].

K Further Discussion and Other Applications

We discuss more about our e-voting system / our blockchain governance system and other possible applications of $\sharp$AMS.

K.1 Further Discussion

We discuss more about our e-voting system / our blockchain governance system and other possible applications.

Infrastructure and authenticated transcription. We assume a (regular) signature scheme $\mathcal{S}$ that is unforgeable, and each node in the network has published a public key of $\mathcal{S}$. For each message to be sent, the sender also generates a signature using their own secret key to prevent the message from being intercepted during transmission.

On-chain. After generating a $\sharp$AMS signature, the leader of the proposal will upload it on the blockchain so that it cannot be altered further.

Vote-and-go. The (non-leader) voters are not involved in the announcement period, and they can leave the system after completing their own part without any interaction during the signing period. In other words, voters are required to participate in the voting process only once. Our round-optimal voting systems, V2 and V3, achieve this property.

Vote-count Concealment. Voters remain unaware of the current vote count (and therefore of the votes of others) until the results are announced. Our round-optimal voting systems, V2 and V3, achieve this property.

K.2 Other Applications of $\sharp$AMS

Linked whistle-blowing [58]**.** Suppose now a citizen wants to reveal a scandal of the government. To avoid the risk of malicious retaliation, he/she may resort to revealing it in anonymous ways, for example, via using a ring signature. Unfortunately, media or journalists may not believe what the whistleblower tells and think that he/she is telling lies. However, they may choose to believe the disclosure if quite a number of citizens confirm it. In this situation, the whistleblower can first gather some supporters and then generate a $\sharp$AMS signature to announce the scandal to the public.

Ad-hoc networks. Ad-hoc network is a decentralized type of wireless network, which does not rely on any preset infrastructure, such as routers or wireless access points. Instead, each node participates in the network by forwarding data to other nodes. Our $\sharp$AMS schemes constructed from chameleon hashes and linkable ring signatures enjoy the advantage of spontaneity, and they are perfectly applicable for the application where several spontaneous nodes (users) want to communicate secretly. By attaching a $\sharp$AMS signature with the message sent out, the receiver is convinced of the authority of the message, and senders remain anonymous.

Table of Contents