Game-Theoretically Fair Distributed Coin Tossing With Private Preferences

Pedro Branco[*], Pratik Soni[†], Sri AravindaKrishnan Thyagarajan[‡], Ke Wu[§]

Abstract

Secure coin-tossing is typically modeled as an *input-less* functionality, where parties with no private inputs jointly generate a fair coin. In the dishonest majority setting, however, a strongly fair coin-tossing protocol is impossible. To circumvent this barrier, recent work has adopted the weaker notion of *game-theoretic fairness*, where adversaries are rational parties with preferences for specific outcomes, seeking to bias the coin in their favor. Yet these preferences may encode secret information, making prior protocols that assume preferences are public, fundamentally incompatible with privacy.

We *initiate* a comprehensive study of *privacy-preserving* game-theoretically fair coin-tossing, where the preferences of honest parties remain private. We propose a simulation-based security framework and a new ideal functionality that reconciles *both* preference-privacy and game-theoretic fairness. A key ingredient is a certifying authority that authenticates each party's preference and publishes only aggregate statistics, preventing misreporting while hiding parties' preferences. The functionality guarantees that every honest party receives an output: either a uniform coin; or, if an adversary deviates, a coin that strictly decreases the adversarial coalition's expected utility.

Within this framework, we construct a protocol realizing our ideal functionality under standard cryptographic assumptions that works for both binary and general m-sided coin-tossing. Our schemes tolerate the same optimal (or nearly optimal) corruption thresholds as the best known protocols with public preferences (Wu-Asharov-Shi, EUROCRYPT '22; Thyagarajan-Wu-Soni, CRYPTO '24). Technically, our protocols combine authenticated preferences with an anonymous communication layer that decouples identities from preference-dependent actions, together with a deviation-penalty mechanism that enforces game-theoretic fairness.

Our work is the first to reconcile game-theoretic fairness with preference privacy, offering new definitional tools and efficient protocols for rational multi-party computation in dishonest majority settings.

[*]Bocconi University, pedrodemelobranco@gmail.com

[†]Kahlert School of Computing, University of Utah, psoni@cs.utah.edu

[‡]School of Computer Science, University of Sydney, t.srikrishnan@gmail.com

[§]Electrical Engineering and Computer Science, University of Michigan, kewucse@umich.edu

1 Introduction

Designing protocols for generating useful randomness in a distributed setting is fundamental to both Cryptography and Distributed Computing. Consider three friends, Alice, Bob, and Charlie, who want to toss a coin over a group call to decide on a cafe to meet. Each prefers one of the only two cafes, but wants to appear neutral while keeping their preference private and trying to bias the outcome towards their preference. This scenario involves two key requirements: (a) *Fairness* stipulates that the chosen coin is uniform (i.e., is 0 with probability 0.5), and (b) *preference-privacy* that requires that no coalition of malicious parties learns any additional information about the honest participants' preferences, beyond the chosen coin.

Coin-tossing is typically formalized as an *input-less* functionality (where parties have no input); hence, all prior works focus exclusively on fairness. When the majority of the parties act honestly, the above notion of fairness that the chosen coin must be uniform (generally referred to as *strong fairness*) can indeed be achieved [Yao82, GMW87, BGW88, CCD88, RB89]. In fact, existing protocols that achieve strong fairness work independently of the parties' preferences, hence vacuously achieving preference-privacy.

However, the situation changes dramatically in the presence of a dishonest majority. In the two-party setting, the celebrated result by Cleve [Cle86] rules out the existence of a strongly fair coin-tossing protocol against a single corruption. Moreover, Cleve's negative result can be strengthened to rule out strong fairness in the presence of a dishonest majority for $n \geq 2$ parties.

Fairness in the Dishonest Majority Setting. In light of this, the recent work of Chung, Guo, Lin, Pass, and Shi [CGL+18a] (henceforth, CGLPS18) developed *game-theoretic* notion of fairness that models adversaries as rational parties who are looking to increase their utilities. In particular, they showed that for several settings where coin-tossing is used as a subroutine, each party has an implicit game-theoretic notion of utility. For example, if P_0 and P_1 were running the coin-tossing protocol to elect a leader amongst them, each party P_b would prefer that the sampled coin be b so that they get elected. More formally, a party P_b gains utility 1 if the outcome matches its preference b or utility 0 otherwise. Then, CGLPS18's game-theoretic fairness requires that any party cannot *increase* their expected utility by deviating from the protocol, compared to its utility in an honest run.[1]

CGLPS18 observed that a variant of Blum's two-party coin-tossing protocol [Blu81] already satisfies this notion of game-theoretic fairness against single corruption, hence circumventing Cleve's impossibility. This led to subsequent works on the feasibility of *multi*-party sampling protocols that achieve game-theoretic fairness against dishonest majority coalitions: for binary coin-toss [WAS21] gave a complete characterization of tolerable corruption thresholds, for leader election (or n-sided coin-toss) [CCWS21, KMSW22] explore round optimal protocols, and [TSW24] give a complete characterization of tolerable corruption thresholds for the general case of sampling from any efficiently sampleable distribution.

Public Preferences - A Privacy Concern. While coin-tossing is typically modeled as an input-less functionality, in the context of game-theoretic fairness, each party actually holds individual preferences as inputs. These preferences could encode information that, depending on the application setting, parties may want to keep secret. Therefore, a truly secure coin-tossing protocol

[1] CGLPS18 refer to this notion as Cooperative-Strategy-Proofness (CSP)-fairness. Other notions of fairness such as maximin fairness considered in the literature become infeasible in the strict dishonest majority setting (our focus), hence considered beyond the scope of this work.

must not only ensure fair coin sampling but also preserve the privacy of honest parties' preferences, akin to secure multi-party computation protocols.

Despite this, all prior works on designing game-theoretically fair protocols completely ignore such privacy concerns. Moreover, the designed protocols inherently require the preferences of each party to be known *publicly*. As an example, consider the binary coin-tossing protocol of [WAS21], where the n parties are grouped according to which of the two outcomes they prefer, and these groups act in different phases of the protocol. Consequently, while being part of a group is necessary for the protocol specification, it inadvertently publicly reveals all parties' preferences.

At a definitional level, game-theoretic fairness does require the parties to have pre-determined preferences to make parties' utility functions well-defined. Yet there is no inherent reason that these preferences must be publicly known. This observation motivates the question:

Can we build n-party protocols for tossing an m-sided coin that satisfies both game-theoretic fairness and preference-privacy against a dishonest majority?

1.1 Our Results

In this work, we answer the above question affirmatively. Surprisingly, even defining a meaningful yet feasible security notion capturing both game-theoretic fairness and preference-privacy is quite non-trivial. We propose a new definitional framework for authenticated-preference coin-tossing and provide our new ideal coin-tossing functionality $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}$ that combines game-theoretic fairness and preference-privacy. Below, we will describe our choice of definition and describe the protocols that realize the ideal functionality $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}$. Finally we conclude with some philosophical discussions on the new definitional framework in Section 1.2.

1.1.1 The Need for Authenticated Preferences

Recall that in the public preference setting, coin-tossing protocols and fairness can be naturally defined with respect to a preference profile. However, this is not the case for private preference. CGLPS18 briefly explored the case of private preference. They considered universal n-party private-preference coin-tossing protocols that is defined to work for every possible preference profile. Unfortunately, CGLPS18 show that a universal protocol that satisfies game-theoretic fairness against malicious adversaries becomes impossible. The obstacle lies in *dynamic misreporting*: in any protocol where parties are free to declare preferences during execution, a malicious coalition can report preferences that differ from their true ones, while still measuring utility against their actual (hidden) preferences. This creates a definitional inconsistency: To evaluate fairness, one must compare the adversary's expected utility to its "honest baseline" utility. But if the coalition can misreport its preferences, the honest baseline no longer reflects what the adversary truly values. As a result, even if a protocol is fair with respect to declared preferences, a coalition can still bias the outcome toward its true preference by strategic misreporting. CGLPS18 formalized this gap and proved that fairness with private preferences is impossible in the malicious model. Feasibility can only be recovered under weaker semi-malicious assumptions, where adversaries are forced to commit to their true preferences.

To overcome this barrier, parties must be held accountable to their declared preferences. Without such accountability, the impossibility of CGLPS18 persists.

We address this by introducing *authenticated preferences*, inspired by prior works on authenticated MPC [DGPS22]. Before joining the protocol, each party interacts with a *certifying authority* that signs the party's stated preference. A preference is considered authentic or certified only if it

carries a signature verifiable under the authority's public verification key. Once certified, a party is bound to this preference throughout the protocol execution, preventing adaptive "switching".

In contrast to prior work on authenticated MPC, we require the authority to be *stateful*: after processing all certification requests, it publishes an aggregate statistics of the reported preferences. For example, in the binary coin-tossing case, the authority releases (n_0, n_1), where n_b is the number of parties that report preference b to the authority. While this assumption may appear strong at first glance, we will explain why publishing aggregates is crucial to feasibility of both preference-privacy and game-theoretic fairness (see Section 1.2).

Naturally, a malicious party may still misreport to the authority at the time of certification. We do not attempt to prevent this entirely, but two factors mitigate its impact. First, in many applications the authority (e.g., a regulator, membership registry, or government agency) can independently validate preferences, making misreporting risky. Second, certification must be obtained before the adversary learns honest parties' preferences, therefore preventing adversaries from misreporting strategically based on any information they learned about honest parties.

Thus, while authentication does not eliminate every form of dishonesty, it decisively rules out the *dynamic misreporting attack* that underlies the impossibility result of CGLPS18. This enables a meaningful definitional framework for game-theoretic fairness with private preferences. We emphasize that to the best of our knowledge, some form of setup or authority is unavoidable in this setting (as dynamic misreporting makes purely distributed solutions impossible). Our model uses a certifying authority in the weakest possible way – it is only involved in an initial setup phase and never in the coin-tossing itself. We direct the reader to Section 1.2 for more discussions.

1.1.2 Defining Fairness and Preference-Privacy

A natural approach for capturing both security properties of fairness and preference-privacy would be to define them independently. Particularly, for preference-privacy, we can adopt the simulation-based definition commonly used in secure multi-party computation and zero-knowledge proofs [Yao82, GMW87, Can01]. On the other hand, game-theoretic fairness is typically captured via a property-based definition that guarantees the joint utility of the adversarial coalition must be negligibly close to the coalition's expected honest utility [CGL+18a].

In this work, we instead take a *unified* approach and propose a new simulation-based definition that reconciles both game-theoretic fairness and privacy. Our definition follows the standard real-and-ideal-world paradigm that is considered the gold standard notion for defining security for any multi-party computation [Yao82, GMW87, Can01]. Specifically, we introduce an ideal coin-tossing functionality $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}$ that appropriately models both game-theoretic fairness and preference-privacy.

Ideal Coin-tossing Functionality with Authenticated Preferences. In Section 3, we formally describe our ideal functionality $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, (n_0, \ldots, n_{m-1})]$ for m-sided coin-tossing. Our ideal functionality is additionally parameterized by the verification key mvk of the certifying authority and the aggregate information $(n_0, \ldots, n_{m-1})$ that is published by the certifying authority, where n_i denotes the number of parties reporting preference i to the authority.[2]

For simplicity, let us consider tossing a binary coin and consider an adversary $\mathcal{A}$ that controls a set $\mathcal{K} \subseteq [n]$ of at most t parties. Let $\mathcal{H} = [n] \setminus \mathcal{K}$ be the set of honest parties. We assume that all parties identities (e.g., public-keys) are publicly known. For such an $\mathcal{A}$, the ideal functionality $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (n_0, n_1)]$ works as follows:

[2] In fact, our ideal functionality only requires n_0 and $n - n_0$ where 0 is the least preferred outcome. We state all aggregates here for simplicity.

1. **Reporting by Honest Parties.** Parties $i \in \mathcal{H}$ sends their $(\mathsf{pref}_i, \sigma_i)$ to the ideal functionality which verifies the validity of σ_i as a signature on (i, pref_i) under key mvk. If σ_i is invalid or $\mathsf{pref}_i \notin \{0,1\}$, then set $\mathsf{pref}_i = \perp$. For $b \in \{0,1\}$, functionality computes h_b as the number of parties $i \in \mathcal{H}$ with $\mathsf{pref}_i = b$, and sets $k_b = n_b - h_b$. It then sends (h_0, h_1) to $\mathcal{A}$ as leakage.[3]
2. **Reporting by Corrupt Parties.** Parties $i \in \mathcal{K}$ send their $(\mathsf{pref}_i, \sigma_i)$ to the ideal functionality which verifies the validity of σ_i as a signature on (i, pref_i) under key mvk. If σ is invalid or $\mathsf{pref}_i \notin \{0,1\}$, then set $\mathsf{pref}_i = \perp$.
3. The functionality proceeds to compute a coin $c' \in \{0,1\}$ that $\mathcal{A}$ does *not* prefer as follows: if $k_0 > k_1$, then set $c' = 1$. Otherwise, set $c' = 0$.
4. Then, the ideal functionality samples a uniform random coin $c \xleftarrow{\$} \{0,1\}$ and sends it to $\mathcal{A}$. If $\mathcal{A}$ accepts the coin, the ideal functionality returns c to all honest parties. Otherwise, it sends the coin c' that $\mathcal{A}$ does *not* prefer to all honest parties.

Our functionality is reminiscent of the ideal coin-tossing functionality that captures simulation-security with abort [Lin16], albeit for authenticated inputs. Specifically, the security with abort notion allows $\mathcal{A}$ to prevent honest parties from learning the coin by "aborting" the ideal functionality. In comparison, our ideal functionality *always* delivers an output to all honest parties, irrespective of whether $\mathcal{A}$ choosing to abort. If $\mathcal{A}$ does not abort, the delivered coin is the uniformly chosen coin c; whereas if $\mathcal{A}$ aborts, the ideal functionality delivers the coin c' that necessarily reduces $\mathcal{A}$'s utility. In addition, our ideal functionality enables embedding preference authentication into the coin-tossing protocol, thus forcing adversaries to use authenticate preferences.

Our goal is to design a protocol that realizes $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (n_0, n_1)]$ in the presence of the certifying authority. Philosophically, realizing $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (n_0, n_1)]$ ensures that no coalition controlled by $\mathcal{A}$ can either increase its expected joint utility or learn any additional information about honest parties' preferences beyond what is revealed by (n_0, n_1) and the computed coin.

1.1.3 Binary Coin-Toss: Realizing $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (n_0, n_1)]$

For a given set of n parties, we use $\mathcal{P}$ to denote the *preference profile*, a tuple mapping parties with their reported preferences. Further, we assume the existence of authenticated pairwise private channels and a broadcast channel. We assume that the parties have already requested for certifications on their preferences, and the aggregate information (n_0, n_1) was already published. Without loss of generality, we assume that 0 is the least preferred outcome (i.e., $n_0 < n_1$).

With the above notation, we design a protocol $\Pi_{n,2}$ that securely realizes $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (n_0, n_1)]$, which allows us to toss a unbounded polynomial number of coins in sequence.

Theorem 1.1 (Binary coin-toss, informal)**.** *Let $\mathcal{P}$ be an arbitrary preference profile for n parties where $n_0 < n_1$. Assuming bounded concurrent zero-knowledge arguments, semantically secure public-key encryption and secure digital signature scheme, there exists a protocol $\Pi_{n,2}$ that securely realizes $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (n_0, n_1)]$ against t corruptions, where*

$$t = \begin{cases} n - \left\lfloor \frac{3}{2} n_0 \right\rfloor, & \text{if } n \geq \frac{7}{2} n_0; \\ \left\lfloor \frac{2}{3} n - \frac{5}{6} n_0 \right\rfloor + \left\lceil \frac{1}{2} n_0 \right\rceil, & \text{otherwise.} \end{cases}$$

[3] At first glance, one might think that any adversary $\mathcal{A}$ controlling a coalition already knows h_b, since it can combine its own preferences with the publicly available aggregation n_b. However, in the simulation-based setting the real-world adversary need not reveal these preferences to the simulator. Although our simulator extracts the coalition's preferences, an adversary that aborts prematurely prevents the simulator from extracting all corrupted parties preferences, leaving the simulator unable to determine the true value of h_b needed to complete the simulation. In our protocols, knowledge of h_b is essential, as the simulator relies on it to set up the honest parties' roles correctly.

Semantically-secure PKE and digital signatures can be instantiated from a variety of concrete algebraic assumptions. Moreover, bounded concurrent zero-knowledge proofs can be constructed from enhanced trapdoor permutations and collision resistant hash functions in [Pas04].

1.1.4 General m-sided Coin-Toss: Realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{Leak}^{\mathsf{agg}}(\cdot)]$

Following the technique of Thyagarajan, Wu, and Soni [TSW24], our protocol can be generalized to multi-sided coin-tossing. Given an arbitrary preference profile $\mathcal{P}$, without loss of generality, assume that 0 is the least preferred outcome. Let n_0 denote the number of players preferring outcome 0.

Theorem 1.2 (m-sided coin-toss, informal)**.** *Let $\mathcal{P}$ be an arbitrary preference profile for n parties where 0 is the least preferred outcome. Assuming bounded concurrent zero-knowledge arguments, semantically secure public-key encryption and secure digital signature scheme, there exists a protocol $\Pi_{n,m}$ that securely realizes $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, n_0, n - n_0]$ against t corruptions, where*

$$t = \begin{cases} \left\lceil \frac{1}{2} n_0 \right\rceil + n - m \cdot n_0, & \text{if } n \geq \frac{4m-1}{2} n_0 \\ \left\lceil \frac{1}{2} n_0 \right\rceil + \left\lfloor \frac{m}{2m-1} n - \frac{3m-1}{2(2m-1)} n_0 \right\rfloor, & \text{otherwise .} \end{cases}$$

As a comparison, for binary coin-tossing, our protocol matches the corruption threshold of [WAS22]; for m-sided coin-tossing, it matches that of [TSW24]. Unlike these prior works, which require public knowledge of all parties' preferences and proves game-theoretic fairness under a property-based definition, our protocol achieves the same thresholds while simultaneously ensuring preference-privacy and realizing a simulation-based notion of fairness, albeit under the assumption of authenticated preferences.

Some Concrete Parameters. The above relation between t, n, n_0 is arguably complex. Below, we give some concrete parameters for better understanding the thresholds.

- When the number of players preferring each outcome is the same, i.e., $n = m \cdot n_0$, our protocol $\Pi_{n,m}$ tolerates any $\lceil \frac{n}{2} \rceil$-sized coalition.

  For binary coin-toss, our protocol $\Pi_{2,m}$ tolerates any $\lceil \frac{n}{2} \rceil$-sized coalition when $n = 2n_0$.
- When n is much larger than $m \cdot n_0$, say $n = 10 \cdot m \cdot n_0$, our protocol tolerates any $0.9n$-sized coalitions.

  For binary coin-toss, our protocol $\Pi_{2,m}$ tolerates any $0.85n$-sized coalition when $n \geq 10n_0$.
- When $n_0 = 1$, our protocol $\Pi_{n,m}$ tolerates $(n - m)$-sized coalition.

  For binary coin-toss, our protocol $\Pi_{2,m}$ tolerates any $(n - 1)$-sized coalition.

1.2 Discussion on our Definition

Our definition introduces a certifying authority that both authenticates preferences and publishes aggregate statistics. While these assumptions may appear strong, we view them as the minimal and natural ingredients needed to overcome the impossibility of fairness with private preferences (as shown by CGLPS18). Below we explain why each aspect of our model is necessary, how it aligns with practice, and why it enables the strongest definitional guarantees currently achievable.

Authenticated preferences. Authentication is essential to rule out the dynamic misreporting attack that underlies the malicious-case impossibility. Without being bound to a certified preference, a coalition could declare different preferences opportunistically and bias the protocol toward its true (hidden) desires, a strategy implicit in Cleve's impossibility. For instance, consider five parties where a coalition of size three has true preferences $(0, 0, 1)$ (jointly preferring 0). The coalition misreports $(0, 1, 1)$ (jointly preferring 1) instead. While Cleve's strategy indicates that the coalition must be able to bias the outcome (towards some bit), game-theoretic fairness rule out all strategies that bias towards 1 based on their reported preferences $(0, 1, 1)$. Therefore, the coalition is able to bias the outcome towards 0, their true joint preferences.

By contrast, authenticated preferences rules out this adaptation and ensures that once a party has obtained a signature on its stated choice, it is committed to that choice throughout the protocol. Of course, parties may still misreport to the authority in the certification stage, but in many applications (e.g., when the authority is a regulator or membership registry) the truthfulness of preferences can be verified. More importantly, adversaries must commit before seeing others' preferences, preventing them from tailoring lies strategically.

On Stateful Authority and Aggregate Information. The role of the authority extends beyond authentication: it is also stateful and publish aggregate statistics (e.g., the number of parties preferring 0 versus 1). If this information is not published, two challenges arise. First, many fair-coin protocols, including ours, internally require aggregate preference information. Absent the authority, the parties would need to compute this information themselves, which in the dishonest-majority setting requires guaranteed output delivery MPC, known to be impossible. Second, one might design protocols that ignore parties that fail to provide valid authenticated preferences (e.g., using commit-and-prove with identifiable abort), but such designs remain vulnerable: the coalition could provide unauthenticated preferences for enough parties (in the coalition) such that the "alive" portion of an adversarial coalition prefers a different value than the full coalition's preference. This "alive" coalition can then perform the Cleve's strategy to bias the outcome towards the whole coalition's preference as described above. The publication of aggregate information prevents this adaptation.

One might ask whether fairness could be achieved with only authentication from the authority, but without requiring it to provide aggregate statistics. While this remains an open theoretical possibility, we are unaware of any prior constructions, and we conjecture that some aggregate information is inherent to any meaningful notion of fairness in this setting. Indeed, even defining a coherent ideal functionality without (n_0, n_1) appears challenging, since the fairness condition must be evaluated against the distribution of preferences.

Coin-tossing in Presence of an Authority. Our model does not delegate coin-tossing itself to the authority, despite its trusted role in setup. The reason is both scalability and practicality: we envision the authority's role as a one-time initialization step, after which parties can independently toss an unbounded sequence of coins. This avoids requiring the authority's continuous availability while still preserving fairness and privacy. In this respect, our use of an authority parallels authenticated MPC, where the authority ensures correct inputs but does not participate in the actual computation.

On Composability. Our framework provides the first simulation-based formulation of game-theoretic fairness with private preferences that enjoys composability. Previous property-based notions lacked such guarantees, limiting their utility as subroutines in larger protocols. By aligning

with the real/ideal paradigm familiar from secure computation, our definition ensures that fair sampling protocols can be modularly embedded into higher-level tasks.

Taken together, these design choices reflect a deliberate balance: while our model imposes additional structure through a certifying authority, this structure appears indispensable for reconciling fairness, privacy, and malicious security. In this sense, we view our framework as the strongest currently feasible approach, and as a foundation for future refinements that may further reduce assumptions.

1.3 Additional Related Work

Exploring the connection between game theory and multi-party computation is not new. A line of work [HT04, KN08, ADGH06, OPRV09, AL11, ACH11] explored rational multi-party computation. However, the utility they consider differs from the line of work on game-theoretic fairness. Specifically, they define the utility based on the assumption that players prefer to compute the function correctly and learn others' secrets while not leaking their secrets.

Recently, with the success of blockchains and digital payments, many works explored a notion called financial fairness [BK14, KMS+16, ADMM16, KB14, KVV16, DEF18, ATM+22, CMST22] based on the use of collaterals. Parties are required to put in collaterals beforehand and get punished if they misbehave. Interestingly, our work and the previous work on game-theoretic fairness [CGL+18b, WAS22, CCWS21, KMSW22, TSW24] guarantees incentive compatibility even without the use of any collateral.

A complementary line of works focus on designing multi-party coin-tossing that achieve security with abort in the dishonest majority setting. In such constructions, the emphasis is on building a round optimal protocol in the plain model that tolerates $n-1$ corruptions. Yao's [Yao82] proposed a constant round protocol for securely computing any two-party functionality (including coin-tossing) against semi-honest adversaries. The following work of [GMW87] then presented a general-purpose compiler for lifting a semi-honest multi-party protocol to a malicious security, but incurs polynomial rounds. Since this a long line of works [BMR90, Pas04, ACJ17, BHP17, IPS08, Wee10, Goy11, IKSS21] (not exhaustively cited) has optimized the round complexity of coin-tossing (and more generally, secure computation), culminating in round optimal protocols based on standard cryptographic assumptions [BGJ+18, HHPV21, RCCG+20].

A line of work has studied how to enforce the integrity of parties' inputs in secure computation via *authenticated inputs*. Dutta et al. [DGPS22] give a general compiler that transforms any honest-majority, linear-secret-sharing based MPC into one with authenticated inputs, achieving strong security guarantees such as identifiable abort without costly, protocol-specific preprocessing. While our focus differs, aiming to reconcile game-theoretic fairness with preference-privacy in dishonest majority coin-tossing, our use of authenticated preferences follows the same insight: binding parties to certified inputs is often essential to overcome fundamental impossibility results.

Organization. In Section 2, we illustrate the overview of the techniques used in our protocol. The formal model is given in Section 3, and the coin-tossing protocol is presented in Section 4.

2 Technical Overview

In this section, we present a brief overview of our techniques. For a better understanding, we will first explain how to construct a binary coin-tossing protocol that securely realizes the ideal functionality against semi-malicious coalitions, who can program its randomness used in the protocol

or abort early, but otherwise follows the protocol honestly. Then we explain how to upgrade to malicious security and multi-sided coin-toss.

Recall that parties get their preference certified by an authority prior to the coin-tossing protocol. We call those who prefer outcome $b \in \{0,1\}$ as b-supporters. If the outcome of the coin-tossing protocol matches the party's authenticated preference, the party gets utility 1 and 0 otherwise. The expected joint utility of a coalition is the sum of all coalition member's expected utility.

Before presenting our protocol, we first outline the public preference protocol from previous works and explain where the preference of each party is being leaked. Then we show how to hide the parties' preferences using anonymous broadcast. Jumping ahead, the anonymous broadcast required in our protocol can be instantiated using only Shamir secret sharing [Sha79].

Why Previous Protocols Leak Preferences. Previous game-theoretic fair protocols [WAS22, TSW24] assumes that each party's preference is publicly known. They work in the following framework: Let G_b denote the set of b-supporters, for $b \in \{0,1\}$.

1. Each 0-supporters first interacts with other 0-supporters in group G_0 to agree on a coin c_0 using Shamir's secret sharing, during which G_1 audits. Similarly, each 1-supporter interacts with other 1-supporters in G_1 to decide a coin c_1 using secret sharing.
2. Groups G_0 reconstructs coin c_0, and then group G_1 reconstructs coin c_1.

The outcome of the protocol is then defined based on c_0 and c_1.

In the above protocol, 0-supporters only interact with 0-supporters, while 1-supporters only interact with 1-supporters via secret sharing. Clearly, protocols with such structure cannot protect parties' preferences: For each party to send messages, they need to know the recipients' identities and preferences. Otherwise, how would they know where to send the message?

Our Protocol: Intuition. In a nutshell, our protocol adds an anonymous network mechanism on top of the previous framework to protect the identity of both the sender and the receiver of any message being sent in the protocol, so that it is infeasible to link a party with its preference. Only the aggregate information is revealed. Specifically, we need the following properties:

1. *Anonymity*: Neither the sender nor the receiver can identify each other's real identity.
2. *Privacy*: For any message, its content, sender identity, and receiver identity remain hidden from all other parties.

Property 1 above ensures that honest parties' preferences are unlinkable via anonymous communication. Property 2 guarantees that all the private messages remain hidden from the adversary to ensure game-theoretic fairness.

Why Natural Approaches Fail. A natural approach is to use an anonymous broadcast channel such as DC-nets [Cha88] that hides the sender's identity for any broadcast messages. A straw-man attempt may instruct a party to send all messages via this anonymous broadcast channel. However, broadcasting all messages directly compromises Property 2. Therefore, we need the senders to first encrypt a message under the receiver's public key for some pubic-key encryption scheme,[4] and then broadcast it through the anonymous broadcast channel. In this way, only the correct receiver can decrypt the message, while no other parties learn the message content, thus achieving Property 2.

There is still one main challenge: how does a sender know who to send messages to? In the previous public-preference protocols [WAS21,TSW24], b-supporters only interact with b-supporters. Even with anonymous broadcast channel, the sender needs to know which public keys corresponds to parties with the same preferences to encrypt the message. Thus, the sender needs to know

[4]This public key is different from the public key that represents parties' identities. This public key is used to encrypt the corresponding message and is independent from the public key representing a party's identity.

the public keys for those who have the same preferences, while honest parties' preferences need to remain hidden.

Our Approach: Masked Profile. To achieve this, we ask each party i to randomly generate a virtual ID vID_i and submits $(\mathsf{vID}_i, \mathsf{pref}_i)$ to the certificate authority to get authentication. This virtual ID will be used throughout the protocol as party i's ID, and we need to hide the mapping between parties' real identities and their virtual identities.

At the beginning of the protocol, all n parties enter a *commit phase* by anonymously broadcasting a tuple of their virtual IDs, an encryption key used to encrypt messages to party i, and its preference $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$. At the end of commit phase, parties agree on a "masked preference profile" $\mathcal{P} = \{(\mathsf{vID}_1, \mathsf{pk}_1, \mathsf{pref}_1), \ldots, (\mathsf{vID}_n, \mathsf{pk}_n, \mathsf{pref}_n)\}$, which itself is a multiset that hides the mapping between party i and its tuple $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$. Some corrupted parties may fail to broadcast a valid tuple and their preferences are set to $\perp$. Therefore, each $\mathsf{pref}_i \in \{0, 1, \perp\}$ in $\mathcal{P}$.

With this masked preference profile, one can imagine a public-preference scenario where the preference of each virtual ID is publicly known, except that now some parties may have $\perp$-preferences. Parties can now interact with others using a similar framework as for public-preferences using virtual IDs and anonymous broadcast channel, but adapt to also account for $\perp$-preferences. For example, if a 0-supporter with virtual ID vID_i wants to send a message to another 0-supporter with virtual ID vID_j, party vID_i encrypts the message using pk_j and broadcast it anonymously. This achieves both desired properties.

Instantiating Anonymous Broadcast. When instantiating the protocol in the real world, we cannot assume the availability of an ideal anonymous broadcast channel that always succeeds. In fact, when defending against majority-sized coalitions, the anonymous broadcast may fail. To handle this, we adopt the identifiable abort approach from [KMSW22]. Consider the following anonymous broadcast with identifiable abort functionality: either the anonymous broadcast succeeds or all honest parties receive the identities of some misbehaving parties. The parties kick out those misbehaved parties and retry the anonymous broadcast with the same message. This functionality guarantees that anonymity is preserved, and the broadcast will eventually succeed, i.e., honest parties' messages will eventually be successfully broadcast. Such ideal functionality can be instantiated using only secret sharing against semi-malicious coalitions.

Our Protocol Description. We now instantiate $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$ using the masked profile and anonymous broadcast with identifiable abort. The protocol proceeds in three phases and is parameterized with two parameters t_0 and t_1. We will explain how to choose the parameters afterwards.

Each party i samples a virtual identifier $\mathsf{vID}_i \xleftarrow{\$} \{0,1\}^\lambda$ and a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ that will be used solely for encryption. It obtains auth_i on $(\mathsf{vID}_i, \mathsf{pref}_i)$ from $\mathsf{agg\text{-}auth}$. Here, the authority's signature binds the virtual identity to the certified preference.

1. *Commit phase:* Each party anonymously broadcasts a tuple $(\mathsf{vID}_i, \mathsf{pref}_i, \mathsf{pk}_i)$. If fail, remove the misbehaved parties and retry with the same message. At the end of this phase, all honest parties agree on a *masked profile*

$$\mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}, \qquad \mathsf{pref}_i \in \{0, 1, \perp\},$$

   This profile reveals only the aggregate information (n_0, n_1), and hides the link between parties' real indentity and $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ by anonymity of the anonymous broadcast.

2. *Sharing phase:* Let $G_b = \{\mathsf{vID}_i : (\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \mathcal{P},\ \mathsf{pref}_i = b\}$ for $b \in \{0,1\}$. Those who have $\perp$-preferences are treated as if they aborted at the beginning of the sharing phase. The protocol now runs *two* group-toss sub-protocol, one for G_0 and one for G_1.

   Each party $\mathsf{vID}_i \in G_b$ randomly picks a secret $s_i \xleftarrow{\$} \{0,1\}$ and generates a (t_b+1)-out-of-n_b Shamir sharing $\{s_{i,j}\}$. Note that although G_b is the group of valid b-supporters, parties generate n_b number of shares using the true aggregation from the certified authority, each party will only send shares to those in G_b. For every recipient $\mathsf{vID}_j \in G_b$, encrypt the shares using the corresponding receivers' public keys and use anonymous broadcast with identifiable abort to broadcast all encrypted shares. Retry until success. Denote G'_b as those who successfully shared.

3. *Reconstruction phase:* Each $\mathsf{vID}_i \in G_b$ decrypts the incoming shares for itself, sums the decrypted shares from parties in G'_b and *anonymously broadcasts* its reconstruction shares to reconstruct $c_b = \oplus_{\mathsf{vID}_i \in G'_b} s_i$. Kick out misbehaving parties and retry the anonymous broadcast until succeed. If the reconstruction of c_0 fails, set $c_0 = 0$; if the reconstruction of c_1 fails, the protocol directly outputs 0. Otherwise if both reconstruction succeed, output $c_0 \oplus c_1$.

In a nutshell, the security follows from the anonymity that hides the mapping between all messages and the senders, thus making the virtual IDs unlinkable from real identities. Only the aggregation information is revealed. The protocol achieves game-theoretic fairness against any coalition satisfying the following three constraints

1. The coalition cannot control both coins.
2. If the coalition can control coin c_1, then the coalition cannot fail the reconstruction of coin c_0.
3. If the coalition can fail the reconstruction of c_1, then it must not prefer 0.

At a high level, Constraint 3 guarantees that a coalition is never incentivized to fail the reconstruction of c_1. Given that c_1 is always successfully reconstructed, Constraints 1 and 2 guarantees that the output is a uniformly random coin. The parameters t_0 and t_1 are chosen to maximize the coalition tolerance.

Upgrade to Malicious Security. To achieve malicious security, we ask each party to prove the validity of their computation in each phase using bounded concurrent zero-knowledge proofs [Pas04]. Specifically,

- In the commit phase, each party i proves that they have a valid certification from the authority for their broadcast preference $(\mathsf{vID}_i, \mathsf{pref}_i)$.
- In the sharing and reconstruction phase, each party proves that the shares are correctly computed.

Extension to m-Sided Coin-Toss. We adopt the standard two-group reduction [TSW24]: treat 0 as the least-preferred outcome and aggregate all other outcomes into oth. The protocol is identical to the above, except that coins are produced by G_0 and G_{oth} and combined as $(c_0 + c_{\mathsf{oth}}) \bmod m$. Our ideal functionality $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}$ leaks only (h_0, h_{oth}); the same anonymity, privacy, and fairness arguments apply, with thresholds chosen to satisfy the feasibility constraints in the main theorem.

3 Model and Preliminary

Notations. We assume each party is a probabilistic polynomial time (PPT) Turing machine. Throughout the paper, we may use $\mathcal{A}$ to denote the corrupted coalition and adversary controlling the coalition interchangeably. We use λ to denote the security parameter. We call a function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ negligible in n if for every $k \in \mathbb{N}$, there exists $n_0 \in \mathbb{N}$ s.t. for every $n \geq n_0$ it holds that $|\mathsf{negl}(n)| \leq n^{-k}$. In this work, $[n]$ denotes the set of integers $\{1, \ldots, n\}$.

We use U_m to denote the uniform distribution over $\{0, ..., m-1\}$. Given two distribution ensembles $\{X_n\}_n$ and $\{Y_n\}_n$, we say that the two ensembles are computationally indistinguishable, denoted as $\{X_n\}_n \approx \{Y_n\}_n$, iff there exists a negligible function $\mathsf{negl}(n)$ such that for any non-uniform PPT adversary $\mathcal{A}$, for any n, we have $|\Pr[x \leftarrow X_n : \mathcal{A}(x) = 1] - \Pr[y \leftarrow Y_n : \mathcal{A}(y) = 1]| \leq \mathsf{negl}(n)$. We use $\equiv$ to denote that two distributions are identical.

3.1 Coin-Tossing Protocol With Certified Authority

In an n-party coin-tossing protocol with a certified authority $\mathsf{agg\text{-}auth}$, each of the n parties first gets its preference certified by the certificate authority. Each party i has a preference pref_i over the outcomes in $\{0, \ldots, m-1\}$. We use the set $\mathcal{P} = \{(\mathsf{ID}_1, \mathsf{pref}_1), \ldots, (\mathsf{ID}_n, \mathsf{pref}_n)\}$ to denote the *preference profile* of the parties, where pref_i denotes party i's preference. Without loss of generality, we assume that 0 is the least preferred outcome, i.e., $n_0 \leq n_\ell$ for any $\ell \in \{0, \ldots, m-1\}$, where n_ℓ denotes the number of ℓ-supporters in the preference profile. We also refer to those non-0-supporters as oth-supporters and use $n_{\mathsf{oth}} = n - n_0$ to denote the number of oth-supporters.

In this work, we consider a certified authority parametrized with a digital signature scheme $\Pi_{\mathrm{Sig}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$, with which the authority signs parties' inputs. The authority $\mathsf{agg\text{-}auth}$ samples a pair of $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \Pi_{\mathrm{Sig}}.\mathsf{Gen}(1^\lambda)$, receives $(\mathsf{ID}_i, \mathsf{pref}_i)$ from party i and sends $\mathsf{auth}_i = \Pi_{\mathrm{Sig}}.\mathsf{Sign}(\mathsf{msk}, (\mathsf{ID}_i, \mathsf{pref}_i))$ to party i. It adds $(\mathsf{ID}_i, \mathsf{pref}_i)$ into a set $\mathcal{P}$ and ignores any messages with duplicate ID_i already in $\mathcal{P}$. At the end, it publishes $(\mathsf{mvk}, n_0, n_{\mathsf{oth}})$ on some public bulletin board, where n_0 and n_{oth} denote the number of 0-preferences and non-0 preferences in $\mathcal{P}$.

With certified preferences, the n parties communicate with each other through a pairwise private channel and a public broadcast channel. We assume that the coin-tossing protocol proceeds in rounds. All the communication channels are synchronous and authenticated, i.e., each message carries the identity of the true sender. A coalition $\mathcal{A}$, also referred to as the adversary for cryptographic analysis, can perform *rushing attacks*: It waits for honest parties' messages in any round r and then decides what messages to send to the honest parties in round r. If a party sends ill-formed messages, it is treated as abort.

We require *public verifiability* of the outcome: at the end of the protocol, the coin outcome is a deterministic, polynomial-time computable function based on the public aggregate information posted by the certified authority $\mathsf{agg\text{-}auth}$ and all messages posted in the broadcast channel.

Utility and Strategies. A party gets utility 1 if the outcome agrees with its certified preference and 0 otherwise. A coalition's utility is the sum of the utilities of all coalition members.

A *malicious* coalition can adopt a mix of the following strategies: 1) program its randomness based on its current view, or 2) abort the protocol in some round r, or 3.) send arbitrary messages in each round r.

Ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, n_0, n_{\mathsf{oth}})]$

// Let $\mathcal{H}$ be the set of honest parties and let $\mathcal{K} = [n] \setminus \mathcal{H}$.

1. **Reporting of Honest Parties Preferences**.
   (a) Each honest party in $\mathcal{H}$, sends $(\mathsf{ID}_i, \mathsf{pref}_i, \mathsf{auth}_i)$ to the ideal functionality. Ideal functionality verifies if auth_i is a valid signature of $(\mathsf{ID}_i, \mathsf{pref}_i)$ under mvk. If not, set $\mathsf{pref}_i = \perp$.
   Let $\mathcal{P}_{\mathcal{H}} = \{(\mathsf{ID}_i, \mathsf{pref}_i)\}_{i \in \mathcal{H}}$ to be the preference profile of the honest parties. Let h_0 and h_1 be the number of 0- and 1-supporters in $\mathcal{P}_{\mathcal{H}}$.
   (b) Ideal functionality sends $\mathsf{Leak}(\mathcal{P}_{\mathcal{H}}) = (h_0, h_{\mathsf{oth}})$ to $\mathcal{A}$.
2. **Reporting of Adversarial Preferences.** Each party in $\mathcal{K}$, sends $(\mathsf{ID}_i, \mathsf{pref}_i, \mathsf{auth}_i)$ to the ideal functionality. Ideal functionality verifies if auth_i is a valid signature under mvk of $(\mathsf{ID}_i, \mathsf{pref}_i)$. If not, set $\mathsf{pref}_i = \perp$.
   Let $\mathcal{P}_{\mathcal{K}} = \{(\mathsf{ID}_i, \mathsf{pref}_i)\}_{i \in \mathcal{K}}$ be the preference profile of the corrupted parties.
3. **Coin Tossing.** Ideal functionality computes two coins c and c' as follows:
   The first coin $c \xleftarrow{\$} \{0, 1, ..., m-1\}$ is computed uniformly at random.
   The second coin c' is set as the lexicalgraphically smallest coin that the coalition does not like, i.e., coalition's joint utility under c' is no more than $\frac{|\mathcal{A}|}{m}$, the honest expected utility.
4. **Output Generation.** Send c to $\mathcal{A}$. If $\mathcal{A}$ sends ok, then c is send to all honest parties in H, else if $\mathcal{A}$ sends fail, then c' is sent to all honest parties.

Figure 1: Ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$ for m-sided coin toss.

3.2 Coin-Tossing Ideal Functionality

We adopt a simulation-based definition to capture both game-theoretic fairness and preference privacy. Consider the following ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$ parametrized with a parameter m, the number of sides of the coin to be tossed, the verification key for checking authentication from $\mathsf{agg\text{-}auth}$ and the published aggregate information n_0 and n_{oth}.

Let $\mathcal{P}_{\mathcal{H}}$ denote the preference profile of honest parties. In the case of the public preference profile considered in previous works [CGL+18c, WAS22, TSW24], where every party's preference is public, one may think of an "all" leakage where $\mathsf{Leak}^{\mathsf{all}}(\mathcal{P}_{\mathcal{H}}) = \mathcal{P}_{\mathcal{H}}$. For private preference, we consider the following leakage in this work:

$$\mathsf{Leak}(\mathcal{P}_{\mathcal{H}}) = (h_0, h_{\mathsf{oth}}),$$

where h_0 and h_{oth} denotes the number of 0- and oth-supporters in $\mathcal{P}_{\mathcal{H}}$.

In Appendix B, we present the previous property-based definition of game-theoretic fairness defined in [CGL+18a] and show that the above ideal functionality implies property-based game-theoretic fairness.

Definition 3.1 (Instantiation of $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$)**.** Given any $m \geq 2$. A protocol securely realizes $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, n_0, n_{\mathsf{oth}}]$ against any t-sized coalition, iff there exists an expected p.p.t. simulator Sim interacting with the above ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{Leak}]$, such that for any non-uniform p.p.t. adversary $\mathcal{A}$ control-

ling t or less number of parties, such that for any preference profile,

$$\{\mathsf{Exec}^{\Pi}_{\mathcal{A}}(1^\lambda, \mathcal{P})\} \approx \{\mathsf{Exec}^{\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}[m,\mathsf{Leak}]}_{\mathsf{Sim}}(1^\lambda, \mathcal{P})\},$$

where $\mathsf{Exec}^{\Pi}_{\mathcal{A}}(1^\lambda, \mathcal{P})$ outputs the honest parties' output as well as the $\mathcal{A}$'s view when interacting in coin toss protocol Π, and $\mathsf{Exec}^{\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}[m,\mathsf{Leak}]}_{\mathsf{Sim}}(1^\lambda, \mathcal{P})$ outputs the honest parties' output as well as the Sim's output.

3.3 Building Blocks

For a better understanding of our construction, we present the syntax of the building block and informally state what kinds of security guarantees they provide. The security of these schemes will be formally defined in Appendix A.

Shamir Secret Sharing. A k-out-of-n secret sharing scheme consists of two algorithms Share and $\mathsf{Reconstruct}$:

- $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s)$: On input a secret s, outputs n shares $s_1, \ldots, s_n$ of the secret.
- $s \leftarrow \mathsf{Reconstruct}(I, (s_i)_{i\in I})$: The reconstruction algorithm takes $(s_i)_{i\in I}$ and computes the secret only if $|I| \geq k$. Else, it outputs fail.

Roughly speaking, the security of Shamir's secret sharing guarantees that any k or more parties can reconstruct the secret, while any less than k parties get no information about the secret. Our construction will make use of the *linearity* of secret sharing: if every party gets a share of s and a share of s', then they can reconstruct $s + s'$ using the sum of their shares.

Digital signature. A digital signature scheme Π_{Sig} consist of three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$:

- $\mathsf{Gen}(1^\lambda)$: takes in the security parameter λ and outputs a key pair $(\mathsf{vk}, \mathsf{sk})$ of verification key vk and signing key sk.
- $\sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk})$: takes in a message m and the signing key sk, produces a signature σ.
- $b \leftarrow \mathsf{Ver}(\mathsf{vk}, \sigma, m)$: takes in a message m, the signature σ, and the verification key vk, output 0 or 1 indicating reject or accept.

A secure digital signature is unforgeable: the adversary, without the knowledge of the signing key, cannot forge a valid signature on some new messages.

Semantically-Secure Public-Key Encryption. Public-key encryption consists of three algorithms:

- $\mathsf{Gen}(1^\lambda)$ takes in the security parameter λ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$: take in the public key and a message msg, outputs a ciphertext ct.
- $\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: take in a ciphertext ct and the secret key sk, output msg.

A semantically secure public-key encryption hides the encrypted message from the adversary.

4 Instantiating $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}[m, \mathsf{mvk}, n_0, n_{\mathsf{oth}}]$

In this section, we present an m-sided coin-toss protocol that realizes $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}[m, \mathsf{mvk}, n_0, n_{\mathsf{oth}}]$ (Figure 1).

At a high level, our protocol follows a structure similar to the public-preference coin-toss protocol against semi-malicious adversaries described in [TSW24]: Parties are partitioned into two groups: the set G_0 of 0-supporters and the set G_{oth} of all other supporters, denoted as oth-supporter. Each b-supporter will run the Shamir secret sharing among its own group G_b to jointly decide a coin c_b for $b \in \{0, \mathsf{oth}\}$. The final output is then defined based on c_0 and c_{oth}.

To achieve privacy and simulation security, we make the following modifications:

- *Masked preference:* To hide parties' preferences, each party i will sample a "virtual ID" vID_i for themselves and use this virtual ID to communicate *anonymously*. The privacy of the preference is achieved by hiding the mapping between the real identities i of a party and its virtual ID vID_i throughout the protocol. Players first computes a "masked" preference profile $\mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}$ and interact with other virtual IDs based on the masked preference profile. The pk_i here is only used for encrypting messages and is independent from a party's real identity.
- *Support undefined preference:* We make the protocol to work even when some corrupted parties' preferences are $\perp$, whereas previous protocol only works when every party's preference is publicly known in clear. This $\perp$-preference accounts for corrupted players' misbehavior when reporting their preferences.
- *Anonymous communication:* We make all the messages public by encrypting the private messages using the receiver's public key for encryption. This allows parties to send all messages using anonymous broadcast and therefore protects the mapping between the real identities i and the corresponding virtual ID vID_i.

4.1 Building Blocks

Bounded Concurrent Zero-Knowledge Proofs. To achieve malicious security, we require the parties to prove validity of their behaviors in each step. We will describe the protocol in a $\mathsf{IdealZK}$-hybrid model, where parties have access to an ideal zero-knowledge proof functionality $\mathsf{IdealZK}$ given below. The functionality $\mathsf{IdealZK}$ either sends $\mathsf{success}$ to *everyone* indicating that the proof is correct, or the identity of the prover/verifier who leads to the failure of the proof. Formally,

Ideal Zero-knowledge Functionality $\mathsf{IdealZK}[x, \mathcal{L}, i, j]$

The functionality involves n parties $1, \ldots, n$, and is parametrized with a statement x, the language $\mathcal{L}$, the prover's identity i and the verifier's identity j.

1. If both the prover i and the verifier j are corrupted, receive a bit b from the prover i. If $b = 1$, send $(\mathsf{success}, i, j)$ to everyone.
2. Receive ok or $\perp$ from the verifier j.
3. If received $\perp$ from the verifier, send (fail, j) to everyone.
4. Receive w or $\perp$ from the prover.
5. If $\mathcal{R}(x, w) = 1$, send $(\mathsf{success}, i, j)$ to everyone. Otherwise send (fail, i) to everyone.

In an n-party $\mathsf{IdealZK}$-hybrid protocol, parties invoke $\mathsf{IdealZK}[x, \mathcal{L}, i, j]$ between every pair of prover and verifier for some NP language $\mathcal{L}$. Th result of the proof, either succeed, or who leads to the failure of the proof, is broadcast to every party. Given an n-party $\mathsf{IdealZK}$-hybrid protocol, $\mathsf{IdealZK}$ can be instantiated using the elegant techniques suggested by Pass [Pas04].

Theorem 4.1. *(Constant-round, bounded concurrent secure computation [Pas04]). Assume the existence of enhanced trapdoor permutations and collision-resistant hash functions. Then, given an n-party $\mathsf{IdealZK}$-hybrid protocol Π^*, in which the number of concurrent calls of $\mathsf{IdealZK}$ is upper bounded by a-priori known bound $\mathsf{poly}(\lambda)$, there exists a real-world protocol Π such that the following hold:*

For every real-world non-uniform p.p.t. adversary $\mathcal{A}$ controlling an arbitrary subset of up to $n-1$ players in Π, there exists a non-uniform p.p.t. adversary $\mathcal{A}^*$ in the protocol Π^*, such that for any input $(x_1, \ldots, x_n)$, every auxiliary string $z \in \{0,1\}^*$,

$$\mathsf{Exec}^{\Pi}_{\mathcal{A}}(1^\lambda, x_1, \ldots, x_n, z) \equiv_c \mathsf{Exec}^{\Pi^*}_{\mathcal{A}^*}(1^\lambda, x_1, \ldots, x_n, z).$$

In the above, the notation $\mathsf{Exec}^{\Pi}_{\mathcal{A}}$ (or $\mathsf{Exec}^{\Pi^*}_{\mathcal{A}^*}$) outputs each honest players' outputs as well as the corrupt players' (arbitrary) outputs. Moreover, The round complexity of Π is at most a constant factor worse than that of Π^*.

This real-world protocol is fulfilled by replacing the $\mathsf{IdealZK}$ instance with the bounded concurrent zero-knowledge proofs. All the zero-knowledge proof messages are posted to the broadcast channel.

Anonymous Broadcast Channel with Identifiable Abort. Throughout the protocol, we need to hide the mapping between virtual IDs and the real identity of parties via anonymous broadcast. However, ideal anonymous broadcast is impossible against corrupted majority. Therefore, parties interact with the following $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ ideal functionality to anonymously broadcast a message. The ideal functionality $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ receives messages from the senders and broadcasts a multiset containing all messages it receives, revealing no information about the sender's identity. If the broadcast fails, parties kick out the misbehaving parties and retry.

$\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$: Anonymous broadcast with t-identifiable abort

- *Parameters*: Let $\mathcal{O}$ be the set of remaining parties, initialized as the set of all parties.
- *Input*: Every party i sends m_i or $\perp$ to $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$.
- *Output*: $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$computes a multiset $\mathsf{Out} = \{m_i\}_{i\in\mathcal{O}}$. Send Out to the adversary $\mathcal{A}$. If $\mathcal{A}$ sends ok, $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ sends Out to every honest party; otherwise, it receives a set $\mathcal{D}$ of at least t number of corrupted IDs.

Figure 2: Anonymous broadcast ideal functionality $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$

Lemma 4.2 (Theorem 6.2 in [KMSW22])**.** *Assuming perfectly binding and computationally hiding commitments. there is a protocol that securely realizes $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ in the $\mathsf{IdealZK}$-hybrid model against $|\mathcal{O}| - t$ corruptions.*

In our protocol, parties will need to invoke $\mathsf{IdealZK}[*, *, \mathsf{vID}_i, \mathsf{vID}_j]$ using provers' and verifiers' virtual identities vID_i and vID_j. Therefore, when instantiating the protocol, all the zero-knowledge proof messages will be routed using $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$, which itself can be instantiated using $\mathsf{IdealZK}[*, *, i, j]$ over real identities for provers i and verifiers j.

4.2 Protocol Description

At the beginning of the protocol, each party i holds its preference pref_i, its authentication auth_i and the aggregate information $(\mathsf{mvk}, n_0, n_{\mathsf{oth}})$ from the certificate authority. Recall that we assume 0 is the least preferred outcome. Our protocol contains three phases:

1. *Commit phase*: parties randomly samples a virtual ID vID and commit to their authenticated preference. At the end of the commit phase, parties agree on a masked preference profile $\mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}$ where each $\mathsf{pref}_i \in \{0, \mathsf{oth}, \perp\}$ and all the virtual IDs are unique.

2. *Sharing phase*: Given the masked preference profile, parties with non-$\perp$ preferences are partitioned into the group of 0-supporters G_0 and the group of oth-supporters G_{oth}. Each group G_b for $b \in \{0, \mathsf{oth}\}$ separately run a sub-protocol to jointly determine a coin c_b.
3. *Reconstruction phase:* Each group G_b reconstruct the coin c_b and the reconstruction outcome determines the final output.

4.2.1 Group Toss Sub-Protocol.

We first describe the sub-protocol for each group G_b for $b \in \{0, \mathsf{oth}\}$, i.e., how each party acts during the sharing and reconstruction phase, given a masked preference profile $\mathcal{P}$. Recall that at the end of the commit phase, parties agree on a masked preference profile $\mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}$, where $\mathsf{pref}_i \in \{0, \mathsf{oth}, \perp\}$. Here, we do not distinguish preferences $1, ..., m-1$, they are all denoted by oth.

Let G_0 be the set of virtual IDs whose preference is 0 and G_{oth} be the set of virtual IDs whose preference is oth. Formally,

$$G_0 = \{\mathsf{vID}_i : (\mathsf{vID}_i, *, 0) \in \mathcal{P}\}, \quad G_{\mathsf{oth}} = \{\mathsf{vID}_i : (\mathsf{vID}_i, *, \mathsf{oth}) \in \mathcal{P}\}, \tag{1}$$

where $*$ stands for wildcards.

NP Languages. We specify the following NP languages that will be used in our protocol. Specifically, parties invoke $\mathsf{IdealZK}$ for $\mathcal{L}_S$ in the sharing phase, and $\mathsf{IdealZK}$ for $\mathcal{L}_R$ in the reconstruction phase to prove correctness of computation.

Language $\mathcal{L}_S$

Given a statement $x_i = (\mathsf{Out}, \{(\mathsf{vID}_j, \widetilde{s}_{i,j})\}_{\mathsf{vID}_j\in G_b}, \mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}, G_b, t_b)$, we say $x_i \in \mathcal{L}_S$ if there exists a witness $w_i = (s_i, \{s_{i,j}\}_{j\in[n_b]}, \mathsf{sk}_i)$ such that

- $(\mathsf{vID}_i, \{(\mathsf{vID}_j, \widetilde{s}_{i,j})\}_{\mathsf{vID}_j\in G_b})$ is an entry in Out.
- $(\mathsf{pk}_i, \mathsf{sk}_i)$ is a valid key pair, $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \mathcal{P}$, and $\mathsf{vID}_i \in G_b$.
- For each $\mathsf{vID}_j \in G_b$, the ciphertext $\widetilde{s}_{i,j}$ is correctly computed using pk_j.
- $\{s_{i,j}\}_{j\in[n_b]}$ forms a valid (t_b+1)-out-of-n_b secret sharing of s_i.

Language $\mathcal{L}_R$

Given a statement $x'_i = (\mathsf{Out}', \mathcal{P}, G_b, \mathcal{I}_b^S, v_i, \mathsf{Out}_s)$, where $\mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}$ and $\mathsf{Out}_s = \{(\mathsf{vID}_i, \widetilde{s}_{j,i})\}_{\mathsf{vID}_i\in G_b, \mathsf{vID}_j\in G_b\setminus\mathcal{I}_b^S}$, we say $x'_i \in \mathcal{L}_R$ if there exists $w'_i = (\{s_{j,i}\}_{\mathsf{vID}_j\in G_b\setminus\mathcal{I}_b^S}, \mathsf{sk}_i)$ such that

- (vID_i, v_i) is an entry in Out'.
- $(\mathsf{pk}_i, \mathsf{sk}_i)$ is a valid key pair, $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \mathcal{P}$, and $\mathsf{vID}_i \in G_b$.
- For each $\mathsf{vID}_j \in G_b \setminus \mathcal{I}_b^S$, the $s_{j,i} = \mathsf{Dec}(\mathsf{sk}_i, \widetilde{s}_{j,i})$ is the correct decryption of $\widetilde{s}_{j,i}$, and $(\mathsf{vID}_i, \widetilde{s}_{j,i})$ is in x_j.
- $v_i = \sum_{j\in G_b\setminus\mathcal{I}_b^S} s_{j,i}$.

Throughout the protocol, we use two sets $\mathcal{I}_b^S$ to keep track of the parties who misbehaved in the committing and sharing phases and and $\mathcal{I}_b^R$ for those misbehaved in the reconstruction phase. The sub-protocols are parametrized with parameters t_0 and t_{oth} to be specified later.

Sub-protocol $\mathsf{GroupToss}^b[m, t_b, \mathcal{P}]$

Ingredients: A public-key encryption scheme $\Pi_{\mathrm{Enc}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

Inputs: Each party i holds its preference pref_i, authentication auth_i, and the aggregate information $(\mathsf{mvk}, n_0, n_{\mathsf{oth}})$ from the certificate authority.

Parameters: The masked profile $\mathcal{P} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{i\in[n]}$ where $\mathsf{pref}_i \in \{0, \mathsf{oth}, \perp\}$ and t_b is a threshold parameter. Every party initializes $\mathcal{I}_b^R$ and $\mathcal{I}_b^S$ as empty sets at the beginning of the protocol.

Protocol: If $|G_b| \le t_b$, every party outputs fail. Otherwise, proceeds as follows.

Sharing phase:

- Every $\mathsf{vID}_i \in G_b$ randomly chooses a secret $s_i \xleftarrow{\$} \{0, \dots, m-1\}$ and computes the shares $\{s_{i,j}\}_{j\in[n_b]}$ using (t_b+1)-out-of-n_b secret sharing. Without loss of generality, let the first $|G_b|$ shares be $\{s_{i,j}\}_{\mathsf{vID}_j\in G_b}$. Let $\widetilde{s}_{i,j} = \mathsf{Enc}(\mathsf{pk}_j, s_{i,j})$ for $\mathsf{vID}_j \in G_b$.
- Initialize $\mathcal{O} = [n]$. Repeat the following until succeed:

  Each $\mathsf{vID}_i \in G_b$ sets $M_i = (\mathsf{vID}_i, \{(\mathsf{vID}_j, \widetilde{s}_{i,j})\}_{\mathsf{vID}_j\in G_b})$ and all other parties $\mathsf{vID}_j \notin G_b$ sets $M_j = \mathbf{0}$ as dummy message. Every party vID_i sends M_i to $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$. If fails, receive a set $\mathcal{D}$ from $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$, set $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$ and retry.

  Let Out be the set of non-dummy messages successfully broadcast.
- Each party vID_i invokes $\mathsf{IdealZK}[x_i, \mathcal{L}_S, \mathsf{vID}_i, \mathsf{vID}_j]$ for each vID_j in $\mathcal{P}$, with statement $x_i = (\mathsf{Out}, \{(\mathsf{vID}_j, \widetilde{s}_{i,j})\}_{\mathsf{vID}_j\in G_b}, \mathcal{P}, G_b, t_b)$ and the witness $w_i = (s_i, \{s_{i,j}\}_{j\in[n_b]}, \mathsf{sk}_i)$ to prove that $x_i \in \mathcal{L}_S$.

  For any $\mathsf{vID}_i \in G_b$, if there exists a vID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_S, \mathsf{vID}_i, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}_b^S$.

  Let Out_s be the set of entries $(\mathsf{vID}_i, \{(\mathsf{vID}_j, \widetilde{s}_{i,j})\}_{\mathsf{vID}_j\in G_b})$ for $\mathsf{vID}_i \in G_b \setminus \mathcal{I}_b^S$ successfully broadcast with valid proofs in Out.

Reconstruction phase:

- Parse each entry in Out_s as $(\mathsf{vID}_j, \{\widetilde{s}_{i,j}\})_{\mathsf{vID}_j\in G_b}$. Each $\mathsf{vID}_i \in G_b$ computes $s_{j,i} = \mathsf{Dec}(\mathsf{sk}_i, \widetilde{s_{j,i}})$ and the sum of decrypted shares: $v_i = \sum_{j\in G_b\setminus\mathcal{I}_b^S} s_{j,i}$.
- Reset $\mathcal{O} = [n]$.[a] Parties repeat the following until succeed:

  Every $\mathsf{vID}_i \in G_b$ sets $M_i' = (\mathsf{vID}_i, v_i)$ and all other parties $\mathsf{vID}_j \notin G_b$ sets $M_j' = \mathbf{0}$ as dummy message. Every party vID_i sends M_i' to $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$. If fails, receive a set $\mathcal{D}$ from $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$, set $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$ and retry.

  Let Out' be the set of non-dummy messages successfully broadcast.
- Each party vID_i invokes $\mathsf{IdealZK}[x_i, \mathcal{L}_R, \mathsf{vID}_i, \mathsf{vID}_j]$ for each vID_j in $\mathcal{P}$, with statement $x_i' = (\mathsf{Out}', \mathcal{P}, G_b, \mathcal{I}_b^S, v_i, \mathsf{Out}_s)$ and witness $w_i' = (\{s_{j,i}\}_{\mathsf{vID}_j\in G_b\setminus\mathcal{I}_b^S}, \mathsf{sk}_i)$ to prove that $x_i' \in \mathcal{L}_R$.

  For any $\mathsf{vID}_i \in G_b$, if there exists a vID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_R, \mathsf{vID}_i, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}_b^R$.
- If $|\mathcal{I}_b^R| > n_b - t_b$, everyone outputs fail. Otherwise, parse each entry in Out' as (vID_i, v_i) and use the shares $\{v_i\}_{i\in G_b\setminus\mathcal{I}_b^R}$ to reconstruct the secret $c_b := \sum_{i\in G_b\setminus\mathcal{I}_b^S} s_i$.

[a] Parties can also continue with the $\mathcal{O}$ stored from the sharing phase. We reset here for simplicity.

While each b-supporter only shares its secret with other b-supporters, it proves the correctness of

its behavior to all parties, including non-b-supporters, by broadcasting the proof. Note that in the above protocol, all messages are broadcast through $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$ (Figure 2). Based on the description of the ideal functionality, the sets $\mathcal{I}_b^S$ and $\mathcal{I}_b^R$ can be uniquely determined from broadcast messages. These two sets are also publicly verifiable, and all honest parties hold the same $\mathcal{I}_b^S$ and $\mathcal{I}_b^R$.

4.2.2 Full Protocol Description

NP Language: We specify the following NP language $\mathcal{L}_{\text{com}}$ used in the commit phase: Given a statement $\widetilde{x}_i = (\widetilde{\mathsf{Out}}, \mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i, \mathsf{mvk})$ where $\mathsf{pref}_i \in \{0, \mathsf{oth}\}$, we say $\widetilde{x}_i \in \mathcal{L}_{\text{com}}$ if $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \widetilde{\mathsf{Out}}$ and

- For $\mathsf{pref}_i = 0$, there exists a witness $(\mathsf{auth}_i, \mathsf{sk}_i)$ such that auth_i is a valid signature of message $(\mathsf{vID}_i, 0)$ under verification key mvk, and sk_i is the valid secret key of pk_i;
- For $\mathsf{pref}_i = \mathsf{oth}$, there exists a witness $(\mathsf{auth}_i, \mathsf{sk}_i)$ such that auth_i is a valid signature of message (vID_i, ℓ) for $\ell \in \{1, ..., m-1\}$ under verification key mvk, and sk_i is the valid secret key of pk_i;

Protocol $\Pi_{n,m}[t_0, t_{\mathsf{oth}}]$

Authentication: Each party i samples $\mathsf{vID}_i \xleftarrow{\$} \{0,1\}^\lambda$ and sends $(\mathsf{vID}_i, \widetilde{\mathsf{pref}}_i)$ to $\mathsf{agg\text{-}auth}$ and gets auth_i, where party i's preference $\widetilde{\mathsf{pref}}_i \in \{0, ..., m-1\}$.

Inputs: If $\widetilde{\mathsf{pref}}_i = 0$, set $\mathsf{pref}_i = 0$; otherwise, set $\mathsf{pref}_i = \mathsf{oth}$. Each party holds $(\mathsf{vID}_i, \mathsf{pref}_i, \mathsf{auth}_i)$ and aggregate information $(\mathsf{mvk}, n_0, n_{\mathsf{oth}})$ from $\mathsf{agg\text{-}auth}$.

Commit phase:

1. Each party i samples a pair of $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Pi_{\text{Enc}}.\mathsf{Gen}(1^\lambda)$.
2. Initialize $\mathcal{O} = [n]$. Repeat the following until the broadcast succeed:

   Every party sends $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ to $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$. If fails, receive a set $\mathcal{D}$ from $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$, set $\mathcal{O} = \mathcal{O} \backslash \mathcal{D}$ and retry.

   Let $\widetilde{\mathsf{Out}}$ be the set of successfully broadcast messages.
3. Each party i invokes $\mathsf{IdealZK}[\widetilde{x}_i, \mathcal{L}_{\text{com}}, \mathsf{vID}_i, j]$ for each j in $[n]$, with $\widetilde{x}_i = (\widetilde{\mathsf{Out}}, \mathsf{vID}_i, \mathsf{pref}_i, \mathsf{pk}_i, \mathsf{mvk})$ and witness $\widetilde{w}_i = (\mathsf{auth}_i, \mathsf{sk}_i)$.

   For any vID_i appeared in $\widetilde{\mathsf{Out}}$, if there exists a j such that $\mathsf{IdealZK}[\widetilde{x}_i, \mathcal{L}_{\text{com}}, \mathsf{vID}_i, j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}^C$.
4. Let $\mathcal{P}_O$ be the preferences with a valid proof in $\widetilde{\mathsf{Out}}$:

$$\mathcal{P}_O = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \widetilde{\mathsf{Out}} : \mathsf{vID}_i \notin \mathcal{I}^C\}.$$

5. Let $n' = |\mathcal{P}_O|$. Let $\mathcal{P}_A = \{\mathsf{vID}'_k, \perp, \perp\}_{k \in [n-n']}$, where vID'_k is the $(\lambda+1)$-bits presentation of k. Let $\mathcal{P} = \mathcal{P}_O \cup \mathcal{P}_A$, and G_0 and G_{oth} as defined in Equation (1).

Sharing phase:

1. *Share*-0*:* Group G_0 runs the sharing phase of $\mathsf{GroupToss}^0[m, t_0, \mathcal{P}]$.
2. *Share*-oth*:* Group G_{oth} runs the sharing phase of $\mathsf{GroupToss}^{\mathsf{oth}}[m, t_{\mathsf{oth}}, \mathcal{P}]$.

Reconstruction phase:

1. *Recon-0:* Group G_0 runs the reconstruction phase of $\mathsf{GroupToss}^0[m, t_0, \mathcal{P}]$. If the sub-protocol outputs fail, set $c_0 = 0$.
2. *Share-*oth*:* Group G_{oth} runs the reconstruction phase of $\mathsf{GroupToss}^{\mathsf{oth}}[m, t_{\mathsf{oth}}, \mathcal{P}]$. If the sub-protocol outputs fail, the protocol outputs 0. Otherwise, output $(c_0 + c_{\mathsf{oth}}) \mod m$.

Below we use k_0 and k_{oth} to denote the number of corrupted 0- and oth-supporters, respectively.

Constraints 4.3. *The threshold k_0 and k_{oth}, and the number of corrupted parties $k = k_0 + k_{\mathsf{oth}}$ the protocol can tolerate is chosen such that the following three constraints are satisfied.*

(C1) The corrupted coalition cannot control both coins: $k_0 + k_{\mathsf{oth}} \leq t_0 + t_{\mathsf{oth}} + 1$.

(C2) If the coalition can control coin c_{oth}, then the coalition cannot fail the reconstruction of coin c_0. Equivalently, if $k_{\mathsf{oth}} \geq t_{\mathsf{oth}} + 1$, then $k_0 < n_0 - t_0$.

(C3) If the coalition can fail the reconstruction of c_{oth}, then it must not prefer 0. That is, if $k_{\mathsf{oth}} \geq n_{\mathsf{oth}} - t_{\mathsf{oth}}$, then $k_0 \leq \frac{k_0 + k_{\mathsf{oth}}}{m}$.

Theorem 4.4. *If the encryption scheme Π_{Enc} is semantically secure, then the above $\Pi_{n,m}[t_0, t_{\mathsf{oth}}]$ securely realizes $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, n_0, n_{\mathsf{oth}}]$ in the $(\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}, \mathsf{IdealZK})$-hybrid world as long as the thresholds t_0, t_{oth}, and the number of corrupted parties $k = k_0 + k_{\mathsf{oth}}$ satisfy Constraints 4.3.*

Proof. We will show that for any non-uniform PPT $\mathcal{A}$ that controls k_0 number of 0-supporter and k_{oth} number of oth-supporter interacting with Π_{toss}, where k_0 and k_{oth} satisfy Constraints 4.3, there exists an adversary Sim interacting with $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, n_0, n_{\mathsf{oth}}]$ such that $\mathcal{A}$'s view in an execution with Π_{toss} is computationally indistinguishable from its view simulated by Sim.

Commit phase: Simulator receives $\mathsf{Leak}(\mathcal{P}_{\mathcal{H}}) = (h_0, h_{\mathsf{oth}})$ from $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}$. Emulate honest parties $i \in \mathcal{H}$ as follows:

- Sample $\mathsf{vID}_i \xleftarrow{\$} \{0,1\}^\lambda$ and key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for each honest party. If there exist duplicate virtual IDs for honest parties, the simulator aborts.

  Let $\mathcal{H}$ be the set of honest virtual IDs. Set $\mathsf{pref}_i = 0$ for the first h_0 number of honest parties and $\mathsf{pref}_i = \mathsf{oth}$ for the remaining honest parties. Let $\mathcal{P}_{\mathcal{H}} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}_{\mathsf{vID}_i \in \mathcal{H}}$.

- Initialize $\mathcal{O}$ as $[n]$ and emulate $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$ until it succeeds as follows:

  Receive $\mathcal{P}_{\mathcal{K}} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}$ from $\mathcal{A}$. Send a multiset $\mathcal{P}_{\mathcal{H}} \cup \mathcal{P}_{\mathcal{K}}$ to $\mathcal{A}$. If received a set $\mathcal{D}$ of size at least t, update $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$ and retry.

  Let $\widetilde{\mathsf{Out}}$ be the set of messages that are successfully broadcast.

- Emulate the $\mathsf{IdealZK}[*, \mathcal{L}_{\mathrm{com}}, *, *]$ as follows:
  - For honest prover vID_i and honest verifier i', send $(\mathsf{success}, \mathsf{vID}_i, i')$ to $\mathcal{A}$;
  - For honest prover vID_i and corrupted verifier j: If received $\perp$ from a corrupted verifier, send (fail, j) to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, j)$ to $\mathcal{A}$;
  - For corrupt prover vID_j and honest verifier i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness $\widetilde{w}_j$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
  - For corrupted prover vID_j and corrupted verifier j', receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

For any vID_i appeared in $\widetilde{\mathsf{Out}}$, if there exists a j such that $\mathsf{IdealZK}[\widetilde{x}_i, \mathcal{L}_{\mathrm{com}}, \mathsf{vID}_i, j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}^C$.

Let $\mathcal{K}$ be the set of corrupted virtual IDs in $\mathcal{P}_{\mathcal{K}}$ but not in $\mathcal{I}^C$. At this moment, for any corrupted $\mathsf{vID}_j \in \mathcal{K}$, the simulator receives a valid witness $\widetilde{w}_j = (\mathsf{auth}_j, \mathsf{sk}_j)$. If $\mathcal{K} \cap \mathcal{H} \neq \emptyset$, i.e., a corrupted virtual ID collides with an honest virtual ID, the simulator aborts.

- Let $\mathcal{P}'_{\mathcal{K}}$ be the corrupted preferences with a valid proof in $\widetilde{\mathsf{Out}}$: $\mathcal{P}'_{\mathcal{K}} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \mathcal{P}_{\mathcal{K}} : \mathsf{vID}_i \in \mathcal{K}\}$. Let $n' = |\mathcal{P}_{\mathcal{H}} \cup \mathcal{P}'_{\mathcal{K}}|$. Let $\mathcal{P}_A = \{\mathsf{vID}'_k, \bot, \bot\}_{k \in [n-n']}$, where vID'_k is the $(\lambda+1)$-bits presentation of k. Let $\mathcal{P} = \mathcal{P}_{\mathcal{H}} \cup \mathcal{P}'_{\mathcal{K}} \cup \mathcal{P}_A$, and G_0 and G_{oth} as defined in Equation (1).
- For any $\mathsf{vID}_j \in \mathcal{K}$, send $(\mathsf{vID}_j, \mathsf{pk}_j, \mathsf{pref}_j, \mathsf{auth}_j)$ to $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$; for other corrupted parties, send $(\mathsf{vID}'_j, \bot, \bot, \bot)$ to $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$.

The simulator gets a coin c from the ideal functionality. At this point, the simulator has a preference profile $\mathcal{P}$ of size n where each $\mathsf{pref}_i \in \{0, \mathsf{oth}, \bot\}$. For all corrupted parties with non-bot preferences, the simulator gets the corresponding secret key sk_j from the extraction algorithm.

Let $\mathcal{H}_0$ be the set of first h_0 number of honest parties whose preferences are 0 and $\mathcal{H}_{\mathsf{oth}}$ be the set of remaining honest parties whose preferences are oth. Let $\mathcal{K}_b$ to denote the set of honest and corrupted b-supporters for $b \in \{0, \mathsf{oth}\}$ whose preference is b in $\mathcal{P}$, respectively. At a high-level, those who have $\bot$-preferences are treated as if they aborted at the beginning of the sharing phase.

Sharing phase: The simulator computes $k_b = n_b - h_b$ for $b \in \{0, \mathsf{oth}\}$, where h_b is the number of honest b-supporters in $\mathcal{P}_{\mathcal{H}}$. Recall that $\mathcal{I}^S_b$ and $\mathcal{I}^R_b$ records the misbehaved b-supporters in the sharing phase and the reconstruction phase, respectively, for $b \in \{0, \mathsf{oth}\}$.

Share-b: Emulate honest parties to simulate execution of the sharing phase in sub-protocol $\mathsf{GroupToss}^b[m, t_b, \mathcal{P}]$. If $|G_b| \leq t_b$, outputs fail for $\mathsf{GroupToss}^b[m, t_b, \mathcal{P}]$.

If $\mathsf{GroupToss}^0[m, t_0, \mathcal{P}]$ outputs fail, set $c_0 = 0$ and proceed to simulate Share-oth step; if $\mathsf{GroupToss}^{\mathsf{oth}}[m, t_{\mathsf{oth}}]$ outputs fail, send no to $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$. Outputs whatever the adversary $\mathcal{A}$ outputs.

Otherwise, proceeds as follows:

- *Sharing:* If $k_b < t_b + 1$, for any $j \in \mathcal{K}_b$, uniformly randomly choose $s_{i,j}$; for any $\mathsf{vID}_{i'} \in \mathcal{H}_b$, set $s_{i,i'} = 0$. Otherwise, chooses a random coin s_i and compute the shares $\{s_{i,j}\}_{j \in [n_b]}$ using (t_b+1)-out-of-n_b Shamir secret sharing.

  For $\mathsf{vID}_j \in G_b$, compute $\widetilde{s}_{i,j} = \mathsf{Enc}(\mathsf{pk}_j, s_{i,j})$.

- *Broadcast:* Emulate $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ until it succeeds as follows:

  For honest party vID_i, set $M_i = (\mathsf{vID}_i, \{(\mathsf{vID}_j, \widetilde{s}_{i,j})\}_{\mathsf{vID}_j \in G_b})$ for $\mathsf{vID}_i \in \mathcal{H}_b$ and $M_i = \mathbf{0}$ for $\mathsf{vID}_i \in \mathcal{H} \setminus \mathcal{H}_b$. Let multiset $\mathcal{M}_{\mathcal{H}} = \{M_i\}_{\mathsf{vID}_i \in \mathcal{H}}$. Receive $\mathcal{M}_{\mathcal{K}} = \{M_j\}$ from $\mathcal{A}$ for corrupted parties. Send the multiset $\mathcal{M}_{\mathcal{H}} \cup \mathcal{M}_{\mathcal{K}}$ to $\mathcal{A}$. If received a set $\mathcal{D}$ from $\mathcal{A}$, update $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$ and retry.

- *Validation:* Emulate the $\mathsf{IdealZK}[*, \mathcal{L}_S, *, *]$ as follows:
  - For honest prover vID_i and honest verifier $\mathsf{vID}_{i'}$, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_{i'})$ to $\mathcal{A}$;
  - For honest prover vID_i and corrupted verifier vID_j: If received $\bot$ from a corrupted verifier, send $(\mathsf{fail}, \mathsf{vID}_j)$ to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_j)$ to $\mathcal{A}$;
  - For corrupt prover vID_j and honest verifier vID_i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\bot$ or witness $\widetilde{w}_j$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
  - For corrupted prover vID_j and corrupted verifier $\mathsf{vID}_{j'}$, receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

For any $\mathsf{vID}_i \in G_b$, if there exists a vID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_S, \mathsf{vID}_i, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}_b^S$.

Reconstruction phase: At this point, for any $\mathsf{vID}_j \in \mathcal{K}_b \setminus \mathcal{I}_b^S$ for $b \in \{0, \mathsf{oth}\}$, the simulator receives a witness from emulating $\mathsf{IdealZK}$, including the secret s_j and shares for honest parties $\{s_{j,i}\}_{i \in \mathcal{H}_b}$.

Recon-0: Emulate honest parties for the reconstruction phase in sub-protocol $\mathsf{GroupToss}^0[m, t_0, \mathcal{P}]$:

1. *Summation:* The simulator computes the reconstruction share for the honest parties $\mathsf{vID}_i \in \mathcal{H}_0$ as follows:
   - If $k_0 \geq t_0 + 1$: simulator has the shares $s_{j,i}$ for all $j \in G_0 \setminus \mathcal{I}_0^S$ and $i \in \mathcal{H}_0$ from the sharing phase. It computes $v_i = \sum_{j \in G_0 \setminus \mathcal{I}_0^S} s_{j,i}$.
   - Otherwise if $k_0 < t_0 + 1$ and $k_{\mathsf{oth}} \geq t_{\mathsf{oth}} + 1$: simulator has the secret s_j for all $j \in (G_0 \cup G_{\mathsf{oth}}) \setminus (\mathcal{I}^S \cup \mathcal{H}_0)$. The simulator chooses $\{s_i\}_{i \in \mathcal{H}_0}$ such that $\sum_{i \in (G_0 \cup G_{\mathsf{oth}}) \setminus \mathcal{I}^S} s_i = c \mod m$. For every $i \in \mathcal{H}_0$, compute a (t_0+1)-out-of-n_0 secret sharing $\{y_{i,j}\}$ of s_i such that for any $j \in \mathcal{K}_0$, the share $y_{i,j} = s_{i,j}$. Set $v_i = \sum_{j \in \mathcal{K}_0 \setminus \mathcal{I}_0^S} s_{j,i} + \sum_{j \in \mathcal{H}_0} y_{j,i}$.
   - Otherwise if $k_0 < t_0 + 1$ and $k_{\mathsf{oth}} < k_{\mathsf{oth}} + 1$: Choose a uniformly random s_i for each $i \in \mathcal{H}_0$ and compute a (t_0+1)-out-of-n_0 secret sharing $\{y_{i,j}\}$ of each s_i such that for any $j \in \mathcal{K}_0$, the share $y_{i,j} = s_{i,j}$. Set $v_i = \sum_{j \in \mathcal{K}_0 \setminus \mathcal{I}_0^S} s_{j,i} + \sum_{j \in \mathcal{H}_0} y_{j,i}$.
2. *Broadcast:* Emulate $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O},t}$ until it succeeds as follows:

   Set $M_i' = (\mathsf{vID}_i, v_i)$ for $\mathsf{vID}_i \in \mathcal{H}_0$ and $M_i' = \mathbf{0}$ for $\mathsf{vID}_i \in \mathcal{H}_{\mathsf{oth}}$. Let $\mathcal{M}_{\mathcal{H}}' = \{M_i'\}_{\mathsf{vID}_i \in \mathcal{H}}$. Receive $\mathcal{M}_{\mathcal{K}}' = \{M_j')0\}$ from $\mathcal{A}$ for corrupted parties. Send the multiset $\mathcal{M}_{\mathcal{K}}' \cup \mathcal{M}_{\mathcal{H}}'$ to $\mathcal{A}$. If received a set $\mathcal{D}$ from $\mathcal{A}$, update $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$ and retry.
3. *Validation:* Emulate the $\mathsf{IdealZK}[*, \mathcal{L}_R, *, *]$ as follows:
   - For honest prover vID_i and honest verifier $\mathsf{vID}_{i'}$, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_{i'})$ to $\mathcal{A}$;
   - For honest prover vID_i and corrupted verifier vID_j: If received $\perp$ from a corrupted verifier, send $(\mathsf{fail}, \mathsf{vID}_j)$ to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_j)$ to $\mathcal{A}$;
   - For corrupt prover vID_j and honest verifier vID_i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness $\widetilde{w}_j$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
   - For corrupted prover vID_j and corrupted verifier $\mathsf{vID}_{j'}$, receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

   For any $\mathsf{vID}_i \in G_0$, if there exists a vID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_R, \mathsf{vID}_i, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}_0^R$.
4. *Reconstruction:* Reconstruct c_0 using $\{v_i\}_{i \in G_0 \setminus \mathcal{I}_0^R}$. If fails, set $c_0 = 0$; otherwise, record the value of c_0.

Recon-oth: Emulate honest parties to simulate execution of the reconstruction phase in sub-protocol $\mathsf{GroupToss}^{\mathsf{oth}}[m, t_{\mathsf{oth}}, \mathcal{P}]$:

1. *Summation:* The simulator computes the reconstruction share for the honest parties:
   - If $k_{\mathsf{oth}} \geq t_{\mathsf{oth}} + 1$: then it must be that $k_0 < t_0 + 1$ according to Constraints 4.3. The simulator Sim would have $s_{j,i}$ for all $j \in G_{\mathsf{oth}} \setminus \mathcal{I}_{\mathsf{oth}}^S$, and it is guaranteed that $\sum_{i \in (G_0 \cup G_{\mathsf{oth}}) \setminus \mathcal{I}^S} s_i = c \mod m$. It computes $v_i = \sum_{j \in G_{\mathsf{oth}} \setminus \mathcal{I}_{\mathsf{oth}}^S} s_{j,i}$.

- Otherwise, choose s_i for each $i \in \mathcal{H}_{\mathsf{oth}}$ such that $c_0 + (\sum_{i \in G_{\mathsf{oth}} \setminus \mathcal{I}^S_{\mathsf{oth}}} s_i) = c \mod m$. Compute a $(t_{\mathsf{oth}}+1)$-out-of-n_{oth} secret sharing $\{y_{i,j}\}$ of s_i such that for any $j \in \mathcal{K}_{\mathsf{oth}}$, the share $y_{i,j} = s_{i,j}$. Set $v_i = \sum_{j \in \mathcal{K}_{\mathsf{oth}} \setminus \mathcal{I}^S_{\mathsf{oth}}} s_{j,i} + \sum_{j \in \mathcal{H}_{\mathsf{oth}}} y_{j,i}$.

2. *Broadcast:* Emulate $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ until it succeeds as follows:

   Set $M'_i = (\mathsf{vID}_i, v_i)$ for $\mathsf{vID}_i \in \mathcal{H}_0$ and $M'_i = \mathbf{0}$ for $\mathsf{vID}_i \in \mathcal{H}_{\mathsf{oth}}$. Let $\mathcal{M}'_{\mathcal{H}} = \{M'_i\}_{\mathsf{vID}_i \in \mathcal{H}}$. Receive $\mathcal{M}'_{\mathcal{K}} = \{M'_j)0\}$ from $\mathcal{A}$ for corrupted parties. Send the multiset $\mathcal{M}'_{\mathcal{K}} \cup \mathcal{M}'_{\mathcal{H}}$ to $\mathcal{A}$. If received a set $\mathcal{D}$ from $\mathcal{A}$, update $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$ and retry.

3. *Validation:* Emulate the $\mathsf{IdealZK}[*, \mathcal{L}_R, *, *]$ as follows:

   - For honest prover vID_i and honest verifier $\mathsf{vID}_{i'}$, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_{i'})$ to $\mathcal{A}$;
   - For honest prover vID_i and corrupted verifier vID_j: If received $\perp$ from a corrupted verifier, send $(\mathsf{fail}, \mathsf{vID}_j)$ to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_j)$ to $\mathcal{A}$;
   - For corrupt prover vID_j and honest verifier vID_i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness $\widetilde{w}_j$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
   - For corrupted prover vID_j and corrupted verifier $\mathsf{vID}_{j'}$, receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

   For any $\mathsf{vID}_i \in G_{\mathsf{oth}}$, if there exists a vID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_R, \mathsf{vID}_i, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, add vID_i to $\mathcal{I}^R_{\mathsf{oth}}$.

4. *Reconstruction:* For those corrupted parties vID_j that failed to broadcast a valid proof, add vID_j to $\mathcal{I}^R_{\mathsf{oth}}$. Reconstruct c_{oth} using $\{v_i\}_{i \in G_{\mathsf{oth}} \setminus \mathcal{I}^R_{\mathsf{oth}}}$. If reconstruction succeeds, send yes to $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$. Otherwise, send no to $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$.

   Output whatever $\mathcal{A}$ outputs.

We first show that the probability that the simulator aborts is negligible.

Claim 4.5. *There exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the probability that the simulator* S *aborts is at most* $\mathsf{negl}(\lambda)$*.*

Proof. The simulator aborts if either 1.) It samples duplicate virtual IDs for honest parties; 2.) If any corrupted virtual ID vID_j conflicts with an honest virtual ID, i.e., if $\mathcal{K} \cap \mathcal{H} \neq \emptyset$. We now compute the probability of these three two separately.

The probability that the simulator samples duplicate virtual ID is $1 - \binom{2^\lambda}{h} \cdot h!/2^{\lambda \cdot h}$, which is negligible.

In the rest of the proof, we focus on proving that the second event happens with a negligible probability. Let $\mathcal{K}\mathsf{id}$ be the set of corrupted virtual IDs signed by the certificate authority. Observe that $\Pr[\mathcal{H} \cap \mathcal{K}\mathsf{id} \neq \emptyset]$ is negligible. We only need to show that conditioned on $\mathcal{H} \cap \mathcal{K}\mathsf{id} = \emptyset$, for any non-uniform PPT $\mathcal{A}$, the probability that $\mathcal{K} \cap \mathcal{H} \neq \emptyset$ is negligible.

Suppose for the sake of contradiction that there exists a non-uniform $\mathcal{A}$ with a non-negligible probability resulting in some $\mathsf{vID}^* \in \mathcal{K} \cap \mathcal{H}$, conditioned on $\mathcal{H} \cap \mathcal{K}\mathsf{id} = \emptyset$. This implies that the adversary is able to produce a valid auth_i for $(\mathsf{vID}^*, *)$ for $\mathsf{vID}^* \notin \mathcal{K}\mathsf{id}$ to send to $\mathsf{IdealZK}$, which breaks the existential unforgeability of signature scheme. Putting together,

$$
\begin{aligned}
&Pr[\mathcal{H} \cap \mathcal{K} \neq \emptyset] \\
=&\Pr[\mathcal{H} \cap \mathcal{K} \neq \emptyset \mid \mathcal{H} \cap \mathcal{K}\mathsf{id} = \emptyset] \cdot \Pr[\mathcal{H} \cap \mathcal{K}\mathsf{id} = \emptyset] + \Pr[\mathcal{H} \cap \mathcal{K} \neq \emptyset \mid \mathcal{H} \cap \mathcal{K}\mathsf{id} \neq \emptyset] \cdot \Pr[\mathcal{H} \cap \mathcal{K}\mathsf{id} \neq \emptyset] \\
=&\mathsf{negl}(\lambda).
\end{aligned}
$$

Therefore, the probability that the simulator aborts is negligible. □

We now show that the joint distribution of honest parties' output and the adversary's view is indistinguishable in the ideal execution when interacting with Sim (denoted as $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A}}$) is identical to the joint distribution of the output of honest parties and the view of the adversary in a real execution (denoted as $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}$), conditioned on the simulator not abort. Consider the following hybrids:

Hybrid Hyb_1: This is an execution of $\Pi_{n,m}[t_0, t_{\mathsf{oth}}]$, where the simulator acts on behalf of all honest parties and interact with the adversary. Moreover, the simulator emulates all the $\mathsf{IdealZK}$ instances for the adversary as follows:

- For honest prover i and honest verifier i', send $(\mathsf{success}, i, i')$ to $\mathcal{A}$;
- For honest prover i and corrupted verifier j: If received $\perp$ from a corrupted verifier, send (fail, j) to $\mathcal{A}$; otherwise, send $(\mathsf{success}, i, j)$ to $\mathcal{A}$;
- For corrupt prover j and honest verifier i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness $\widetilde{w}_j$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
- For corrupted prover j and corrupted verifier j', receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

We have $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}} \equiv \mathsf{Hyb}_1$ by definition.

Hybrid Hyb_2: The simulator behaves almost the same as in Hyb_1, except that now the simulator receives leakage $\mathsf{Leak} = (h_0, h_{\mathsf{oth}})$ from the ideal functionality and randomly sample virtual IDs for each honest parties. We have $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_1$.

Hybrid Hyb_3: The simulator behaves almost the same as in Hyb_4 except the following: In Share-b step, the simulator computes the shares for honest parties $\mathsf{vID}_i \in \mathcal{H}_b$ as in Hyb_4, but the encryptions $\{\widetilde{s}_{i,j}\}_{\mathsf{vID}_j \in G_b}$ are computed differently. For $b \in \{0, \mathsf{oth}\}$,

- For honest $\mathsf{vID}_i \in \mathcal{H}_b$ and corrupted $\mathsf{vID}_j \in \mathcal{K}_b$, let $\widetilde{s}_{i,j} = \mathsf{Enc}(\mathsf{pk}_j, s_{i,j})$.
- For honest $\mathsf{vID}_i \in \mathcal{H}_b$ and $\mathsf{vID}_{i'} \in \mathcal{H}_b$, let $\widetilde{s}_{i,j} = \mathsf{Enc}(\mathsf{pk}_j, 0)$.

We have $\mathsf{Hyb}_3 \approx \mathsf{Hyb}_2$ by semantic security of public-key encryption scheme.

Next, we show that $\mathsf{Hyb}_3 \equiv \mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A}}$, which concludes the proof. By security of Shamir's secret sharing, the adversary's view in both experiments are identical. We only need to show that for any fixed adversary's view, honest parties' outputs are also identical in both experiments.

By constraint (C3), if the reconstruction of c_{oth} fails, and Sim sends no to $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$. Then $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$ uses $k_0 = n_0 - h_0$ and $k_{\mathsf{oth}} = n_{\mathsf{oth}} - h_{\mathsf{oth}}$ to determine the lexicographically smallest outcome that the coalition does not like, which is 0. Therefore, if the reconstruction of c_{oth} fails, honest parties' output will be 0 in both experiments.

In the following, we will only focus on proving that if the reconstruction of c_{oth} succeeds, the honest parties' output in both worlds will be c, the coin generated by $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}$. There are two cases:

- If $t_{\mathsf{oth}} \geq k_{\mathsf{oth}} + 1$, by constraints (C1) and (C2), it must be the case that $t_0 < k_0 + 1$ and c_0 must be successfully reconstructed. Therefore, by the description of *Recon-0* step, $\{s_i\}_{i \in \mathcal{H}_0}$ are chosen such that $(\sum_{i \in (G_{\mathsf{oth}} \cup G_0) \setminus \mathcal{I}^S} s_i) \mod m = c$. Therefore, if both c_0 and c_{oth} are successfully reconstructed, honest parties' output should be c.
- If $t_{\mathsf{oth}} < k_{\mathsf{oth}} + 1$, by the description of the *Recon-*oth step, $\{s_i\}_{i \in \mathcal{H}_{\mathsf{oth}}}$ are chosen such that $(c_0 + \sum_{j \in G_{\mathsf{oth}} \setminus \mathcal{I}^S_{\mathsf{oth}}} s_j) \mod m = c$. If the reconstruction of c_{oth} succeeds, the honest parties will output c in both experiments.

Therefore, by hybrid argument, $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A}} \approx \mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}$. □

Previous work [TSW24] presented the optimal solutions of t_0, t_{oth} and k that satisfies Constraints 4.3. The optimal parameters are given in the table below. For $m = 2$, the coalition tolerance matches the optimal tolerance in [WAS22].

Case	k_0	t_{oth}	t
If $n_{\mathsf{oth}} \geq \frac{4m-3}{2} n_0$	$\lfloor \frac{n_0}{2} \rfloor$	$n_{\mathsf{oth}} - (m-1)n_0$	$\lceil \frac{n_0}{2} \rceil + n_{\mathsf{oth}} - (m-1)n_0$
Otherwise	$\lfloor \frac{n_0}{2} \rfloor$	$\left\lfloor \frac{m}{2m-1} n_{\mathsf{oth}} - \frac{m-1}{2(2m-1)} n_0 \right\rfloor$	$\lceil \frac{n_0}{2} \rceil + \left\lfloor \frac{m}{2m-1} n_{\mathsf{oth}} - \frac{m-1}{2(2m-1)} n_0 \right\rfloor$

Table 1: Optimal parameter choice for m-sided coin-toss Π instantiating $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}$. Here, n_b denotes the number of b-supporters for $b \in \{0, \mathsf{oth}\}$ receiving certification from the certified authority.

References

[ACH11] Gilad Asharov, Ran Canetti, and Carmit Hazay. Towards a game theoretic view of secure computation. In *Eurocrypt*, 2011. 1.3

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. Cryptology ePrint Archive, Paper 2017/402, 2017. 1.3

[ADGH06] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Halpern. Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In *PODC*, 2006. 1.3

[ADMM16] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and undefinedukasz Mazurek. Secure multiparty computations on bitcoin. *Commun. ACM*, 59(4):76–84, March 2016. 1.3

[AL11] Gilad Asharov and Yehuda Lindell. Utility dependence in correct and fair rational secret sharing. *Journal of Cryptology*, 24(1), 2011. 1.3

[ATM+22] Lukas Aumayr, Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, Pedro Moreno-Sanchez, and Matteo Maffei. Sleepy channels: Bi-directional payment channels without watchtowers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 179–192, New York, NY, USA, 2022. Association for Computing Machinery. 1.3

[BGJ+18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal mpc. In *Annual International Cryptology Conference*, pages 459–487. Springer, 2018. 1.3

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988. 1

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, page 645–677, Berlin, Heidelberg, 2017. Springer-Verlag. 1.3

[BK14] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *CRYPTO*, pages 421–439, 2014. 1.3

[Blu81] Manuel Blum. Coin flipping by telephone. In *CRYPTO*, 1981. 1

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 503–513, New York, NY, USA, 1990. Association for Computing Machinery. 1.3

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001. 1.1.2

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988. 1

[CCWS21] Kai-Min Chung, T.-H. Hubert Chan, Ting Wen, and Elaine Shi. Game-theoretic fairness meets multi-party protocols: The case of leader election. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 3–32, Cham, 2021. Springer International Publishing. 1, 1.3

[CGL+18a] Kai-Min Chung, Yue Guo, Wei-Kai Lin, Rafael Pass, and Elaine Shi. Game theoretic notions of fairness in multi-party coin toss. In *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part I*, page 563–596, Berlin, Heidelberg, 2018. Springer-Verlag. 1, 1.1.2, 3.2, B

[CGL+18b] Kai-Min Chung, Yue Guo, Wei-Kai Lin, Rafael Pass, and Elaine Shi. Game theoretic notions of fairness in multi-party coin toss. In *TCC*, 2018. 1.3

[CGL+18c] Kai-Min Chung, Yue Guo, Wei-Kai Lin, Rafael Pass, and Elaine Shi. Game theoretic notions of fairness in multi-party coin toss. In *TCC*, 2018. 3.2, B.1

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988. 2

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 364–369, 1986. 1

[CMST22] Hao Chung, Elisaweta Masserova, Elaine Shi, and Sri AravindaKrishnan Thyagarajan. Rapidash: Foundations of side-contract-resilient fair exchange. Cryptology ePrint Archive, Paper 2022/1063, 2022. https://eprint.iacr.org/2022/1063. 1.3

[DEF18] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 967–984, New York, NY, USA, 2018. Association for Computing Machinery. 1.3

[DGPS22] Moumita Dutta, Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Compute, but verify: Efficient multiparty computation over authenticated inputs. Cryptology ePrint Archive, Paper 2022/1648, 2022. 1.1.1, 1.3

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *ACM symposium on Theory of computing (STOC)*, 1987. 1, 1.1.2, 1.3

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 695–704, 2011. 1.3

[HHPV21] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. *Journal of Cryptology*, 34(3):19, 2021. 1.3

[HT04] Joseph Halpern and Vanessa Teague. Rational secret sharing and multiparty computation. In *STOC*, 2004. 1.3

[IKSS21] Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. On the round complexity of black-box secure mpc. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*, pages 214–243. Springer, 2021. 1.3

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 572–591, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 1.3

[KB14] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 30–41. ACM, 2014. 1.3

[KMS+16] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 839–858. IEEE Computer Society, 2016. 1.3

[KMSW22] Ilan Komargodski, Shin'ichiro Matsuo, Elaine Shi, and Ke Wu. log*-round game-theoretically-fair leader election. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 409–438, Cham, 2022. Springer Nature Switzerland. 1, 1.3, 2, 4.2

[KN08] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In *TCC*, 2008. 1.3

[KVV16] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *ACM CCS*, 2016. 1.3

[Lin16] Yehuda Lindell. How to simulate it - a tutorial on the simulation proof technique. Cryptology ePrint Archive, Paper 2016/046, 2016. 1.1.2

[OPRV09] Shien Jin Ong, David C. Parkes, Alon Rosen, and Salil P. Vadhan. Fairness with an honest minority and a rational majority. In *TCC*, 2009. 1.3

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, 2004. 1.1.3, 1.3, 2, 4.1, 4.1

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85. ACM, 1989. 1

[RCCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part II 18*, pages 291–319. Springer, 2020. 1.3

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. 2

[TSW24] Sri AravindaKrishnan Thyagarajan, Pratik Soni, and Ke Wu. Game-theoretically fair distributed sampling. In *Annual International Cryptology Conference*, pages 207–239. Springer, 2024. 1, 1.1.4, 1.1.4, 1.3, 2, 2, 2, 3.2, 4, 4.2.2

[WAS21] Ke Wu, Gilad Asharov, and Elaine Shi. A complete characterization of game-theoretically fair, multi-party coin toss. *IACR Cryptol. ePrint Arch.*, page 748, 2021. 1, 1, 2

[WAS22] Ke Wu, Gilad Asharov, and Elaine Shi. A complete characterization of game-theoretically fair, multi-party coin toss. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 120–149, Cham, 2022. Springer International Publishing. 1.1.4, 1.3, 2, 3.2, 4.2.2

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 531–540, 2010. 1.3

[Yao82] Andrew C Yao. Protocols for secure computations. In *FOCS*, 1982. 1, 1.1.2, 1.3

Supplementary Materials

A Preliminary

A.1 Shamir Secret Sharing.

A k-out-of-n secret sharing scheme consists of two algorithms Share and $\mathsf{Reconstruct}$

- $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s)$: On input a secret s, outputs n shares $s_1, \ldots, s_n$ of the secret.
- $s \leftarrow \mathsf{Reconstruct}(I, (s_i)_{i \in I})$: The reconstruction algorithm takes $(s_i)_{i \in I}$ and reconstructs the secret only if $|I| \geq k$. Otherwise, the reconstruction fails and output fail.

A k-out-of-n Shamir secret sharing scheme satisfies the following properties:

- **Correctness**: For any secret s and any sets $I \subset \{1, \ldots, n\}$ such that $|I| \geq k$,

$$\Pr[(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s) : \mathsf{Reconstruct}(I, (s_i)_{i \in I}) = s] = 1.$$

- **Security**: For any secrets s and s', for any sets I such that $I \subset \{1, \ldots, n\}$, and $|I| \leq k - 1$,

$$\{(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s) : (s_i)_{i \in I}\} \approx \{(s'_1, \ldots, s'_n) \leftarrow \mathsf{Share}(s') : (s'_i)_{i \in I}\}.$$

  Informally, this means that if a coalition has only $k-1$ shares, then they learn no information about the secret.

- **Linearity**: For any secrets s and s', for any sets $I \subset \{1, \ldots, n\}$ such that $|I| \geq k$, the following holds:

$$\Pr\left[\begin{array}{l}(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s), \\ (s'_1, \ldots, s'_n) \leftarrow \mathsf{Share}(s')\end{array} : \mathsf{Reconstruct}(I, (s_i + s'_i)_{i \in I}) = s + s'\right] = 1.$$

  Roughly speaking, this means that one can reconstruct $s + s'$ using the sum of the shares of s and s'.

In this work, we consider Shamir's secret sharing on a finite field $\mathbb{F}_q$ for some prime q. Specifically, we will consider a finite field of size $q > n \cdot m$ in out m-sided coin-toss protocol among n parties. The sharing algorithm randomly chooses a degree $k-1$ polynomial $p(\cdot)$ such that $p(0) = s$. The reconstruction algorithm takes in k or more number of shares and interpolates the polynomial to recover s.

A.2 Secure Public-Key Encryption.

A public-key encryption consists of three algorithms:

- $\mathsf{Gen}(1^\lambda)$ takes in the security parameter λ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$: take in the public key and a message msg, outputs a ciphertext ct.
- $\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: take in a ciphertext ct and the secret key sk, output msg.

A semantically secure public-key encryption satisfies the following properties:

- **Correctness**: $\Pr[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) : \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{msg})) = \mathsf{msg}] = 1$.

- **Security**: We say that the encryption scheme is semantically secure iff

$$\left| \Pr\left[\begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{msg}_0,\mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda,\mathsf{pk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{msg}_0) \end{array} : \mathcal{A}(1^\lambda,\mathsf{ct}) = 1 \right] \right.$$
$$\left. - \Pr\left[\begin{array}{c} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\mathsf{msg}_0,\mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda,\mathsf{pk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{msg}_1) \end{array} : \mathcal{A}(1^\lambda,\mathsf{ct}) = 1 \right] \right| \leq \mathsf{negl}(n)$$

A.3 Digital Signature Scheme

A digital signature scheme Π_{Sig} consists of the following algorithms:

- $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$: takes in the security parameter λ and outputs a key pair $(\mathsf{vk},\mathsf{sk})$ of verification key vk and signing key sk.
- $\sigma \leftarrow \mathsf{Sign}(m,\mathsf{sk})$: takes in a message m and the signing key sk, produces a signature σ.
- $b \leftarrow \mathsf{Ver}(\mathsf{vk},\sigma,m)$: takes in a message m, the signature σ, and the verification key vk, output 0 or 1 indicating reject or accept.

A secure digital scheme satisfies the following properties

- **Correctness**: Except with negligible probability over $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, it must be that for any message m, $\mathsf{Ver}(\mathsf{vk},m,\mathsf{Sign}(\mathsf{sk},m)) = 1$.
- **Security**: A digital scheme is existentially unforgeable under chosen-message attack iff

$$\Pr\left[\begin{array}{c} (\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m^*,\sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(1^\lambda,\mathsf{vk}) \end{array} : \mathsf{Ver}(\mathsf{vk},m^*,\sigma) = 1 \right] \leq \mathsf{negl}(n)$$

B $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}$ Implies CSP Fairness and Preference Privacy

CSP-fairness. CSP-fairness was first introduced in [CGL+18a]. It states that no coalition can increase its own expected utility except for a negligible term, no matter how it deviates from the protocol. For a coin-toss protocol Π with trusted setup Setup and a coalition $\mathcal{A}$, let $\Pi^{(S_\mathcal{A},H_{-\mathcal{A}})}(\mathsf{pp},\mathsf{sk}_1,\ldots,\mathsf{sk}_n)$ denote an execution of protocol Π when every party holds public parameter pp, each party holds sk_i as private parameter, the coalition $\mathcal{A}$ adopts strategy $S_\mathcal{A}$ and other parties behave honestly. Each trace tr sampled from the random execution $\Pi^{(S_\mathcal{A},H_{-\mathcal{A}})}(\mathsf{pp},\mathsf{sk}_1,\ldots,\mathsf{sk}_n)$ assigned a joint utility $\mathsf{util}_\mathcal{A}(tr)$ of coalition $\mathcal{A}$. We use $\mathsf{util}_\mathcal{A}(S_\mathcal{A},H_{-\mathcal{A}})$ to denote the expected utility of the coalition:

$$\begin{aligned} &\mathsf{util}_\mathcal{A}(S_\mathcal{A},H_{-\mathcal{A}}) \\ :=&\mathbb{E}\left[(\mathsf{pp},\mathsf{sk}_1,\ldots,\mathsf{sk}_n) \leftarrow \mathsf{Setup}(1^\lambda,\mathcal{P}), tr \leftarrow \Pi^{(S_\mathcal{A},H_{-\mathcal{A}})}(\mathsf{pp},\mathsf{sk}_1,\ldots,\mathsf{sk}_n) : \mathsf{util}_\mathcal{A}(tr)\right]. \end{aligned}$$

Definition B.1 (Property-based definition, CSP fairness [CGL+18c])**.** An m-sided coin-toss protocol Π with trusted setup Setup for a preference profile $\mathcal{P}$ is *cooperative-strategy-proofness* (CSP-fairness) against any t-sized coalition iff for any coalition $\mathcal{A}$ of size no more than t, for any non-uniform PPT strategy $S_\mathcal{A}$ that $\mathcal{A}$ adopts, there exists a negligible function $\mathsf{negl}(\cdot)$, such that $\mathsf{util}_\mathcal{A}(S_\mathcal{A},H_{-\mathcal{A}}) \leq \mathsf{util}_\mathcal{A}(H_\mathcal{A},H_{-\mathcal{A}}) + \mathsf{negl}(\lambda)$, where $H_\mathcal{A}$ denotes the honest strategy of $\mathcal{A}$.

Remark B.2 ($\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$ (Figure 1) implies Definition B.1)**.** The ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$ guarantees Definition B.1. The outcome is either a uniformly random coin chosen by the ideal functionality, or a coin is such that the corrupted coalition's joint utility is at most $\frac{|\mathcal{A}|}{m}$. Clearly, in the ideal world where parties have access to $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$, the corrupted coalition has no strategy to increase its joint utility. Therefore, if a real-world protocol securely instantiates this ideal functionality, it also satisfies Definition B.1.