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 does not increase 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 binary scheme matches the corruption threshold of Wu-Asharov-Shi (EUROCRYPT '22); our multi-sided scheme builds on Thyagarajan-Wu-Soni (CRYPTO '24) with a corruption guarantee depending on the coalition's preference counts. 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.

*IE 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

Contents

1 Introduction **3**
- 1.1 Our Results . . . 4
  - 1.1.1 The Need for Authenticated Preferences . . . 4
  - 1.1.2 Defining Fairness and Preference-Privacy . . . 6
  - 1.1.3 Binary Coin-Toss: Realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (N_0, N_1)]$ . . . 7
  - 1.1.4 General m-sided Coin-Toss: Realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}]$ . . . 7
  - 1.1.5 Security Preserving Compiler from Public- to Private-Preferences. . . . 8
- 1.2 Discussion on our Definition . . . 9
- 1.3 Additional Related Work . . . 10
- 1.4 Technical Overview . . . 11

2 Model and Definitions **14**
- 2.1 Preliminaries . . . 14
- 2.2 Coin-Tossing Protocols with Certified Preferences . . . 16
- 2.3 CSP-Fair Coin-Tossing Ideal Functionality . . . 18
- 2.4 Comparing Property- and Simulation-Based CSP-Fairness . . . 19

3 A Compiler from Public to Private Preferences **21**
- 3.1 Building Blocks . . . 21
- 3.2 The Compiler . . . 23
  - 3.2.1 Formation of the Masked Preference Profile . . . 23
  - 3.2.2 Emulating the Source Protocol Π . . . 24
- 3.3 Security of the Compiler . . . 25
- 3.4 Efficiency of the Compiler . . . 29

4 Protocol Securely Realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pub}}]$ **31**

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+18] (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 *multi*-party sampling protocols that achieve game-theoretic fairness against dishonest majority coalitions: for binary coin-toss [WAS22] 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 [WAS22], 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.

This privacy concern arises naturally in leader election and, more generally, multi-candidate selection. Prior works [WAS22,CCWS21,KMSW22,TSW24] typically consider the setting in which each party prefers to be selected itself, making the preference profile effectively public. In broader selection settings, however, a party may privately support another candidate, and revealing this preference may expose sensitive strategic, political, or commercial alignments. When an external authority already certifies participation or reported preferences, it can authenticate each party's preference and publish only aggregate statistics. The parties can then repeatedly run the selection protocol without revealing which party supports which candidate.

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 barrier by introducing *authenticated preferences*, inspired by prior work on authenticated MPC [DGPS22]. Before the protocol begins, each party declares a preference to a certifying authority, which signs the pair consisting of the party's identity and its declared preference. The resulting certified preference serves as the fixed benchmark for defining that party's utility: a party receives utility 1 when the protocol output matches its certified preference and utility 0 otherwise. Thus, the authority need not determine a party's subjective "true" preference; rather, it binds the party to a single preference before the protocol begins. In applications where preferences correspond to externally verifiable attributes, the authority may additionally validate the declaration or impose accountability for false declarations, but such application-level validation is separate from our cryptographic guarantee.

We additionally require the authority to be *stateful* during this one-time certification phase. After processing all certification requests, it publishes aggregate statistics $(N_0, \ldots, N_{m-1})$, where N_b is the number of parties certified with preference b. These aggregates are fixed before the coin-tossing protocol begins and remain the benchmark for fairness even if a corrupted party later withholds its certificate or submits an invalid one. Such a submission may be treated as $\perp$ within that execution, but it does not remove the party's certified preference from the published aggregate or from the coalition's utility. In particular, corrupted parties cannot change the relevant preference distribution by selectively causing some coalition members to be treated as $\perp$.

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.

Role of the Authority. Finally, we emphasize that the authority does not assign or reshuffle the virtual identities used by our protocol. That step is performed distributively by the parties during the commit phase, where parties anonymously broadcast their virtual identities and preferences together with zero-knowledge proofs that these values are properly certified. The authority's role is therefore limited to fixing the certified-input benchmark and publishing its aggregate; the cryptographic protocol provides preference privacy, enforces consistency with certified preferences, and realizes game-theoretically fair sampling against malicious coalitions. After the certification phase, the authority does not participate in the coin-tossing protocol.

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+18].

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 $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}$ that appropriately models both game-theoretic fairness and preference-privacy.

Ideal Coin-tossing Functionality with Authenticated Preferences. In Section 2, we formally describe our ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}]$ for m-sided coin-tossing. Our ideal functionality is parameterized by the verification key mvk of the certifying authority and the aggregate information $\mathbf{N} = (N_0, \ldots, N_{m-1})$ 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 $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, \mathbf{N} = (N_0, N_1)]$ works as follows:

1. **Reporting by Honest Parties.** Parties $i \in \mathcal{H}$ sends their $(i, \mathsf{pref}_i, \sigma_i)$ 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 $\mathsf{leak} = (h_0, h_1)$ 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_i is invalid or $\mathsf{pref}_i \notin \{0, 1\}$, then the functionality records the submitted value as $\perp$. This does not alter the aggregate (N_0, N_1) fixed during certification or, importantly, the aggregate k_0 and k_1 that will be used to evaluate the coalition's honest utility.

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.

[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.

[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.

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.

For the general m-sided coin toss, the leakage leak the ideal functionality sends to $\mathcal{A}$ at the end of the honest parties reporting phase is $\mathbf{h} = (h_0, ..., h_{m-1})$ as the honest aggregate, where h_ℓ is the number of honest parties preferring outcome ℓ.

Our goal is to design a protocol that realizes $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}]$ in the presence of the certifying authority. Philosophically, realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}]$ 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 $\mathbf{N}$ and the computed coin.

1.1.3 Binary Coin-Toss: Realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (N_0, N_1)]$

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 $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (N_0, N_1)]$, 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 $\Pi[2, \mathsf{mvk}, (N_0, N_1)]$ that securely realizes $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[2, \mathsf{mvk}, (N_0, N_1)]$ against t corruptions, where*

$$t = \begin{cases} n - \left\lfloor \frac{3}{2} N_0 \right\rfloor, & \text{if } n \geq \frac{7}{2} N_0; \\ \left\lfloor \frac{2}{3} n - \frac{5}{6} N_0 \right\rfloor + \left\lceil \frac{1}{2} N_0 \right\rceil, & \text{otherwise.} \end{cases}$$

Semantically-secure PKE and digital signatures can be instantiated from concrete algebraic assumptions, and 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 $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}]$

Following the technique of Thyagarajan, Wu, and Soni [TSW24], our protocol can be generalized to multi-sided coin-tossing. Let $\mathbf{N} = (N_0, ..., N_{m-1})$ be the aggregat published by the authority. Given an arbitrary preference profile $\mathcal{P}$, without loss of generality, assume that $N_0 \leq N_\ell$ for $\ell \in \{0, ..., m-1\}$. 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*

$\Pi[m, \mathsf{mvk}, \mathbf{N}]$ *that securely realizes* $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}]$ *against a family of corruptions* $\mathcal{K}$ *containing* k_ℓ *number of parties preferring outcome* ℓ *such that*

$$|\mathcal{K}| \leq \lfloor N_0/2 \rfloor + t^*, \text{ and either } |\mathcal{K}| - k_0 < (n - N_0) - t^* \text{ or } k_0 \leq k_\ell \text{ for } \ell \in \{0, ..., m-1\}$$

where t^* *is as follows:*

$$t^* = \begin{cases} n - m \cdot N_0, & \text{if } n \geq \frac{4m-1}{2} N_0 \\ \left\lfloor \frac{m}{2m-1} n - \frac{3m-1}{2(2m-1)} N_0 \right\rfloor, & \text{otherwise.} \end{cases}$$

As a comparison, for binary coin-tossing, our protocol matches the corruption threshold of [WAS22]; for m-sided coin-tossing, it provides the guarantee above, using the construction of [TSW24].[4] 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 binary threshold and the stated multi-sided corruption family while simultaneously ensuring preference-privacy and realizing a simulation-based notion of fairness, albeit under the assumption of authenticated preferences.

1.1.5 Security Preserving Compiler from Public- to Private-Preferences.

Both Theorem 1.1 and Theorem 1.2 are obtained through a general compiler that transforms any position-oblivious protocol realizing our public-preference ideal functionality into one realizing the corresponding private-preference ideal functionality. Informally, position-obliviousness means that the protocol does not depend on which party occupies which index or position. The transformation incurs at most an additive $O(n)$-rounds in round complexity and polynomial overhead in communication and computation. Crucially, the compiler preserves the adversarial resilience of the underlying protocol: if the public-preference protocol is secure against a coalition $\mathcal{K} \subseteq [n]$, then the compiled private-preference protocol remains secure against the same coalition.

To state our result more formally, we first introduce an extension of our ideal functionality that captures both the public- and the private-preference cases. Specifically, we parameterize the ideal functionality via a leakage function Leak such that the ideal-functionality reveals leakage $\mathsf{Leak}(\mathcal{P}_\mathcal{H})$ on the honest preference profile $\mathcal{P}_\mathcal{H}$ to the adversary $\mathcal{A}$ at the end of the honest reporting phase. The public-preference ideal functionality is then instantiated via the function $\mathsf{Leak}_{\text{pub}}$ that leaks the full honest preference profile, where as in the private-preference case, leakage $\mathsf{Leak}_{\text{pri}}$ reveals only the honest aggregate $\mathbf{h} = (h_0, ..., h_{m-1})$. We can now state the theorem behind our compiler and overview ideas behind it in Section 1.4.

Theorem 1.3 (Compiler, informal)**.** *Assuming bounded concurrent zero-knowledge arguments, semantically secure public-key encryption and secure digital signature scheme, there is a compiler* Comp *that takes in a position-oblivious public-preference protocol* Π *securely instantiating* $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pub}}]$ *against a family* Γ *of corruptions and outputs* $\mathsf{Comp}(\Pi)$ *such that* $\mathsf{Comp}(\Pi)$ *securely instantiates* $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pri}}]$ *against a family* Γ *of corruptions.*

[4] Prior works in this line of work state threshold-corruption guarantees: after fixing the overall preference profile and deriving a threshold t, the protocol is claimed to be secure against every coalition $\mathcal{K} \subseteq [n]$ with $|\mathcal{K}| \leq t$, irrespective of the preferences' of parties in $\mathcal{K}$. Our guarantee is more refined. Even for a fixed overall preference profile, not every coalition of a given size is necessarily tolerated; whether a coalition $\mathcal{K}$ can be handled depends additionally on the preference profile induced by the parties in $\mathcal{K}$. Our multi-sided coin-tossing protocol builds on the construction of [TSW24], which claims a threshold-corruption guarantee of the former type. We observe that its security analysis does not establish security for all coalitions below the stated threshold, and discuss this issue in Section 4. The corrected statement instead characterizes security as a function of both the overall preference profile and the preferences of the corrupted coalition, which is the form inherited by our theorem statement.

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 authority is stateful only during the one-time certification phase: after certifying the parties' declared preferences, it publishes the corresponding aggregate statistics, such as (N_0, N_1). These aggregates are fixed before the coin-tossing protocol begins and determine the preference profile with respect to which fairness and coalition utility are evaluated. Consequently, if a corrupted party later withholds its certificate or submits an invalid one, its submitted value may be treated as $\perp$, but its certified preference remains included in the fixed aggregate and in the coalition's utility. Thus, corrupted parties cannot change the relevant preference distribution by selectively excluding some members of the coalition—the "alive coalition" attack that would arise if fairness were evaluated only with respect to successfully submitted preferences.

Replacing the stateful authority would require a distributed mechanism that privately establishes a common aggregate preference profile, ensures agreement on this aggregate despite a malicious majority, and prevents selective withholding from changing it. Whether the same guarantees can be achieved without such an authority remains an interesting open direction.

Division between the Authority and Cryptography. The authority fixes the certified-input benchmark: it certifies one declared preference per party and publishes the resulting aggregate statistics. Our formal guarantees are defined relative to these certified preferences; as in standard MPC, the cryptographic protocol does not determine whether a supplied input reflects an external ground truth. In applications, the authority may use independent information or accountability mechanisms to validate declarations, but this is application-specific. Once the certified inputs are fixed, cryptography provides the remaining guarantees: it binds parties to their certified preferences during execution, hides the association between parties and preferences, distributively reshuffles the virtual identities, and realizes fair sampling with preference privacy against malicious coalitions.

The authority neither performs the virtual-identity reshuffling nor participates in the subsequent coin-tossing executions. Moreover, despite an admittedly powerful authority, our model does not delegate coin-tossing itself to the authority and its involvement is limited to the one-time initialization step. 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 real-ideal formulation of game-theoretic fairness with private preferences. In addition to capturing fairness and preference privacy within a unified ideal functionality, this formulation supports the standard *sequential modular composition* guarantee of stand-alone secure computation. In particular, by the standard subroutine-substitution theorem of Canetti [Can00], a protocol that securely realizes $\mathcal{F}_{\mathsf{toss}}^{\mathsf{auth}}$ can replace an ideal invocation of this functionality within a higher-level protocol. Consequently, our protocol may be invoked a polynomial number of times sequentially, or used as a sequential subroutine in a larger protocol, in this standard stand-alone sense. By contrast, the earlier property-based formulations do not by themselves provide such a simulation-based substitution guarantee. We emphasize that we do not establish concurrent general composition or universal composability; such stronger guarantees require a framework designed to preserve security under arbitrary concurrent composition [Can01, Lin03].

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+18, 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 1.4, we illustrate the overview of the techniques used in our protocol. The formal model and definitions are given in Section 2, and the compiler is presented in Section 3. We then present the public-preference protocol instantiating the public-preference ideal functionality in Section 4.

1.4 Technical Overview

In this section, we present an overview of our compiler from public to private preferences coin tossing. For better understanding, we use binary coin tossing from [WAS22] as an illustrating source protocol to explain why existing public-preference protocols leak the parties' preferences. We then describe how our compiler emulates this public-preference protocol over a masked preference profile, and explain why this transformation preserves simulation security. Finally, we describe how to instantiate the compiler with public-preference protocols for binary and general m-sided coin tossing. For simplicity, we use $\mathcal{F}_{\text{pub}}$ and $\mathcal{F}_{\text{pri}}$ to denote the public- and private-preference ideal functionality when the parameters are clear.

Recall that parties have their preferences certified by an authority before the coin-tossing protocol begins. We call a party who prefers outcome $b \in \{0,1\}$ a b-supporter. If the outcome of the coin-tossing protocol matches the party's authenticated preference, the party obtains utility 1, and otherwise obtains utility 0. The expected joint utility of a coalition is the sum of its members' expected utilities.

Before presenting our protocol, we first outline the binary public preference protocol and explain where the preference of each party is being leaked. Then we show how our compiler hides the parties' preferences using anonymous broadcast. Jumping ahead, the anonymous broadcast required in our protocol can be instantiated using Shamir secret sharing [Sha79] and bounded-concurrent zero-knowledge proofs [Pas04].

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 receivers' identities and preferences. Otherwise, how would they know where to send the message?

Our Compiler: Intuition. In a nutshell, our compiler places an anonymous communication layer over a public-preference source protocol. The source protocol is executed using fresh virtual identities $\mathsf{vID}_1, ..., \mathsf{vID}_n$, so that it remains infeasible to link a virtual identity, its publicly revealed preference, or any of its messages to the corresponding real identity. Only the aggregate preference information is revealed. For this purpose, we want the following two properties:

1. *Anonymity*: The adversary cannot link an honest virtual sender or receiver to its real identity.
2. *Privacy*: The adversary cannot learn the content of any private messages between honest senders and honest receivers.

Property 1 above ensures that honest parties' preferences are unlinkable via anonymous communication. Property 2 guarantees that all the honest-to-honest private messages remain hidden from the adversary to maintain the security of the source protocol.

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,[5] 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 public-preference binary CSP-fair coin tossing, 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 the public keys for those who have the same preferences.

Our Approach: Masked Profile. To achieve this, we ask each party i to randomly generate a virtual identity vID_i and an encryption key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, and to obtain a certification of $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ from the authority. The virtual identity vID_i will serve as party i's identity in the source protocol, while the mapping between real parties and virtual identities remains hidden throughout the execution via anonymous broadcast.

At the beginning of the compiled protocol, the parties anonymously broadcast their certified tuples $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ and prove that they possess both a valid certification and the secret key corresponding to pk_i. The valid tuples form a public *masked preference profile* $\widehat{\mathcal{P}} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)\}$. This profile reveals the preference associated with every successfully reported virtual identity, but not the real identity who owns that virtual identity and preference.

Some corrupted parties may fail to report a valid certified tuple. The public aggregate $\mathbf{N}$ published by the authority determines exactly how many certified preferences of each type are missing. The compiler therefore inserts *dummy virtual identities* with the exact same number of missing preferences into $\widehat{P}$ and treats them as parties that abort at the beginning of the source protocol. Consequently, the masked profile $\widehat{\mathcal{P}}$ has the same aggregate $\mathbf{N}$ as the certified preference profile, even though the adversary may withhold some of its certified preferences.

With this masked profile, the parties can execute the public-preference source protocol as if they have the public preference profile with respect to virtual IDs. For example, if the source protocol instructs each 0-supporter to share a secret among all 0-supporters, each vID_i with preference 0 will now share a secret among all virtual IDs with 0-preference in $\widehat{\mathcal{P}}$. Whenever a virtual ID vID_i needs to send a message $\mathsf{msg}_{i,j}$ to a virtual ID vID_j, it encrypts the message under vID_j's public key

[5] 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.

pk_j and broadcast the ciphertext using anonymous broadcast. Thus, the source protocol can use the public masked profile exactly as before, while the ownership of its protocol identities remains hidden.

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.

Description of the Compiler: Emulating the Source Protocol. Now that the parties anonymously broadcast their certified preferences and forms a masked preference profile $\widehat{\mathcal{P}}$, we describe how our compiler emulates the source protocol. Let Π be the public-preference source protocol. We require that the source protocol to be *position-oblivious*, i.e., the execution of Π only depends on the preference profile $\widehat{P}$ and the protocol identity vID_i, it does not depend on the i-th position in the preference profile. For example, in the binary CSP-fair coin tossing we described, permuting the preference profile does not change the protocol's execution.

Once the masked profile $\widehat{\mathcal{P}}$ has been formed, the compiler emulates Π round by round using the virtual identities in $\widehat{\mathcal{P}}$. The inserted dummy identities are treated as having aborted before the first source-protocol round.

In each round, the owner of an active virtual identity vID_i evaluates the next-message function of Π with respect to its virtual identity and its current source-protocol view. This produces one private message for each source-protocol receiver and one public broadcast message. The party encrypts every private message $\mathsf{msg}_{i,j}$ intended for an active virtual identity vID_j under pk_j and get ciphertext $\mathsf{ct}_{i,j}$, places a dummy ciphertext in the position of each initially aborted identity, and leaves the source-protocol broadcast message in the clear. It then anonymously broadcasts the resulting message vector $(\mathsf{vID}_i, (\mathsf{ct}_{i,1}, ..., \mathsf{ct}_{i,n}, \mathsf{msg}_{i,\mathrm{all}}))$, where $\mathsf{msg}_{i,\mathrm{all}}$ is the broadcast message vID_i is supposed to send.

For each anonymously broadcast message

$$\big(\mathsf{vID}_i, (\mathsf{ct}_{i,1}, \ldots, \mathsf{ct}_{i,n}, \mathsf{msg}_{i,\mathsf{all}})\big),$$

the sender proves in zero knowledge that it knows the secret key corresponding to the public key pk_i associated with vID_i, and that it knows plaintexts $\mathsf{msg}_{i,j}$ and randomness used in encryption consistent with all ciphertexts in the broadcast vector. This guarantees that every accepted anonymous contribution can be interpreted as one well-formed message from a source-protocol identity. The proof does not enforce that the plaintexts were generated by the honest next-message function: a corrupted virtual identity may choose arbitrary plaintexts, exactly as a corrupted party may choose arbitrary messages in the source protocol.

Each active virtual identity decrypts the ciphertext addressed to it and uses the resulting plaintext, together with the public broadcast messages, to update its emulated source-protocol view. The parties continue in this manner until the source protocol terminates.

Why the Compiler Is Secure. The security proof relies on two observations. First, anonymous broadcast hides the mapping between real identities and virtual identities. Therefore, a simulator that learns only the honest preference aggregate can sample fresh virtual identities and assign the honest preferences to them that are consistent with those aggregate. Since the source protocol is position-oblivious, this hidden assignment does not affect its execution. The security of messages are then guaranteed by the security of public key encryption.

Second, the simulator interprets the extracted messages as the actions of an induced adversary against the public-preference source protocol. The induced adversary controls both the successfully reported corrupted virtual identities and the inserted identities that abort initially. Their total certified preference vector is exactly

$$\mathbf{k} = \mathbf{N} - \mathbf{h},$$

the aggregate of the physical corrupted coalition. The simulator may therefore invoke the security of the source protocol. Moreover, because the real and virtual corrupted coalitions have the same preference counts, the public- and private-preference ideal functionalities select the same least-preferred outcome whenever the adversary requests failure. Therefore, taking any source protocol Π that instantiates public-preference ideal functionality against a family Γ of corruption, our compiler $\mathsf{Comp}(\Pi)$ returns a protocol that instantiates private-preference ideal functionality against a family Γ of corruption.

Source Protocol. It remains to show that there are position-oblivious protocols that instantiate the public-preference ideal functionality. The aforementioned binary CSP-fair coin-tossing from [WAS22] is position oblivious and can be made to instantiate the public-preference ideal functionality against any t-sized coalition for t in Theorem 1.1. For m-sided coin, we can adopt the two-group reduction of [TSW24]. Assume that $N_0 \leq N_\ell$ for any $\ell \in \{0, ..., m-1\}$. Separate 0-supporters into a group G_0 and all other parties into a group G_{oth}. The protocol generates coins c_0 and c_{oth} for the two groups and combines them as $(c_0 + c_{\mathsf{oth}}) \bmod m$. We show that this protocol is position-oblivious and can be adapted to instantiate $\mathcal{F}_{\text{pub}}$ against a specific family Γ of corruptions. Feeding these protocols as source protocol into the compiler gives us protocols instantiating $\mathcal{F}_{\text{pri}}$.

2 Model and Definitions

Notation. We assume that every party is a probabilistic polynomial-time (PPT) Turing machine. Throughout the paper, we use $\mathcal{A}$ to denote the adversary controlling a coalition of corrupted parties. We use λ for the security parameter and $[n] = \{1, \ldots, n\}$. We also write $[m]_0 = \{0, \ldots, m-1\}$ and use U_m for the uniform distribution over $[m]_0$.

A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible in λ if, for every $c \in \mathbb{N}$, there is $\lambda_0 \in \mathbb{N}$ such that $\mathsf{negl}(\lambda) \leq \lambda^{-c}$ for every $\lambda \geq \lambda_0$. Two distribution ensembles $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable, denoted by $\{X_\lambda\}_{\lambda \in \mathbb{N}} \approx_c \{Y_\lambda\}_{\lambda \in \mathbb{N}}$, if, for every non-uniform PPT distinguisher D, there is a negligible function negl_D such that, for every λ,

$$|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]| \leq \mathsf{negl}_D(\lambda).$$

We use $\equiv$ to denote identical distributions.

2.1 Preliminaries

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:

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

  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.

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

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

  where we require $|\mathsf{msg}_0| = |\mathsf{msg}_1|$.

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.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$: takes in a message m and the signing key sk, produces a signature σ.
- $b \leftarrow \mathsf{Ver}(\mathsf{vk}, m, \sigma)$: 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 the following bound holds for every non-uniform PPT adversary $\mathcal{A}$, where Q is the set of messages $\mathcal{A}$ queried before:

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

2.2 Coin-Tossing Protocols with Certified Preferences

In an n-party coin-tossing protocol with certified preferences, every party first has its preference certified. Each party $i \in [n]$ has a preference $\mathsf{pref}_i \in [m]_0$. We use ID_i for the protocol identity under which party i is certified. This identity need not be the party's physical identity: it could be a label, a public key, or a freshly sampled virtual identity. When the context is clear, we may call party i as party ID_i interchangeably.

A *preference profile* is the ordered labeled list

$$\mathcal{P} = ((\mathsf{ID}_1, \mathsf{pref}_1), \ldots, (\mathsf{ID}_n, \mathsf{pref}_n)).$$

The order corresponds i, while the labels ID_i are the identities used by the protocol. For a subset $S = \{i_1 < \cdots < i_s\} \subseteq [n]$, its restricted profile is the ordered sublist $\mathcal{P}_S = ((\mathsf{ID}_{i_1}, \mathsf{pref}_{i_1}), \ldots, (\mathsf{ID}_{i_s}, \mathsf{pref}_{i_s}))$.

If $\mathsf{pref}_i = \ell$, we call party i an ℓ-supporter. Given a preference profile $\mathcal{P}$, define the aggregate of the profile as

$$N_\ell(\mathcal{P}) = \left|\{i \in [n] : \mathsf{pref}_i = \ell\}\right| \quad \text{and} \quad \mathbf{N}(\mathcal{P}) = (N_0(\mathcal{P}), \ldots, N_{m-1}(\mathcal{P})).$$

The authority who certifies the preference is denoted as $\mathsf{agg\text{-}auth}$. The authority is parameterized by a digital-signature scheme $\Pi_{\mathrm{Sig}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$. It acts as follows.

$\mathsf{agg\text{-}auth}(\Pi_{\mathrm{Sig}})$

- It samples $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \Pi_{\mathrm{Sig}}.\mathsf{Gen}(1^\lambda)$, initializes an empty ordered preference profile, and initializes an empty set of registered parties.
- Upon a request $(\mathsf{ID}_i, \mathsf{pref}_i)$ from party i, the authority accepts the request only if i has not previously registered and ID_i has not previously been used. Thus, each party can obtain at most one certificate.
- For an accepted request, it sends $\mathsf{auth}_i = \Pi_{\mathrm{Sig}}.\mathsf{Sign}(\mathsf{msk}, (\mathsf{ID}_i, \mathsf{pref}_i))$ to party i and stores $(\mathsf{ID}_i, \mathsf{pref}_i)$ in i-th slot of $\mathcal{P}$.

- At the end, it publishes $(\mathsf{mvk}, \mathbf{N}(\mathcal{P}))$ on a public bulletin board.

After certification, parties communicate through pairwise private channels and a public broadcast channel. The channels are synchronous and authenticated with respect to the protocol identities ID_i recorded in $\mathcal{P}$: every message carries the identity of the sender.

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 agg-auth and all messages posted in the broadcast channel. All protocols run for a polynomially bounded number of rounds and have polynomially bounded message complexity and message lengths.

Next-message function. We denote an m-sided coin-tossing protocol with certified authority by $\Pi[m, \mathsf{mvk}, \mathbf{N}]$, where $(\mathsf{mvk}, \mathbf{N})$ is the public output of agg-auth. We describe honest execution in next-message form. At the start of the execution, party ID_i samples a private random tape ω_i. In round r, it evaluates

$$(\mathsf{msg}_i^r, \mathsf{out}_i^r) \leftarrow \mathsf{next}_\Pi\big(1^\lambda, \mathsf{ID}_i, r, \mathsf{view}_i^{<r}; \omega_i\big). \tag{1}$$

Here, the outgoing-message list is

$$\mathsf{msg}_i^r = ((\mathsf{ID}_1, \mathsf{msg}_{i,1}^r), \ldots, (\mathsf{ID}_n, \mathsf{msg}_{i,n}^r), (\mathsf{all}, \mathsf{msg}_{i,\mathsf{all}}^r)),$$

where msg_i^r is indexed by recipient's protocol identities ID_j, and instructs ID_i to send $\mathsf{msg}_{i,j}^r$ to ID_j. Recipient all denotes broadcast. If in round r, party ID_i does not need to send message to party ID_j, then $\mathsf{msg}_{i,1}^r = \mathsf{dum}$ as dummy message. The value out_i^r is either a protocol output or $\perp$ if the party has not terminated.

The local view $\mathsf{view}_i^{<r}$ contains all information available to ID_i at the beginning of round r: its private input, secret keys, and setup information $(\mathsf{pref}_i, \mathsf{auth}_i)$, the public setup $(\mathsf{mvk}, \mathbf{N})$, and its complete local transcript of messages sent and received in earlier rounds.

The indices $i \in [n]$ are used only as bookkeeping positions in the security experiment. The protocol identifies parties, addresses messages, and interprets the preference profile solely through the certified protocol identities ID_i.

Definition 2.1 (Position obliviousness)**.** We say that a protocol Π is *position-oblivious* if the following holds. Let $\mathcal{P} = ((\mathsf{ID}_1, \mathsf{pref}_1), \ldots, (\mathsf{ID}_n, \mathsf{pref}_n))$ and π be an arbitrary permutation over $[n]$. Let $\mathcal{P}'$ be a permutation of $\mathcal{P}$: $\mathcal{P}' = ((\mathsf{ID}_{\pi(1)}, \mathsf{pref}_{\pi(1)}), \ldots, (\mathsf{ID}_{\pi(n)}, \mathsf{pref}_{\pi(n)}))$. Consider any adversarial $\mathcal{A}$ that controls the same set of protocol identities $\mathcal{K} = \{\mathsf{ID}_k\}$ in the two executions. After matching parties' local views and outputs by protocol identity, the joint distributions of the public transcript, the parties' local views and outputs, and the adversary's output are identical in the executions with $\mathcal{P}$ and $\mathcal{P}'$.

Thus, the execution of Π depends only on the labeled preference profile and not on the bookkeeping position occupied by each record.

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

A malicious party need not follow the prescribed next-message function. We consider *rushing* adversary, meaning that in every round r, it may wait for all honest messages from that round before choosing its round-r actions. It may choose its randomness as a function of its current view,

send arbitrary messages in any round, or abort in any round. After aborting, it sends no further protocol messages; the protocol specifies how missing messages are interpreted.

Given a coalition $\mathcal{K}$ with certified preference aggregate $\mathbf{k} = (k_0, ..., k_{m-1})$, i.e., the number of ℓ-supporters in $\mathcal{K}$ is k_ℓ. We denote the *lease preferred outcome* of the coalition as

$$c_{\min}(\mathbf{k}) = \min \arg\min_{\ell} \{k_\ell\}, \tag{2}$$

where the first min means breaking ties lexicographically.

2.3 CSP-Fair Coin-Tossing Ideal Functionality

We define a CSP-fair coin-tossing functionality parameterized by a leakage function Leak. Let $\mathcal{H}$ be the set of honest parties, let $\mathcal{P}_\mathcal{H}$ be their preference profile. Let $h_\ell = N_\ell(\mathcal{P}_\mathcal{H})$ be the number of honest ℓ-supporters and $\mathbf{h} = \mathbf{N}(\mathcal{P}_\mathcal{H}) = (h_0, \ldots, h_{m-1})$ be the honest aggregates. The two leakage functions used in this paper are

$$\mathsf{Leak}_{\text{pub}}(\mathcal{P}_\mathcal{H}) = \mathcal{P}_\mathcal{H} \qquad \text{and} \qquad \mathsf{Leak}_{\text{pri}}(\mathcal{P}_\mathcal{H}) = \mathbf{h}.$$

Thus, $\mathsf{Leak}_{\text{pub}}$ reveals the full honest profile, whereas $\mathsf{Leak}_{\text{pri}}$ reveals only its preference counts.

Ideal functionality $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}]$

Let $\mathcal{H}$ be the set of honest parties and let $\mathcal{K}$ be the set of corrupted parties.

1. **Reporting of honest parties' preferences.** Every honest party $i \in \mathcal{H}$ sends $(\mathsf{ID}_i, \mathsf{pref}_i, \mathsf{auth}_i)$ with a valid certificate auth_i to the functionality. Let $\mathcal{P}_\mathcal{H} = ((\mathsf{ID}_i, \mathsf{pref}_i))_{i \in \mathcal{H}}$ be the honest preference profile and compute the aggregates $\mathbf{h} = (h_0, ..., h_{m-1})$, where $h_\ell = N_\ell(\mathcal{P}_\mathcal{H})$. Send $\mathsf{Leak}(\mathcal{P}_\mathcal{H})$ to $\mathcal{A}$.

2. **Reporting by corrupted parties.** 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 $\mathsf{pref}_i = \perp$. For each $\ell \in [m]_0$, set $k_\ell = N_\ell - h_\ell$ as the aggregates for corrupted parties. Let $\mathbf{k} = (k_0, ..., k_{m-1})$.

3. **Coin tossing.** Sample $c \xleftarrow{\$} [m]_0$ uniformly at random. Set c' to the least-preferred coin $c_{\min}(\mathbf{k})$ from Equation (2).

4. **Output generation.** Send c to $\mathcal{A}$. If $\mathcal{A}$ sends ok, send c to all honest parties. If $\mathcal{A}$ sends fail, send c' to all honest parties.

Figure 1: The m-sided authenticated coin-tossing functionality.

The above ideal functionality ensures that sending fail cannot increase the coalition's expected utility, since it will end up with the least preferred outcome based on the certified preferences. Importantly, the least preferred outcome is determined by the aggregate published by authority $\mathbf{N}$ and the honest aggregate $\mathbf{h}$; it does not depend on how corrupted parties report their preferences.

Secure Realization. Our security guarantees may depend not only on the total number of corrupted parties, but also on their certified preferences. For each certified aggregate $\mathbf{N} = (N_0, \ldots, N_{m-1})$,

we let $\Gamma_{\mathbf{N}}$ denote a set of admissible corrupted preference-count vectors $\mathbf{k} = (k_0, \ldots, k_{m-1})$, where $0 \le k_\ell \le N_\ell$ for every $\ell \in [m]_0$. A coalition $\mathcal{K}$ is said to be Γ-admissible for a preference profile $\mathcal{P}$ if

$$\mathbf{N}(\mathcal{P}_{\mathcal{K}}) \in \Gamma_{\mathbf{N}(\mathcal{P})}.$$

This formulation allows the tolerated corruptions to depend on the preference composition of the coalition rather than only on its cardinality. In particular, any two coalitions with the same certified preference-count vector are either both admissible or both inadmissible, regardless of the identities of their members. The usual threshold model is recovered as the special case in which $\Gamma_{\mathbf{N}}$ contains all those vectors satisfying $\sum_{\ell=0}^{m-1} k_\ell \le t$.

Definition 2.2 (Secure realization)**.** Let $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ be an admissible corruption family. We say that $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ securely realizes $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}]$ with respect to Γ if, for every coalition $\mathcal{K} \subseteq [n]$ and every non-uniform PPT real-world adversary $\mathcal{A}$ controlling $\mathcal{K}$, there is a PPT ideal-world simulator Sim such that, for every certified preference profile $\mathcal{P}$ satisfying $\mathbf{N}(\mathcal{P}) = \mathbf{N}$ for which $\mathcal{K}$ is Γ-admissible for $\mathcal{P}$, for any auxiliary input $z \in \{0,1\}^*$,

$$\left\{\mathsf{Exec}_{\mathcal{A}}^{\Pi}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}, z)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathsf{Exec}_{\mathsf{Sim}}^{\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m,\mathsf{mvk},\mathbf{N};\mathsf{Leak}]}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}, z)\right\}_{\lambda\in\mathbb{N}},$$

where $\mathsf{Exec}_{\mathcal{A}}^{\Pi}(1^\lambda, m, \mathsf{mvk}, \mathcal{P}, z)$ outputs the honest parties' output as well as the $\mathcal{A}$'s view when interacting in coin toss protocol Π where parties' preference profile is $\mathcal{P}$, and $\mathsf{Exec}_{\mathsf{Sim}}^{\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m,\mathsf{mvk},\mathbf{N};\mathsf{Leak}]}(1^\lambda, m, \mathsf{mvk}, \mathcal{P}, z)$ outputs the honest parties' output as well as the Sim's output. In the above, $\mathcal{P}$ is used to setup the experiment. It is not an input feed to the adversary $\mathcal{A}$ or simulator Sim.

2.4 Comparing Property- and Simulation-Based CSP-Fairness

Previous works use a property-based definition to capture CSP-fairness. For a certified preference profile $\mathcal{P}$, let $\mathsf{util}_{\mathcal{A}}^{\Pi[S_{\mathcal{A}}, H_{-\mathcal{A}}]}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P})$ denote the expected sum of the utilities of the coalition controlled by $\mathcal{A}$, when it uses strategy $S_{\mathcal{A}}$ and all other parties follow the honest strategy. Utilities are evaluated on the publicly verifiable coin outcome.

Definition 2.3 (Property-based CSP-fairness [CGL+18])**.** An m-sided coin-tossing protocol $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ is *CSP-fair* with respect to $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ if, for every certified profile $\mathcal{P}$ with aggregate $\mathbf{N}$, every Γ-admissible coalition $\mathcal{K}$, and every non-uniform PPT strategy $S_{\mathcal{A}}$ controlling $\mathcal{K}$, its expected utility satisfies

$$\mathsf{util}_{\mathcal{A}}^{\Pi[S_{\mathcal{A}}, H_{-\mathcal{A}}]}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}) \le \mathsf{util}_{\mathcal{A}}^{\Pi[H_{\mathcal{A}}, H_{-\mathcal{A}}]}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}) + \mathsf{negl}(\lambda),$$

where $H_{\mathcal{A}}$ is the honest strategy for the coalition and the bound holds for every auxiliary input, with a negligible function depending on the adversary.

Theorem 2.2 Implies Theorem 2.3. Property-based CSP-fairness constrains only the coalition's expected utility, whereas secure realization of the specified functionality constrains the full joint distribution of the honest output and the adversary's view. In fact, we show that if a protocol achieves simulation-based CSP-fairness, then it is property-based CSP-fair.

Lemma 2.4. *If a* $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ *securely realizes* $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}]$ *with respect to* Γ*. Then* Π *is CSP-fair with respect to* Γ*.*

Proof. Fix a certified preference profile $\mathcal{P}$, a Γ-admissible coalition $\mathcal{K}$ with aggregate $\mathbf{k} = (k_0, \ldots, k_{m-1})$, and a non-uniform PPT strategy $S_{\mathcal{A}}$. The honest expected utility is $\frac{1}{m}\sum_{\ell=0}^{m-1} k_\ell$, up to a negligible term. By secure realization, the real outcome and adversary's view are computationally indistinguishable from an ideal execution with some simulator Sim. The functionality samples $c \xleftarrow{\$} [m]_0$; whether Sim replies ok or fail, the coalition's utility is at most k_c, since failure yields $k_{c_{\min}(\mathbf{k})} = \min_\ell k_\ell$. Thus the ideal expected utility is at most $\frac{1}{m}\sum_{\ell=0}^{m-1} k_\ell$. The utility is efficiently computable and bounded by the polynomially bounded number of parties, so computational indistinguishability transfers this bound to the real execution up to a negligible term. □

Theorem 2.3 Does Not Imply Theorem 2.2. The converse statement, however, is not true.

Lemma 2.5. *There exists protocol* $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ *that satisfies Theorem 2.3 (property-based CSP-fairness) with respect to some* Γ*, but does not realize* $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pri}}]$ *with respect to the same* Γ*.*

Proof. Fix $m = 3$, let $\Gamma_{\mathbf{N}}^{(1)}$ consist of all feasible vectors $\mathbf{k}$ with $k_0 + k_1 + k_2 \le 1$. In other words, $\Gamma_{\mathbf{N}}^{(1)}$ allows at most one corrupted party in total. Fix $n \ge 2$ and an aggregate $\mathbf{N}$ with $N_0 \ge 1$. Let $\Pi'[3, \mathsf{mvk}, \mathbf{N}]$ be a protocol that securely realizes $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[3, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pri}}]$ with respect to $\Gamma^{(1)}$. Define Π that adds an initial broadcast round on Π' as follows.

- Honest parties send a fixed dummy message. A party with preference 0 may instead send a special abort message $(\mathsf{abort}^*, \mathsf{ID}_i, 0, \mathsf{auth}_i)$ and abort.
- *Special branch*: If there is a special abort message $(\mathsf{abort}^*, \mathsf{ID}_i, 0, \mathsf{auth}_i)$ in round 1 such that auth_i is a valid signature under mvk of $(\mathsf{ID}_i, 0)$, all honest parties output 2 and terminate.
- Otherwise parties run Π', treating any party that sent a special abort message in round 1 as abort at the beginning of Π. Other malformed initial messages are ignored.

Π satisfies Theorem 2.3 (property-based). An all-honest execution of Π runs Π', so each party has expected utility $1/3$. Consider a strategy for one corrupted party. Except with negligible probability of forging a valid signature of the certificate authority, only a certified 0-supporter can trigger the special branch, where its utility is 0. Construct a PPT strategy for Π' that internally executes the additional initial round, continues with the given strategy if the special branch does not occur, and otherwise follows the honest strategy in Π'. The initial honest messages are fixed and reveal no information about the subsequent execution. On the special branch, the replacement strategy has nonnegative utility; on every other branch its utility is unchanged. Hence the original strategy's expected utility in Π is at most that of this strategy in Π', plus the negligible probability of breaking signature, and is at most $1/3 + \mathsf{negl}(\lambda)$ by Theorem 2.4. This argument also works for 1- and 2-supporters, which cannot trigger the special branch without breaking signatures. Thus Π is property-based CSP-fair.

Π' does not satisfy Theorem 2.2 (simulation-based). Consider a corrupted 0-supporter who always sends its valid certification in the first round and trigger the special branch. Honest parties output 2 with probability 1 (up to any negligible signature forgeability error). In every corresponding ideal execution, $\mathbf{k} = (1, 0, 0)$, regardless of the simulator's reports. The tie-breaking rule in Equation (2) gives $c_{\min}(\mathbf{k}) = 1$. Therefore an ideal output of 2 requires both $c = 2$ and a reply of ok, and

$$\Pr[\mathsf{out}_{\mathrm{real}} = 2] = 1 - \mathsf{negl}(\lambda), \qquad \Pr[\mathsf{out}_{\mathrm{ideal}} = 2] \le \frac{1}{3}.$$

A distinguisher can distinguish the two worlds by checking honest parties' output and has a non-negligible advantage against every simulator. □

The same argument applies with $\mathsf{Leak}_{\mathrm{pub}}$ in place of $\mathsf{Leak}_{\mathrm{pri}}$.

3 A Compiler from Public to Private Preferences

Let $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ be an admissible corruption family as defined in Section 2.3. This section gives a protocol transformation

$$\mathsf{Comp} : \Pi \longmapsto \mathsf{Comp}(\Pi)$$

with the following simulation-preserving guarantee:

$$\Pi[m, \mathsf{mvk}, \mathbf{N}] \text{ realizes } \mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pub}}] \text{ with respect to } \Gamma$$
$$\implies \quad \mathsf{Comp}(\Pi)[m, \mathsf{mvk}, \mathbf{N}] \text{ realizes } \mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pri}}] \text{ with respect to } \Gamma.$$

At a high level, the compiler lets each party i samples a virtual ID vID_i and communicate anonymously using their virtual IDs. The parties run the public preference profile protocol with respect to their virtual IDs and reveal the preference of each virtual ID publicly. We call the preference profile with respect to the virtual IDs as the *masked preference profile*. However, the mapping between parties' real identities and virtual ID are hidden by anonymous communication throughout the protocol, therefore hiding the preference of each party i.

3.1 Building Blocks

Bounded Concurrent Zero-Knowledge Proofs. To achieve malicious security, we require the parties to prove validity of their reported preference. 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 $\mathcal{L}$. The 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 3.1. *(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, $\mathcal{F}^{\mathcal{O}}_{\mathsf{anon}}$ sends Out to every honest party; otherwise, it receives a set $\mathcal{D}$ of at least t number of real corrupted IDs and sends $\mathcal{D}$ to every honest party.

The anonymous broadcast ensures that after at most n/t number of retries, the broadcast eventually succeeds, and all honest parties' messages will appear in the eventual output. Additionally, each party can only send at most one message M_i, and the identity of the sender is hidden.

Lemma 3.2 (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 with virtual IDs, one can understand the virtual ID as session labels: All the zero-knowledge proof messages will be routed using $\mathcal{F}^{\mathcal{O}}_{\mathsf{anon}}$, which itself can be instantiated using $\mathsf{IdealZK}[*, *, i, j]$ between real identities for provers i and verifiers j. In the compiler, we simply set $t = 1$ and ignore the t parameter when describing the ideal functionality.

3.2 The Compiler

Fix a set of parties $\{1, \ldots, n\}$ with preferences. The compiler first forms a masked preference profile using parties' sampled virtual IDs and then emulate the source protocol Π over that masked profile. To make the notation cleaner, we use $\mathcal{F}_{\text{pub}} := \mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pub}}]$ and $\mathcal{F}_{\text{pri}} := \mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pri}}]$.

3.2.1 Formation of the Masked Preference Profile

Certifying Preferences. Every party i samples a fresh virtual identity $\mathsf{vID}_i \xleftarrow{\$} \{0,1\}^\lambda$ and a public-secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \Pi_{\text{Enc}}.\mathsf{Gen}(1^\lambda)$. This key pair will be used for other parties to encrypt messages send to party i. Now the identity parties will be using to get certification and to interact is $\mathsf{ID}_i = (\mathsf{vID}_i, \mathsf{pk}_i)$. It sends $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ to $\mathsf{agg\text{-}auth}$ for its preference $\mathsf{pref}_i \in [m]_0$ and receive auth_i. The certified protocol identity is $ID_i = (vID_i, pk_i)$. When no confusion arises, we use vID_i as a shorthand label for this identity, with pk_i recovered from the masked profile. The resulting certified ordered profile is

$$\mathcal{P} = ((\mathsf{vID}_1, \mathsf{pref}_1), ..., (\mathsf{vID}_n, \mathsf{pref}_n)).$$

The authority publishes $(\mathsf{mvk}, \mathbf{N}(\mathcal{P})) = (\mathsf{mvk}, \mathbf{N})$.

Anonymous Preference Reporting. Each party uses the anonymous broadcast to reveal its certified preference $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$, and use $\mathsf{IdealZK}$ for language $\mathcal{L}$ described at the bottom.

Initialize $\mathcal{O} = [n]$.

1. Repeat the following until the broadcast succeed:

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

   Let Out be the multiset of successfully broadcast messages.

2. Each party i invokes $\mathsf{IdealZK}[x_i, \mathcal{L}, \mathsf{vID}_i, j]$ for each j in $[n]$, with $x_i = (\mathsf{Out}, \mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i, \mathsf{mvk})$ and witness $w_i = (\mathsf{auth}_i, \mathsf{sk}_i)$.

   For any vID_i appeared in Out, if there exists a j such that $\mathsf{IdealZK}[x_i, \mathcal{L}, \mathsf{vID}_i, j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, i.e., the prover fails to prove the statement, remove $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ from Out.

3. Let $\mathcal{P}_O$ be the set of preferences $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ appearing in Out with *distinct* vID_i. Let $(N'_0, ..., N'_{m-1}) = \mathbf{N}(\mathcal{P}_O)$ be the aggregate of $\mathcal{P}_O$ and n' be the size of $\mathcal{P}_O$.

4. For $\ell \in [m]_0$, let $k'_\ell = N_\ell - N'_\ell \geq 0$. Let $\mathcal{P}_I = ((\mathsf{vID}'_k, \perp, \mathsf{pref}'_k))_{k \in [n-n']}$ be a preference profile such that vID'_k is the $(\lambda+1)$-bits representation of k, and that $\mathbf{N}(\mathcal{P}_I) = (k'_0, ..., k'_{m-1})$. Let $\widetilde{\mathcal{P}} = \mathcal{P}_O \cup \mathcal{P}_I$ and sort lexicographically based on virtual IDs. Let $\mathcal{V}_I$ denote the set of virtual IDs in $\mathcal{P}_I$ and $\mathcal{V}$ denote the set of virtual IDs in $\widetilde{\mathcal{P}}$.

Language $\mathcal{L}$: Given a statement $x_i = (\mathsf{Out}, \mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i, \mathsf{mvk})$ where $\mathsf{pref}_i \in [m]_0$, we say $x_i \in \mathcal{L}$ if $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i) \in \mathsf{Out}$ if there exists a witness $(\mathsf{auth}_i, \mathsf{sk}_i)$ such that

- auth_i is a valid signature of message $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ under verification key mvk;

- sk_i is a valid secret key of pk_i.

At the end of the preference reporting phase, each honest party holds the same masked preference $\mathcal{P}$ such that $\mathbf{N}(\widetilde{\mathcal{P}}) = \mathbf{N}(\mathcal{P}) = \mathbf{N}$. Additionally, those whose virtual identities are in $\mathcal{I}$ are corrupted parties who failed to report a certified preference. These parties will be treated as aborting at the beginning of the emulation.

3.2.2 Emulating the Source Protocol Π

The compiler now uses the public masked profile $\widetilde{P}$ to emulate the source protocol Π. At the beginning of emulating the protocol, each party holds its $(\mathsf{vID}_i, \mathsf{pref}_i, \mathsf{auth}_i)$, the public setup information $\mathsf{mvk}, \mathbf{N}$, and a masked profile $\widetilde{\mathcal{P}} = \mathcal{P}_O \cup \mathcal{P}_I$. We assume publicly known polynomials R as the upper bound on the number of rounds and L on the length of each message in Π. Without loss of generality, we assume all messages in Π are of length L after padding.[6]

During the emulation, parties will need to invoke $\mathsf{IdealZK}$ to prove that they compute the encryptions correctly. They use $\mathsf{IdealZK}$ for the language $\mathcal{L}'$ described at the bottom of the box. The compiler emulates $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ as follows.

Treat all virtual IDs in $\mathcal{V}_\mathcal{I}$ as aborting at the beginning of $\Pi[m, \mathsf{mvk}, \mathbf{N}]$.
In each source-protocol round $r = 1, ..., R$,

- Party vID_i computes $(\mathsf{msg}_i^r, \mathsf{out}_i^r) \leftarrow \mathsf{next}_\Pi\big(1^\lambda, \mathsf{vID}_i, r, \mathsf{view}_i^{<r}; \omega_i\big)$, where

  $$\mathsf{msg}_i^r = ((\mathsf{vID}_1, \mathsf{msg}_{i,1}^r), \ldots, (\mathsf{vID}_n, \mathsf{msg}_{i,n}^r), (\mathsf{all}, \mathsf{msg}_{i,\mathsf{all}}^r)).$$

  Party vID_i adds $(\mathsf{msg}_i^r, \mathsf{out}_i^r)$ to $\mathsf{view}_i^{<r+1}$ and computes $\mathsf{ct}_{i,j}^r = \mathsf{Enc}_{\mathsf{pk}_j}(\mathsf{msg}_{i,j}^r; \rho_{i,j}^r)$ for $\mathsf{vID}_j \notin \mathcal{I}$ and $\mathsf{ct}_{i,k}^r = \mathsf{dum}$ for $\mathsf{vID}_k \in \mathcal{V}_\mathcal{I}$. Let

  $$M_i^r = ((\mathsf{vID}_1, \mathsf{ct}_{i,1}^r), \ldots, (\mathsf{vID}_n, \mathsf{ct}_{i,n}^r), (\mathsf{all}, \mathsf{msg}_{i,\mathsf{all}}^r)).$$

- Each party vID_i repeat the following until succeed:

  Sends (vID_i, M_i^r) to $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O}}$. If fails, receive a set $\mathcal{D}$, set $\mathcal{O} = \mathcal{O} \setminus \mathcal{D}$, and retry.

  Let Out^r be the multiset of all successfully broadcast messages.

- Each party i invokes $\mathsf{IdealZK}[x_i', \mathcal{L}', \mathsf{vID}_i, \mathsf{vID}_j]$ for each vID_j in $\mathcal{V}$, with statement $x_i' = (\mathsf{Out}^r, r, \widetilde{P}, \mathcal{I}, \mathsf{vID}_i, M_i^r)$ and witness $w_i = (\mathsf{sk}_i, \{\mathsf{msg}_{i,j}^r; \rho_{i,j}^r\}_{j \in [n]})$.

  For any $(\mathsf{vID}_i, *)$ appeared in Out^r, if there exists a vID_j such that $\mathsf{IdealZK}[x_i', \mathcal{L}', \mathsf{vID}_i, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_i)$, remove the pair (vID_i, M_i^r) from Out^r.

- If $\mathsf{out}_i^r \neq \perp$, party vID_i records that output out_i but continues sending the prescribed dummy message vector through round R.

- Otherwise, for each $(\mathsf{vID}_j, M_j^r, *)$ with distinct vID_j in Out^r, party vID_i computes $\overline{\mathsf{msg}}_{j,i}^r = \mathsf{Dec}_{\mathsf{sk}_i}(\mathsf{ct}_{j,i}^r)$, adds $\overline{\mathsf{msg}}_{j,i}^r$ and $\mathsf{msg}_{j,\mathsf{all}}^r$ in $\mathsf{view}_i^{<r+1}$. If a $\mathsf{vID}_j \in \mathcal{V} \setminus \mathcal{V}_\mathcal{I}$ did not appear in Out^r, then treat it as vID_j abort in round r in Π.

[6] Throughout the construction and proof, self-addressed messages are delivered locally in plaintext; their broadcast slots contain only dum and are omitted from encryption, decryption, and ciphertext-consistency checks.

At the end, each party vID_i outputs its recorded output out_i.

Language $\mathcal{L}'$: Given a statement $x'_i = (\mathsf{Out}^r, r, \widetilde{P}, \mathcal{I}, \mathsf{vID}_i, M_i^r)$, we say that $x'_i \in \mathcal{L}'$ if there exists a witness $w_i = (\mathsf{sk}_i, \{\mathsf{msg}_{i,j}^r, \rho_{i,j}^r\}_{j\in[n]})$ such that the following hold:

- $(\mathsf{vID}_i, \mathsf{pk}_i, *) \in \widetilde{P}$ and that sk_i is a valid secret key of pk_i;
- Parse $M_i^r = ((\mathsf{vID}_1, \mathsf{ct}_{i,1}^r), \ldots, (\mathsf{vID}_n, \mathsf{ct}_{i,n}^r), (\mathsf{all}, \mathsf{msg}_{i,\mathsf{all}}^r))$. The pair $(\mathsf{vID}_i, M_i^r) \in \mathsf{Out}^r$ of round r, and that each $\mathsf{ct}_{i,j}^r = \mathsf{Enc}_{\mathsf{pk}_j}(\mathsf{msg}_{i,j}^r, \rho_{i,j}^r)$ for $\mathsf{vID}_j \notin \mathcal{V}_\mathcal{I}$ and $\mathsf{ct}_{i,k}^r = \mathsf{dum}$ for $\mathsf{vID}_k \in \mathcal{V}_\mathcal{I}$.

3.3 Security of the Compiler

Recall that

$$\mathsf{Leak}_{\mathrm{pub}}(\mathcal{P}_H) = \mathcal{P}_H \qquad \text{and} \qquad \mathsf{Leak}_{\mathrm{pri}}(\mathcal{P}_H) = \mathbf{h} = (h_0, \ldots, h_{m-1}),$$

where h_ℓ is the number of honest ℓ-supporters.

Theorem 3.3 (Security of the compiler)**.** *Assume the certification signature is existentially unforgeable under chosen-message attacks, the public-key encryption is semantically secure. Let $\Gamma = \{\Gamma_\mathbf{N}\}_\mathbf{N}$ be an admissible corruption family. Suppose that the source protocol $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ is position-oblivious and securely realizes $\mathcal{F}_{\mathrm{pub}}$ with respect to Γ. Then $\mathsf{Comp}(\Pi)[m, \mathsf{mvk}, \mathbf{N}]$ securely realizes $\mathcal{F}_{\mathrm{pri}}$ with respect to Γ in the $(\mathcal{F}_{\mathsf{anon}}^\mathcal{O}, \mathsf{IdealZK})$-hybrid model.*

Proof. Fix a non-uniform PPT adversary $\mathcal{A}$ controlling a set $\mathcal{K} \subseteq [n]$, and let $\mathcal{H} = [n] \setminus \mathcal{K}$. We next show a simulator $\mathsf{Sim}_{\mathrm{pri}}$ interacting with $\mathcal{F}_{\mathrm{pri}}$ that simulates $\mathcal{A}$'s view. The simulator works as follows.

Preference reporting. After the honest parties report to $\mathcal{F}_{\mathrm{pri}}$, the simulator receives $\mathsf{Leak}_{\mathrm{pri}}(\mathcal{P}_H) = \mathbf{h}$. Emulate honest parties $i \in \mathcal{H}$ as follows:

- Sample $\mathsf{vID}_i \leftarrow \{0,1\}^\lambda$ and key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for honest parties. If there exists duplicate virtual IDs for honest parties, the simulator aborts.
- Let $\mathcal{V}_\mathcal{H}$ denote the set of honest virtual IDs. Create a public preference profile $\widehat{\mathcal{P}}_\mathcal{H} = \{(\mathsf{vID}_i, \mathsf{pk}_i, \widehat{\mathsf{pref}}_i)\}_{i\in\mathcal{H}}$ such that $\mathbf{N}(\widehat{\mathcal{P}}_\mathcal{H}) = \mathbf{h}$.
- Initialize $\mathcal{O}$ as $[n]$ and emulate $\mathcal{F}_{\mathsf{anon}}^\mathcal{O}$ until it succeeds as follows:

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

  Otherwise, let $\widehat{\mathsf{Out}}$ be the set of messages that are successfully broadcast.
- Emulate the $\mathsf{IdealZK}[*, \mathcal{L}, *, *]$ 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 k: If received $\perp$ from a corrupted verifier, send (fail, k) to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, k)$ to $\mathcal{A}$;
  - For corrupt prover vID_k and honest verifier i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness $w_k = (\mathsf{auth}_k, \mathsf{sk}_k)$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
  - For corrupted prover vID_k and corrupted verifier k', receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

For any vID_k appeared in $\widehat{\mathsf{Out}}$, if there exists a j such that $\mathsf{IdealZK}[x_k, \mathcal{L}, \mathsf{vID}_k, j]$ outputs $(\mathsf{fail}, \mathsf{vID}_k)$, i.e., the prover fails to prove the statement, remove $(\mathsf{vID}_k, \mathsf{pk}_k, \widehat{\mathsf{pref}}_k)$ from $\widehat{\mathsf{Out}}$.

Let $\mathcal{V}_\mathcal{K}$ be the set of corrupted virtual IDs in $\widehat{\mathsf{Out}}$ and

$$\widehat{\mathcal{P}}_\mathcal{K} = \{(\mathsf{vID}_k, \mathsf{pk}_k, \widehat{\mathsf{pref}}_k)\}_{\mathsf{vID}_k \in \mathcal{V}_\mathcal{K}}.$$

Then $\widehat{\mathsf{Out}} = \widehat{\mathcal{P}}_\mathcal{K} \cup \widehat{\mathcal{P}}_\mathcal{H}$. At this moment, for any corrupted $\mathsf{vID}_j \in \mathcal{V}_\mathcal{K}$, the simulator receives a valid witness $w_k = (\mathsf{auth}_k, \mathsf{sk}_k)$.

If $\mathcal{V}_\mathcal{K} \cap \mathcal{V}_\mathcal{H} \neq \emptyset$, i.e., a corrupted virtual ID who passes the proof collides with an honest virtual ID, the simulator aborts.

- Let $\mathcal{P}_O$ be the preferences $(\mathsf{vID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ left in $\widehat{\mathsf{Out}}$. Let $(N'_0, ..., N'_{m-1}) = \mathbf{N}(\mathcal{P}_O)$ be the aggregate of $\mathcal{P}_O$ and n' be the size of $\mathcal{P}_O$. For $\ell \in [m]_0$, let $k'_\ell = N_\ell - N'_\ell \geq 0$.

  Let $\widehat{\mathcal{P}}_\mathcal{I} = \{(\mathsf{vID}'_k, \perp, \mathsf{pref}'_k)\}_{k \in [n-n']}$ be a preference profile such that vID'_k is the $(\lambda+1)$-bits presentation of k, and that $\mathbf{N}(\widehat{\mathcal{P}}_\mathcal{I}) = (k'_0, ..., k'_{m-1})$. Let $\mathcal{V}_\mathcal{I}$ be the set of virtual IDs in $\widehat{\mathcal{P}}_I$

  Let $\widehat{\mathcal{P}} = \widehat{\mathcal{P}}_\mathcal{H} \cup \widehat{\mathcal{P}}_\mathcal{K} \cup \widehat{\mathcal{P}}_\mathcal{I}$ and sort lexicographically based on virtual IDs.

- For any $\mathsf{vID}_k \in \mathcal{V}_\mathcal{K}$, send $(\mathsf{vID}_k, \mathsf{pk}_k, \widehat{\mathsf{pref}}_k, \mathsf{auth}_k)$ to $\mathcal{F}_{\text{pri}}$; send invalid report $\perp$ for any $\mathsf{vID}'_j \in \mathcal{V}_\mathcal{I}$ to $\mathcal{F}_{\text{pri}}$.

- Receive a coin c from $\mathcal{F}_{\text{pri}}$.

Induced Source Protocol Adversary $\mathcal{B}$. Let $\mathcal{V} = \mathcal{V}_\mathcal{H} \cup \mathcal{V}_\mathcal{K} \cup \mathcal{V}_\mathcal{I}$ in $\widehat{\mathcal{P}}$ be the set of identities running the source protocol with respect to a masked preference profile $\widehat{\mathcal{P}}$. Define a PPT source-protocol adversary $\mathcal{B}$ that controls identities in $\mathcal{V}_\mathcal{K} \cup \mathcal{V}_\mathcal{I}$. It takes in an auxiliary input encoding $\mathcal{A}$'s state at the end of the preference-reporting phase and the complete reporting transcript. It starts from $\mathcal{A}$'s state at the end of the preference-reporting phase and implements the following interface between the source execution and its internal copy of $\mathcal{A}$.

- All identities in $\mathcal{V}_\mathcal{I}$ abort at the beginning of the protocol.

- For each source protocol round r, the adversary $\mathcal{B}$ receives for each $\mathsf{vID}_i \in \mathcal{V}_\mathcal{H}$: 1.) the broadcast message $\mathsf{msg}^r_{i,\mathsf{all}}$; and 2.) each private message $\mathsf{msg}^r_{i,k}$ to a corrupted source identity $\mathsf{vID}_k \in \mathcal{V}_\mathcal{K}$.

  It sets $\mathsf{msg}^r_{i,i'} = \mathbf{0}$ for every pair of honest vID_i and $\mathsf{vID}_{i'}$. For each $\mathsf{vID}_i \in \mathcal{V}_\mathcal{H}$ and $\mathsf{vID}_j \notin \mathcal{V}_\mathcal{I}$, $\mathcal{B}$ samples randomness $\rho^r_{i,j}$ and computes $\mathsf{ct}^r_{i,j} = \mathsf{Enc}_{\mathsf{pk}_j}(\mathsf{msg}^r_{i,j}; \rho^r_{i,j})$. For each $\mathsf{vID}_j \in \mathcal{V}_\mathcal{I}$, set $\mathsf{ct}^r_{i,j} = \mathsf{dum}$. It forms M^r_i for each $\mathsf{vID}_i \in \mathcal{V}_\mathcal{H}$.

- Adversary $\mathcal{B}$ then emulate $\mathcal{F}^\mathcal{O}_{\mathsf{anon}}$ and receives $\{(\mathsf{vID}_k, M^r_k)\}_{\mathsf{vID}_k \in \mathcal{V}_\mathcal{K}}$ from $\mathcal{A}$. It uses $\{(\mathsf{vID}_i, M^r_i)\}_{\mathsf{vID}_i \in \mathcal{V}_\mathcal{H}}$ as honest inputs and sends the resulting multiset union $\{(\mathsf{vID}_j, M^r_j)\}_{\mathsf{vID}_j \in \mathcal{V}_\mathcal{H} \cup \mathcal{V}_\mathcal{K}}$ to $\mathcal{A}$. If received a set $\mathcal{D}$ of size at least t, kick out those in $\mathcal{D}$ and retry until succeed. Let the output be Out^r.

- Emulate the $\mathsf{IdealZK}[*, \mathcal{L}', *, *]$ 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_k: If received $\perp$ from a corrupted verifier, send $(\mathsf{fail}, \mathsf{vID}_k)$ to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_k)$ to $\mathcal{A}$;

– For corrupt prover vID_k and honest verifier vID_i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness $w'_k = (\mathsf{sk}_k, \{\mathsf{msg}^r_{k,j}, \rho^r_{k,j}\}_{j\in\mathcal{V}})$ received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.

– For corrupted prover vID_k and corrupted verifier $\mathsf{vID}_{k'}$, receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

For any vID_k appeared in Out^r, if there exists a j such that $\mathsf{IdealZK}[x'_k, \mathcal{L}', \mathsf{vID}_k, \mathsf{vID}_j]$ outputs $(\mathsf{fail}, \mathsf{vID}_k)$, i.e., the prover fails to prove the statement, remove (vID_k, M^r_k) from Out^r, and $\mathcal{B}$ treats vID_k as aborted only if no accepted tuple remains for that identity. Missing tuples also cause source abort.

- Adversary $\mathcal{B}$ then sends $\mathsf{msg}^r_{k,i}$ for each $\mathsf{vID}_k \in \mathsf{Out}^r$ and $\mathsf{vID}_i \in \mathcal{V}_\mathcal{H}$ from the witness w'_k, and broadcasts $\mathsf{msg}^r_{k,\mathsf{all}}$ from the accepted M^r_k.
- Honest termination produces only the prescribed dummy messages through round R, with no extra output notification to $\mathcal{A}$.

At the end of the execution, $\mathcal{B}$ includes only the complete final view and output of its internal copy of $\mathcal{A}$ as a distinguished component of its own view. Let $\mathsf{Proj}_\mathcal{A}(\cdot)$ denote the deterministic polynomial time projection that extracts this component.

By the construction of $\widehat{\mathcal{P}}$, we have $\mathbf{N}(\widehat{P}) = \mathbf{N}$ and that $\mathbf{N}(\mathcal{P}_\mathcal{K} \cup \mathcal{P}_\mathcal{I}) = \mathbf{N} - \mathbf{h}$. If $\mathcal{K}$ is Γ-admissible for $\mathcal{P}$, then $\mathcal{V}_\mathcal{K} \cup \mathcal{V}_\mathcal{I}$ is Γ-admissible for $\widehat{\mathcal{P}}$. For the source-security experiment, place honest records first and corrupted or dummy records last, fixing the corrupted bookkeeping positions; position obliviousness preserves the execution. By security implies that there exists a PPT simulator $\mathsf{Sim}_{\mathrm{pub}}$, such that for any auxiliary input z,

$$\mathsf{Exec}^{\mathcal{F}_{\mathrm{pub}}}_{\mathsf{Sim}_{\mathrm{pub}}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \widehat{P}, z) \approx_c \mathsf{Exec}^{\Pi}_{\mathcal{B}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \widehat{P}, z). \tag{3}$$

Emulation of the Source Protocol The remaining behavior of $\mathsf{Sim}_{\mathrm{pri}}$, after the preference reporting phase, is as follows. It invokes $\mathsf{Sim}_{\mathrm{pub}}$ with auxiliary input z^* encoding $\mathcal{A}$'s state at the end of the preference-reporting phase and the preference-reporting transcript, and internally emulates $\mathcal{F}_{\mathrm{pub}}$ for $\mathsf{Sim}_{\mathrm{pub}}$ as follows:

- Send $\widehat{\mathcal{P}}_\mathcal{H}$ to $\mathsf{Sim}_{\mathrm{pub}}$. $\mathsf{Sim}_{\mathrm{pri}}$ has this profile from the preference reporting phase.
- Receive $\widehat{\mathcal{P}}_\mathcal{K}$ and $\widehat{\mathcal{P}}_\mathcal{I}$ from $\mathsf{Sim}_{\mathrm{pub}}$. The corrupted certified preference-count vector used by the functionality is nevertheless $\mathbf{k} = \mathbf{N} - \mathbf{h}$.
- Send c to $\mathsf{Sim}_{\mathrm{pub}}$. If received ok, send ok to $\mathcal{F}_{\mathrm{pri}}$. Otherwise, send fail to $\mathcal{F}_{\mathrm{pri}}$. Output $\mathsf{Proj}_\mathcal{A}(\mathsf{Exec}^{\mathcal{F}_{\mathrm{pub}}}_{\mathsf{Sim}_{\mathrm{pub}}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \widehat{P}))$.

Claim 3.4. *There exists a negligible function* $\mathsf{negl}(\cdot)$*, such that the probability that the simulator* $\mathsf{Sim}_{\mathrm{pri}}$ *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 $\mathcal{V}_\mathcal{K} \cap \mathcal{V}_\mathcal{H} \neq \emptyset$. 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 $\Pr[\mathcal{V}_{\mathcal{H}} \cap \mathcal{K}\mathsf{id} \neq \emptyset]$ is negligible. We only need to show that conditioned on $\mathcal{V}_{\mathcal{H}} \cap \mathcal{K}\mathsf{id} = \emptyset$, for any non-uniform PPT $\mathcal{A}$, the probability that $\mathcal{V}_{\mathcal{K}} \cap \mathcal{V}_{\mathcal{H}} \neq \emptyset$ is negligible.

Suppose for the sake of contradiction that there exists a non-uniform $\mathcal{A}$ with a non-negligible probability resulting in some $\mathsf{vID}^* \in \mathcal{V}_{\mathcal{K}} \cap \mathcal{V}_{\mathcal{H}}$, conditioned on $\mathcal{V}_{\mathcal{H}} \cap \mathcal{K}\mathsf{id} = \emptyset$. 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,

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

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

Next, we use hybrid argument to show that, conditioned on $\mathsf{Sim}_{\mathrm{pri}}$ not abort,

$$\mathsf{Exec}_{\mathcal{A}}^{\mathsf{Comp}(\Pi)}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}) \approx_c \mathsf{Exec}_{\mathsf{Sim}_{\mathrm{pri}}}^{\mathcal{F}_{\mathrm{pri}}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}).$$

Hybrid Hyb_1: This is an execution of $\mathsf{Comp}(\Pi)[m, \mathsf{mvk}, \mathbf{N}]$, 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 $\mathcal{A}$ 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_k: If received $\perp$ from a corrupted verifier, send $(\mathsf{fail}, \mathsf{vID}_k)$ to $\mathcal{A}$; otherwise, send $(\mathsf{success}, \mathsf{vID}_i, \mathsf{vID}_k)$ to $\mathcal{A}$;
- For corrupt prover vID_k and honest verifier vID_i, send ok to $\mathsf{IdealZK}$ for the honest verifier, and forward $\perp$ or witness w_k received from $\mathcal{A}$ to $\mathsf{IdealZK}$. Send the output of $\mathsf{IdealZK}$ to $\mathcal{A}$.
- For corrupted prover vID_k and corrupted verifier $\mathsf{vID}_{k'}$, receive a bit from $\mathcal{A}$ and send the output of $\mathsf{IdealZK}$.

Output honest parties' outputs and $\mathcal{A}$'s view. We have $\mathsf{Exec}_{\mathcal{A}}^{\mathsf{Comp}(\Pi)}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}) \equiv \mathsf{Hyb}_1$ by definition.

Hybrid Hyb_2: The simulator behaves almost the same as in Hyb_1, except that now the simulator receives leakage $\mathbf{h}$ from the ideal functionality and randomly sample virtual IDs for each honest parties. We have $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_1$ by the anonymity of $\mathcal{F}_{\mathsf{anon}}^{\mathcal{O}}$, the exchangeability of the freshly sampled honest identities and keys under the initialization above, and position obliviousness of the source protocol.

Hybrid Hyb_3: This is almost the same as Hyb_2 except that now all honest-to-honest messages during the emulation encrypts $\mathbf{0}$. Honest source views are updated using the original plaintexts, and corrupted plaintexts are extracted from the $\mathcal{L}'$ witnesses. We have $\mathsf{Hyb}_3 \approx \mathsf{Hyb}_2$ by semantic security of public-key encryption scheme.

Hybrid Hyb_4: Replace the execution after preference-reporting in Hyb_3 by a real source-execution of Π on $\widehat{\mathcal{P}}$ against $\mathcal{B}$ with auxiliary input z^* encoding $\mathcal{A}$'s state at the end of the preference-reporting phase and the preference-reporting transcript. Outputs the honest parties' outputs and $\mathsf{Proj}_{\mathcal{A}}(\mathsf{Exec}_{\mathcal{B}}^{\Pi}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \widehat{P}, z^*))$. We have $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$ by definition of $\mathcal{B}$.

Hybrid Hyb_5: Replace the real source-execution of Π against $\mathcal{B}$ with auxiliary input z^* with the execution generated by $\mathsf{Sim}_{\mathrm{pub}}$ interacting with $\mathbb{F}_{\mathrm{pub}}$. Outputs the honest parties' outputs and $\mathsf{Proj}_{\mathcal{A}}(\mathsf{Exec}^{\mathcal{F}_{\mathrm{pub}}}_{\mathsf{Sim}_{\mathrm{pub}}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \widehat{P}, z^*))$. We have $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$ by Equation (3).

In the rest of the proof we show that $\mathsf{Hyb}_5 \equiv \mathsf{Exec}^{\mathcal{F}_{\mathrm{pri}}}_{\mathsf{Sim}_{\mathrm{pri}}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P})$. The view of $\mathcal{A}$ are identical in both experiments, thus we focus on proving that the honest parties outcome is the same. Since the corrupted parties $\mathcal{K}$ controlled by $\mathcal{A}$ and $\mathcal{V}_{\mathcal{K}} \cup \mathcal{V}_{\mathcal{I}}$ controlled by $\mathcal{B}$ have the same aggregate $\mathbf{k} = \mathbf{N} - \mathbf{h}$, they have the same least preferred outcomes. Therefore, $\mathsf{Sim}_{\mathrm{pub}}$ sends ok, both functionalities deliver c; if it sends fail, both functionalities deliver $c_{\min}(\mathbf{k})$.

Thus, conditioned on $\mathsf{Sim}_{\mathrm{pri}}$ not abort, we have $\mathsf{Exec}^{\mathsf{Comp}(\Pi)}_{\mathcal{A}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P}) \approx_c \mathsf{Exec}^{\mathcal{F}_{\mathrm{pri}}}_{\mathsf{Sim}_{\mathrm{pri}}}(1^\lambda, m, \mathsf{mvk}, \mathbf{N}, \mathcal{P})$. The theorem follows by combining the above and Claim 3.4. □

Corollary 3.5 (Real-world instantiation)**.** *Assume enhanced trapdoor permutations, collision-resistant hash functions, perfectly binding and computationally hiding commitments, unforgeable digital signature under chosen-message attack, semantically secure public-key encryption. Let $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ be an admissible corruption family. There exists a compiler $\mathsf{Comp}_{\mathrm{real}}(\cdot)$ such that for every source protocol $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ that is position-oblivious and securely realizes $\mathcal{F}_{\mathrm{pub}}$ with respect to Γ, Then $\mathsf{Comp}_{\mathrm{real}}(\Pi)[m, \mathsf{mvk}, \mathbf{N}]$ securely realizes $\mathcal{F}_{\mathrm{pri}}$ with respect to Γ.*

Proof. By Theorem 3.3, Theorem 3.1, and Lemma 3.2. □

3.4 Efficiency of the Compiler

We next quantify the overhead introduced by the compiler. Let Π be an R-round source protocol with per-party local computation comp and total communication comm bits. Here, comm is measured after applying the padding convention in Section 3.2.2. Namely, it includes all private and broadcast messages appearing in the source-protocol next-message vectors. Any computation or communication used to instantiate subroutines of the source protocol is also included in comp and comm.

Let $R_{\mathrm{anon}}, \mathsf{comm}_{\mathrm{anon}}(q, L), \mathsf{comp}_{\mathrm{anon}}(q, L)$ denote the round complexity, total communication, and local computation of instantiating $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ among q alive parties and L-bits messages, and $R_{\mathrm{zk}}, \mathsf{comm}_{\mathrm{zk}}, \mathsf{comp}_{\mathrm{zk}}$ be the corresponding cost of instantiating $\mathsf{IdealZK}$.

Round Complexity. Let t be the identifiable-abort parameter of $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$. Each failed invocation identifies and removes at least t corrupted real parties. Consequently, over the entire compiled execution, the number of failed anonymous-broadcast calls satisfies

$$R' \le \frac{|\mathcal{K}|}{t} \le \left\lfloor \frac{n}{t} \right\rfloor.$$

In the compiler described above we take $t = 1$, and hence $R' \le n - 1$.

The compiler invokes $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ at most $R + R' + 1$ times, including one successful call for masked-profile formation, one successful call for each source round, and at most R' failed calls.

It makes at most $n^2(R+1)$ calls to $\mathsf{IdealZK}$: n^2 concurrent calls during masked-profile formation and at most n^2 concurrent calls in each source round. Thus the total round complexity is

$$R^* = R_{\mathrm{anon}}(n,) \cdot (R + 1 + R') + R_{\mathrm{zk}} \cdot (R + 1).$$

Total Communication Complexity. For each source round $r \in [R]$, let

$$C_r := \sum_{i\in[n]} \left(\left|\mathsf{msg}^r_{i,\mathsf{all}}\right| + \sum_{j\in[n]} \left|\mathsf{msg}^r_{i,j}\right| \right),$$

be the total communication complexity in round r, so that $\sum_{r=1}^{R} C_r = \mathsf{comm}$. Assume, for simplicity, that virtual identities and public keys have length $O(\lambda)$ and that encrypting an ℓ-bit plaintext produces a ciphertext of length $\ell + O(\lambda)$. Then the total input length to $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ call used to form the masked preference profile is $B_0 = O\big(n(\lambda + \log m)\big)$, and the total input length of the $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ call emulating source round r is $B_r = O\big(C_r + n^2\lambda\big)$. The additive $n^2\lambda$ term accounts for ciphertext expansion, virtual-identity labels, and the message-vector representation. Define $B_{\max} := \max_{0\le r\le R} B_r$. Since all communication are performed via anonymous broadcast, the total number of bits submitted to $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$, including the failed tries, is at most

$$\begin{aligned}\mathsf{comm}^* &\le \sum_{r=0}^{R} B_r + R' \cdot B_{\max} \\ &= O\left(\mathsf{comm} + Rn^2\lambda + R' \cdot B_{\max}\right).\end{aligned}$$

Thus the total communication is at most $\mathsf{comm}^* + (R' + R + 1)\mathsf{comm}_{\mathrm{anon}}(n, \mathsf{comm}^*) + n^2(R + 1)\mathsf{comm}_{\mathrm{zk}}$.

Local Computation. Let $\mathsf{comp}_{\mathrm{PKE}}$ upper-bound the local computation cost of one encryption/ decryption of a padded source-protocol message. Each party performs one key generation and, in every source round, at most n encryptions and n decryption. Note that parties do not recompute messages from failed tries of $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$. Thus the per-party local computation is

$$\mathsf{comp}^* = O\big(\mathsf{comp} + Rn\,\mathsf{comp}_{\mathrm{PKE}}\big) + (R' + R + 1)\mathsf{comp}_{\mathrm{anon}}(n, \mathsf{comm}^*) + n(R + 1)\mathsf{comp}_{\mathrm{zk}}.$$

We next describe the cost when $\mathcal{F}^{\mathcal{O},t}_{\mathsf{anon}}$ is instantiated using the identifiable-abort anonymous-broadcast construction of [KMSW22]. Their construction runs in constant rounds, among q parties with an L-bit input per party incurs, in the $\mathsf{IdealZK}$-hybrid model, it incurs $O(q^2L)$ number of $\mathsf{IdealZK}$ calls, $O(\lambda q^3 L)$ bits of total communication and $O(q^2 b)$ number of field operations per party. Consequently, the compiled protocol in the $\mathsf{IdealZK}$-hybrid model makes

$$O\big(n^2(R + 1) + n \cdot \mathsf{comm}^*\big)$$

calls to $\mathsf{IdealZK}$ in total, and has total communication $O(\lambda n^2 \mathsf{comm}^*)$
and $O\left(\mathsf{comp} + Rn\,\mathsf{comp}_{\mathrm{PKE}} + n\mathsf{comm}^*\right)$ per-party computation.

Finally, suppose that each invocation of $\mathsf{IdealZK}$ is instantiated by a bounded-concurrent zero-knowledge protocol with a constant number of rounds, total communication $\mathsf{comm}_{\mathsf{zk}}$, and local computation $\mathsf{comp}_{\mathsf{zk}}$ for each participating prover or verifier. Its resulting local computation is therefore

$$O\left(\mathsf{comp} + Rn\,\mathsf{comp}_{\mathrm{PKE}} + n\mathsf{comm}^* + \big(n(R + 1) + \mathsf{comm}^*\big)\mathsf{comp}_{\mathsf{zk}}\right).$$

The resulting total communication is

$$O\left(\lambda n^2 \mathsf{comm}^* + Q_{\mathsf{ZK}}\,\mathsf{comm}_{\mathsf{zk}}\right).$$

Because all zero-knowledge calls belonging to one compiler phase are run concurrently, a constant-round realization of $\mathsf{IdealZK}$ preserves the round bound

$$O\left(R + \frac{n}{t}\right).$$

4 Protocol Securely Realizing $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pub}}]$

For completeness, we prove that a variant of the protocols from [WAS22, TSW24], denoted as $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ with parameters t_0 and t_{oth}, securely realizes $\mathcal{F}_{\text{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\text{pub}}]$ with respect to $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ defined as the intersection of the following three sets:

$$\Gamma^1_{\mathbf{N}} := \{(k_0, ..., k_{m-1}) : \sum_{\ell=0}^{m-1} k_\ell \leq t_0 + t_{\mathsf{oth}} + 1\}; \tag{4}$$

$$\Gamma^2_{\mathbf{N}} := \{(k_0, ..., k_{m-1}) : \sum_{\ell=0}^{m-1} k_\ell \leq N_0 - t_0 + t_{\mathsf{oth}}\}; \tag{5}$$

$$\Gamma^3_{\mathbf{N}} := \{(k_0, ..., k_{m-1}) : k_{\mathsf{oth}} < N_{\mathsf{oth}} - t_{\mathsf{oth}} \text{ or } k_0 \leq k_\ell \text{ for } \ell \in [m]_0\}; \tag{6}$$

$$\Gamma_{\mathbf{N}} = \{(k_0, ..., k_{m-1}) \in \Gamma^1_{\mathbf{N}} \cap \Gamma^2_{\mathbf{N}} \cap \Gamma^3_{\mathbf{N}} : 0 \leq k_\ell \leq N_\ell \text{ for } \ell \in [m]_0\}, \tag{7}$$

where $N_{\mathsf{oth}} = \sum_{\ell=1}^{m-1} N_\ell$ and $k_{\mathsf{oth}} = \sum_{\ell=1}^{m-1} k_\ell$.

At a high level, parties are grouped into G_0 and G_{oth} based on whether their preference is 0. Without loss of generality, we assume $N_0 \leq N_\ell$ for all $\ell \in [m]_0$. Each group G_b for $b \in \{0, \mathsf{oth}\}$ decides a random coin c_b. The final outcome is then determines by c_0 and c_{oth}. The protocol starts with a commit phase in which parties report their preferences. If $N_0 = 0$, then $G_0 = \emptyset$, and the protocol simply lets G_{oth} decide a random coin.

We denote the parties identities in the protocol as ID_i for $i \in [n]$. When simply running a public-preference protocol, this identity ID_i could just be i and distinct for each party; when running as a source protocol for private-preference protocol, this ID_i would be the virtual ID vID_i sampled by parties themselves. Now the identity parties will be using to get certification and to interact is $(\mathsf{ID}_i, \mathsf{pk}_i)$. At the beginning of the protocol, each party holds a certified preference $(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ with signature auth_i from $\mathsf{agg\text{-}auth}$, where party i's preference $\mathsf{pref}_i \in [m]_0$.

Protocol $\Pi[m, \mathsf{mvk}, \mathbf{N}; t_{\mathsf{oth}}]$

Initialize $\mathcal{I}$ as an emptyset.

Commit phase:

1. Each party broadcast $(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$. Let Out be the multiset of all broadcast messages.
2. Each party i invokes $\mathsf{IdealZK}[x_i, \mathcal{L}, \mathsf{ID}_i, \mathsf{ID}_j]$ for each ID_j for $j \in [n]$, with $x_i = (\mathsf{Out}, \mathsf{ID}_i, \mathsf{pk}_i, \mathsf{pref}_i, \mathsf{mvk})$ and witness $\widetilde{w}_i = (\mathsf{auth}_i, \mathsf{sk}_i)$.

   For any ID_i appeared in Out, if there exists an ID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}, \mathsf{ID}_i, \mathsf{ID}_j]$ outputs $(\mathsf{fail}, \mathsf{ID}_i)$, i.e., the prover fails to prove the statement, add ID_i to $\mathcal{I}$ and remove $(\mathsf{ID}_i, \mathsf{pk}_i, \mathsf{pref}_i)$ from Out.
3. Let G_0 be the set of ID_i left in Out with preference 0 and G_{oth} be the set of remaining ID_i in Out. Let $\mathcal{P}_O$ be an ordered preference profile in Out by lexicographic order by ID_i.

Sharing phase:

1. *Share*-0*:* Group G_0 runs the sharing phase of $\mathsf{GroupToss}^0[m, N_0, t_0; \mathcal{P}_O]$.
2. *Share*-oth*:* Group G_{oth} runs the sharing phase of $\mathsf{GroupToss}^{\mathsf{oth}}[m, N_{\mathsf{oth}}, t_{\mathsf{oth}}; \mathcal{P}_O]$.

Reconstruction phase:

1. *Recon*-0*:* Group G_0 runs the reconstruction phase of $\mathsf{GroupToss}^0[m, N_0, t_0; \mathcal{P}_O]$. If the sub-protocol outputs fail, set $c_0 = 0$.

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

The subprotocols $\mathsf{GroupToss}^b[m, N_b, t_b; \mathcal{P}_O]$ are defined as follows in Figure 2.

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

Inputs: Each party i holds its preference pref_i, authentication auth_i, and the aggregate information $(\mathsf{mvk}, \mathbf{N})$ from $\mathsf{agg\text{-}auth}$ and preference profile $\mathcal{P}_O$ from the commit phase.

Parameters: Initializes $\mathcal{I}_b^R$ and $\mathcal{I}_b^S$ as empty sets at the beginning of the protocol.

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

Sharing phase:

- Every $\mathsf{ID}_i \in G_b$ randomly chooses a secret $s_i \xleftarrow{\$} [m]_0$ 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 $\{s_{i,j}\}_{\mathsf{ID}_j\in G_b}$. Let $\widetilde{s}_{i,j} = \mathsf{Enc}_{\mathsf{pk}_j}(s_{i,j}, \rho_{i,j})$ for $\mathsf{ID}_j \in G_b$ and fresh randomness $\rho_{i,j}$.

- Every party broadcast $(\mathsf{ID}_i, \{\widetilde{s}_{i,j}\}_{\mathsf{ID}_j\in G_b})$. Let Out^S be the multiset of broadcast messages. If an $\mathsf{ID}_i \in G_b$ failed to broadcast, add ID_i to $\mathcal{I}_b^S$.

- Each party ID_i invokes $\mathsf{IdealZK}[x_i^S, \mathcal{L}_S, \mathsf{ID}_i, \mathsf{ID}_j]$ with each ID_j for $j \in [n]$, with statement $x_i^S = (\mathsf{Out}^S, \{(\mathsf{ID}_j, \widetilde{s}_{i,j})\}_{\mathsf{ID}_j\in G_b}, \mathcal{P}_O, G_b, t_b)$ and the witness $w_i^S = (s_i, \{s_{i,j}\}_{j\in[N_b]}, \{\rho_{i,j}\}_{\mathsf{ID}_j\in G_b})$ to prove that $x_i \in \mathcal{L}_S$.

  For any $\mathsf{ID}_i \in G_b$, if there exists a ID_j such that $\mathsf{IdealZK}[x_i^S, \mathcal{L}_S, \mathsf{ID}_i, \mathsf{ID}_j]$ outputs $(\mathsf{fail}, \mathsf{ID}_i)$, i.e., the prover fails to prove the statement, add ID_i to $\mathcal{I}_b^S$ and remove $(\mathsf{ID}_i, \{\widetilde{s}_{i,j}\}_{\mathsf{ID}_j\in G_b})$ from Out^S.

Reconstruction phase:

- Parse each entry in Out^S as $(\mathsf{ID}_i, \{\widetilde{s}_{i,j}\}_{\mathsf{ID}_j\in G_b})$ for $\mathsf{ID}_i \in G_b \setminus \mathcal{I}_b^S$. Each $\mathsf{ID}_i \in G_b$ computes $s_{j,i} = \mathsf{Dec}(\mathsf{sk}_i, \widetilde{s}_{j,i})$ and the sum of decrypted shares: $v_i = \sum_{j\in G_b\setminus\mathcal{I}_b^S} s_{j,i}$.

- Broadcast (ID_i, v_i). Let Out^R be the multiset of broadcast messages. If an $\mathsf{ID}_i \in G_b$ failed to broadcast, add ID_i to $\mathcal{I}_b^R$.

- For each $j \in [n]$, party ID_i invokes $\mathsf{IdealZK}[x_i^R, \mathcal{L}_R, \mathsf{ID}_i, \mathsf{ID}_j]$ with statement $x_i^R = (\mathsf{Out}^R, \mathsf{Out}^S, \mathcal{P}_O, G_b, \mathcal{I}_b^S)$ and witness $w_i' = (\{s_{j,i}\}_{\mathsf{ID}_j\in G_b\setminus\mathcal{I}_b^S}, \mathsf{sk}_i)$ to prove that $x_i^R \in \mathcal{L}_R$.

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

- If $|G_b\setminus\mathcal{I}_b^R| < t_b+1$, everyone outputs fail. Otherwise, parse each entry in Out^R as (ID_i, v_i) and use the shares $\{v_i\}_{i\in G_b\setminus\mathcal{I}_b^R}$ to reconstruct the secret $c_b := \sum_{i\in G_b\setminus\mathcal{I}_b^S} s_i$.

Figure 2: Sub-protocol $\mathsf{GroupToss}$

The languages $\mathcal{L}_S$ and $\mathcal{L}_R$ used in the sub-protocol are as follows.

Language $\mathcal{L}_S$

Given a statement $x_i^S = (\mathsf{Out}^S, \{(\mathsf{ID}_j, \widetilde{s}_{i,j})\}_{\mathsf{ID}_j \in G_b}, \mathcal{P}_O, G_b, t_b)$, we say $x_i \in \mathcal{L}_S$ if there exists a witness $w_i = (s_i, \{s_{i,j}\}_{j\in[N_b]}, \{\rho_{i,j}\}_{\mathsf{ID}_j \in G_b})$ such that

- $(\mathsf{ID}_i, \{(\mathsf{ID}_j, \widetilde{s}_{i,j})\}_{\mathsf{ID}_j \in G_b})$ is an entry in Out^S.
- For each $\mathsf{ID}_j \in G_b$, the ciphertext $\widetilde{s}_{i,j}$ is correctly computed using pk_j and $\rho_{i,j}$.
- $\{s_{i,j}\}_{j\in[N_b]}$ forms a valid (t_b+1)-out-of-N_b secret sharing of $s_i \in [m]_0$.

Language $\mathcal{L}_R$

Given a statement $x_i^R = (\mathsf{Out}^R, \mathsf{Out}^S, \mathcal{P}_O, G_b, \mathcal{I}_b^S)$, where $\mathsf{Out}^S = (\mathsf{ID}_i, \{\widetilde{s}_{i,j}\}_{\mathsf{ID}_j \in G_b})$ for $\mathsf{ID}_i \in G_b \setminus \mathcal{I}_b^S$, we say $x_i^R \in \mathcal{L}_R$ if there exists $w_i^R = (\{s_{j,i}\}_{\mathsf{ID}_j \in G_b \setminus \mathcal{I}_b^S}, \mathsf{sk}_i)$ such that

- (ID_i, v_i) is an entry in Out^R.
- For each $\mathsf{ID}_j \in G_b \setminus \mathcal{I}_b^S$, the $s_{j,i} = \mathsf{Dec}(\mathsf{sk}_i, \widetilde{s}_{j,i})$ is the correct decryption of $\widetilde{s}_{j,i}$ under sk_i, which is a valid secret key for pk_i.
- $v_i = \sum_{j \in G_b \setminus \mathcal{I}_b^S} s_{j,i}$.

Theorem 4.1. *Assume the certification signature is existentially unforgeable under chosen-message attacks, the encryption scheme Π_{Enc} is semantically secure, then $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ with parameters t_0 and t_{oth} securely realizes $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pub}}]$ with respect to $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ defined in Equation (7) in the $\mathsf{IdealZK}$-hybrid model. Moreover, $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ is position-oblivious.*

Proof. Position-obliviousness follows from the protocol description that each party's view only depends on its preference and protocol identity ID_i. We will show that for any non-uniform PPT $\mathcal{A}$ that controls a coalition $\mathcal{K}$ with aggregates $\mathbf{k} = (k_0, ..., k_{m-1})$. There exists an adversary Sim interacting with $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, N_0, N_{\mathsf{oth}}]$ such that for any preference profile $\mathcal{P}$ such that $\mathbf{N}(\mathcal{P}) = \mathbf{N}$ and that $\mathcal{K}$ is Γ-admissible with respect to $\mathcal{P}$, $\mathcal{A}$'s view in an execution with $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ is computationally indistinguishable from its view simulated by Sim. For cleaner notation, we use $\mathcal{F}_{\mathrm{pub}}$ to denote $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pub}}]$.

For all the $\mathsf{IdealZK}$ emulation, the simulator acts as follows:

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

Commit phase: Simulator receives $\mathcal{P}_{\mathcal{H}}$ from $\mathcal{F}_{\mathrm{pub}}$. Emulate honest parties $i \in \mathcal{H}$ as follows:

- Send $\mathcal{P}_{\mathcal{H}}$ to $\mathcal{A}$ and receives $\mathcal{P}_{\mathcal{K}}$ from $\mathcal{A}$. Let $\mathsf{Out} = \mathcal{P}_{\mathcal{H}} \cup \mathcal{P}_{\mathcal{A}}$ be the broadcast preferences.
- Emulate the $\mathsf{IdealZK}[*, \mathcal{L}, *, *]$. For any ID_k appeared in Out, if there exists an ID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}, \mathsf{ID}_k, \mathsf{ID}_j]$ outputs $(\mathsf{fail}, \mathsf{ID}_k)$, i.e., the prover fails to prove the statement, add ID_k

to $\mathcal{I}$ and remove $(\mathsf{ID}_k, \mathsf{pk}_k, \mathsf{pref}_k)$ from $\mathcal{P}_{\mathcal{K}}$. Let $\mathcal{K}'$ be the set of corrupted IDs in $\mathcal{P}_{\mathcal{K}}$ but not in $\mathcal{I}$. If $\mathcal{K}' \cap \mathcal{H} \neq \emptyset$, the simulator aborts.

Otherwise, let $\mathcal{P}_O = \mathcal{P}_{\mathcal{H}} \cup \mathcal{P}_{\mathcal{K}}$ be the remaining valid preferences. At this moment, for any corrupted $\mathsf{ID}_j \in \mathcal{K}'$, the simulator receives a valid witness $w_j = (\mathsf{auth}_j, \mathsf{sk}_j)$ by emulating $\mathsf{IdealZK}$.

- For any $\mathsf{ID}_j \in \mathcal{K}'$, send $(\mathsf{ID}_j, \mathsf{pk}_j, \mathsf{pref}_j, \mathsf{auth}_j)$ to $\mathcal{F}_{\text{pub}}$; for other corrupted parties, send $(\mathsf{ID}'_j, \perp, \perp, \perp)$ to $\mathcal{F}_{\text{pub}}$.

The simulator gets a coin c from the ideal functionality.

Let $\mathcal{H}_b = \mathcal{H} \cap G_b$ and $\mathcal{K}_b = \mathcal{K}' \cap G_b$ to denote the set of honest and corrupted b-supporters for $b \in \{0, \mathsf{oth}\}$ whose preference is b in $\mathcal{P}_O$, respectively. At a high-level, those who have $\perp$-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 $\mathcal{I}_b^S$ and $\mathcal{I}_b^R$ 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 $\mathsf{GroupToss}^b[m, N_b, t_b, \mathcal{P}_O]$. If $|G_b| \leq t_b$, outputs fail for $\mathsf{GroupToss}^b[m, N_b, t_b, \mathcal{P}_O]$.

If $\mathsf{GroupToss}^0[m, N_0, t_0, \mathcal{P}_O]$ 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 fail to $\mathcal{F}_{\text{pub}}$. 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 $\mathsf{ID}_{i'} \in \mathcal{H}_b$, 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 $\mathsf{ID}_j \in G_b$, compute $\widetilde{s}_{i,j} = \mathsf{Enc}(\mathsf{pk}_j, s_{i,j})$.
- *Broadcast:* Each party ID_i broadcast $(\mathsf{ID}_i, \{(\mathsf{ID}_j, \widetilde{s}_{i,j})\}_{\mathsf{ID}_j \in G_b})$. Let Out^S be the multiset of broadcast messages.
- *Validation:* Emulate $\mathsf{IdealZK}[*, \mathcal{L}_S, *, *]$. For any $\mathsf{ID}_k \in G_b$, if there exists a ID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_S, \mathsf{ID}_k, \mathsf{ID}_j]$ outputs $(\mathsf{fail}, \mathsf{ID}_k)$, i.e., the prover fails to prove the statement, add ID_k to $\mathcal{I}_b^S$.

Reconstruction phase: At this point, for any $\mathsf{ID}_j \in \mathcal{K}_b \setminus \mathcal{I}_b^S$ 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 $\mathsf{ID}_i \in \mathcal{H}_0$ 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 $k_{\mathsf{oth}} < t_{\mathsf{oth}} + 1$: 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:* Party ID_i broadcast (ID_i, v_i). Let Out_0^R denote the multiset of broadcast messages.
3. *Validation:* Emulate the $\mathsf{IdealZK}[*, \mathcal{L}_R, *, *]$. For any $\mathsf{ID}_k \in G_0$, if there exists a ID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_R, \mathsf{ID}_k, \mathsf{ID}_j]$ outputs $(\mathsf{fail}, \mathsf{ID}_i)$, i.e., the prover fails to prove the statement, add ID_k 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 Equation (7). 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 $c_0 + (\sum_{i \in G_{\mathsf{oth}} \setminus \mathcal{I}_{\mathsf{oth}}^S} s_i) = c \mod m$. 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 $v_i = \sum_{j \in \mathcal{K}_{\mathsf{oth}} \setminus \mathcal{I}_{\mathsf{oth}}^S} s_{j,i} + \sum_{j \in \mathcal{H}_{\mathsf{oth}}} y_{j,i}$.
2. *Broadcast:* Party ID_i broadcast (ID_i, v_i). Let $\mathsf{Out}_{\mathsf{oth}}^R$ denote the multiset of broadcast messages.
3. *Validation:* Emulate the $\mathsf{IdealZK}[*, \mathcal{L}_R, *, *]$. For any $\mathsf{ID}_i \in G_{\mathsf{oth}}$, if there exists a ID_j such that $\mathsf{IdealZK}[x_i, \mathcal{L}_R, \mathsf{ID}_i, \mathsf{ID}_j]$ outputs $(\mathsf{fail}, \mathsf{ID}_i)$, i.e., the prover fails to prove the statement, add ID_i to $\mathcal{I}_{\mathsf{oth}}^R$.
4. *Reconstruction:* For those corrupted parties ID_j that failed to broadcast a valid proof, add ID_j to $\mathcal{I}_{\mathsf{oth}}^R$. Reconstruct c_{oth} using $\{v_i\}_{i \in G_{\mathsf{oth}} \setminus \mathcal{I}_{\mathsf{oth}}^R}$. If reconstruction succeeds, send ok to $\mathcal{F}_{\mathrm{pub}}$. Otherwise, send fail to $\mathcal{F}_{\mathrm{pub}}$.

   Output whatever $\mathcal{A}$ outputs.

The probability of simulator aborts is negligible since otherwise the adversary $\mathcal{A}$ is able to break the unforgeability of the signature scheme used by $\mathsf{agg\text{-}auth}$. We now show that, conditioned on the simulator not abort, the joint distribution of honest parties' output and the adversary's view is indistinguishable in the ideal execution when interacting with Sim (denoted as $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Ideal}}$) is identical to the joint distribution of the output of honest parties and the view of the adversary in a real execution (denoted as $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real}}$), conditioned on the simulator not abort. Consider the following hybrids:

Hybrid Hyb_1: This is an execution of $\Pi[m, \mathsf{mvk}, \mathbf{N}; t_{\mathsf{oth}}]$, 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 specified earlier. We have $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Real}} \equiv \mathsf{Hyb}_1$ by definition.

Hybrid Hyb_2: The simulator behaves almost the same as in Hyb_1 except the following: In Share-b step, the simulator computes the shares for honest parties $\mathsf{ID}_i \in \mathcal{H}_b$ as in Hyb_1, but the encryptions $\{\widetilde{s}_{i,j}\}_{\mathsf{ID}_j \in G_b}$ are computed differently. For $b \in \{0, \mathsf{oth}\}$ with $k_b < t_b + 1$ (all other ciphertexts remain unchanged),

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

Honest reconstruction still uses the original shares. We have $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_1$ by semantic security of public-key encryption scheme.

Next, we show that $\mathsf{Hyb}_2 \equiv \mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A}}$, 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.

If the reconstruction of c_{oth} fails, and Sim sends fail to $\mathcal{F}_{\mathrm{pub}}$. Then $\mathcal{F}_{\mathrm{pub}}$ uses $\mathbf{k} = \mathbf{N} - \mathbf{h}$ to determine the least preferred outcome of $\mathcal{K}$, which is 0 by Equation (6). 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 $\mathcal{F}_{\mathrm{pub}}$. There are two cases:

- If $k_{\mathsf{oth}} \geq t_{\mathsf{oth}} + 1$, it must be the case that $k_0 < t_0 + 1$ by Equation (4), and c_0 must be successfully reconstructed by Equation (5). 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 $k_{\mathsf{oth}} < t_{\mathsf{oth}} + 1$, 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, $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A}} \approx \mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}$. □

Corollary 4.2 (Real-world public-preference protocol)**.** *Assume enhanced trapdoor permutations, collision-resistant hash functions, perfectly binding and computationally hiding commitments, unforgeable digital signature under chosen-message attack, semantically secure public-key encryption. Let $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ be an admissible corruption family defined in Equation (7). Then the above protocol $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ securely realizes $\mathcal{F}_{\mathrm{pub}}$ with respect to Γ.*

Proof. By Theorem 4.1 and Theorem 3.1. □

Corollary 4.3 (Real-world private-preference protocol)**.** *Assume enhanced trapdoor permutations, collision-resistant hash functions, perfectly binding and computationally hiding commitments, unforgeable digital signature under chosen-message attack, semantically secure public-key encryption. Let $\Gamma = \{\Gamma_{\mathbf{N}}\}_{\mathbf{N}}$ be an admissible corruption family defined in Equation (7). Then applying the compiler in Section 3.2 on the above $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ securely realizes $\mathcal{F}_{\mathrm{pri}}$ with respect to Γ.*

Proof. By Corollary 3.5 and Corollary 4.2. □

Corollary 4.4 (Parameter selection of t_0 and t_{oth})**.** *Assume the assumptions in Corollary 4.3. With the parameter selection in Table 1, for $m = 2$, applying the compiler in Section 3.2 on the above $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ securely realizes $\mathcal{F}^{\mathsf{auth}}_{\mathrm{toss}}[m, \mathsf{mvk}, \mathbf{N}; \mathsf{Leak}_{\mathrm{pri}}]$ with respect to $\Gamma_{\mathbf{N}}$*

$$\Gamma^{\mathrm{thresh}}_{\mathbf{N}} = \{(k_0, k_1) : k_0 + k_1 \leq t\},$$

which is optimal by [WAS22].

For $m > 2$, the parameter choices in Table 1 gives the admissible corruption family in Theorem 1.2.

Case	t_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 $\Pi[m, \mathsf{mvk}, \mathbf{N}]$ instantiating $\mathcal{F}_{\mathrm{toss}}^{\mathsf{auth}}$.

Proof. When $m = 2$, the protocol $\Pi[2, \mathsf{mvk}, \mathbf{N}]$ becomes a variant of the protocol proposed in [WAS22]. When choosing t_0 and t_{oth} as in Table 1, the admissible corruption structure in Equation (7) contains the above threshold structure. the optimality follows by [WAS22]. □

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

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, January 2000. 1.2

[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, 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, 1.3

[CGL+18] 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, 1.3, 2.3

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988. 1.4

[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, 1.3, 1.4, 3.2, 3.4

[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

[Lin03] Yehuda Lindell. General composition and universal composability in secure multiparty computation. Cryptology ePrint Archive, Paper 2003/141, 2003. 1.2

[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, 1.4, 3.1, 3.1

[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. 1.4

[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, 1.1.4, 1.1.4, 4, 1.3, 1.4, 1.4, 4

[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, 1.1.4, 1.3, 1.4, 1.4, 1.4, 4, 4.4, 4

[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