Asymptotically Optimal Adaptive Asynchronous Common Coin and DKG with Silent Setup

Hanwen Feng* Qiang Tang*

The University of Sydney

Abstract. This paper presents the first optimal-resilient, adaptively secure asynchronous common coin protocol with $O(\lambda n^2)$ communication complexity and $O(1)$ rounds, requiring only a public silent setup. Our protocol immediately implies a sequence of quadratic-communication, constant-round asynchronous Byzantine agreement protocols and asynchronous distributed key generation with a silent setup. Along the way, we formulate a new primitive called asynchronous subset alignment and introduce a simple framework to reason about specific composition security suitable for asynchronous common coin, which may be of independent interest.

1 Introduction

Coin flipping [15], also known as a common coin, is a classical problem with significant modern applications [24, 32, 38, 48, 53, 60, 67]. At a high level, a group of n parties jointly execute a common coin protocol, where all participants agree on one value that is both unpredictable to the adversary and indistinguishable from a uniformly sampled value. A closely related problem is distributed key generation (DKG) [7, 22, 37, 47, 54, 58, 71], in which each of the honest participants obtains a share of a secret key. It is well known that a DKG protocol immediately yields a common coin protocol.

In this work, we investigate the multiparty coin-flipping (and DKG) problem in an *asynchronous setting*, where an adversary has the power to arbitrarily delay and reschedule messages between honest participants. Our primary focus is on improving the communication and round complexities of the protocol relative to the group size n. Beyond its theoretical interest, asynchronous coin-flipping (and DKG) protocols are a critical component of asynchronous consensus protocols [4, 52, 53, 67, 70], which are central to blockchain applications [14, 46]. These blockchain systems often involve a large number of participants, making scalability a key concern.

There has been a recent surge of interest in asynchronous common coins [10, 27, 75] and DKG protocols [2, 3, 29, 42, 61]. Unfortunately, *all* of them suffer from high communication complexity of at least $O(n^3)$. The only exception is the seminal work of Cachin, Kursawe, and Shoup [20], which achieves $O(\lambda n^2)$ via a unique threshold signature, but this inherently requires a *private* setup. [1] In a private setup, a trusted party disperses correlated secrets to each participant, which is significantly stronger than a common public setup, such as a public bulletin board, CRS, or PKI. Besides the drawbacks of private setups in distributed systems, the basic motivation of DKG is to eliminate such private setups. The following major question remains open,

Can we construct an asynchronous common coin (and DKG) protocol, that is with only $O(\lambda n^2)$ communication, secure against strongly adatpive adversary corrupting up to $\lceil \frac{n}{3} \rceil - 1$ parties, but without relying on private setups?

[1] We summarize all existing asynchronous coin-flipping, including DKG protocols, in Table 1, which clearly demonstrates the current status. Here, λ is the computational security parameter, denoting length of the coin, or size of signature etc. More detailed explanations of broader developments are provided in Appendix A.

Table 1. Comparison with optimal-resilient *asynchronous* common coin protocols[2]

Protocol	Type	Comm.Cost	Round	Adaptive?	Assumption	Setup
CKS [20]	`Strong`	$O(\lambda n^2)$	1	✓ [†]	RO,DDH,co-CDH [†]	Private Setup
DDL+ [27]	`Weak`	$O(\lambda n^3)$	$O(1)$	✗	RO	SecChan
FKT [75]	`Weak`$^+$	$O(\lambda n^3 \log n)$	$O(1)$	✗	DLog	SecChan
BBB+ [10]	`Weak`$^+$	$O(\lambda n^3 \log n)$ [‡]	$O(1)$	✓	RO	SecChan
KMS [61]	`Strong`	$O(\lambda n^4)$	$O(n)$	✗	RO,DDH	PKI
DYX+ [31]	`Strong`	$O(\lambda n^3)$	$O(\log n)$	✗	RO,DDH	PKI
AJM+ [3]	`Strong`	$O(\lambda n^3)$	$O(1)$	✓ [*]	RO,AGM,COMDL	PKI
GLL+ [42]	`Strong`	$O(\lambda n^3)$	$O(1)$	✓ [*]	RO,AGM,COMDL	PKI
Bingo [2]	`Strong`	$O(\lambda n^3)$	$O(1)$	✓	RO,AGM,q-SDH	CRS, SecChan
Ours	`Strong`	$O(\lambda n^2)$	$O(1)$	✓	RO, GGM	CRS,PKI[**]

– `Type`: `Strong` coins ensure *agreement* with $1 - \mathsf{negl}(\lambda)$ probability. `Weak` coins ensure agreement with a constant probability. `Weak`$^+$ coins ensure agreement with $(1 - \mathsf{negl}(\lambda) - \delta)$ probability for some adjustable small δ.
– `Comm.Cost` measures the *expected* number of bits sent by all honest nodes, where λ is the security parameter; The costs of CKS and `Weak` or `Weak`$^+$ coins are exact.
[‡] BBB+ supports batching, so that after amortization, the communicaiton cost for each coin can be reduced to $O(\lambda n^2 \log n)$. We measured the cost of a single-shot coin generation in the table.
– `Round` measures the *expected* number of rounds within which all honest nodes can terminate. The rounds of CKS and `Weak` or `Weak`$^+$ coins are exact.
– `Adapptive?` asks if the protocol is strongly adaptive secure [1].
[†]CKS is strongly adaptive secure if using an adaptively secure unique threshold signature scheme, for example, the scheme from [28] under DDH and co-CDH assumptions.
[*]AJM+ and GLL+ proved the static security of their protocol, but their results can be adaptively secure by using a component from [9].
– `Assumption`: RO: Random Oracle; AGM: Algebraic Group Model [39]; GGM: Generic Group Model [66, 74]; q-SDH: q-Strong Diffie-Hellman Assumption [16, 57]; COMDL: Co-One-More Discrete Logarithm Assumption [9].
– `Setup`: SecChan: Secure Channels; CRS: Common Reference String. Note that [2, 3, 31, 42, 61] may also need a uniformly distributed CRS, which can be implied by RO.
[**] Our instantiation requires $O(\lambda n)$-sized public keys due to the current construction of underlying primitive from [45], incurring $O(\lambda n^3)$ setup communication due to reading all keys. This cost can be immediately reduced to $O(\lambda n^2)$ with the help of a key curator [43], which can deterministically (and even verifiably) aggregate the public keys before returning them to participants. See Remark 2 for details.

1.1 Our Contributions

In this work, we answer this question affirmatively.

Primary result: asymptotically optimal common coin without private setup. We present an asynchronous *strong* common coin protocol, ensuring that, except with negligible probability, all honest nodes can output the same value in every instance. This value is both unpredictable and unbiased, even in the presence of a computationally bounded adversary capable of adaptively corrupting up to $f = \lceil \frac{n}{3} \rceil - 1$ nodes. The communication complexity and round complexity of our

[2] This table only includes computational asynchronous coin protocols, there are also asynchronous coins against information theoretic adversary, which, naturally have even higher complexity. Also, there are many research on synchronous common coin and DKG protocols, most of them are also with cubic communication; only until very recently, we have the first synchronous coin and DKG protocols with subcubic communication [7, 37], still, both of them are not yet matching $O(\lambda n^2)$ communication, and with $O(n)$ rounds.

protocol are $O(\lambda n^2)$ and $O(1)$ in expectation, respectively. Our protocol asymptotically matches the performance of Cachin et al.'s common coin [20] but requires only a silent setup. [3]

Direct implications: optimal Asynchronous Byzantine Agreements without private setup and near-optimal ADKG protocol. All existing asymptotically optimal asynchronous consensus protocols assume a quadratic communication asynchronous coin (or use [20] directly via private setup). This includes the binary version (ABA) [69], the multi-valued version (MBA) [70], the multi-valued validated version (MVBA) [65], and the asynchronous common subset (ACS) [20, 65]. Our common coin protocol can be directly plugged in, providing the first set of these asynchronous consensus protocols without requiring a private setup.

Furthermore, very recently, [36] introduced an ADKG protocol in the *coin-aided* model, [4] which enjoys strong adaptive security (assuming adaptively secure coins), $O(\kappa\lambda n^2)$ communication complexity, and $O(1)$ round complexity, where κ and λ are the statistical and computational security parameters, respectively. Instantiating the coin oracle in [36] with our common coin protocol directly yields a full-fledged ADKG protocol with the same asymptotic complexity under a silent setup. In contrast, all existing ADKG protocols [2, 3, 31, 42, 61], as outlined in Table 1, require at least $O(\lambda n^3)$ communication cost.

A simple framework for analyzing specific composition security. Since our common coin protocols involve several sub-protocols as components, we need to ensure that all components remain secure when composed. Most previous works on computational asynchronous common coins appear to have overlooked this issue. While the security properties of each component are defined and analyzed in a stand-alone setting, their security in composition is often taken for granted, and the final proofs are built upon this implicit assumption. [5]

We introduce a simple framework for analyzing the composability of specific protocols (avoiding the strong UC framework, which might be too heavy; see Sect.4.2 for further discussion), which may be of independent interest. At a high level, we abstract the information an adversary could possibly learn from a protocol instance Π_A as an oracle $\mathcal{O}$. If a protocol instance Π_B remains secure even when the adversary has access to $\mathcal{O}$, then Π_B is secure in composition with Π_A. Our analysis method is essentially a generalization of the work by Lindell, Lysyanskaya, and T. Rabin [64]. Details are presented in Sect.4.2.

2 Technical Overview

Our goal is to construct a communication-optimal strong common coin in the asynchronous setting, which, except with negligible probability, ensures the following properties: (1) *termination*, meaning that all honest nodes eventually terminate with a value; (2) *agreement*, meaning that all honest nodes terminate with the same value; and (3) *unpredictability* and *bias-resistance*, meaning that the agreed value is unpredictable and pseudorandom to the adversary.

[3] In a silent setup, each party only posts *once* their public key (e.g., writes to the public bulletin board or sends it to a trusted party). This generalizes the conventional PKI. At the same time, a silent setup remains a public setup that disallows the distribution of secrets to individual parties, as discussed in Remark 1.

[4] We sketch this result in Appendix E.

[5] Indeed, we show a concrete construction of asynchronous verifiable secret sharing (AVSS) [23, 30] (a primitive fundamental to most existing asynchronous coin protocols), which satisfies all properties in the stand-alone setting but becomes *insecure* when two instances of it are run together (see Sect.4.2). While our example does not imply that existing constructions are insecure, particularly when their components indeed provide security guarantees beyond the stand-alone setting, the security analysis does require more careful attention.

In the following, we walk through the challenges and our solutions.

Existing construction methods. Most existing asynchronous coins start with a *weak* coin, where both *agreement* and *unpredictability* and *bias-resistance* may hold only with constant probability. They construct certain asynchronous Byzantine consensus protocols based on a weak common coin protocol, enabling the application of conventional "commit-and-reveal" methods to achieve a strong common coin.

A weak coin, in turn, is typically constructed through a delicate combination of an information gathering (Gather) primitive [3, 23, 27] and a random ranking mechanism [3, 23, 27, 42] (Steps 1-3 below).

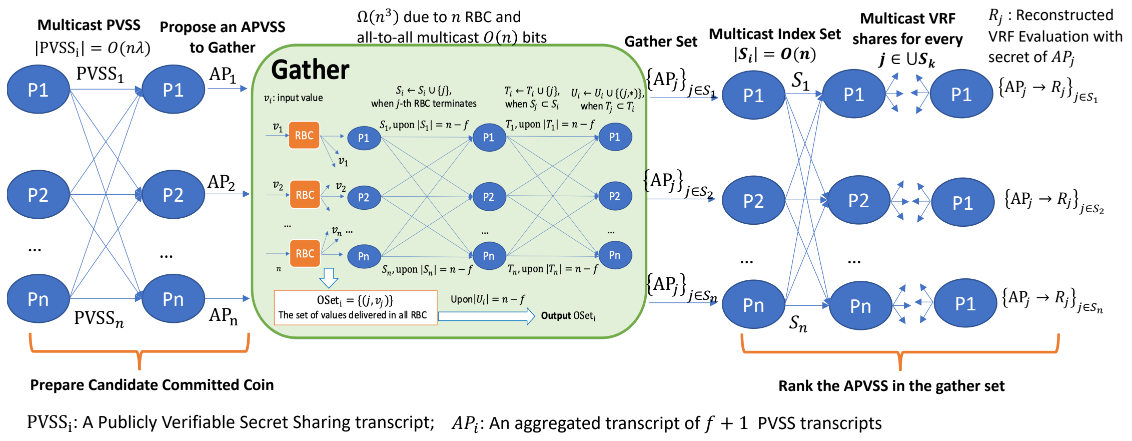

Multicast PVSS
$|\mathrm{PVSS_i}| = O(n\lambda)$
Propose an APVSS to Gather
$\Omega(n^3)$ due to n RBC and all-to-all multicast $O(n)$ bits
Gather Set
Multicast Index Set $|S_i| = O(n)$
Multicast VRF shares for every $j \in \cup S_k$
R_j : Reconstructed VRF Evaluation with secret of AP_j
PVSS_1
PVSS_n
AP_1
AP_2
AP_n
Gather
v_i: input value
$S_i \leftarrow S_i \cup \{j\}$, when j-th RBC terminates
$T_i \leftarrow T_i \cup \{j\}$, when $S_j \subset S_i$
$U_i \leftarrow U_i \cup \{(j,*)\}$, when $T_j \subset T_i$
S_1, upon $|S_1| = n - f$
T_1, upon $|T_1| = n - f$
S_n, upon $|S_n| = n - f$
T_n, upon $|T_n| = n - f$
RBC
$\mathrm{OSet_i} = \{(j, v_j)\}$
The set of values delivered in all RBC
Upon $|U_i| = n - f$
Output $\mathrm{OSet_i}$
$\{\mathrm{AP}_j\}_{j\in S_1}$
$\{\mathrm{AP}_j\}_{j\in S_2}$
$\{\mathrm{AP}_j\}_{j\in S_n}$
S_1
S_n
$\{\mathrm{AP}_j \rightarrow R_j\}_{j\in S_1}$
$\{\mathrm{AP}_j \rightarrow R_j\}_{j\in S_2}$
$\{\mathrm{AP}_j \rightarrow R_j\}_{j\in S_n}$
P1
P2
Pn
Prepare Candidate Committed Coin
Rank the APVSS in the gather set
$\mathrm{PVSS_i}$: A Publicly Verifiable Secret Sharing transcript; AP_i: An aggregated transcript of $f+1$ PVSS transcripts

Fig. 1. The Weak Common Coin Protocol from Abraham et al. [3]

- **Step 1: Prepare a "committed candidate coin."** Even in an asynchronous setting, every node can still receive at least $n - f$ commitments provided by distinct nodes, and any subset of $f + 1$ commitments is sufficient to define a "candidate" random coin. At least one commitment originates from an honest node, ensuring that the final aggregated value remains unpredictable to the adversary. In existing works, every node is responsible for forming such a "committed candidate coin" (referred to simply as a "candidate" hereafter), which can be naively described by a set of $f + 1$ commitments received by the node or an aggregation of those commitments. [6] Note that alternative instantiations, such as the scheme in [42], also exist; however, the essence of this step is that each node proposes a committed random value, with the committed value being unpredictable even to the node that proposed the commitment.
- **Step 2: Gather.** This is typically the most complicated component. At this stage, every node has a candidate, and the next question is which candidate the network will decide on. Ideally, if every honest node could see the same subset of candidate committed coins, they could apply a deterministic rule to pick one and open it as the coin. However, achieving this requires asynchronous consensus—which itself requires a coin to begin with. The *Gather* mechanism is introduced to address the absence of full-fledged asynchronous consensus. *Gather* cannot ensure

[6] For example, if the underlying "commitment" supports aggregation, such as using aggregatable publicly verifiable secret sharing as in [3].

that the subsets received by different honest nodes are identical, but it guarantees the existence of a large common subset:

  - *Binding Core*: Let $\{\mathsf{OSet}_i\}_{i\in\mathcal{H}}$ represent the output sets of all honest nodes $\{P_i\}_{i\in\mathcal{H}}$. There exists a *core* set of $n-f$ values that is a subset of **every** OSet_i for $i \in \mathcal{H}$ (where $\mathcal{H}$ is the index set of all honest nodes). Furthermore, the core set is *binding*, meaning it is already determined at the moment the first honest node generates its output set.

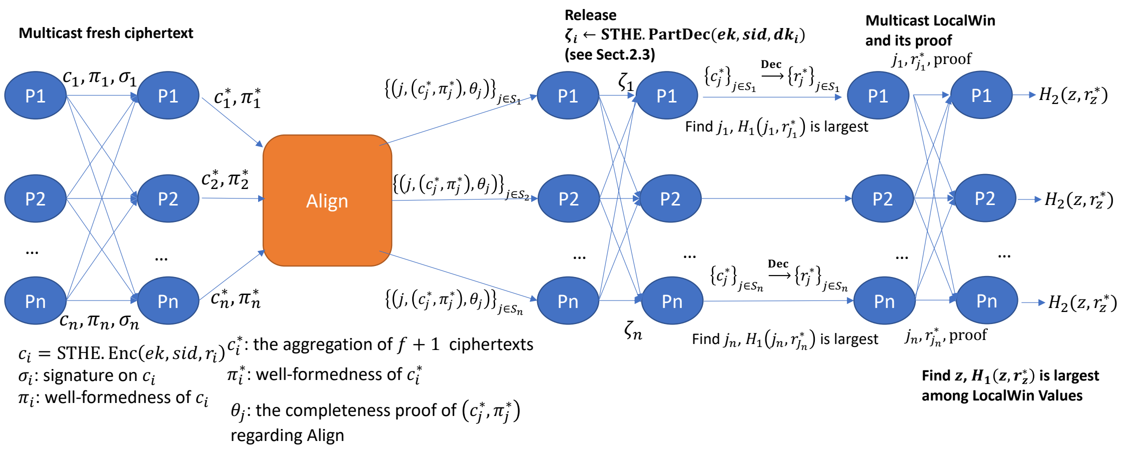
  In addition to the conventional termination and validity requirements, *Gather* ensures that every node can propose at most *one* value (in this application, a candidate).

- **Step 3: "Rank" the candidates.** At this stage, there are at most n candidates, with $n-f$ appearing in the output sets of all honest nodes. In this step, the network ranks these candidates such that every node simply decides on the commitment with the highest rank in its output set. The ranking process is deterministic yet unpredictable before a certain point, ensuring a probability of roughly $\frac{n-f}{n}$ that the highest-ranked commitment is in the core set. If this occurs, every honest node will decide on it and take its opening as the common coin.
  Care must be taken to ensure that the ranking process does not grant the adversary additional power to create a committed candidate outside the core set with a high rank [3, 27, 42].
- **Step 4: Compiling a weak coin into a strong coin.** After constructing a weak common coin, we need to build certain asynchronous consensus protocols that can imply an asynchronous "bulletin board," such as multivalued validated Byzantine agreement (MVBA) [3, 4, 19, 65]. This step represents the final technical challenge. While conventional MVBA protocols assume the existence of a strong common coin, weak-coin-based constructions with special care or alternative designs [2,3,42] are often employed.

Cubic communication in *each* step. Unfortunately, *every step* described above incurs $\Omega(n^3)$ communication complexity.

- *Cost for candidate committed coins.* In existing protocols, the "commitment" scheme supporting forced opening is realized using (a few variants of) asynchronous verifiable secret sharing (AVSS) [2,30]. An adaptively secure AVSS incurs $O(\lambda n^2)$ communication cost, leading to $\Omega(\lambda n^3)$ cost for preparing candidate committed coins, as $\Omega(n)$ instances of AVSS are utilized.
- *Cost for Gather.* The $\Omega(n^3)$ cost in Gather arises for *multiple* reasons. First, if the size of a candidate is $O(n)$, then Gather incurs $\Omega(n^3)$ cost because each node receives n candidates, resulting in $\Omega(n^2)$ bits. Second, the Gather protocols in [3, 27] involve n instances of reliable broadcast (RBC) [18] and an instance of a so-called index-Gather protocol [27] to determine which RBC instances should be included in the output set. Both the n RBC instances and the index-Gather incur $\Omega(n^3)$ communication cost. Notably, this $\Omega(n^3)$ term persists even when the candidate size is reduced to $O(1)$.
  We also note that a weaker form of Gather, known as the Weak Core Set, has been studied in [42], which only requires that the output sets of $f+1$ honest nodes contain the binding core. However, the above arguments regarding the $\Omega(n^3)$ communication cost still apply.
- *Cost for ranking.* The ranking step requires opening *all* $O(n)$ candidates. Even if each node multicasts only $O(\lambda)$ bits for each candidate, the total cost still amounts to $O(\lambda n^3)$.
- *Cost for converting a weak coin to a strong coin via MVBA.* Most existing MVBA protocols fall into two categories: (1) Protocols such as [3,27] that use a weak leader election subroutine, which, with constant probability, selects the same leader ι for all nodes (and thus can be implemented using a weak common coin). However, the corresponding MVBA schemes still incur $\Omega(\lambda n^3)$

communication complexity, even when ignoring the cost of the weak coin. (2)Protocols such as [4, 19, 65] that use an *ideal* leader election subroutine. While some of these protocols [4, 65] achieve quadratic communication complexity (excluding the cost of leader election), implementing leader election using a weak coin requires additional effort. The state-of-the-art compiler [42] involves n instances of reliable broadcast (RBC), which inherently incurs $\Omega(n^3)$ communication complexity.



Multicast fresh ciphertext
P1
P2
...
Pn
c_1, π_1, σ_1
c_n, π_n, σ_n
c_1^*, π_1^*
c_2^*, π_2^*
c_n^*, π_n^*
Align
$\{(j, (c_j^*, \pi_j^*), \theta_j)\}_{j\in S_1}$
$\{(j, (c_j^*, \pi_j^*), \theta_j)\}_{j\in S_2}$
$\{(j, (c_j^*, \pi_j^*), \theta_j)\}_{j\in S_n}$
Release
$\zeta_i \leftarrow$ STHE.PartDec(ek, sid, dk_i)
(see Sect.2.3)
ζ_1
ζ_n
$\{c_j^*\}_{j\in S_1} \xrightarrow{\text{Dec}} \{r_j^*\}_{j\in S_1}$
Find j_1, $H_1(j_1, r_{j_1}^*)$ is largest
$\{c_j^*\}_{j\in S_n} \xrightarrow{\text{Dec}} \{r_j^*\}_{j\in S_n}$
Find j_n, $H_1(j_n, r_{j_n}^*)$ is largest
Multicast LocalWin and its proof
$j_1, r_{j_1}^*$, proof
$j_n, r_{j_n}^*$, proof
$H_2(z, r_z^*)$
Find z, $H_1(z, r_z^*)$ is largest among LocalWin Values
$c_i =$ STHE.Enc(ek, sid, r_i)
σ_i: signature on c_i
π_i: well-formedness of c_i
c_i^*: the aggregation of $f+1$ ciphertexts
π_i^*: well-formedness of c_i^*
θ_j: the completeness proof of (c_j^*, π_j^*) regarding Align

Fig. 2. Our Weak Common Coin Protocol

2.1 Our solutions to achieve quadratic communication

We reduce the communication complexity of all four steps using a set of techniques that enable an asynchronous strong common coin with quadratic communication complexity. The execution flow of our solution is illustrated in Fig. 2.

STHE: for improving Step 1 and Step 3. As we discussed, existing asynchronous common coin protocols exclusively use an asynchronous verifiable secret sharing protocol as the "commitment", which leads to $\Omega(n^3)$ communication cost for all nodes to contribute their commitments. We shift to another tool, threshold encryption, to realize this purpose.

Conventionally, threshold encryption/decryption requires a private setup. Recently, this perspective has changed due to Garg et al.'s silent-setup threshold encryption (STE) scheme [45]. This new advancement enables a $O(\lambda)$-size commitment scheme supporting forced opening without private setup. However, a "plain" STE scheme is not sufficient for our purpose, as costs incurred due to proposing a candidate committed coin and ranking these candidates are still $O(\lambda n^3)$. For fully enjoying the benefits of an STE scheme, we consider a variant of it called STE with *tag-homomorphism* (STHE) (for aggregation), also with strengthend security, which has the following useful features:

- *Tag-homomorphism*: The encryption algorithm takes an additional input, *tag*, such that ciphertexts under the same *tag* can be homomorphically aggregated into a single compact ciphertext. In our protocol, *tag* corresponds to the instance identifier `sid`. Without this property, a candidate committed coin would require $f+1$ ciphertexts for its description. With tag-homomorphism, these ciphertexts can be aggregated into a single compact ciphertext representing the candidate.

- *Tag-decryption*: During the decryption phase, each node only needs to release an $O(\lambda)$-sized partial decryption key for the corresponding *tag* (instead of $O(n\lambda)$-sized shares). Once enough partial decryption keys are released, they can be aggregated into a complete decryption key for the *tag*. This key allows the decryption of all ciphertexts under that *tag*. This property reduces the ranking phase to $O(\lambda n^2)$-bit communication cost. Without tag-decryption, every ciphertext would need to be decrypted individually, resulting in $O(\lambda n^3)$-bit communication complexity.

Besides the more powerful functionalities, we define much *stronger security* for STHE. In particular, the adversary is allowed to obtain the partial decryption keys of honest nodes and can perform adaptive corruption, while neither of these adversarial abilities are considered in [45]. A formal definition for STHE is provided in Sect.4.1, and a construction (adapted from [45]) is presented in Appendix C.

Now, a candidate committed coin is represented as an aggregation of $f+1$ ciphertexts (each encrypting a random number) contributed by distinct nodes. To ensure that the value encrypted by such an aggregated ciphertext remains unknown to the adversary, we leverage a constant-size SNARK [49] to prove that a ciphertext c^* is an aggregation of $f+1$ ciphertexts c_j, each signed by a distinct node. This guarantees that the decryption r^* of c^* is formed as the sum of fresh randomness values $r_{i i \in S}$, with r_z among them, chosen by some honest P_z.

However, a subtle issue arises: while r_z is unknown to the adversary $\mathcal{A}$, $\mathcal{A}$ might create a ciphertext c' on behalf of corrupted nodes such that the decryption of c' is related to r_z (e.g., $-r_z$), thereby revealing the final r^* to $\mathcal{A}$. To prevent such malleability attacks, we ensure that every fresh ciphertext c_j is accompanied by a validity proof π_j with respect to a simulation-extractable NIZK system [62]. Additionally, the SNARK proof must demonstrate that each fresh ciphertext being aggregated has a corresponding validity proof.

With these measures, we prove that the value encrypted by any ciphertext c^* with the specified SNARK proof remains unpredictable to the adversary, even under adaptive corruption. In particular, we carefully apply the *single inconsistency party* proof technique [22, 28] to handle adaptive corruption without relying on memory erasures. More details are available in Sect. 6.1.

Align : a further weaker version of Gather (for Step 2). While the communication cost associated with the underlying commitment scheme can be reduced using STHE, the cubic cost of the Gather protocol still persists. As previously discussed, existing Gather protocols incur $\Omega(n^3)$-bit communication regardless of the commitment size. Achieving a quadratic-communication Gather protocol appears to be a very challenging task, if not impossible. Specifically, when a node outputs a set in Gather, it must be assured that a binding core set of $n-f$ values will be included in the output sets of all other nodes. To achieve this, it seems necessary for each node to share its intended output with all other nodes, which inherently requires $\Omega(n^3)$ communication.

On the other hand, a weaker version of Gather, known as the Weak Core Set, has been studied in [42]. This weaker version only requires that the output sets of $f+1$ honest nodes contain the core set. However, similar communication barriers persist even for this weaker version.

Fortunately, we found that the guarantees provided by Gather or the Weak Core Set are unnecessarily strong. Recall that the purpose of Gather is to enable a constant probability for all honest nodes to decide on the globally highest-ranked candidate. First, as also observed in [42], once $f+1$ honest nodes have seen the globally highest-ranked candidate, it is straightforward to ensure that

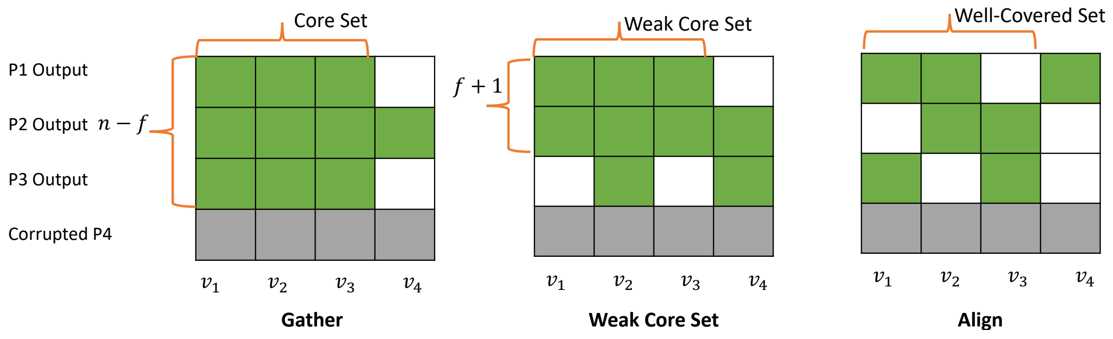


Fig. 3. The output guarantees in Gather, Weak Core Set, and Align

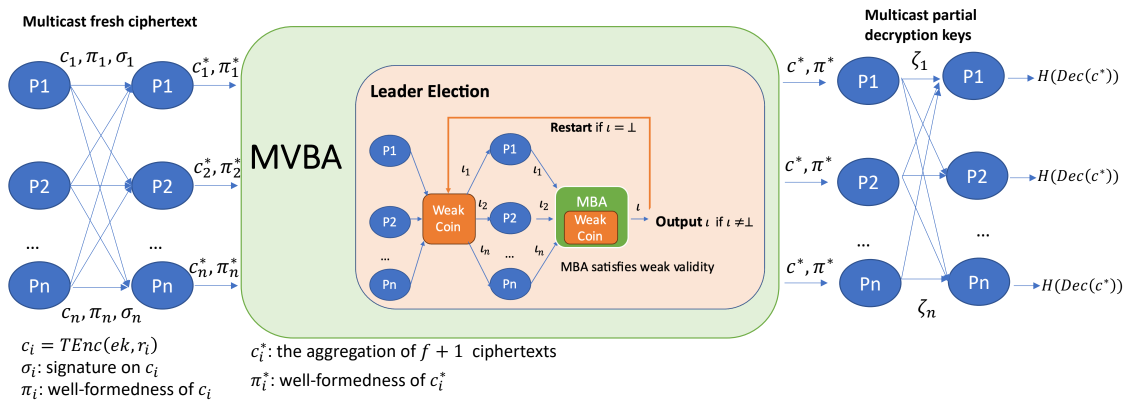
all honest nodes can decide on this candidate by performing an additional round of multicast for amplification. Second, we further observe that to ensure $f+1$ nodes see the globally highest-ranked candidate with a certain probability, it is not necessary for the output sets of $f+1$ nodes to contain a core set. Instead, this can be guaranteed by the existence of a large "well-covered" subset WS, where every element is "well-covered," i.e., included in the output sets of at least $f+1$ honest nodes. Unlike the Weak Core Set, elements in WS can be covered by *different* subsets of $f+1$ nodes.

Based on this observation, we introduce a new primitive termed Asynchronous Subset Alignment (Align). Fig.3 provides an example with four nodes, illustrating how the guarantees of Gather, the Weak Core Set, and Align differ. In addition to the existence of a "well-covered" set, Align also ensures the existence of a *binding cover*, a property previously studied in [27] as a characteristic of Gather. The binding cover ensures that all elements verifiable as valid Align (or Gather) outputs are known at the time the first node generates an output. This property is crucial to prevent an adversary from injecting new candidates once the ranking phase has started. A formal definition of Align is presented in Sect.5.1.

Most importantly, we demonstrate that our weakening enables a solution with $O(1)$ rounds and $O(\lambda n^2)$ communication complexity. Our protocol builds upon the provable broadcast (PB) primitive [4,20,52], introduced by Cachin et al. [20] and recently extensively used in the construction of communication-efficient MVBA protocols [4, 52, 65]. Specifically, we carefully apply n parallel chains of three consecutive PB instances to realize Align. Notably, using fewer PB instances would result in a security issue. The details of our Align protocol are discussed in Sect.5.2.

Simpler conversion from weak coin to leader election (for Step 4). After constructing a quadratic-communication weak common coin, we build a strong common coin through the following two steps:(1) Construct an MVBA protocol with quadratic communication complexity using our weak common coin; (2) Use the MVBA to enable the network to agree on a specific candidate committed coin, such that all honest nodes can obtain the coin value by opening this candidate, which, in our case, involves decrypting the ciphertext. The overall execution flow of our strong common coin protocol is illustrated in Fig. 4.

The main technical gap lies in the MVBA component. Specifically, while there are quadratic-communication MVBA protocols, they all rely on a quadratic-communication leader election subroutine, which ensures agreement on the election outcome with a probability of $1 - \mathsf{negl}(\lambda)$. [42] proposed a compiler from a weak common coin to leader election; however, this approach incurs an $O(\lambda n^3)$-bit communication cost.



Multicast fresh ciphertext
c_1, π_1, σ_1
c_n, π_n, σ_n
c_1^*, π_1^*
c_2^*, π_2^*
c_n^*, π_n^*
P1
P2
Pn
...
$c_i = TEnc(ek, r_i)$
σ_i: signature on c_i
π_i: well-formedness of c_i
MVBA
Leader Election
Restart if $\iota = \perp$
Weak Coin
MBA
Weak Coin
Output ι if $\iota \neq \perp$
MBA satisfies weak validity
c_i^*: the aggregation of $f+1$ ciphertexts
π_i^*: well-formedness of c_i^*
Multicast partial decryption keys
c^*, π^*
ζ_1
ζ_n
$H(Dec(c^*))$

Fig. 4. Asynchronous Strong Common Coin

To address this gap, we propose a much simpler compiler built upon multivalued Byzantine agreement (MBA) [70]. After receiving a value $\iota_i \in [n]$ from a weak common coin instance with output space $[n]$, each honest node P_i inputs ι_i into the MBA protocol. This ensures that, except with negligible probability, the network agrees on the same $\iota \in [n]$. If the MBA protocol returns $\perp$, the process is simply re-run until the MBA produces a valid output. A prudent reader may observe that the MBA protocol [70] itself requires randomization, which can be readily supplied by concurrent instances of weak common coins. A detailed construction and analysis are provided in Sect. 7.1. [7]

3 Models, Goals and Preliminary

Notations. Throughout the paper, let λ represent the bit length of the cryptographic security parameter. The notation $[i, n]$ denotes the set $\{i, i+1, \ldots, n\}$, where i and n are integers with $i < n$. We may abbreviate $[1, n]$ as $[n]$. For a set $\{x_1, x_2, \ldots, x_n\}$ and a sequence $(x_1, x_2, \ldots, x_n)$, we denote them as $\{x_i\}_{i\in[n]}$ and $(x_i)_{i\in[n]}$, respectively, for brevity. A function $f(n)$ is considered negligible in n, denoted by $f(n) \leq \text{negl}(n)$, if for every positive integer c, there exists an n_0 such that for all $n > n_0$, $f(n) < n^{-c}$. Conversely, a non-negligible function satisfies $f(n) > \text{negl}(n)$. For a set $\mathbb{X}$, the notation $x \leftarrow\!\!\$\ \mathbb{X}$ signifies sampling x uniformly from $\mathbb{X}$. Given a distribution X, $x \leftarrow X$ denotes sampling x from X. For a probabilistic algorithm A, $A(x_1, x_2, \ldots; r)$ represents the result of running A with inputs $x_1, x_2, \ldots$ and random coins r. We use $y \leftarrow A(x_1, x_2, \ldots)$ to denote randomly choosing r and computing $A(x_1, x_2, \ldots; r)$ to obtain the output y. We assume that every protocol instance is assigned a unique identifier `sid`, and the identifier of an instance serves as the prefix for the identifiers of its subroutines. The concatenation of `sid` and $\texttt{sid}_0$ is denoted as $\langle \texttt{sid}, \texttt{sid}_0 \rangle$, where `sid` is a prefix of this concatenated identifier.

Network and Adversary Models. We consider an *asynchronous* network where nodes are pairwise connected via *authenticated* channels, and we assume the identities of all participating nodes are publicly known. The adversary cannot omit, change, or inject messages sent by honest nodes.

[7] We note that our compiler, apart from the weak common coin instance, does not require any setup or cryptographic operations. We expect it to be useful in hash-only settings or information-theoretical settings.

However, the adversary can reorder the messages and arbitrarily delay them, though it cannot delay them indefinitely. We focus on optimal resilience, meaning that the total number of corrupted nodes is at most $f \leq \lceil \frac{n}{3} \rceil - 1$.

We consider the *adaptive* adversaries which can adaptively corrupt nodes at any time during the protocol execution. Once a node is corrupted, the adversary gains access to its local state and controls its subsequent behavior. During execution, the nodes that have remained honest up to a given point are referred to as *so-far-honest* nodes, while those that remain honest until the end of execution are called *forever-honest* nodes. For simplicity, when we refer to honest nodes in this paper, we mean *forever-honest* nodes. In particular, an adaptive adversary can perform "after-fact-removal" attacks [1,68], *i.e.*, corrupt a node and *remove* all messages the node just sent before they are delivered.

Silent Setup. Following [44,45], we consider a setup phase to be *silent* if the setup can be performed with the help of a public bulletin board, to which each participant has only one-time writing access (and unbounded reading access). It is easy to observe that the conventional PKI setup and common reference string (CRS) setup fall into this category. In this work, we consider a slightly more general setup phase that includes both the PKI and CRS setups and can be described by the following three algorithms:

- $\mathsf{CRS}(1^\lambda, n, t) \rightarrow \mathtt{crs}$. On input the security parameter λ, the number of participants n, and the threshold number t, it generates a common reference string $\mathtt{crs}$. This algorithm can be run by a trusted entity, and $\mathtt{crs}$ is published on the bulletin board and made available to everyone.
- $\mathsf{KeyGen}(\mathtt{crs}, i) \rightarrow ((pk_i, \mathsf{hint}_i), sk_i)$. A user P_i locally runs the algorithm to generate its own key pair (pk_i, sk_i) along with a hint information hint_i. While sk_i is kept private, (pk_i, hint_i) is published on the bulletin board. Notably, (pk_i, hint_i) can be viewed as an extended public key of P_i.
- $\mathsf{GroupPKGen}(\mathtt{crs}, (pk_1, \mathsf{hint}_1), \ldots, (pk_n, \mathsf{hint}_n)) \rightarrow pk$. This is a deterministic algorithm such that every participant can read the public keys $((pk_1, \mathsf{hint}_1), \ldots, (pk_n, \mathsf{hint}_n))$ and compute the same group public key pk.

For notational simplicity, we denote the above process as

$$\mathsf{SilSetup}(1^\lambda, n, t) \rightarrow (\mathtt{crs}, pk, (pk_i)_{i \in [n]}, (sk_i)_{i \in [n]}).$$

Furthermore, an adversary $\mathcal{A}$ is allowed to corrupt some participants during the setup phase (after seeing the CRS) and generate keys on behalf of the corrupted parties (after seeing the public keys of uncorrupted parties). We denote an execution of $\mathsf{SilSetup}$ involving $\mathcal{A}$ as

$$\mathsf{SilSetup}^{\mathcal{A}}(1^\lambda, n, t) \rightarrow (\mathtt{crs}, pk, (pk_i)_{i \in [n]}; \mathsf{state}_{\mathcal{A}}; (sk_i)_{i \in [n] \setminus \mathcal{C}_{\mathsf{init}}}),$$

where $\mathcal{C}_{\mathsf{init}} \subset [n]$ is the set of initially corrupted parties' identities, and $\mathsf{state}_{\mathcal{A}}$ is the state of $\mathcal{A}$ after the setup.

Remark 1. As discussed in [44], a silent setup does not enable the participants to run a DKG during the setup phase. In particular, even for the non-interactive DKG protocols [50,58], the parties need to write to the bulletin board *twice*. First, they must post their public keys to the bulletin board. Then, based on the public keys of all other nodes, they generate transcripts, which also need to be posted to the bulletin board. In contrast, a silent setup allows only one-time writing access to the bulletin board, which enables the nodes to join the system "asynchronously": when each node arrives, they only need to read the CRS and post their public key; the participants do not need to coordinate with each other to complete the setup.

Remark 2 (Registered PKI Setup with a Key Curator). Looking ahead, our coin construction requires each node to generate $O(\lambda n)$-sized hint information. If all honest nodes read all hints and run $\mathsf{GroupPKGen}$ locally, the setup incurs $O(\lambda n^3)$ communication cost. This cost can be reduced to $O(\lambda n^2)$ under a registered PKI setup with a key curator [43], which collects (pk_i, hint_i) from participants, runs $\mathsf{GroupPKGen}$ locally, and returns $(pk, (pk_i)_{i\in[n]})$ to all participants.

We note that key curators have been extensively studied in registration-based encryption [43], representing a strictly weaker setup assumption than a trusted key generator which is needed by identity-based encryption [17] and the unique threshold signature schemes. A key curator is fully transparent, holds no secret information, and performs only deterministic tasks.

Design Goal: Common Coin. We aim at designing a silent-setup common coin protocol with optimal resilience and optimal communication. Specifically, a common coin protocol provides a random source observable by all participants but unpredictable by an adversary [20], which can be formally defined as follows.

Definition 1 (Common Coin). *Let Π be a protocol involving n participants in which each P_i does not have an input and will be instructed to output a value $v_i \in V$. We say Π is an (n, f, φ) common coin protocol for $\varphi \in (0, 1]$, if, in any instance of Π, and for any PPT adversary $\mathcal{A}$ which can corrupt up to f parties, the following properties are ensured:*

- ***Termination:*** *When all honest nodes are activated in this instance, except with negligible probability, every honest party P_i eventually outputs a value $r_i \in V$.*
- *φ-**Fairness:** With a probability of at least $\varphi - \mathsf{negl}(\lambda)$, the protocol has a fair outcome, satisfying:*
  - *Agreement: There is a value $v \in V$ such that $v_i = v$ for any $i \in \mathcal{H}$, where $\{P_i\}_{i\in\mathcal{H}}$ is the set of honest parties.*
  - *Unpredictability and bias-resistance: Let $\mathsf{State}_{\mathcal{A}}^{before}$ denote the internal state of $\mathcal{A}$ before $n - f - |\mathcal{C}|$ so-far-honest nodes participate in the protocol, where $|\mathcal{C}|$ is the number of corrupted parties. For any PPT distinguisher $\mathcal{D}$ and $u \leftarrow\$ V$,*

$$\left| \Pr[\mathcal{D}(\mathsf{State}_{\mathcal{A}}^{before}, v) = 1] - \Pr[\mathcal{D}(\mathsf{State}_{\mathcal{A}}^{before}, u) = 1] \right| \leq \mathsf{negl}(\lambda).$$

Remark 3. When $\varphi = 1$, the above definition captures the multivalued version of the cryptographic *strong* common coin [20], which ensures termination, agreement, unpredictability, and bias-resistance except with negligible probability. When $\varphi < 1$, this definition can be seen as a multivalued and cryptographically secure version of the *weak* common coin by Feldman and Micali [35], and it can trivially imply a protocol satisfying their definition, as we show in Lemma 14 in Appendix B.1.

3.1 Other Cryptographic and Consensus Tools

Non-interactive (Zero-Knowledge) Argument An argument system Φ for an NP relation $\mathcal{R}$ can be described by the following three algorithms.

- $\mathsf{Setup}(1^\lambda)$ generates a common reference string $\texttt{crs}$.
- $\mathsf{Prove}(\texttt{crs}, x, w)$ on inputs a statement x and its witness w such that $(x, w) \in R$, produces a proof π.
- $\mathsf{Vrfy}(\texttt{crs}, x, \pi)$ validates the proof π for x.

  We consider the following properties of an argument system Φ.

- *Completeness.* Φ is complete, if for any $(x, w) \in \mathcal{R}$, $\texttt{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, it holds that

$$\Pr[\mathsf{Vrfy}(\texttt{crs}, x, \mathsf{Prove}(\texttt{crs}, x, w)) = 1] = 1$$

.

- *Knowledge Soundness.* Φ is knowledge-sound, if for any PPT adversary $\mathcal{A}$, there is an PPT algorithm $E_\mathcal{A}$, such that

$$\Pr\begin{bmatrix}\texttt{crs} \leftarrow \mathsf{Setup}(1^\lambda), (x, \pi) \leftarrow \mathcal{A}(\texttt{crs}), w \leftarrow E_\mathcal{A}(x, \pi) : \\ \mathsf{Vrfy}(\texttt{crs}, x, \pi) = 1 \wedge (x, w) \notin R\end{bmatrix} \leq \mathsf{negl}(\lambda).$$

- *Zero Knowledge.* Φ is zero-knoweldge, if there is a pair of simulator algorithms $\{\mathsf{SimSetup}, \mathsf{SimProve}\}$, and for any PPT adversary $\mathcal{A}$, it holds that

$$\left| | \Pr\begin{bmatrix}\texttt{crs} \leftarrow \mathsf{Setup}(1^\lambda) : \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_0(\cdot)}(\texttt{crs})\end{bmatrix} - \Pr\begin{bmatrix}(\texttt{crs}, \texttt{tk}) \leftarrow \mathsf{SimSetup}(1^\lambda) : \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}_1(\cdot)}(\texttt{crs})\end{bmatrix}\right| \leq \mathsf{negl}(\lambda),$$

  where $\mathcal{O}_0$ on input $(x, w) \in \mathcal{R}$ returns $\pi \leftarrow \mathsf{Prove}(\texttt{crs}, x, w)$, and $\mathcal{O}_1$ returns $\pi \leftarrow \mathsf{SimProve}(\texttt{crs}, \texttt{tk}, x)$.
- *Simulation Extractability.* Φ is simulation extractable if, in addition to the zero-knowledge simulation algorithm, $\{\mathsf{SimSetup}, \mathsf{SimProve}\}$, there is a PPT extractor algorithm SimExt, such that, for any PPT adversary $\mathcal{A}$, the following equation holds, where $\mathcal{O}$ returns $\mathsf{SimProve}(\texttt{crs}, \texttt{tk}, x)$, and $\mathsf{Hist}_\mathcal{O}$ records all x queried to $\mathcal{O}$.

$$\Pr\begin{bmatrix}(\texttt{crs}, \texttt{tk}) \leftarrow \mathsf{SimSetup}(1^\lambda), (x, \pi) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\texttt{crs}) \;\; s.t. x \notin \mathsf{Hist}_{\mathcal{O}_1} \\ w \leftarrow \mathsf{SimExt}(\texttt{crs}, \texttt{tk}, x, \pi) : \mathsf{Vrfy}(\texttt{crs}, x, \pi) = 1 \wedge (x, w) \notin R\end{bmatrix} \leq \mathsf{negl}(\lambda).$$

In addition, Φ is *succinct* if the proof size, *i.e.*, $|\pi|$, is at most logarithmic in $|w|$ (the witness size).

In this paper, we consider the following two types of non-interactive argument systems with different combinations of properties.

- NIZK: Satisfy the completeness, the zero-knowledge property, and the simulation extractability. Due to technical reasons, which will become clear when demonstrating our coin protocol, we require NIZK in the standard model, which admits various standard instantiations, such as Groth-Sahai [51] (plus generic compiler for simulation extractability [62]), that work for a wide range of relations and even all NP relations.
- SNARK: Satisfy the completeness, knowledge soundness, and the succinctness. In particular, there are SNARK schemes, such as Groth-16 [49] and [41], that enjoy *constant* proof size and work for all NP relations.

Multi-valued Byzantine Agreement (MBA). In an (n, t, ℓ)-MBA protocol, there are n parties $P_1, \ldots, P_n$, each P_i having an ℓ-bit initial input v_i. Against any adversary $\mathcal{A}$ that corrupts up to t parties, a secure MBA ensures the following properties:

- **Validity.** If all honest parties share the same input v, they all eventually output v.
- **Agreement.** If an honest party P_i outputs v, then all honest party eventually outputs v.
- **Termination.** All honest parties eventually produce an output message.

Multi-valued Validated Byzantine Agreement (MVBA). In an (n, t, ℓ)-MVBA protocol with a predicate $\mathsf{Predicate} : \{0,1\}^\ell \rightarrow \{0,1\}$, there are n parties $P_1, \ldots, P_n$, each P_i having an ℓ-bit initial input v_i. Against any adversary $\mathcal{A}$ that corrupts up to t parties, a secure MVBA ensures the following properties:

- **External Validity.** If an honest party outputs v, then it must be that $\mathsf{Predicate}(v) = 1$.
- **Agreement.** If an honest party P_i outputs v, then all honest party eventually outputs v.
- **Termination.** If at least $n - f$ honest nodes are activated in the instance, and the input v_i of each P_i is valid, *i.e.*, $\mathsf{Predicate}(v_i) = 1$, then all honest nodes eventually output a value.

Provable Broadcast. In an instance of a provable broadcast (PB) protocol, there is a designated party, the sender, who has an input message v to be delivered to the network. When v is "successfully" delivered, the sender can obtain a proof ρ, which can be verified by an algorithm $\mathsf{PB}[\mathtt{sid}].\mathsf{Vrfy}$, where $\mathtt{sid}$ is the session ID of the instance. In addition, an external predicate $\mathsf{Predicate}$ on the message may be considered, such that only messages satisfying this predicate shall be delivered. Following Abraham et al. [4], an (n, f)-PB protocol shall satisfy the following properties except with a negligible probability, for any session ID $\mathtt{sid}$, and any PPT adversary controlling up to f nodes.

- **Integreity.** An honest node delivers at most one message.
- **External Validity.** If an honest node delivers v', then $\mathsf{Predicate}(v') = 1$.
- **Abandon-ability.** An honest node can call a function $\mathsf{Abandon}(\mathtt{sid})$ of the instance, such that this node will not deliver any message for this instance.
- **Termination.** If the sender is honest, and the input v satisfies $\mathsf{Predicate}(v) = 1$, then any honest party eventually delivers v for this instance if it does not call $\mathsf{Abandon}(\mathtt{sid})$. In addition, if no honest party called $\mathsf{Abandon}(\mathtt{sid})$, then the sender can obtain a proof ρ, such that $\mathsf{PB}.\mathsf{Vrfy}[\mathtt{sid}](v, \rho) = 1$.
- **Provability.** For any two pairs (v, ρ) and (v', ρ'), if $\mathsf{PB}[\mathtt{sid}].\mathsf{Vrfy}(v', \rho') = \mathsf{PB}[\mathtt{sid}].\mathsf{Vrfy}(v, \rho) = 1$ it follows that $v' = v$. In addition, if a (potentially corrupted) sender can output a valid proof ρ for v, then at least $f + 1$ forever-honest nodes delivered v in this instance.

Instantiation. There is a simple construction based on threshold signature [20] to realize PB with $O(n(\ell + \lambda))$ communication complexity and 2 rounds. In particular, the sender first multicasts the message v to all recipients. Upon receiving a message v from the sender, a recipient will deliver the message and return a partial signature on v to the sender if the message satisfies the predicate. After receiving $2f + 1$ partial signatures, the sender can aggregate them into a constant-size threshold signature, which is the delivery proof ρ for v. We detail this construction in Algorithm 5 in Appendix B.2. Note that the threshold signature can be realized using a silent setup threshold signature (STS) scheme, for example, [26]. We recall the formal definition of STS in Appendix.B.3.

4 Our Tools

In this section, we introduce a few new tools for our common coin protocols. In particular, in Sect.4.1, we present our core cryptographic tool, which is a variant of the silent threshold encryption

(STE) from [45] but with stronger security guarantees that are necessary for our coin protocols. In Sect.4.2, we introduce a simple framework for analyzing specific compositions.

4.1 Silent Threshold Encryption with Tag-homomorphism.

A threshold encryption scheme allows a group of nodes to jointly decrypt a ciphertext under their group public key, while an adversary controlling up to a threshold number of nodes cannot break the secrecy guarantee. It is called a silent threshold encryption scheme if its setup phase is silent (as in Remark 1). In addition, we consider *tag-homomorphism*, which means the encryption algorithm takes an extra input *tag*, and ciphertexts under the same public key and the same tag can be homomorphically evaluated.

Syntax. Formally, an (n, t) silent threshold tag-homomorphic encryption (STHE) scheme consists of the following algorithms:

- $\mathsf{SilSetup}(1^\lambda, n, t) \to (\mathtt{crs}, ek, (ek_i)_{i\in[n]}, (dk_i)_{i\in[n]})$. This is the silent setup phase of STHE. Here, ek and dk denote the public encryption key and the secret decryption key, respectively. $\mathtt{crs}$ is the implicit input to all the following algorithms.
- $\mathsf{Enc}(ek, tag, m)$ Encrypts the plaintext m under ek and tag, producing a ciphertext c. In particular, we assume the plaintext space $\mathcal{M}$ is a commutative group with an operation $\oplus$.
- $\mathsf{Eval}(ek, tag, \{c_i\}_{i\in S})$ On input a set of ciphertexts under the same encryption key ek and tag, outputs a ciphertext c^*.
- $\mathsf{PartDec}(ek, tag, dk_i)$ Generates a partial decryption key ζ_i for ciphertexts under ek and tag.
- $\mathsf{PartVrfy}(ek, tag, ek_i, \zeta_i)$ Verifies whether ζ_i is a partial decryption key generated by node i.
- $\mathsf{DkAgg}(ek, tag, \{\zeta_i\}_{i\in S})$ Aggregates a set of valid partial decryption keys into a decryption key dk_{tag} for tag.
- $\mathsf{Dec}(ek, tag, c, dk_{tag})$ Decrypts a ciphertext c using the decryption key dk_{tag}.

Remark 4. Compared with the syntax STE established in [45], the STHE introduced above includes the notion of a tag, which serves as an additional input to the encryption algorithm and enables two essential features: (1) *Tag-homomorphism*, allowing ciphertexts encrypted under the same tag to be homomorphically aggregated; (2) *Tag-based decryption*, enabling all participants to release a decryption key for a specific tag to decrypt all ciphertexts under that tag.

Correctness. Let $\mathcal{A}$ be any PPT adversary corrupting up to t participants. Let $\{\mathsf{crs}, ek, (ek_i)_{i\in[n]}\} \leftarrow \mathsf{SilSetup}^{\mathcal{A}}(1^\lambda, n, t)$. The following properties hold:

- *Partial Verification Correctness:* For any tag and $i \notin \mathcal{C}_{\mathsf{init}}$, it holds that
$$\Pr[\mathsf{PartVrfy}(ek, tag, ek_i, \mathsf{PartDec}(ek, tag, dk_i))] = 1.$$
- *Decryption Correctness:* For any tag, m, and γ, such that $c = \mathsf{Enc}(ek, tag, m; \gamma)$, and for $t+1$ partial decryption keys $\{\zeta_j\}_{j\in S}$ where $\mathsf{PartVrfy}(ek, tag, ek_j, \zeta_j) = 1$, it holds that
$$\Pr[\mathsf{Dec}(ek, tag, c, \mathsf{DkAgg}(ek, tag, \{\zeta_j\}_{j\in S}))] = m.$$
- *Evaluation Correctness:* For any $\{c_i = \mathsf{Enc}(ek, tag, m_i; \gamma_i)\}_{i\in S}$ and $c^* \leftarrow \mathsf{Eval}(ek, tag, \{c_i\}_{i\in S})$, there exists γ^* such that
$$c^* = \mathsf{Enc}(ek, tag, \bigoplus_{i\in S} m_i; \gamma^*).$$

Semantic Security. At a high level, the semantic security captures that any PPT adversary cannot learn any non-trivial information from a ciphertext as long as the adversary does not know enough partial decryptions for the tag of the ciphertext. We further allow the adversary to see the partial decryptions produced by honest participants for any specific tags, which captures the fact that the decryption process for other tags will not undermine the secrecy of ciphertexts under the current tag. In Fig.5, we use $\mathcal{O}_{\mathsf{PartD}}$ to formalize such an ability. In addition, for an adaptive adversary, we consider a corruption oracle $\mathcal{O}_{\mathsf{Corr}}$ through which the adversary can obtain the secret key, as well as the randomness previously used to generate the partial decryption keys; The adversary can decide which participant to corrupt based on the query results of $\mathcal{O}_{\mathsf{PartD}}$ and the challenge oracle $\mathcal{O}_b$.

Formally, we say an (n, t)-STHE scheme satisfies the adaptively semantic security if for any PPT adversary $\mathcal{A}$, it holds that, for every $\lambda \in \mathbb{N}$ and every $tag^* \in \{0,1\}^*$,

$$\mathsf{Adv}^{\mathrm{aind}}_{\mathcal{A},tag^*}(\lambda) = |\Pr[\mathsf{aIND}_0^{\mathcal{A},tag^*}(1^\lambda, n, t)] - \Pr[\mathsf{aIND}_1^{\mathcal{A},tag^*}(1^\lambda, n, t)]| \leq \mathsf{negl}(\lambda). \tag{1}$$

```
$\mathsf{aIND}_b^{\mathcal{A},tag^*}(1^\lambda, n, f)$
$\mathcal{C}, \mathsf{Tags}, \mathsf{State}_1, \ldots, \mathsf{State}_n \leftarrow \emptyset$
$(\mathsf{crs}, ek, (ek_i)_{i\in[n]}; \mathsf{state}_{\mathcal{A}}; (dk_i)_{i\in[n]\setminus\mathcal{C}_{\mathsf{init}}})$
    $\leftarrow \mathsf{SilSetup}^{\mathcal{A}}(1^\lambda, n, t), \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\mathsf{init}}$
$1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Corr}}(\cdot), \mathcal{O}_{\mathsf{PartD}}(\cdot), \mathcal{O}_b(\cdot)}(\mathsf{state}_{\mathcal{A}})$

$\mathcal{O}_{\mathsf{Corr}}(i)$
if $i \in [n] \setminus \mathcal{C} \wedge |\mathcal{C}| < t$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$
    return $(dk_i, \mathsf{State}_i)$

$\mathcal{O}_{\mathsf{PartD}}(tag, i)$
if $tag \neq tag^* \wedge i \notin \mathcal{C}$
    $\mathsf{Tags} \leftarrow \mathsf{Tags} \cup \{tag\}$
    $\gamma \leftarrow \mathsf{Rand}_{\mathsf{PartD}}$
    $\zeta_i \leftarrow \mathsf{PartDec}(ek, tag, dk_i; \gamma)$
    $\mathsf{State}_i \leftarrow \mathsf{State}_i \cup \{(tag, i, \gamma, \zeta_i)\}$
    return $\zeta_i$}

$\mathcal{O}_b(m_0, m_1)$
return $c_b \leftarrow \mathsf{Enc}(ek, tag^*, m_b)$
```

Fig. 5. Security Game of STHE.

Instantiation. We show an instantiation of STHE in Appendix C, which is a variant of the silent threshold encryption scheme in [45]. In summary, the ciphertext size and the partial decryption size are both $O(\lambda)$, independent of the number of participants n. While the CRS size and public key size are $O(\lambda n)$, each node can obtain this information from the setup phase.

4.2 A Simple Framework for Analyzing Specific Composition

In this section, we introduce a framework to analyze the security of a protocol instance when it is composed of certain other instances in a particular manner.

What might go wrong if not formally proving composition security? In most existing works in the domain of asynchronous common coins or asynchronous consensus, the protocol security is defined with a list of properties. A protocol usually has several components, each with a list of properties as its security definition. These properties usually focus on a single instance, and the components are proven to satisfy these properties in the single instance setting. When analyzing the security of the final protocol, existing works usually show a reduction from certain properties of the final protocol to certain properties of certain components. However, there is a loophole in such a security proof since the following question has not been addressed: Do the components still satisfy the security properties when they are composed in the final protocol?

We show a concrete example of a protocol that satisfies all security properties in the stand-alone setting becoming insecure when two instances of the same protocol run together. Our example is an artificial modification of the asynchronous verifiable secret sharing (AVSS) protocol by Das et al. [30]. Das et al.'s AVSS works as follows:

- The dealer samples a random polynomial f whose $f(0)$ is the secret, sends every receiver P_i a share $f(i)$ and a polynomial commitment [57] com to f, along with some proof showing $f(i)$ is the corret evaluation w.r.t. the committed polynomial.
- Each receiver P_i, after verifying $f(i)$ is an evaluation at i of the committed polynomial, signs com, and returns the signature to the dealer.
- Then, the dealer, after collecting $n-f$ valid signatures, will *broadcast* (via a RBC [18]) the commitment com, all $n-f$ signatures, and the remaining $f(j)$ for all P_j which has not returned the signature.

To show what might go wrong, let us focus on the following important property of AVSS.

- *Completeness of AVSS*: If an honest node terminates in the AVSS instance, then every honest node can deliver a correct share.

Note that the AVSS scheme described above satisfies the completeness because the valid shares are either available via the broadcast or the corresponding receivers have confirmed the delivery by their signature.

Now, let us assume the underlying signature scheme is a one-time signature, which ensures that an adversary cannot forge a signature of vk_i when only seeing one signature of P_i, while the signature can be easily forged after seeing two signatures under vk_i. It is easy to see that the protocol can still guarantee single-instance security. However, when two instances using the same PKI setup run together, the adversary can forge the signature of honest nodes without sending corresponding shares, which violates completeness.

Note that AVSS is fundamental to almost all asynchronous common coin protocols, and more importantly, each common coin instance will invoke several AVSS instances. Therefore, a single-instance-secure AVSS is not secure in the composition of coin protocols, which indicates the composition security for asynchronous coins and consensus deserves a more serious treatment.

Why not UC? On the other hand, a well-established path for ensuring composition security is to prove each component is secure in the Universal Composable framework [21]. However, as our construction leverages certain "non-UC" components, such as a succinct non-interactive argument knowledge (SNARK) [49], it is unlikely to have a simple UC proof without heavy modeling. In addition, due to the nature of asynchronous protocols, where the messages from different components cannot be scheduled at different times, the trivial composition theorems, like that for sequential composition, cannot apply.

A simple case. Let $\mathit{\Pi}[\mathtt{sid}]$ and $\mathit{\Pi}'[\mathtt{sid}']$ be two protocol instances running together. Let $\mathcal{A}$ be an adversary against this composition. Note that $\mathcal{A}$ can access information via two means: one is to observe the communication transcripts, whether they are sent directly to the corrupted parties or between two honest nodes; another is to corrupt an honest node so that all its internal states will be leaked to the adversary. In particular, if $\Pi[\texttt{sid}]$ and $\Pi'[\texttt{sid}']$ involve some common participants, corrupting a common participant in one instance triggers corruption in the other instance.

We model the view of $\mathcal{A}$ in composition as a joint random variable $\mathsf{VIEW}_{\mathcal{A}} = (\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}, \mathsf{VIEW}_{\mathcal{A},\Pi'[\texttt{sid}']})$, where $\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}$ (resp. $\mathsf{VIEW}_{\mathcal{A},\Pi'[\texttt{sid}]}$) consists the communication transcripts and internal states available to $\mathcal{A}$ in Π (resp. Π'). Then, we have the following facts.

Lemma 1. *$\Pi[\texttt{sid}]$ remains secure in the composition with $\Pi'[\texttt{sid}']$, if for any $\mathcal{A}$ against the composition, there is an algorithm $\mathcal{S}$ whose complexity is polynomial in $\mathcal{A}$'s complexity, such that at any point of the execution, the distribution of $\mathsf{VIEW}_{\mathcal{A}} = (\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}, \mathsf{VIEW}_{\mathcal{A},\Pi'[\texttt{sid}']})$ is identical to $(\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}, \mathcal{S}(\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}))$.*

Proof. Assume that there is an adversary $\mathcal{A}$ which can violate the certain security property of $\Pi[\texttt{sid}]$ in the composition. Then, we can turn $\mathcal{A}$ into a $\mathcal{B}$, which violates the security of $\Pi[\texttt{sid}]$ in the stand-alone setting. $\mathcal{B}$ runs a copy of $\mathcal{A}$ in the following manner: it serves as a relay between $\mathcal{A}$ and the instance $\Pi[\texttt{sid}]$ (so $\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}$ is available to both $\mathcal{A}$ and $\mathcal{B}$), and provides $\mathsf{VIEW}_{\mathcal{A},\Pi'[\texttt{sid}']} = \mathcal{S}(\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]})$ to $\mathcal{A}$. Since $(\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}, \mathsf{VIEW}_{\mathcal{A},\Pi'[\texttt{sid}']}) = (\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}, \mathcal{S}(\mathsf{VIEW}_{\mathcal{A},\Pi[\texttt{sid}]}))$, $\mathcal{A}$ can violate the security of $\Pi[\texttt{sid}]$, so can $\mathcal{B}$. $\square$

The above lemma basically states the facts: if an instance $\Pi'[\texttt{sid}']$ cannot give any "new" information to an adversary against Π, then $\Pi[\texttt{sid}]$ remains secure when it is composed with $\Pi'[\texttt{sid}']$. A typical example of such a composition is the composition of information-theoretical consensus protocols, where the input and output of honest parties are all available to the adversary, and honest nodes do not keep any secret internal states, which makes it trivial to generate the adversary's view. Indeed, Lindell et al. [64] showed that all information-theoretical Byzantine agreement protocols are secure under concurrent composition.

Composition with signing oracles. There are scenarios in which the messages in one instance cannot be generated using the view of another instance. For example, consider $\Pi[\texttt{sid}]$ and $\Pi'[\texttt{sid}']$ that are using the same digital signature scheme under the same PKI setup to authenticate communication. Then, due to the security of the digital signature scheme, it is infeasible to generate a message sent by an honest node P_i in $\Pi'[\texttt{sid}']$, given the view of the adversary in $\Pi[\texttt{sid}]$.

The barrier can be overcome if a signing oracle w.r.t. P_i is available. Intuitively, if $\Pi[\texttt{sid}]$ remains secure when the adversary has access to the signing oracles of the honest participants, then it remains secure in the composition with $\Pi'[\texttt{sid}']$ whose communication transcripts can be generated by the adversary with the access to the signing oracles. Of course, the signing oracles cannot be unrestricted; Otherwise, $\mathcal{A}$ may leverage these oracles to forge messages in $\Pi[\texttt{sid}]$, so that $\Pi[\texttt{sid}]$ cannot be secure against such an adversary. However, with unique identifiers $\texttt{sid}$ and $\texttt{sid}'$, we can force every message in $\Pi[\texttt{sid}]$ and $\Pi'[\texttt{sid}']$ to carry the corresponding identifer. Then, $\Pi[\texttt{sid}]$ remains secure, when $\mathcal{A}$ can access signing oracles that do not sign messages starting with $\texttt{sid}$, while such signing oracles, are sufficient for helping to generate messages in $\Pi'[\texttt{sid}']$. Lindell et al. cites LindellLR06 have shown that a variety of authenticated Byzantine agreement instances are secure under concurrent composition when each instance has a unique identifier.

General oracle-enhanced case. We consider a more general case, where $\mathcal{A}$ and $\mathcal{S}$ can access oracles beyond the signing oracles. Formally, we consider a set of oracles $\mathbb{O} = \{\mathcal{O}_z\}$, where each $\mathcal{O}_z$ takes input in the form of $(\texttt{sid}, i, \cdot)$, where $\texttt{sid}$ is an identifier of a protocol instance, and i is the index of an uncorrupted node. There is a general restriction that an adversary attacking an instance with $\mathtt{sid}$ can only query $\mathcal{O}_z$ with $(\mathtt{sid}', i, \cdot)$ s.t. $\mathtt{sid}' \neq \mathtt{sid}$. In addition, notice that an adaptive adversary may corrupt a party P_i during the protocol execution so that all internal states of P_i used to generate the previous messages must be leaked to $\mathcal{A}$ as well. To capture this, we consider that, $\mathbb{O}$, by default, includes a special corruption oracle $\mathbb{O}_{\mathsf{Corr}}$, that is, when an adversary $\mathcal{A}$ corrupts P_i, the query i to $\mathbb{O}_{\mathsf{Corr}}$ is automatically triggered, such that $\mathbb{O}_{\mathsf{Corr}}$ returns all internal secrets and randomness of all $\mathcal{O}_z$ that have been used to responde the queries in the form of $(\cdot, i, \cdot)$.

Furthermore, notice that there might be some information from an instance $\Pi'[\mathtt{sid}']$ that cannot be captured by oracles, for example, the inputs to $\Pi'[\mathtt{sid}']$. Therefore, we also consider an auxiliary string Aux, which is available to both the adversary against $\Pi[\mathtt{sid}]$ and the simulator algorithm $\mathcal{S}$. Formally, we can extend Lemma 1 to the oracle-enhanced case.

Lemma 2. *Let $\Pi[\mathtt{sid}]$ and $\Pi'[\mathtt{sid}']$ be two protocol instances running together. Let Aux be an auxiliary string. $\Pi[\mathtt{sid}]$ remains secure in the composition with $\Pi'[\mathtt{sid}']$, if $\Pi[\mathtt{sid}]$ remains secure in the stand-alone setting even when the adversary has access to oracles in $\mathbb{O}$ and* Aux*, and for any $\mathcal{A}$ against the composition, there is an algorithm $\mathcal{S}$ whose complexity is polynomial in $\mathcal{A}$'s complexity and which has access to oracles in $\mathbb{O}$ and* Aux*, such that at any point of the execution, the distribution of $\mathsf{VIEW}_{\mathcal{A}} = (\mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}, \mathsf{VIEW}_{\mathcal{A},\Pi'[\mathtt{sid}']})$ is identical to $(\mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}, \mathcal{S}^{\mathsf{Aux},\mathbb{O}}(\mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}))$.*

Proof. The proof is similar to the proof for Lemma 1. Assume that there is an adversary $\mathcal{A}$ which can violate the certain security property of $\Pi[\mathtt{sid}]$ in the composition. Then, we can turn $\mathcal{A}$ into a $\mathcal{B}^{\mathbb{O}}$ which violates the security of $\Pi[\mathtt{sid}]$ in the stand-alone setting. $\mathcal{B}$ runs a copy of $\mathcal{A}$ in the following manner: it serves as a relay between $\mathcal{A}$ and the instance $\Pi[\mathtt{sid}]$ (so $\mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}$ is available to both $\mathcal{A}$ and $\mathcal{B}$), and provides $\mathsf{VIEW}_{\mathcal{A},\Pi'[\mathtt{sid}']} = \mathcal{S}^{\mathbb{O}}(\mathsf{Aux}, \mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]})$ to $\mathcal{A}$. Since $(\mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}, \mathsf{VIEW}_{\mathcal{A},\Pi'[\mathtt{sid}']}) = (\mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}, \mathcal{S}^{\mathbb{O}}(\mathsf{Aux}, \mathsf{VIEW}_{\mathcal{A},\Pi[\mathtt{sid}]}))$, $\mathcal{A}$ can violate the security of $\Pi[\mathtt{sid}]$, so can $\mathcal{B}^{\mathbb{O}}$. □

Oracle-emulatable protocols. Note that in Lemma 2, we show that a protocol instance $\Pi[\mathtt{sid}]$ is secure in the composition with a protocol instance $\Pi'[\mathtt{sid}']$, if $\Pi[\mathtt{sid}]$ is secure when $\mathbb{O}$ is available to $\mathcal{A}$, and $\Pi'[\mathtt{sid}']$ can be "emulated" using oracles in $\mathbb{O}$ and the view of $\mathcal{A}$ in $\Pi[\mathtt{sid}]$. However, whether $\Pi'[\mathtt{sid}']$ can be "emulated" using $\mathbb{O}$ and the view of $\mathcal{A}$ in $\Pi[\mathtt{sid}]$ needs to be checked w.r.t. $\Pi[\mathtt{sid}]$. Instead, we would like to have more general results regarding the "emulatability" of $\Pi'[\mathtt{sid}']$, independent of which $\Pi[\mathtt{sid}]$ will be composed with. In the following, we define $\mathbb{O}$-emulatable instances. In particular, we would like to cover the case that under the same setup, there are multiple instances of the same protocol, which will be helpful in arguing the composition security for the instances using the same setup.

Definition 2 ($\mathbb{O}$-emulatable protocol). *Let $\Pi.\mathsf{Setup}[\mathtt{sid}]$ be an instance of the setup phase of protocol Π, and let $\Pi[\langle \mathtt{sid}, \mathtt{sid}_0 \rangle]$ be an instance under the setup of $\Pi.\mathsf{Setup}[\mathtt{sid}]$. We say $\Pi[\langle \mathtt{sid}, \mathtt{sid}_0 \rangle]$ is $\mathbb{O}$-emulatable under $\Pi.\mathsf{Setup}[\mathtt{sid}]$, if for an arbitrary protocol instance $\Pi'[\mathtt{sid}']$ running in an arbitary composition with* $\mathsf{Align.Setup}[\mathtt{sid}]$ *and* Align*, and any PPT adversary $\mathcal{A}$ against the composition, there exists a PPT algorithm $\mathcal{S}$ which can query the oracles in $\mathbb{O}$ with messages starting with $\mathtt{sid}$, such that the view of $\mathcal{A}$ in the composition, which consists of*

$$(\mathsf{VIEW}_{\mathcal{A},\Pi'[\mathtt{sid}']}, \mathsf{VIEW}_{\mathcal{A},\Pi.\mathsf{Setup}[\mathtt{sid}]}, \mathsf{VIEW}_{\mathcal{A},\Pi[\langle \mathtt{sid}, \mathtt{sid}_0 \rangle]}),$$

distributes identically with

$$(\mathsf{VIEW}_{\mathcal{A},\Pi'[\mathtt{sid}']}, \mathsf{VIEW}_{\mathcal{A},\Pi.\mathsf{Setup}}, \mathcal{S}^{\mathbb{O}}(\{v_i\}_{i \in \mathcal{H}}, \mathsf{VIEW}_{\mathcal{A},\Pi'[\mathtt{sid}']}, \mathsf{VIEW}_{\mathcal{A},\Pi.\mathsf{Setup}})).$$

*If removing Π.*Setup$[sid]$ *from above definition, then* $\Pi[\langle sid, sid_0\rangle]$ *is* $\mathbb{O}$*-emulatable.*

How we analyze the composition security for components in this paper. We use the framework established above to ensure the composition security of components in this paper. Specifically, we additionally prove two facts about every component. First, this component remains secure even given a set of oracles to the adversary. Second, all information an adversary can learn from an instance of the component can be emulated using a set of oracles. Therefore, when arguing whether a component A is secure in a composition, we only need to check whether the rest of the composition can be emulated using the set of oracles with which the component A remains secure.

As most of the composition analysis for components in this paper is easy to follow and more like a sanity check, we defer all composition analyses to a dedicated section in the Appendix. D.

5 Asynchronous Subset Alignment

One of the main tools in our design of communication-efficient asynchronous coin protocol is a new primitive termed *Asynchronous Subset Alignment* (Align). It can be seen as a (strictly) weaker form of the well-studied Gather problem [3, 27] or the Weak Core Set problem [42], while it is actually sufficient for weak coin constructions as we will soon demonstrate in the next section.

5.1 Definition

Following the intuition discussed in Sect.2, we formally define the primitive as follows. We discuss why each property is needed in remarks after the definition.

Syntax. Let Π be a protocol involving n participants in which each P_i has an input v_i and will be instructed to output a set OSet_i which either is empty or consists of tuples in the form of (j, v_j, θ_j). We say Π is an (n, f)-secure Align with a predicate function $\mathsf{Predicate} : \{0,1\}^* \to \{0,1\}$, if it is equipped with an algorithm $\mathsf{Align.Vrfy} : (\mathtt{sid}, j, v_j, \theta_j) \to \{0,1\}$, such that in any instance $\Pi[\mathtt{sid}]$, and for any PPT adversary $\mathcal{A}$ which can corrupt up to f nodes, the *termination*, *external validity*, and $f+1$-*alignment* properties are ensured.

For facilitating a formal definition, we use the following variables: (1) $\mathcal{H}_{\mathtt{first}} \subset [n]$, so that $\{P_i\}_{i\in\mathcal{H}_{\mathtt{first}}}$ is the set of all so-far-honest nodes at the time the first honest P_i outputted; (2) $\mathcal{H}_{\mathsf{end}} \subset [n]$, so that $\{P_i\}_{i\in\mathcal{H}_{\mathsf{end}}}$ is the set of all forever-honest nodes; (3) $\mathsf{State}_{\mathcal{A}}^{\mathtt{end}}$, which is the state of $\mathcal{A}$ at the time all honest nodes have outputted in the instance; (4) $\mathsf{State}_i^{\mathtt{first}}$, which is the state of P_i at the time that the first honest node generated an output. With these notions, we define the properties as follows.

- *Termination.* When every honest participant P_i is activated on $\mathtt{sid}$ with a valid input v_i such that $\mathsf{Predicate}(v_i) = 1$, except with a negligible probability, every honest participant P_i eventually outputs a set OSet_i which is either $\emptyset$ or in the form of $\{(j, v_j, \theta_j)\}_{j\in S_i}$ for some $S_i \subset [n]$.
- *External Validity.* When an honest node P_i outputs a non-empty $\mathsf{OSet}_i = \{(j, v_j, \theta_j)\}_{j\in S_i}$, it satisfies that for all $j \in S_i$, $\mathsf{Align.Vrfy}(\mathtt{sid}, j, v_j, \theta_j) = 1$ and $\mathsf{Predicate}(v_j) = 1$.
- $(f+1)$-*Alignment.* There is a polynomial-time algorithm CSGen, which on the input of $\{\mathsf{State}_i^{\mathsf{first}}\}_{i\in\mathcal{H}_{\mathsf{first}}}$, outputs a set $\mathsf{CoverSet} = \{(j, v_j)\}_{j\in\mathbb{I}_c}$ so that $n - f \le |\mathsf{CoverSet}| = |\mathbb{I}_c| \le n$,

and a subset $\mathsf{WS} = \{(j, v_j)\}_{j \in \mathbb{I}_s} \subset \mathsf{CoverSet}$ s.t. $|\mathbb{I}_s| = n - f$. For any PPT algorithm $\mathcal{A}'$, it holds that

$$\Pr\left[\begin{array}{l} \mathsf{CSGen}(\{\mathsf{State}_i^{\mathtt{first}}\}_{i \in \mathcal{H}_{\mathtt{first}}}) \to (\mathsf{CoverSet}, \mathsf{WS}) \\ \mathcal{A}'(\mathsf{State}_{\mathcal{A}}^{\mathsf{end}}, \{\mathsf{OSet}_i\}_{i \in \mathcal{H}_{end}}) \to (j', v', \theta') : \\ \left(\exists (j, v_j) \in \mathsf{WS}, \text{s.t.} \sum_{i \in \mathcal{H}_{\mathsf{end}}} \mathsf{isCovered}(\mathsf{OSet}_i, (j, v_j)) < f + 1\right) \\ \bigvee (\mathsf{Align.Vrfy}(\mathtt{sid}, j', v', \theta') = 1 \wedge (j', v') \notin \mathsf{CoverSet}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{isCovered}(\mathsf{OSet}_i, (j, v_j)) = 1$, if and only if there exists some θ_j s.t. $(j, v_j, \theta_j) \in \mathsf{OSet}_i$.

Remark 5. By properly defining Predicate, the external validity ensures the outputs in Align are "meaningful" to its application scenarios. Looking ahead, in our weak common coin protocol, we expect Align to return sets of valid aggregated ciphertexts. Without this property, each node may return a set of invalid ciphertexts that cannot be decrypted at all, which is useless for our application even if the other properties are all ensured.

Remark 6. The $(f+1)$-alignment captures that although the output sets OSet_i's of different nodes can be different, all of them are "well-aligned" according to certain constraints which are determined when the first honest generated its output set. In particular:

- There is a $(n - f)$-sized well-covered subset WS of CoverSet, such that for every (j, v_j) in the subset, it must be contained in the output set of at least $f + 1$ forever honest nodes.
- There is binding cover CoverSet determined at the time the first honest node generated its output, such that all valid values w.r.t. Align.Vrfy have been contained in it. It implies the output set OSet_j of any honest P_j is a subset of CoverSet.

Remark 7. The existence of both the well-covered subset WS and the binding cover set CoverSet is fundamental to our weak common coin protocol.

As we sketched in Sect.2, in our weak common coin protocol, after Align, there will be a ranking phase to rank all values in the output set of Align, and it is important to ensure that there is a constant probability that all nodes can decide on the highest-ranked value. So, intuitively, as WS has $n - f$ values, and CoverSet has at most n values, there is a good probability that WS has the value with the highest rank. So, $f + 1$ honest nodes will immediately decide on this value. As ensured by CoverSet, the adversary cannot inject new values with a valid proof w.r.t. Align.Vrfy, so all honest nodes can decide on the highest-ranked value with the help of the $f + 1$ honest nodes.

It is also important to ensure that both CoverSet and WS must be determined at the time that the first honest node generates an output. This is because when an honest node finishes Align, it may start to assist in ranking the values (in our weak common coin, it helps decrypt all ciphertexts). If CoverSet and WS have not been decided at that point, $\mathcal{A}$ may, according to the ranking information, maliciously delay messages so that the highest-ranked value will never be contained in WS.

5.2 Our Construction

Intuition. There is an implicit requirement from the $(f + 1)$-alignment property that for every node j, there is at most one value v_j such that (j, v_j) has a valid proof with respect to Align.Vrfy. A natural idea is to use provable broadcast (PB) [4,20] to ensure this property. We recall the definition of PB in Sect. 3.1 and provide a threshold-signature-based construction in Appendix B.2. At a high level, a sender broadcasts its message v to the network, and it is ensured that for each instance, the sender can obtain a delivery proof for at most one value v, perfectly aligning with this security requirement. Additionally, the existence of a proof for a value v indicates that at least $f+1$ honest nodes have delivered v in PB.

In the following, we start with a simple PB-based protocol, analyze where it fails, and gradually strengthen it to satisfy all properties of Align.

First, assume that each node broadcasts its input value to the network via PB. After obtaining a delivery proof for its input, it multicasts the proof. A node can terminate when receiving $n-f$ valid proofs. However, although the value provided by each node will be unique, there will be no *well-covered set* that can be determined at the time the first honest node outputs.

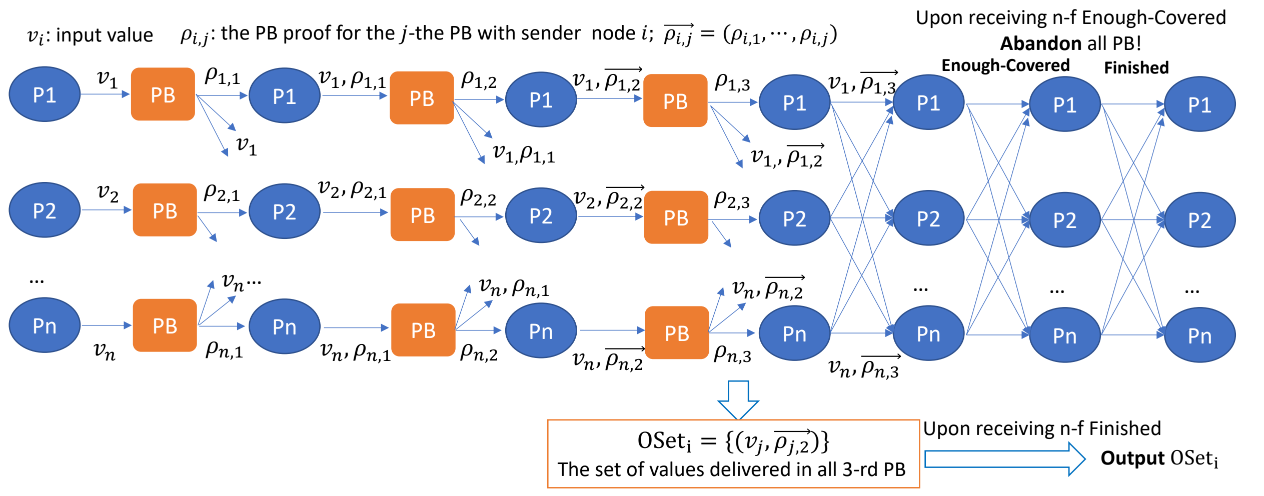
Next, consider a solution with two consecutive PB instances. Specifically, after obtaining a proof for the first PB instance, a sender invokes the second PB to broadcast *both* the proof ρ_1 and the input value v. Receivers deliver this message only when ρ_1 is a valid proof for v with respect to the first PB instance. Then, after receiving the proof ρ_2 for the second PB, the sender multicasts the value v along (ρ_1, ρ_2) to the network. Note that a proof ρ_2 for v with respect to the second PB instance means that at least $f+1$ honest nodes have delivered the value v in the second PB instance. Meanwhile, in the second PB instance with sender P_j, the values delivered by all honest nodes must be the same value v, since v has the delivery proof for the first PB instance with P_j. Thus, if each honest node outputs values delivered in the "second" PB instances upon seeing $n-f$ values with proofs for the "second" PB instances, the existence of a *well-covered set* is guaranteed. Specifically, the $n-f$ values with delivery proofs of the "second" PB instances must be well-covered, and $\mathsf{Align.Vrfy}$ verifies if the values contain the corresponding proofs for PB.

However, there are still security issues regarding the binding cover property (part of $(f+1)$-alignment). So far, we cannot decide all values with PB proofs at the moment that the first honest node outputs. Even if the node that outputs stops responding to the PB instances (or "abandons" the PB instances), there are still enough nodes in the system to produce a PB proof. Therefore, we add a few rounds of communication (called the "tail" phase) after the "second" PB. With the tail phase, an honest node terminates when $f+1$ honest nodes have abandoned all PB instances while ensuring that everyone can terminate even if many honest nodes have abandoned the PB instances.

Nonetheless, even with all these measures, we cannot guarantee that the size of the binding cover is bounded by n. A corrupted node can send $f+1$ different values in the "first" PB instance to $f+1$ honest nodes before they abandon the PBs. When the first honest node outputs, the corrupted node can still select any of those $f+1$ values to finish the first PB, causing the selected value to appear in some honest nodes' output sets. Since there are up to f corrupted nodes, the number of values that may appear in the output sets is $f(f+1)$, which exceeds n.

To address this issue, we introduce an extra PB instance to "fully lock" the input value to the "first" PB. This PB instance is added before the "first" PB instance discussed above, resulting in our protocol with n parallel chains of three consecutive PB instances. The execution flow of the protocol is described in Fig. 6.

Construction. With a silent setup phase for enabling PB, we present the pseudocode of our Align protocol in Algorithm 1. Our protocol works as follows:



v_i: input value
$\rho_{i,j}$: the PB proof for the j-the PB with sender node i; $\overrightarrow{\rho_{i,j}} = (\rho_{i,1}, \cdots, \rho_{i,j})$
Upon receiving n-f Enough-Covered
Abandon all PB!
Enough-Covered
Finished
P1
P2
Pn
PB
$\mathsf{OSet_i} = \{(v_j, \overrightarrow{\rho_{j,2}})\}$
The set of values delivered in all 3-rd PB
Upon receiving n-f Finished
Output $\mathsf{OSet_i}$

Fig. 6. Asynchronous Subset Alignment Align

- *First PB:* At Line 4-5, every honest node P_i broadcasts its input v_i via the PB instance $\mathsf{PB}[\langle \mathtt{sid}, i, 1\rangle]$, whose predicate is the external predicate of Align.
- *Second PB:* At Line 6-7, upon obtaining a proof $\rho_{i,1}$ w.r.t $\mathsf{PB}[\langle \mathtt{sid}, i, 1\rangle]$, P_i broadcasts $(v_i, \rho_{i,1})$ via $\mathsf{PB}[\langle \mathtt{sid}, i, 2\rangle]$, whose predicate is that $\rho_{i,1}$ should be a valid delivery proof for v_i w.r.t. $\mathsf{PB}[\langle \mathtt{sid}, i, 1\rangle]$. In addition, at Line 12-13, when P_i delivers $(v_j, \rho_{j,1})$ in $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle]$, it adds (j, v_j) to Lock_i.
- *Thrid PB:* At Line 8-9, upon obtaining a proof $\rho_{i,2}$ w.r.t $\mathsf{PB}[\langle \mathtt{sid}, i, 2\rangle]$, P_i broadcasts $(v_i, \rho_{i,1}, \rho_{i,2})$ via $\mathsf{PB}[\langle \mathtt{sid}, i, 3\rangle]$, whose predicate is that $\rho_{i,2}$ should be a valid delivery proof for $(v_i, \rho_{i,1})$ w.r.t. $\mathsf{PB}[\langle \mathtt{sid}, i, 2\rangle]$. In addition, at Line 14-15, when P_i delivers $(v_j, \rho_{j,1}, \rho_{j,2})$ in $\mathsf{PB}[\langle \mathtt{sid}, j, 3\rangle]$, it adds $(j, v_j, \rho_{j,1}, \rho_{j,2})$ to OSet_i.
- *Tail Phase:* At line 10-11, upon obtaining a proof $\rho_{i,3}$ w.r.t $\mathsf{PB}[\langle \mathtt{sid}, i, 3\rangle]$, P_i multicasts $(\mathtt{sid}, \text{COVERED}, v_i, \rho_{i,1}, \rho_{i,2}, \rho_{i,3})$. At line 16-20, after receiving valid COVERED messages from at least $n - f$ distinct nodes, P_i multicasts $(\mathtt{sid}, \text{ENOUGH-COVERED})$. Otherwise, it will multicast this message when it receives $(\mathtt{sid}, \text{ENOUGH-COVERED})$ from at least $f + 1$ distinct nodes.
- *Abandon:* An honest P_i abandons all PB instances after receiving $(\mathtt{sid}, \text{ENOUGH-COVERED})$ from at least $n - f$ distinct nodes. In the meanwhile, $(\mathtt{sid}, \text{FINISHED})$ is multicasted to the network.
- *Output:* Finally, upon receiving the FINISHED messages from $n - f$ distinct nodes, P_i outputs OSet_i.

Algorithm 1 The Align protocol with identifier $\mathtt{sid}$ and $\mathsf{Predicate}$ for P_i

```
1: $\mathsf{Locked}_i, \mathsf{OSet}_i, \mathsf{Covered}_i \leftarrow \emptyset$
2: for $j = 1$ to $n$ do
3:   Initialize $\mathsf{PB}[\langle \mathtt{sid}, j, \textsc{Num}\rangle]$ with the predicate $\mathsf{Pred}_{\textsc{Num}}$, for $\textsc{Num} = 1, 2, 3$

4: upon activated with input $v_i$ s.t. $\mathsf{Predicate}(v_i) = 1$ do
5:   Invoke $\mathsf{PB}[\langle \mathtt{sid}, i, 1\rangle]$ as the sender with input $v_i$.
6:   wait until $\mathsf{PB}[\langle \mathtt{sid}, i, 1\rangle]$ returns $\rho_{i,1}$
7:     Invoke $\mathsf{PB}[\langle \mathtt{sid}, i, 2\rangle]$ as the sender with input $(v_i, \rho_{i,1})$
8:     wait until $\mathsf{PB}[\langle \mathtt{sid}, i, 2\rangle]$ returns $\rho_{i,2}$
9:       Invoke $\mathsf{PB}[\langle \mathtt{sid}, i, 3\rangle]$ as the sender with input $(v_i, \rho_{i,1}, \rho_{i,2})$
10:      wait until $\mathsf{PB}[\langle \mathtt{sid}, i, 3\rangle]$ returns $\rho_{i,3}$
11:        Multicast $(\mathtt{sid}, \textsc{Covered}, v_i, \rho_{i,1}, \rho_{i,2}, \rho_{i,3})$

12: upon $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle]$ delivers $(v_j, \rho_{j,1})$ do
13:   $\mathsf{Locked}_i \leftarrow \mathsf{Locked}_i \cup \{(j, v_j)\}$

14: upon $\mathsf{PB}[\langle \mathtt{sid}, j, 3\rangle]$ delivers $(v_j, \rho_{j,1}, \rho_{j,2})$ do
15:   $\theta_j \leftarrow (\rho_{j,1}, \rho_{j,2})$, $\mathsf{OSet}_i \leftarrow \mathsf{OSet}_i \cup \{(j, v_j, \theta_j)\}$

16: upon receiving $(\mathtt{sid}, \textsc{Covered}, v_j, \rho_{j,1}, \rho_{j,2}, \rho_{j,3})$ from $P_j$ for the first time do
17:   if $\mathsf{PB}[\langle \mathtt{sid}, j, 3\rangle].\mathsf{Vrfy}((v_j, \rho_{j,1}, \rho_{j,2}), \rho_{j,3}) = 1$ then
18:     $\mathsf{Covered}_i \leftarrow \mathsf{Covered}_i \cup \{(j, v_j)\}$
19:   if $|\mathsf{Covered}_i| = n - f$ and the $\textsc{Enough-Covered}$ has not been sent then
20:     Multicast $(\mathtt{sid}, \textsc{Enough-Covered})$

21: upon receiving $(\mathtt{sid}, \textsc{Enough-Covered})$ from at least $f+1$ distinct nodes and $P_i$ has not sent $\textsc{Enough-Covered}$
    do
22:   Multicast $(\mathtt{sid}, \textsc{Enough-Covered})$

23: upon receiving $(\mathtt{sid}, \textsc{Enough-Covered})$ from at least $n - f$ distinct nodes do
24:   for $j = 1$ to $n$ do
25:     Call $\mathsf{PB}.\mathsf{Abandon}(\langle \mathtt{sid}, j, \textsc{Num}\rangle)$, for $\textsc{Num} = 1, 2, 3$.
26:   Multicast $(\mathtt{sid}, \textsc{Finished})$

27: upon receiving $(\mathtt{sid}, \textsc{Finished})$ from at least $n - f$ distinct nodes do
28:   return $\mathsf{OSet}_i$
29:
    Algorithm $\mathsf{Align}.\mathsf{Vrfy}(\mathtt{sid}, j, v_j, \theta_j)$
30:   Parse $\theta_j \rightarrow (\rho_{j,1}, \rho_{j,2})$, and return $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle]((v_j, \rho_{j,1}), \rho_{j,2})$
31:
    Predicate $\mathsf{Pred}_{\textsc{Num}}$
32:   $\mathsf{Pred}_1(v) = 1$ iff $\mathsf{Predicate}(v) = 1$
33:   $\mathsf{Pred}_2(j, v_j, \rho_{j,1}) = 1$ iff $\mathsf{PB}[\langle \mathtt{sid}, j, 1\rangle].\mathsf{Vrfy}(v_j, \rho_{j,1}) = 1$
34:   $\mathsf{Pred}_3(j, v_j, \rho_{j,1}, \rho_{j,2}) = 1$ iff $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle].\mathsf{Vrfy}((v_j, \rho_{j,1}), \rho_{j,2}) = 1$
```

Efficiency analysis. Note that a PB instance incurs $O(n(\ell + \lambda))$-bit communication cost and 2 rounds when broadcasting ℓ bits. And its proof size is $O(\lambda)$. Since our protocol involves three consecutive PB instances and three additional rounds after PB, it requires 9 rounds to terminate. It is easy to verify the communication complexity is $O(n^2(\ell + \lambda))$.

Instantiation and assumptions. The Align protocol is built upon a PB protocol, for which we sketched an instantiation in Appendix.B.2 based on a silent threshold signature scheme [26]. The silent threshold signature scheme is proven to be secure under the q-SDH assumption [16, 57] and the co-CDH assumption in the algebraic group model (AGM) and the random oracle model. Thus, our Align protocol inherites the assumptions needed by the silent threshld signature scheme [26].

Security analysis. We establish the security of our Align protocol in the following theorem.

Theorem 1 (Align). *Assuming the security of the underlying PB, the* Align *protocol in Algorithm 1 ensures the properties of the termination, external validity, and* $f+1$*-alignment.*

We proceed with the proof by analyzing the properties in the following lemmas.

Lemma 3 (Align termination). *Our Align protocol ensures termination, i.e., every honest node* P_i *can eventually terminate with a set* OSet_i*, which can be empty.*

Proof. We show that every honest node P_i can proceed with the instruction of returning OSet_i (line 28). Note that if an honest node P_j called $\mathsf{PB.Abandon}$ (line 25), then it must have received the ENOUGH-COVERED message from at least $n-f$ nodes, while at least $f+1$ among them are forever-honest. Therefore, every honest node can receive at least $f+1$ ENOUGH-COVERED messages. Following the instruction at line 21, all honest nodes will eventually multicast the ENOUGH-COVERED message. Hence, all honest nodes will multicast the FINISHED message (line 26), which grants the execution of the instruction at line 28. It leaves us to discuss whether there exists an honest node that can receive $n-f$ ENOUGH-COVERED messages, and we prove this by showing contradictions. In particular, if no honest node saw $n-f$ ENOUGH-COVERED messages, then no honest node called $\mathsf{PB.Abandon}$. Following the termination property of PB, all honest senders will eventually obtain the broadcast proofs. It follows that all honest nodes can receive $n-f$ "valid" COVERED messages and then broadcast the ENOUGH-COVERED messages, which grants that an honest node to observe $n-f$ ENOUGH-COVERED messages. □

Lemma 4 (Align External Validity). *The protocol* Align *satisfies the external validity.*

Proof. According to the protocol description, an honest node P_i will add a tuple (j, v_j, θ_j) to the set OSet_i only when $\mathsf{PB}[\langle \mathtt{sid}, j, 3\rangle]$ delivers $(v_j, \rho_{j,1}, \rho_{j,2})$. Ensured by the external validity of PB, it follows that $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle].\mathsf{Vrfy}(v_j, \rho_{j,1}, \rho_{j_2}) = 1$, which implies that at least $f+1$ honest nodes delivered $(v_j, \rho_{j,1})$ in $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle]$. Following a similar arguement, at least $f+1$ honest nodes delivered v_j in $\mathsf{PB}[\langle \mathtt{sid}, j, 1\rangle]$, which implies $\mathsf{Predicate}(v_j) = 1$. □

Lemma 5 (Align $f+1$-Alignment). *Our Align protocol satisfies the* $f+1$*-alignment.*

Proof. First, we describe the algorithm CSGen, which on input of $\{\mathsf{State}_i^{\mathsf{first}}\}_{i\in\mathcal{H}_{\mathsf{first}}}$ outputs a cover $\mathsf{CoverSet} = \{(j, v_j)\}_{j\in\mathbb{I}_c}$ and a well covered subset $\mathbb{I}_s$. In particular, let $\{\mathsf{Locked}_j\}_{j\in\mathcal{H}_{\mathsf{first}}}$ be the local

Locked sets of so-far-honest nodes the time that the first honest nodes generated an output, which are contained in $\{\mathsf{State_i^{first}}\}_{i\in\mathcal{H}_{\mathsf{first}}}$. CSGen will output

$$\mathsf{CoverSet} = \bigcup_{j\in\mathcal{H}} \mathsf{Locked}_j = \{(j, v_j)\}_{j\in\mathbb{I}_c}$$

Regarding the well-covered subset $\mathbb{I}_w$, CSGen checks the $\{\mathsf{Covered}_i\}_{i\in\mathcal{H}_{\mathtt{first}}}$ (also available in the states), and finds any i' such that $|\mathsf{Covered}_{i'}| = n - f$. It outputs $\mathsf{WS} = \mathsf{Covered}_{i'}$.

Now we argue that $\mathsf{CoverSet}$ are well-formed. Namely, $|\mathsf{CoverSet}| = |\mathbb{I}_c| \leq n$, which implies that there are no $(j, v_j), (j, v'_j) \in \mathsf{CoverSet}$ such that $v_j \neq v'_j$. Since each $(z, v_z) \in \mathsf{Locked}_j$, P_j delivers $(v_z, \rho_{z,1})$ in $\mathsf{PB}[\langle \mathtt{sid}, z, 2\rangle]$. According the predicate Pred_2 of $\mathsf{PB}[\langle \mathtt{sid}, z, 2\rangle]$, it holds that $\mathsf{PB}[\langle \mathtt{sid}, z, 1\rangle].\mathsf{Vrfy}(v_z, \rho_{z,1}) = 1$. Ensured by the provability of PB, if two honest parties deliver $(v_z, \rho_{z,1})$ and $(v'_z, \rho'_{z,1})$ in $\mathsf{PB}[\langle \mathtt{sid}, z, 2\rangle]$, respectively, it must follow $v_z = v'_z$. Therefore, $|\mathsf{CoverSet}| \leq n$.

Next, we show that if the adversary $\mathcal{A}'$ outputs some (j, v_j, θ_j) such that $\mathsf{Align.Vrfy}(\mathtt{sid}, j, v_j, \theta_j) = 1$, (j, v_j) must belong to $\mathsf{CoverSet}$. Note that when P_i outputs, at least $f + 1$ forver-honest nodes have multicasted the FINISHED message. According to line 23-25, these $f + 1$ nodes have called $\mathsf{PB.Abandon}$ for all PB instances. On the other hand, if (j, v_j) is not included in $\mathsf{CoverSet}$, then no honest party has delivered v_j in $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle]$; In addition, there are $f + 1$ forever-honest nodes who have abandoned all PB instances, then, according to the provability of PB, the sender cannot obtain a proof $\rho_{j,2}$ for the value v_j. Therefore, due to the absence of the proof, no tuple in the form of $(j, v_j, \rho_{j,1}, \cdot)$ can be delivered in $\mathsf{PB}[\langle \mathtt{sid}, j, 3\rangle]$, so (j, v_j) will not be included in OSet_i of any honest P_i.

Then, we argue that CSGen can always find some $i' \in \mathcal{H}_{\mathsf{first}}$ such that $|\mathsf{Covered}_{i'}| = n - f$, and $\mathsf{Covered}_{i'}$ is a subset of $\mathsf{CoverSet}$. Note that an honest node P_i outputs OSet_i when it has seen $n - f$ FINISHED messages, while at least $f + 1$ of them are sent by forever-honest nodes, which implies at least $f+1$ forever honest nodes have seen $n-f$ ENOUGH-COVERED messages. Then, at least one honest $P_{i'}$ sent the ENOUGH-COVERED message because of $|\mathsf{Covered}_{i'}| = n - f$. For every $(j, v_j) \in \mathsf{Covered}_{i'}$, according to Line 17, there exists a proof showing it has been delivered by at least $f + 1$ forever honest nodes in $\mathsf{PB}[\langle \mathtt{sid}, j, 3\rangle]$, and it follows that it was delivered by at least $f + 1$ forever honest nodes in $\mathsf{PB}[\langle \mathtt{sid}, j, 2\rangle]$. Therefore, according to Line 12-13, (j, v_j) must been included in at the Locked_i of at least $f + 1$ forever honest P_i's, which makes $(j, v_j) \in \mathsf{CoverSet}$

Finally, we prove that for every $j \in \mathbb{I}_s$, (j, v_j) must appear in the output set of at least $f+1$ forver honest nodes. For every $(z, v_z) \in \mathsf{Covered}_{i'}$, there exists a tuple $(v_z, \rho_{z,1}, \rho_{z,2})$ that was delivered by $\mathsf{PB}[\langle \mathtt{sid}, z, 3\rangle]$, such that $\mathsf{PB}[\langle \mathtt{sid}, z, 2\rangle].\mathsf{Vrfy}((v_z, \rho_{z,1}), \rho_{z,2}) = 1$, Ensured by the provability of PB, at least $f + 1$ forever-honest nodes have delivered $(v_z, \rho_{z,1})$ in $\mathsf{PB}[\langle \mathtt{sid}, z, 2\rangle]$. Therefore, (z, v_z) is included in at least $f + 1$ forever-honest nodes' OSet. □

6 Communication-Optimal Asynchronous Weak Coin

Following the intuition from Sect.2, we present our asynchronous weak common coin protocol with $O(\lambda n^2)$ communication complexity and $O(1)$ round complexity.

6.1 The Construction

Building blocks: Our scheme is built upon the following ingredients.

STHE. An (n, f)-secure STHE scheme $\mathsf{STHE} = \{\mathsf{STHE.SilSetup}, \mathsf{STHE.Enc}, \mathsf{STHE.Eval}, \mathsf{STHE.PartDec}, \mathsf{STHE.PartVrfy}, \mathsf{STHE.DkAgg}, \mathsf{STHE.Dec}\}$, introduced in Sect.4.1, is assumed in our weak coin protocol. For notational convenience, we denote its message space and the randomness space by $\mathsf{Msg}_{\mathsf{te}}$ and $\mathsf{Rand}_{\mathsf{te}}$, respectively.

Digital signature. A standard-model digtial signature scheme $\mathsf{DS} = \{\mathsf{DS.KeyG}, \mathsf{DS.Sign}, \mathsf{DS.Vrfy}\}$ is assumed. DS shall satisfy the conventional *existential unforgeability under chosen message attacks*, which could have various instantiations, for example, the Boneh-Boyen signature [16].

Argument systems. We need a $\mathsf{NIZK} = \{\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Vrfy}\}$ for the following relation:

$$\mathcal{R}_{\mathsf{enc}} := \{((ek, tag, c, aux), (m, \gamma)) : c = \mathsf{STHE.Enc}(ek, tag, m; \gamma)\}, \tag{2}$$

which demonstrates that a ciphertext c is well-formed under ek and tag.

We further need a $\mathsf{SNARK} = \{\mathsf{SNARK.Setup}, \mathsf{SNARK.Prove}, \mathsf{SNARK.Vrfy}\}$ for the following relation:

$$\begin{aligned}\mathcal{R}_{\mathsf{agg}} :=\{&(n, f, (vk_i)_{i\in[n]}, ek, tag, \mathtt{crs}_{\mathsf{enc}}, c^*), \{(c_{i_j}, \pi_{\mathsf{enc},i_j}, \sigma_{i_j})\}_{j\in[t+1]} :\\ &c^* = \mathsf{STHE.Eval}(ek, tag, \{c_{i_j}\}_{j\in[f+1]}), \text{ and } \forall j \in [f+1], \mathsf{DS.Vrfy}(vk_{i_j},\\ &(tag, c_{i_j}), \sigma_{i_j}) = 1 \wedge \mathsf{NIZK.Vrfy}(\mathtt{crs}_{\mathsf{enc}}, (ek, tag, c_{i_j}, i_j), \pi_{\mathsf{enc},i_j}) = 1\}.\end{aligned} \tag{3}$$

SNARK is used to show a ciphertext c^* is obtain by honestly aggregating $f + 1$ ciphertexts that are signed by distinct users. As we introduced in Sect.3.1, NIZK shall satisfy the completeness, simulation extractability, and zero-knowledge property, while SNARK ensures the completeness and knowledge soundness and provides $O(1)$-sized proofs. In addition, as the verification algorithm of NIZK is a part of the relation $\mathcal{R}_{\mathsf{agg}}$, we require NIZK to be secure in the standard model.

Align. Our weak coin protoocl uses our (n, f)-secure Align protocol as a subroutine. In particular, the external predicate $\mathsf{Predicate}$, is parameterized by $(\mathtt{crs}_{\mathsf{agg}}, n, f, (vk_i)_{i\in[n]}, ek, \mathtt{sid}, \mathtt{crs}_{\mathsf{enc}})$, and $\mathsf{Predicate}(c^*, \pi^*) = 1$, iff

$$\mathsf{SNARK.Vrfy}(\mathtt{crs}_{\mathsf{agg}}, (n, f, (vk_i)_{i\in[n]}, ek, \mathtt{sid}, \mathtt{crs}_{\mathsf{enc}}, c^*), \pi^*) = 1. \tag{4}$$

Hash functions. We explicitly use two hash functions $H_1 : \{0,1\}^* \to \mathsf{TO}$ and $H_2 : \{0,1\}^* \to \mathsf{V}$, which will be modeled as random oracles in the security analysis. Here, TO is a totally ordered set with space greater than $2^{2\lambda}$, so that the probability of two independent sampling in TO yielding the same value is negligibly small. V is the space of coin values.

The protocol description. With the ingredients introduced above, we present our weak common coin protocol in Algorithm 2. It works under a silent setup, where (1) $\mathsf{STHE.SilSetup}$ has been finished, so every P_i has the key pair (ek_i, dk_i), while $(ek, (ek_i)_{i\in[n]})$ and the CRS $\mathtt{crs}_{\mathsf{te}}$ are available to everyone; (2) the CRS setups for both NIZK and SNARK have been done, so $\mathtt{crs}_{\mathsf{enc}}$ and $\mathtt{crs}_{\mathsf{agg}}$ are publicly available; (3) the PKI setup for DS is finished, so that everyone has its own key pair (vk_i, sk_i) and knows the public keys of others; (4) the silent setup needed by Align is also finalized.

At a high level, our protocol works as follows.

- *Contribute a random ciphertext.* At line 2-5, each user P_i encrypts a fresh chosen randomness under ek and the session ID $\mathtt{sid}$ to a ciphertext c_i, proves the well-formedness of c_i, and signs it. The ciphertext, c_i, the proof π_i, and the signature σ_i, are multicasted to the network.

Algorithm 2 The asynchronous weak coin protocol $\mathsf{WeakCoin}$ with identifier $\mathtt{sid}$ for P_i

```
1: Initialize: $\mathsf{Candidates}, \mathsf{LocWins} \leftarrow \emptyset$, and $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali} \rangle]$ instance.
2: upon activated with $\mathtt{sid}$ do
3:   $r_i \leftarrow\$ \mathsf{Msg}_{\mathsf{te}}, \gamma_i \leftarrow\$ \mathsf{Rand}_{\mathsf{te}}, c_i \leftarrow \mathsf{STHE.Enc}(ek, \mathtt{sid}, r_i; \gamma_i)$
4:   $\sigma_i \leftarrow \mathsf{DS.Sign}(sk_i, (\mathtt{sid}, c_i))$ $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathtt{crs}_{\mathsf{enc}}, (ek, \mathtt{sid}, c_i, i), (r_i, \gamma_i))$
5:   Multicast $(\mathtt{sid}, \textsc{Fresh}, c_i, \sigma_i, \pi_i)$
6: upon receiving $(\mathtt{sid}, \textsc{Fresh}, c_j, \sigma_j, \pi_j)$ from node $P_j$ for the first time do
7:   if $\mathsf{DS.Vrfy}(vk_j, (\mathtt{sid}, c_j), \sigma_j) = 1 \wedge \mathsf{NIZK.Vrfy}(\mathtt{crs}_{\mathsf{enc}}, (ek, \mathtt{sid}, c_j, j), \pi_j) = 1$ then
8:     $\mathsf{Fresh} \leftarrow \mathsf{Fresh} \cup \{(j, c_j, \sigma_j, \pi_j)\}$
9:   if $|\mathsf{Fresh}| = f + 1$ and has not invoked $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali} \rangle]$ then
10:    $c_i^* \leftarrow \mathsf{STHE.Eval}(ek, \mathtt{sid}, \{c_j\}_{(j,\dots) \in \mathsf{Fresh}})$          ▷ Aggregate then prove
11:    $\pi_i^* \leftarrow \mathsf{SNARK.Prove}(\mathtt{crs}_{\mathsf{agg}}, (n, f, (vk_i)_{i\in[n]}, ek, \mathtt{sid}, \mathtt{crs}_{\mathsf{enc}}, c^*), \mathsf{Fresh})$
12:    Invoke $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali} \rangle]$ with input $(c_i^*, \pi_i^*)$
13: upon $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali} \rangle]$ returns $\mathsf{OSet}_i = \{(z, c_z^*, \pi_z^*, \theta_z^*)\}$ do
14:   $\mathsf{Candidates} \leftarrow \mathsf{OSet}_i$
15:   if $\mathsf{Candidates} = \emptyset$ then
16:     Multicast $(\mathtt{sid}, \textsc{LocalWin}, \perp)$
17:   $\zeta_i \leftarrow \mathsf{STHE.PartDec}(ek, \mathtt{sid}, dk_i)$
18:   Multicast $(\mathtt{sid}, \mathsf{PartDec}, \zeta_i)$
19: upon receiving $(\mathtt{sid}, \textsc{PartDec}, \zeta_j)$ from $P_j$ for the first time do
20:   if $\mathsf{STHE.PartVrfy}(ek, \mathtt{sid}, pk_j, \zeta_j) = 1$ then
21:     $\mathsf{ParK} \leftarrow \mathsf{ParK} \cup \{(j, \zeta_j)\}$
22:   if $|\mathsf{ParK}| = n - f$ and has not sent $\textsc{LocalWin}$ messsage then
23:     $dk_{\mathtt{sid}} \leftarrow \mathsf{STHE.DkAgg}(ek, \mathtt{sid}, \{\zeta_k\}_{(k,)\in \mathsf{ParK}})$          ▷ decryption key
24:     for $(z, (c_z^*, \pi_z^*), \theta_z) \in \mathsf{Candidates}$ do
25:       $r_z^* \leftarrow \mathsf{STHE.Dec}(ek, \mathtt{sid}, dk_{\mathtt{sid}}, c_z^*)$, and $\mathsf{to}_z \leftarrow H_1(\mathtt{sid}, z, r_z^*)$
26:     Pick $a$ such that $\mathsf{to}_a = \mathsf{MAX}(\{\mathsf{to}_a\}_{(a,)\in \mathsf{Candidates}})$.
27:     Multicast $(\mathtt{sid}, \textsc{LocalWin}, (a, c_a^*, \pi_a^*, \theta_a))$
28: upon receiving $(\mathtt{sid}, \textsc{LocalWin}, (a, c_a^*, \pi_a^*, \theta_a))$ from $P_j$ for the first time do
29:   if $(a, c_a^*, \pi_a^*, \theta_a) = \perp$ then $\mathsf{LocWins} \leftarrow \mathsf{LocWins} \cup \{(a, \perp)\}$
30:   if $\mathsf{Align.Vrfy}(\mathtt{sid}, a, (c_a^*, \pi_a^*), \theta_a) = 1$ then $r_a^* \leftarrow \mathsf{STHE.Dec}(ek, \mathtt{sid}, dk_{\mathtt{sid}}, c_a^*)$
31:     $\mathsf{LocWins} \leftarrow \mathsf{LocWins} \cup \{(a, r_a^*, \mathsf{to}_a = H_1(\mathtt{sid}, a, r_a^*))\}$
32:   if $|\mathsf{LocWins}| = n - f$ then
33:     Decide $z$ such that $\mathsf{to}_z = \mathsf{Max}(\{\mathsf{to}_a\})$
34:     Return $H_2(\mathtt{sid}, z, r_z^*)$, where $H_2$ is a random oracle whose range is $V$.
```

- *Aggregate $f+1$ signed ciphertext.* At line 6-11, each user P_i collects $f+1$ ciphertexts which each has a well-formedness proof and a valid signature. Then, it aggregates those ciphertexts to c_i^*, and provides a proof π_i^* showing aggregation correctness. Note that any ciphertext c^* with a valid proof is an aggregation of $f+1$ ciphertexts, which contains at least one ciphertext contributed by the forever-honest node, so the encrypted value is unknown to the adversary before the decryption starts.
- *Reach a set of well-covered aggregated ciphertexts.* At line 12, each honest P_i provides (c_i^*, π_i^*) as input to Align, and it obtains a set $\mathsf{Candidates} = \{(j, c_j^*, \pi_j^*, \theta_j)\}$ at line 13-14, where θ_j is a proof w.r.t. Align, showing (c_j^*, π_j^*) is in the outcome of the Align.
- *Decide the "winner" among local candidates.* At lines 17-18, each honest P_i provides its partial decryption key w.r.t. the tag `sid`, and then at lines 19-23, it obtains the full decryption key for the tag `sid`. Then, at lines 24-25, P_i decrypts all ciphertexts in $\mathsf{Candidates}$ and then hashes the plaintexts into a total order set. At lines 26-27, P_i picks the ciphertext c_a^* whose decryption result yields the largest hash value as the "local winner", and then multicasts c_a^* with its validity proof in a LocalWin message. Note if $\mathsf{Candidates}$ is empty, P_i multicasts $\perp$ as a placeholder at lines 15-16.
- *Decide the output.* At lines 28-33, P_i collects $n-f$ LocalWin messages and then applies the same rule to decide the winner among all the ciphertexts carried by this textsc LocalWin messages. Applying another random oracle to the decryption of the winner ciphertext, P_i obtains the output value.

Efficiency Analysis. According to Algorithm 2, and as clearly shown in Fig.2, the protocol $\mathsf{WeakCoin}$ has 3 more rounds besides Align which in turn has 9 rounds. Hence, $\mathsf{WeakCoin}$ requires 12 rounds. In addition, since the input size to Align is merely $O(\lambda)$, the communication complexity due to Align is $O(\lambda n^2)$. Then, it is easy to verify the overall communication complexity is $O(\lambda n^2)$.

Instantiation and Assumptions. The STHE scheme is developed in Appendix.C, which is secure in the generic group model (GGM) and the random oracle model. The GGM and random oracle model indeed have subsumed the assumptions needed for other components. Specifically, the Align protocol requires the q-SDH assumption and the co-CDH assumption in the AGM, which can be implied by GGM, according to [56]. We can instantiate the underlying digital signature scheme with the Boneh-Boyen signature [16] and the SNARK with Groth16 [49], whose assumptions are all subsumed by GGM. Regarding the NIZK, it is easy to see that R_{enc} falls in the family of relations supported by the Groth-Sahai proof system [51]. So, a NIZK with simulation extractability for R_{enc} can be realized by applying the generic transformation [62] to the Groth-Sahai [51], and the assumptions needed for the NIZK can be subsumed by GGM.

6.2 The Security Analysis

We establish the security results of our $\mathsf{WeakCoin}$ in Theorem 2. Before proceeding to it, in the following, we consider a security game, which we call Random Aggregation Unpredictability (raUNP). Basically, this security notion captures all security guarantees that $\mathsf{WeakCoin}$ needs from the underlying STHE, NIZK, DS, and SNARK. It says that any PPT adversary $\mathcal{A}$ cannot produce an aggregated ciphertext c^* along with a valid SNARK proof π^* while knowing the decryption of c^*. Here π^* shows c^* is an honest aggregation of $f+1$ ciphertext from $f+1$ distinct nodes, i.e., the

relation R_{agg} in Eq.3. The rest of the proofs for Theorem 2 can be built upon the raUNP security without giving a security reduction to the security of each cryptographic component.
Random aggregation unpredictability. For facilitating the security analysis, we consider the *Random Aggregation Unpredictability* (raUNP) security game $\mathsf{Exp}^{\mathsf{raUNP}}_{\mathcal{A}}$ in Fig.7. In Lemma.6, we prove that the probability of winning the security is negligible for any PPT adversary.

```
$\mathsf{Exp}^{\mathcal{A},tag^*}_{\mathsf{raUNP}}(1^\lambda, n, f)$
$\mathcal{C}, \mathsf{Tags}, \mathsf{State}_1, \ldots, \mathsf{State}_n \leftarrow \emptyset$.Generate $\mathsf{CRS} := (\mathtt{crs}_{\mathsf{enc}}, \mathtt{crs}_{\mathsf{agg}}, \mathtt{crs}_{\mathsf{sthe}})$
$(\mathcal{C}_{\mathsf{Init}}, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathsf{CRS}), \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\mathsf{Init}}$
for $i \notin \mathcal{C}_{\mathsf{init}}$, Generate $(ek_i, dk_i), (vk_i, sk_i)$, provide public keys to $\mathcal{A}$, and wait $\mathcal{A}$
  to provide public keys of corrupted parties. Let $ek$ be the group encryption key
for $i \in [n] \setminus \mathcal{C}_{\mathsf{init}}$
  $\gamma_i \leftarrow \mathsf{Rand}_{\mathsf{te}}, r_i \leftarrow \mathsf{Msg}_{\mathsf{te}}, \hat{\gamma}_i \leftarrow \mathsf{Rand}_{\mathsf{zk}}, \bar{\gamma}_j \leftarrow \mathsf{Rand}_{\mathsf{sig}}, c_i \leftarrow \mathsf{STHE.Enc}(ek, tag^*, r_i; \gamma_i)$
  $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathtt{crs}_{\mathsf{enc}}, (ek, tag^*, c_i, i), (r_i, \gamma_i); \hat{\gamma}_i); \sigma_i \leftarrow \mathsf{DS.Sign}(sk_i, (tag^*, c_i); \bar{\gamma}_i)$
$(c^*, \pi^*, r^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\widetilde{\mathsf{Corr}}}(\cdot), \mathcal{O}_{\widetilde{\mathsf{PartD}}}(\cdot), \mathcal{O}_{\widetilde{\mathsf{Sign}}}(\cdot)}(\mathsf{state}_{\mathcal{A}}, (c_i, \pi_i, \sigma_i)_{i \in [n] \setminus \mathcal{C}_{\mathsf{init}}})$
if $\mathsf{SNARK.Vrfy}(\mathtt{crs}_{\mathsf{agg}}, (n, f, (vk_i)_{i \in [n]}, ek, tag^*, \mathtt{crs}_{\mathsf{enc}}, c^*), \pi^*) = 1$
  $\wedge\, r^* = \mathsf{STHE.Dec}(ek, dk_{tag^*}, c^*)$ then
  return 1   else return 0

$\mathcal{O}_{\widetilde{\mathsf{Sign}}}(i, (tag, \mathtt{msg}))$
if $tag \neq tag^* \wedge i \notin \mathcal{C}$
  $\mathsf{Tags} \leftarrow \mathsf{Tags} \cup \{tag\}, \gamma \leftarrow\!\!\$\ \mathsf{Rand}_{\mathsf{Sign}}$
  $\sigma_i \leftarrow \mathsf{DS.Sign}(sk_i, (tag, \mathtt{msg}); \gamma)$
  $\mathsf{State}_i \leftarrow \mathsf{State}_i \cup \{(\mathsf{Sign}, i, tag, \mathtt{msg}, \gamma, \sigma_i)\}$
  return $\sigma_i$

$\mathcal{O}_{\widetilde{\mathsf{PartD}}}(tag, i)$
if $tag \neq tag^* \wedge i \notin \mathcal{C}$
  $\mathsf{Tags} \leftarrow \mathsf{Tags} \cup \{tag\}, \gamma \leftarrow\!\!\$\ \mathsf{Rand}_{\mathsf{PartD}}$
  $\zeta_i \leftarrow \mathsf{PartDec}(ek, tag, dk_i; \gamma)$
  $\mathsf{State}_i \leftarrow \mathsf{State}_i \cup \{(\mathsf{PartD}, i, tag, \gamma, \zeta_i)\}$
  return $\zeta_i$

$\mathcal{O}_{\widetilde{\mathsf{Corr}}}(i)$
if $i \in [n] \setminus \mathcal{C} \wedge |\mathcal{C}| \leq f$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$; return $dk_i, sk_i, r_i, \gamma_i, \hat{\gamma}_i, \bar{\gamma}_i, \mathsf{State}_i$
```

Fig. 7. ra-UNP Security Game.

Lemma 6. *Assume* STHE, NIZK, SNARK, *and* DS *satisfy the security properties outlined in Sect.6.1. For any PPT adversary $\mathcal{A}$, any $\lambda \in \mathbb{N}$, any positive integers f and $n = 3f + 1$ that are polynomial in λ, any $tag^* \in \{0,1\}^*$, it holds that*

$$\mathsf{Adv}^{\text{raunp}}_{\mathcal{A},tag^*}(\lambda) := \Pr[\mathsf{Exp}^{\mathcal{A},tag^*}_{\mathsf{raUNP}}(1^\lambda, n, f) = 1] \leq \mathsf{negl}(\lambda),$$

where the experiment is defined in Fig.7.

Proof. We prove this lemma by using the following hybrid experiments. We use $\mathtt{Win}_k$ to denote that **Hybrid** k returns 1.
Hybrid 0. It is the experiment defined in Fig.7.

Hyrbid 1. It is almost identical to **Hybrid 0**, except that it has an additional rule:

- In the begining, uniformly sample $\bar{i} \leftarrow\$ [n]$. If, during the protocol execution, $\mathcal{A}$ chooses to corrupt $P_{\bar{i}}$, then abort the experiment.

At the point of $\mathcal{A}$'s view, as long as the experiment does not abort, then it is identical to **Hybrid 0**. Since the choice of $\bar{i}$ is independent of $\mathcal{A}$'s view, the probability that $\mathcal{A}$ chooses to corrupt $\bar{i}$ is bounded by f/n. Thus, $\Pr[\mathtt{Win}_1] \geq \Pr[\mathtt{Win}_0] \cdot (\frac{n-f}{n})$.

Hybrid 2. It is almost identical to Hybrid 1, except that it uses the simulated setup of NIZK to generate the CRS $\mathsf{crs}_{\mathsf{enc}}$ along with the trapdoor $\mathtt{tk}$, and $\pi_{\bar{i}}$ is generated with the simulated prover algorithm using $\mathtt{tk}$.

Ensured by the zero-knowledge of NIZK, we have $|\Pr[\mathtt{Win}_2] - \Pr[\mathtt{Win}_1]| \leq \mathsf{negl}(\lambda)$.

Hybrid 3. It is almost identical to Hybrid 2, except that under the condition that Hybrid 2 does not abort, after $\mathcal{A}$ provides the tuple (c^*, π^*, r^*), it additionally applies the extractor algorithm $E_{\mathcal{A}}$, whose existence is granted by the knowledge soundness of SNARK, to extract the knowledge $\{j, c_j, \sigma_j, \pi_j\}_{j \in \mathbb{I}'}$ for some $\mathbb{I}' \subset [n]$ and $|\mathbb{I}'| = f+1$. For every $j \in \mathbb{I}'$, check whether $\mathsf{DS.Vrfy}(vk_j, \sigma_j, c_j) = 1$ and $\mathsf{NIZK.Vrfy}(\mathsf{crs}_{\mathsf{enc}}, (ek, tag^*, c_j, j), \pi_j) = 1$. It aborts if any checking does not pass.

If Hybrid 3 does not abort due to the new checking procedure, then at the point of $\mathcal{A}$'s view, it is identical to Hybrid 2. Ensured by the knowledge soundness of SNARK, the probability that the extraction fails is negligible. Therefore, $|\Pr[\mathtt{Win}_3] - \Pr[\mathtt{Win}_2]| \leq \mathsf{negl}(\lambda)$.

Hybrid 4. It is almost identical to Hybrid 3, except that under the condition that Hybrid 3 does not abort, after extracting the knowledge $\{j, c_j, \sigma_j, \pi_j\}_{j \in \mathbb{I}'}$, it additionally checks whether $\bar{i} \in \mathbb{I}'$, and if not, it aborts.

At the point of $\mathcal{A}$'s view, as long as the experiment does not abort due to the new checking procedure, then it is identical to Hybrid 3. Notice that $\mathcal{A}$ has to include at least one corrupted index into $\mathbb{I}'$, and the choice of $\bar{i}$ is independent of $\mathcal{A}$'s view. Hence, the probability that $\mathcal{A}$ does not include $\bar{i}$ into $\mathbb{I}'$ is bounded by $\frac{1}{n-f}$. It folloiws that $\Pr[\mathtt{Win}_4] \geq \Pr[\mathtt{Win}_3] \cdot (\frac{1}{n-f})$.

Hybrid 5. It is almost identical to Hybrid 4, except that under the condition that Hybrid 4 does not abort, for any $j \in \mathbb{I}' \cap \mathcal{C}$, it additionally applies the simulation extractor SimExt, whose existence is granted by the simulation extractability of NIZK, to extract the knowledge of (r_j, γ_j) w.r.t. c_j for all $j \in \mathbb{I}' \setminus \{\bar{i}\}$, such that $\mathsf{STHE.Enc}(ek, tag^*, r_j; \gamma_j) = c_j$. If the extraction fails, it aborts.

Notice that since the statement (ek, tag^*, c_j, j) w.r.t. NIZK contains the identity j. Therefore, for all $j \notin \bar{i}$, the statement is not equal to the statement of the simulated proof. Hence, ensured by simulation extractability of NIZK, the probability that the extraction fails is negligible. It follow that $|\Pr[\mathtt{Win}_5] - \Pr[\mathtt{Win}_4]| \leq \mathsf{negl}(\lambda)$.

Hybrid 6. It is almost identical to Hybrid 5, except that under the condition that Hybrid 5 does not abort, it additionally checks whether the tuple $(\bar{i}, c_{\bar{i}}, \sigma_{\bar{i}}, \pi_{\bar{i}})$, extracted by $E_{\mathcal{A}}$, contains the same ciphertext $c_{\bar{i}}$ that was honestly generated and provided to $\mathcal{A}$ in the experiment. If not, it aborts.

Note that the party $P_{\bar{i}}$ is not corrupted throughout the execution, and $(\mathtt{sid}, c_{\bar{i}})$ has a valid signature under $vk_{\bar{i}}$. Due to the unforgeability of DS, the probability that $c_{\bar{i}}$ is different from the honestly generated ciphertext is negligible. Thus, $|\Pr[\mathtt{Win}_6] - \Pr[\mathtt{Win}_5]| \leq \mathsf{negl}(\lambda)$.

Following the above arguments, we can see that $\Pr[\mathtt{Win}_6] \geq \frac{\Pr[\mathtt{Win}_0]}{n} - \mathsf{negl}(\lambda)$. Therefore, if $\Pr[\mathtt{Win}_0]$ is non-negligible, then $\Pr[\mathtt{Win}_6]$ must be non-negligible.

On the other hand, assuming there is a PPT adversary $\mathcal{A}$ such that $\Pr[\mathtt{Win}_6] > \mathsf{negl}(\lambda)$, we can build a PPT adversary $\mathcal{B}$ against the adaptive semantic security of STHE, such that $\mathsf{Adv}^{\mathrm{aind}}_{\mathcal{B}}(\lambda) > \mathsf{negl}(\lambda)$. $\mathcal{B}$ works as follows.

- **Setup:** $\mathcal{B}$ uniformly samples $\bar{i} \leftarrow\$ [n]$, runs a copy of $\mathcal{A}$ and finishes the setup phase in the following way: for the setup w.r.t. STHE, $\mathcal{B}$ just forwards the messages it receives in the aIND experiment; for the setup w.r.t. the digital signature DS, and SNARK, $\mathcal{B}$ follows the specifications in Fig.7. For NIZK, it runs the simulation setup $\mathsf{NIZK.SimSetup}$ to generate the CRS and its trapdoor $\mathtt{tk}$. If $\mathcal{A}$ requests to corrupt the party $P_{\bar{i}}$, it returns $b' \leftarrow\$ \{0,1\}$ and aborts.
- **Prepare ciphertexts, proofs, and signatures:** For every $j \in [n] \setminus (\mathcal{C}_{\mathsf{Init}} \cup \{\bar{i}\})$, it generates the ciphertext c_j, the proof π_j, and the signature σ_j by following the specifications in Fig.7. Regarding the ciphertext $c_{\bar{i}}$, it uniformly sampels $m_0, m_1 \leftarrow\$ \mathsf{Msg_{te}}$, queries the oracle $\mathcal{O}_b$ with (m_0, m_1) in the experiment aIND, and assigns $c_{\bar{i}} \leftarrow c_b$ which is returned by $\mathcal{O}_b$. Then, it generates a simulated proof $\pi_{\bar{i}}$ using $\mathtt{tk}$, and signs it honestly. All $(c_i, \pi_i, \sigma_i)_{i \in [n] \setminus \mathcal{C}_{\mathsf{init}}}$ are then provided to $\mathcal{A}$.
- **Respond queries to $\mathcal{O}_{\widetilde{\mathsf{PartD}}}$:** $\mathcal{B}$ just forwards these requeries to $\mathcal{O}_{\mathsf{PartD}}$ in the experiement aIND.
- **Respond queries to $\mathcal{O}_{\widetilde{\mathsf{Sign}}}$:** For a query $(i, (tag, \mathtt{msg}))$, $\mathcal{B}$ responds it honestly as specified in Fig.7, using the signing key of P_i.
- **Respond queries to $\mathcal{O}_{\widetilde{\mathsf{Corr}}}$:** When $\mathcal{A}$ queries $\mathcal{O}_{\widetilde{\mathsf{Corr}}}$ with i, if $i = \bar{i}$, it returns $b' \leftarrow\$ \{0,1\}$ and aborts. Otherwise, it requries $\mathcal{O}_{\mathsf{Corr}}$ in the aIND experiment with i and obtains the decryption key dk_i along $\mathsf{State}_{\mathsf{PD},i}$. Then, it returns $(dk_i, sk_i, \gamma_i, \hat{\gamma}_i, \bar{\gamma}_i, \mathsf{State}_i)$ to $\mathcal{A}$, where sk_i is the signing key of P_i, $\gamma_i, \hat{\gamma}_i, \bar{\gamma}_i$ are the randomness for generating the ciphertext, the proof, and the signature; State_i contains the randomness used to respond each query to $\mathcal{O}_{\widetilde{\mathsf{Sign}}}$ and $\mathcal{O}_{\widetilde{\mathsf{PartD}}}$.
- **Output:** After $\mathcal{A}$ returns (c^*, π^*, r^*), it applies the extractor $E_{\mathcal{A}}$ to obtain the list $\{j, c_j, \sigma_j, \pi_j\}_{j \in \mathbb{I}'}$ as specified in Hybrid 3, applies the extractor SimExt to obtain the plaintext and randomness $(r_j, \gamma_j)_{j \in \mathbb{I}' \setminus \bar{i}}$ as specified in Hybrid 5. It will return $b' \leftarrow\$ \{0,1\}$ and aborts under the same conditions of Hybrid 6. If not abort, then it calculates $r'_{\bar{i}} \leftarrow r^* \ominus \bigoplus_{j \in \mathbb{I}' \setminus \{\bar{i}\}} r_j$. Then, if $r'_{\bar{i}} = m_0$, $\mathcal{B}$ returns 0; If $r'_{\bar{i}} = m_1$, $\mathcal{B}$ returns 1. In any other cases, it returns $b' \leftarrow\$ \{0,1\}$.

It is easy to verify that $\mathcal{B}$ perfectly simulates the experiment Hybrid 6 in $\mathcal{A}$'s view. For notational simplicity, we still use $\mathtt{Win}_6$ to denote the following event: $\mathcal{B}$ does not abort, and $\mathcal{A}$ returns (c^*, π^*, r^*) such that π^* is a valid proof for c^* w.r.t. SNARK and r^* is the decryption of c^*.

Now, we analyze $\mathsf{Adv}^{\mathrm{aind}}_{\mathcal{B}}(\lambda) = |\Pr[\mathsf{aIND}^{\mathcal{B}}_1] - \Pr[\mathsf{aIND}^{\mathcal{B}}_0]|$. Note that due to the *evaluation correctness* of STHE, when $\mathtt{Win}_6$ happens, there exists γ^* such that $c^* = \mathsf{STHE.Enc}(ek, tag^*, \bigoplus_{j \in \mathbb{I}'} r_j \oplus m_b)$. Then, due to the decryption correctness, we have $r^* = \bigoplus_{j \in \mathbb{I}'} r_j \oplus m_b$. It follows that $\Pr[\mathsf{aIND}^{\mathcal{B}}_1 | \mathtt{Win}_6] = 1$, while $\Pr[\mathsf{aIND}^{\mathcal{B}}_0 | \mathtt{Win}_6] = 0$. On the other hand, there is a special event $\mathtt{SE}$ that $\mathcal{A}$ returns (c^*, π^*, r^*) but $r^* = \bigoplus_{j \in \mathbb{I}'} r_j \oplus m_{1-b}$. It is easy to verify that $\Pr[\mathsf{aIND}^{\mathcal{B}}_1 | \mathtt{SE}] = 0$, while $\Pr[\mathsf{aIND}^{\mathcal{B}}_0 | \mathtt{SE}] = 1$. However, since m_{1-b} is uniformly sampled and indepdent of $\mathcal{A}$'s view, so $\Pr[\mathtt{SE}] = \frac{1}{|\mathsf{Msg_{te}}|}$. In addition, under the condition that $\neg(\mathtt{SE} \cup \mathtt{Win}_6)$, $\mathcal{B}$ simply returns a uniformly sampled bit, so $\Pr[\mathsf{aIND}^{\mathcal{B}}_1 | \neg(\mathtt{SE} \cup \mathtt{Win}_6)] = \Pr[\mathsf{aIND}^{\mathcal{B}}_0 | \neg(\mathtt{SE} \cup \mathtt{Win}_6)] = \frac{1}{2}$. Putting the above facts together, we have:

$$\begin{aligned} |\Pr[\mathsf{aIND}^{\mathcal{B}}_1] - \Pr[\mathsf{aIND}^{\mathcal{B}}_0]| =& |\Pr[\mathsf{aIND}^{\mathcal{B}}_1 | \mathtt{Win}_6] \Pr[\mathtt{Win}_6] + \Pr[\mathsf{aIND}^{\mathcal{B}}_1 | \mathtt{SE}] \Pr[\mathtt{SE}] \\ &- \Pr[\mathsf{aIND}^{\mathcal{B}}_0 | \mathtt{Win}_6] \Pr[\mathtt{Win}_6] - \Pr[\mathsf{aIND}^{\mathcal{B}}_0 | \mathtt{SE}] \Pr[\mathtt{SE}]| \\ =& |\Pr[\mathtt{Win}_6] - \Pr[\mathtt{SE}]| \end{aligned}$$

If $\Pr[\texttt{Win}_6]$ is non-negligible, then $\mathsf{Adv}^{\mathrm{aind}}_{\mathcal{B}}(\lambda)$ is also non-negligible. Under the assumption that STHE satisfies the adaptive semnatic security for (n, f), $\mathsf{Adv}^{\mathrm{aind}}_{\mathcal{B}}(\lambda)$ is negligible for any PPT $\mathcal{B}$, so $\Pr[\texttt{Win}_6]$ (and thus $\Pr[\texttt{Win}_0]$) must also be negligible. □

Security of our weak coin protocol. We establish the following theorem for our weak coin protocol in Algorithm 2. In particular, following the composition security lemmas in Sect.4.2, we prove an "enhanced" version of the security, where the adversary against the instance $\mathsf{WeakCoin}[\mathtt{sid}]$ is given additional access to the following oracles:

- $\mathcal{O}_{\widehat{\mathsf{PartD}}}$: On input $(\mathtt{sid}', i)$, if $\mathtt{sid}' \neq \mathtt{sid}$, returns $\mathsf{STHE.PartD}(ek, \mathtt{sid}', dk_i)$
- $\mathcal{O}_{\widehat{\mathsf{Sign}}}$: On input $(\mathtt{sid}', i, \mathtt{msg})$, if $\mathtt{sid}' \neq \mathtt{sid}$, returns $\mathsf{DS.Sign}(sk_i, (\mathtt{sid}', \mathtt{msg}))$.
- Oracles in $\mathbb{O}_{\mathsf{Align}}$, defined in Lemma.16 for the underlying Align.

We denote the set of the above oracles as well as the corruption oracle by $\mathbb{O}_{\mathsf{WeakCoin}}$.

Theorem 2. *The* $\mathsf{WeakCoin}$ *protocol in Algorithm.2 satisfies the termination and φ-fairness for $\varphi = \frac{n-f}{n}$, even when the adversary has additional access to the oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$.

We prove the termination in Lemma.7 and the φ-fairness in Lemma 8, respectively. We argue in Proposition 6 in Appendix.D that Align remains secure in the composition.

Lemma 7 (Termination of $\mathsf{WeakCoin}$). $\mathsf{WeakCoin}$ *satisfies the termination property, even when the adversary has additional access to oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$.

Proof. We prove the termination by showing every honest node can execute the code of Line 32-34 in Algorithm 2. For Lines 1-5, every honest node can perform the task without issues. For 6-12, due to the completeness of the underlying NIZK and the correctness of the underlying DS, every honestly generated proof π_j and signature σ_i can be verified, so that $(j, c_j, \sigma_j, \pi_j)$ carried by the FRESH message will be included in the set Fresh. Since all honest nodes have multicasted the FRESH message, the condition in Line 9 can be satisfied, and every honest node can execute Line 9-12.

Every honest node P_i will invoke the $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$ with (c^*_i, π^*_i). Due to the completeness of SNARK, π^*_i is a valid proof w.r.t. c^*_i and the public parameters, so (c^*_i, π^*_i) satisfies the predicate of $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$. Since all honest nodes provide valid inputs to $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$, every honest node P_i can terminate in $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$ with a set OSet_i (which can be empty). Therefore, Lines 13-14 and Lines 17-18 can be executed by all honest nodes. Following the correctness of STHE, all honest nodes can receive enough valid partial decryption keys, so Lines 19-21 can be executed by all honest nodes. In addition, every honest node will execute either Line 15-16 or Line 22-27, and due to the $f+1$-alignment property of Align, at least $f+1$ honest nodes execute Line 22-27. Ensured by the correctness of STHE and termination of Align, all honestly generated LOCALWIN messages will be included in the set $\mathsf{LocWins}$. When $|\mathsf{LocWins}| = n - f$, at least one of element in $\mathsf{LocWins}$ carries $(a, r^*_a, to_a) \neq \perp$. Thus, every honest node can execute Line 34 and return a value.

Note that giving $\mathcal{A}$ access to oracles in $\mathbb{O}_{\mathsf{WeakCoin}}$ does not violate any security properties of the underlying primitives that we need in the above analysis. □

Lemma 8 ($\frac{n-f}{n}$-Fairness). $\mathsf{WeakCoin}$ *satisfies the φ-Fairness property with $\varphi = \frac{n-f}{n}$, even when the adversary has additional access to oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$.

Proof. We first describe an event that happens with the probability φ, conditioned on which the agreement, the unpredictability, and the bias-resistance are granted. Due to the $(f+1)$-alignment property (binding cover property and well-covered set property), at the first time that an honest node P_i receives an output from $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$, there exists the well-covered set $\mathsf{WS} = \{(j,(c^*_j,\pi^*_j),\cdot)\}_{j\in\mathbb{I}_s}$ whose size is $n-f$, as well as the binding cover $\mathsf{CoverSet} = \{(j,(c^*_j,\pi^*_j),\cdot)\}_{j\in\mathbb{I}_c}$ whose size is at most n. Note that we have $\mathsf{WS} \subset \mathsf{CoverSet}$ (while well-formedness proofs are ignored). With this notion, we can define the following event:

- $\mathtt{Fair}$: Let $dk_{\mathtt{sid}}$ be the decryption key obtained by an honest node at Line 23. Let $r^*_j = \mathsf{STHE.Dec}(ek, \mathtt{sid}, dk_{\mathtt{sid}}, c^*_j)$, and let $\mathsf{to}_j = H_1(\mathtt{sid}, r^*_j)$, for $j \in \mathbb{I}_c$. In this event, there exists an index $j_{\max} \in \mathbb{I}_s$, such that $\mathsf{to}_{j_{\max}}$ is the largest value among all $\{\mathsf{to}_j\}_{j\in\mathbb{I}_c}$.

In Proposition 2, we show $\mathtt{Fair}$ happens with a probability of at least $\varphi - \mathsf{negl}(\lambda)$, despite any attacking strategy. Then, we prove that agreement and unpredictability in Proposition 3 and Proposition 4, respectively. These proofs rely on the fact that the random oracles H_1 and H_2 had not been queried with $(\mathtt{sid}, j, r^*_j)$ for any $j \in \mathbb{I}_c$ until the first time that an honest node finishes $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$, which we proved in Proposition 1. □

Proposition 1. *At the first time that an honest node P_i finishes the* $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$ *instance, the probability that* H_1 *or* H_2 *has been queried with* $(\mathtt{sid}, j, r^*_j)$ *for any* $j \in \mathbb{I}_c$ *is negligibly small.*

Proof. We prove this lemma by contradiction. Assume that there exists a PPT $\mathcal{A}$, such that in execution of $\mathsf{WeakCoin}[\mathtt{sid}]$, there is a non-negligible probability that H_1 or H_2 has been queried with the specified $(\mathtt{sid}, j, r^*_j)$ at the first time that an honest node P_i finishes the $\mathsf{Align}[\mathtt{sid}|\mathtt{ali}]$ instance. Then, we can build a PPT adversary $\mathcal{B}$ which can win the raUNP security game in Fig.7, i.e., $\mathsf{Exp}^{\mathcal{B}}_{\mathsf{raUNP}}(1^\lambda, n, f) = 1$, with a non-negligible probability.

We outline the strategy of $\mathcal{B}$ as follows, which runs a copy of $\mathcal{A}$ and tries to simulate the execution of our weak coin protocol until an honest node finishes the Align subroutine.

- *Setup Phase:* $\mathcal{B}$ forwards the messages it received in the raUNP security game to $\mathcal{A}$ and forwards the corruption requests and public keys of corrupted parties from $\mathcal{A}$ to the security game. In addition, the setup for Align is run by $\mathcal{B}$. Initialize two sets $\mathcal{H} \subset [n]$ and $\mathcal{C} \subset [n]$ to track the identities of honest parties and corrupted parties, respectively.
- *Protocol Simulation:* Note that in the raUNP security game, $\mathcal{B}$ receives $(c_i, \pi_i, \sigma_i)_{i\in\mathcal{C}_{\mathsf{init}}}$. Then, $\mathcal{B}$ acts on the behalf of every so-far-honest node P_i to multicast $(\mathtt{sid}, \textsc{Fresh}, c_i, \sigma_i, \pi_i)$. Next, $\mathcal{B}$ honestly follows the protocol specification to act on behalf of all honest nodes. Random oracle queries are responded to in a standard way, and $\mathcal{B}$ maintains two tables Hist_1 and Hist_2 of the query records.
- *Handle Oracle Queries:* Note that $\mathcal{A}$ can query the additional oracles in

$$\mathbb{O}_{\mathsf{weakcoin}} = \{\mathcal{O}_{\widehat{\mathsf{PartD}}}, \mathcal{O}_{\widehat{\mathsf{Sign}}}\} \cup \mathbb{O}_{\mathsf{Align}}.$$

  When $\mathcal{A}$ queries $\mathcal{O}_{\widehat{\mathsf{PartD}}}$ (resp. $\mathcal{O}_{\widehat{\mathsf{Sign}}}$) with $(\mathtt{sid}', i)$ (resp. $(\mathtt{sid}', i, \mathtt{msg})$) such that $i \in \mathcal{H}$ and $\mathtt{sid}' \neq \mathtt{sid}$, $\mathcal{B}$ queries $\mathcal{O}_{\widetilde{\mathsf{PartD}}}$ (resp. $\mathcal{O}_{\widetilde{\mathsf{Sign}}}$)in the raUNP security game with $(\mathtt{sid}', i)$ (resp. (resp. $(i, (\mathtt{sid}', \mathtt{msg}))$)) and forwards the message from $\mathcal{O}_{\widetilde{\mathsf{PartD}}}$ to $\mathcal{A}$. Regarding queries to oracles in $\mathbb{O}_{\mathsf{Align}}$, $\mathcal{B}$ answers them honestly by using the secret states of honest parties w.r.t. $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$.

- *Handle Corruption:* When $\mathcal{A}$ corrupts P_i, $\mathcal{B}$ checks if the number of corrupted parties has reached the limit. If not, $\mathcal{B}$ queries $\mathcal{O}_{\widetilde{\mathsf{Corr}}}$ with i, which returns $(dk_i, sk_i, r_i, \gamma_i, \hat{\gamma}_i, \bar{\gamma}_i, \mathsf{State_i})$. Then, it returns the tuple along with other secrets and randomness for SNARK and Align to $\mathcal{A}$.
- *Output:* At the first time that an honest node finishes Align, $\mathcal{B}$ obtains the binding cover $\mathsf{CoverSet} = \{(j, (c_j^*, \pi_j^*), \cdot)\}_{j \in \mathbb{I}_c}$ as well as the set of well-covered elements WS. As ensured by Align, (c_j^*, π_j^*) shall satisfy the predicate of Align, i.e.,

$$\mathsf{SNARK.Vrfy}(\mathtt{crs}_{\mathsf{agg}}, (n, f, (vk_i)_{i \in [n]}, ek, tag^*, \mathtt{crs}_{\mathsf{enc}}, c_j^*), \pi_j^*) = 1.$$

  Then, $\mathcal{B}$ extracts all records in Hist_1 and Hist_2 with the input form $(\mathtt{sid}, j, r_j^*)$ for $j \in \mathbb{I}_c$, and uniformly picks one record $(\mathtt{sid}, z, r_z^*)$ from them. It outputs (c_z^*, π_z^*, r_z^*) in the raUNP security game.

It is easy to verify, in $\mathcal{A}$'s view, that the execution simulated by $\mathcal{B}$ is identical to a real execution of $\mathsf{WeakCoin}[\mathtt{sid}]$. Therefore, there is a non-negligible probability that a tuple $(\mathtt{sid}, j, r_j^*)$, where $j \in \mathbb{I}_c$ and r_j^* is the decryption of c_j^*, has been issued to H_1 and H_2. Moreover, since the number of query records must be polynomial in λ, the probability that $\mathcal{B}$ happens to pick this tuple is at least $\frac{1}{\mathsf{poly}(\lambda)}$. There would be a non-negligible probability that $\mathcal{B}$ wins in the raUNP security game, which contradicts Lemma 6. □

Proposition 2. *The event* $\mathtt{Fair}$ *(defined in Lemma 8) happens with a probability of at least* $\varphi - \mathsf{negl}(\lambda)$.

Proof. For ease of discussion, we consider the following event:

- $\mathtt{EarlyQ}$: H_1 or H_2 has been queried with $(\mathtt{sid}, j, r_j^*)$ for any $j \in \mathbb{I}_c$ before the first time an honest node finishes $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali} \rangle]$.

As shown in Lemma 1, we know $\Pr[\mathtt{EarlyQ}] \leq \mathsf{negl}(\lambda)$. Note that at the first time an honest node finishes $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali} \rangle]$, $\mathsf{WS} = \{(j, c_j^*, \cdot)\}_{j \in \mathbb{I}_s}$ and $\mathsf{CoverSet} = \{j, c_j^*, \cdot\}_{j \in \mathbb{I}_c}$ have been decided, so whether a tuple $(j, c_j^*, \cdot)$ is included in WS and $\mathsf{CoverSet}$ is independent of the values $\{\mathsf{to}_j = H_1(\mathtt{sid}, j, r_j^*)\}$. Therefore, for each $j \in \mathbb{I}_c$, $\Pr[\mathsf{to}_j = \mathsf{Max}(\{\mathsf{to}_i\}_{i \in \mathbb{I}_c}) | \neg\mathtt{EarlyQ}] = \frac{1}{|\mathbb{I}_c|}$, and $\Pr[\mathsf{Max}(\{\mathsf{to}_i\}_{i \in \mathbb{I}_c}) \in \{\mathsf{to}_i\}_{i \in \mathbb{I}_s} | \neg\mathtt{EarlyQ}] = \frac{n-f}{|\mathbb{I}_c|}$, where the probability is taken over the choice of the random function inside the random oracle H_1. I.e., $\Pr[\mathtt{Fair} | \neg\mathtt{EarlyQ}] = \frac{n-f}{|\mathbb{I}_c|}$. Therefore,

$$\begin{aligned} \Pr[\mathtt{Fair}] &\geq \Pr[\mathtt{Fair} | \neg\mathtt{EarlyQ}] \Pr[\neg\mathtt{EarlyQ}] \\ &\geq \frac{n-f}{|\mathbb{I}_c|} \cdot (1 - \mathsf{negl}(\lambda)) \geq \frac{n-f}{n} - \mathsf{negl}(\lambda) \end{aligned}$$

where the probability is taken over the choice of the random function inside the random oracle H_1, the randomness of the adversary $\mathcal{A}$, and the randomness of all honest nodes. □

Proposition 3. *Under the condition that the event* $\mathtt{Fair}$ *(defined in Lemma8) happens, all honest nodes output the same value.*

Proof. Let $i_{\mathsf{max}} \in \mathbb{I}_s$ be the index such that $\mathsf{to}_{i_{\mathsf{max}}}$ is the largest value among all $\{\mathsf{to}_j\}_{j \in \mathbb{I}_c}$. Ensured by the *well-covered set* property of Align, the tuple $(i_{\mathsf{max}}, (c_{i_{\mathsf{max}}}^*, \pi_{i_{\mathsf{max}}}^*), \theta_{i_{\mathsf{max}}})$ is in the output sets of at least $f+1$ forever-honest parties, and all of these $f+1$ honest parties will multicast $(\texttt{sid}, \textsc{LocalWin}, (i_{\mathsf{max}}, c^*_{i_{\mathsf{max}}}, \pi^*_{i_{\mathsf{max}}}, \theta_{i_{\mathsf{max}}}))$. Hence, after receiving $n-f$ LOCAL-WIN messages, all honest nodes can receive a message carrying $(\texttt{sid}, \textsc{LocalWin}, (i_{\mathsf{max}}, c^*_{i_{\mathsf{max}}}, \pi^*_{i_{\mathsf{max}}}, \theta_{i_{\mathsf{max}}}))$. Meanwhile, due to the *binding cover* property of Align, all tuples passing the $\mathsf{Align.Vrfy}$ check must have been included in $\mathsf{CoverSet} = \{(j, c^*_j, \pi^*_j)\}_{j\in\mathbb{I}_c}$. Thus, all honest nodes must decide on $\mathsf{to}_{i_{\mathsf{max}}}$ is the largest value among all $\{H_1(\texttt{sid}, j, r^*_j)\}_{j\in\mathbb{I}_c}$, which ensures the agreement property v. ☐

Proposition 4. *Under the condition that the event* $\texttt{Fair}$ *happens, the output v of an honest node is unpredictable to $\mathcal{A}$. Namely, let* $\mathsf{State}^{before}_{\mathcal{A}}$ *be the internal state of $\mathcal{A}$ before $n-f-|\mathcal{C}|$ honest nodes participating* $\mathsf{WeakCoin}[sid]$*, then, for any PPT algorithm $\mathcal{D}$, and $u \leftarrow\$ V$,*

$$|\Pr[\mathcal{D}(\mathsf{State}^{before}_{\mathcal{A}}, v) = 1] - \Pr[\mathcal{D}(\mathsf{State}^{before}_{\mathcal{A}}, u) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Let $\mathsf{State}'_{\mathcal{A}}$ be the state of $\mathcal{A}$ at the first time that an honest node finishes $\mathsf{Align}[\langle\texttt{sid}, \texttt{ali}\rangle]$. Since when an honest node finishes $\mathsf{Align}[\langle\texttt{sid}, \texttt{ali}\rangle]$, $n-f-|\mathcal{C}|$ nodes must have already participated in the instance, we can assume, without loss of generality, that $\mathsf{State}^{\mathsf{before}}_{\mathcal{A}}$ has been encoded in $\mathsf{State}'_{\mathcal{A}}$. Therefore, it would suffice to show $|\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1] - \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, u) = 1]| \leq \mathsf{negl}(\lambda)$.

Recall the proof of Lemma 2 where we defined an event $\texttt{EarlyQ}$, in which H_1 or H_2 has been queried with $(\texttt{sid}, j, r^*_j)$ for any $j \in \mathbb{I}_c$ before the first time an honest node finishes $\mathsf{Align}[\langle\texttt{sid}, \texttt{ali}\rangle]$. Under the condition $\neg\texttt{EarlyQ}$, the index i_{max} is determined independent of H_2. Therefore, $\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, H_2(\texttt{sid}, i_{\mathsf{max}}, r^*_{i_{\mathsf{max}}})) = 1|\texttt{EarlyQ}] = \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, u) = 1]$. Under the condition that $\texttt{Fair}$ happens, as we analyzed in Lemma 3, every honest node outputs $v = H_2(\texttt{sid}, i_{\mathsf{max}}, r^*_{i_{\mathsf{max}}})$. Hence, $\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1|\neg\texttt{EarlyQ}] = \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, u) = 1]$. Note that

$$\begin{aligned}\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1] =& \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1|\neg\texttt{EarlyQ}]\Pr[\neg\texttt{EarlyQ}] \\ &+ \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1|\texttt{EarlyQ}]\Pr[\texttt{EarlyQ}]\end{aligned}$$

According to Lemma 1, $\Pr[\neg\texttt{EarlyQ}] \geq 1 - \mathsf{negl}(\lambda)$. Therefore,

$$\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1] = (1 - \mathsf{negl}(\lambda))\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, u) = 1] + \mathsf{negl}(\lambda).$$

It follows that $|\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1] - \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, u) = 1]| \leq \mathsf{negl}(\lambda)$. ☐

7 Optimal Asynchronous Strong Common Coin with Silent Setup

Following the discussion in Sect.2, we "compile" a quadratic-communication weak common coin protocol into a quadratic-communication strong common coin protocol via two steps. First, as in Sect.7.1, we present a leader election protocol based on our weak common coin protocol, which is sufficient for instantiating the existing quadratic-communication MVBA protocols that assume a leader election subroutine. Next, as in Sect.7.2, a strong common coin can be easily built on the silent-setup quadratic-communication MVBA.

7.1 Optimal Leader Election with Silent Setup

Leader Election Definition. We present a definition for leader election as follows, which is adapted from [42]. In particular, compared with a weak common coin protocol with the output space of $[n]$, a leader election can ensure the *agreement* with $1 - \mathsf{negl}(\lambda)$ probability.

Definition 3. *Let Π be a protocol involving n participants in which each P_i does not have input and will be instructed to output an index ι in $[n]$. We say Π is an (n, f, ϕ) leader election for $\phi \in (0, 1]$, if, in any instance of Π, and for any PPT adversary $\mathcal{A}$ which can corrupt up to f parties, the following properties are ensured.*

- ***Termination**: When all honest nodes are activated in this instance, except with a negligible probability, every honest node P_i eventually outputs a value $\iota \in [n]$.*
- ***Agreement**: If two honest parties outputs ι and ι', respectively, then $\iota = \iota'$.*
- *ϕ-**Unpredictability:** Let $\mathsf{State}_{\mathcal{A}}^{before}$ be the internal state of $\mathcal{A}$ before $(n - f - |\mathcal{C}|)$ honest parties are activated in this instance. Let ι be the output of an honest node. Then, for any PPT algorithm $\mathcal{D}$, it follows that*

$$\Pr[\mathit{BadSet} \leftarrow \mathcal{D}(\mathsf{State}_{\mathcal{A}}^{before}) : \iota \notin \mathit{BadSet} \subset [n]] \geq \phi(\frac{n - |\mathit{BadSet}|}{n}) - \mathsf{negl}(\lambda).$$

It can capture the essence that there is always a constant probability of electing a "good" leader.

Multivalued Byzantine Agreement with Silent Setup. As an ingredient to our leader election protocol, we study a full-fledged instantiation of multivalued Byzantine Agreement (MBA) (the definition recalled in Sect.3.1).

MBA has been studied extensively in the (weak)-coin-aided model. In particular, Mostefaoui and Raynal [70] presented a communication-optimal MBA protocol in the (weak)-coin-aided model, where ℓ is the bit length of the input to MBA. We summarize their results in the following lemma.

Lemma 9 (Coin-aided MBA, [70]). *Assume there is a binary weak coin protocol with termination and ϕ-fairness for any constant $\phi \in (0, \frac{1}{2}]$, as per [35] (cf. Appendix.B.1). Then, there is a secure MBA protocol using the binary weak coin as subroutines, which, except the weak coin subroutines, is deterministic and information-theoretically secure and has the communication complexity of $O(\ell n^2)$ and the round complexity of $O(1)$ in expectation, where ℓ is the bit length of an input.*

In Theorem 2, we demonstrate a weak coin protocol for a general output space V with termination and $\frac{n-f}{n}$-fairness, as per Def.1. As we have shown in Lemma 14, our results imply a binary weak coin with termination and $\frac{n-f}{2n}$-fairness as per Feldman and Micali [35]. Therefore, a straightforward approach to a communication-optimal MBA with a silent setup is to use multiple instances of our binary weak coin protocol under the same setup to realize their weak coin subroutine. We denote such a protocol by MBA, and its setup is $\mathsf{WeakCoin.Setup}$. Since our weak coin protocol is not guaranteed to be universally composable, we demonstrate the security of MBA in Lemma 19 in Appendix.D.

The Construction. Now, we are ready to present our leader election protocol with a silent setup. Its setup phase is exactly $\mathsf{WeakCoin.Setup}$. After the setup, the following tasks are iterated until honest nodes output: First, a weak coin instance with the output space of $[n]$ is invoked. Next, after receiving ρ_i from the weak coin instance, an honest node P_i passes ρ_i to an MBA instance. Then, if the MBA instance returns $\rho \neq \perp$ to P_i, then P_i returns ρ as the output of the leader election. Otherwise, P_i will join the next iteration. Here, the MBA instances are from Lemma 19, which use binary weak coin instances under the setup of $\mathsf{WeakCoin.Setup}$. We describe the pseudocode in Algorithm 3, while the execution flow is outlined in a part of Fig.4.

Algorithm 3 The asynchronous leader election protocol Leader with identifier $\mathtt{sid}$ for P_i

```
1: for $k = 1$ to $\infty$ do
2:   Invoke $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$
3:   wait until receiving $\rho_i$ from $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$
4:     Invoke $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{ba}, k\rangle]$ with input $\rho_i$.
5:   wait unitil receiving $\rho'$ from $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{ba}, k\rangle]$
6:     if $\rho' \neq \perp$ then
7:       return $\rho'$
```

Theorem 3 (Leader Election). *Under the setup* $\mathsf{WeakCoin.Setup}$*, the leader election protocol* $\mathsf{Election}$ *in Algorithm 3 satisfies the termination and* φ*-unpredictability for* $\varphi = \frac{n-f}{n}$*. Its communication complexity and round complexity are* $\mathcal{O}(\lambda n^2)$ *and* $\mathcal{O}(1)$ *in expectation, respectively.*

Proof. We argued why the components, $\mathsf{WeakCoin}$ and MBA, remain secure in the composition in Proposition 7 in Appendix.D. Based on this fact, we prove the properties as follows.
Termination: First, following the termination property of $\mathsf{WeakCoin}$ and MBA, every honest node can advance to the next iteration if it has not returned a value. Then, if the $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$ returns the same value ρ to all honest nodes, following the weak validity of MBA, $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{ba}, k\rangle]$ must returns ρ to all honest nodes, so the protocol can terminate. In addition, ensured by the φ-fairness of MBA, each $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$ can return the same value to all honest nodes with an independent probability of at least $\varphi - \mathsf{negl}(\lambda)$. So, the probability that $\mathsf{Election}[\mathtt{sid}]$ does not terminate until k-th iteration is $(1-(\varphi-\mathsf{negl}(\lambda)))^{k-1}\cdot(\varphi-\mathsf{negl}(\lambda))$, which decreses exponentially in k. Therefore, $\mathsf{Election}[\mathtt{sid}]$ can evenutally terminate. In addition, it is expected to terminate in

$$k^* = \sum_{k=1}^{\infty} k \cdot (\varphi - \mathsf{negl}(\lambda)) \cdot (1-(\varphi-\mathsf{negl}(\lambda)))^{k-1} \approx 2$$

Since each iteration takes $O(1)$ rounds and $O(\lambda n^2)$-bit communication cost in expectation, the overall communication and round complexity are $\mathcal{O}(\lambda n^2)$ and $\mathcal{O}(1)$ in expectation, respectively
Agreement: It is naturally ensured by the agreement of MBA.
φ-unpredictability: Note that there is a fair event, denoted by $\mathtt{Fair}_k$ w.r.t. $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$, such that $\Pr[\mathtt{Fair}_k] \geq \varphi - \mathsf{negl}(\lambda)$, and under which all honest nodes output the same ρ, and any PPT algorithm, given the state $\mathsf{State}_{\mathcal{A}}^{\mathtt{before}}$ of $\mathcal{A}$ before $n - f - |\mathcal{C}|$ honest nodes are activated in $\mathsf{Election}[\mathtt{sid}]$, cannot distinguish ρ and uniformly sampled ρ' from $[n]$. Therefore,

$$\begin{aligned}
&\Pr[\mathcal{D}(\mathsf{State}_{\mathcal{A}}^{\mathtt{before}}) \to \mathsf{BadSet} : \rho \notin \mathsf{BadSet}] \\
\geq &\Pr[\mathcal{D}(\mathsf{State}_{\mathcal{A}}^{\mathtt{before}}) \to \mathsf{BadSet} : \rho \notin \mathsf{BadSet} | \mathtt{Fair}_k] \Pr[\mathtt{Fair}_k] \\
\geq &(\frac{n - \mathsf{BadSet}}{n} - \mathsf{negl}(\lambda))(\varphi - \mathsf{negl}(\lambda)) = \frac{n - \mathsf{BadSet}}{n} \cdot \varphi - \mathsf{negl}(\lambda).
\end{aligned}$$

Achieving Multivalued Byzantine Validated Agreement with Silent Setup. As an ingredient to our strong common coin protocol, we study a full-fledged instantiation of multivalued Byzantine Agreement (MVBA) with silent setup (the definition recalled in Sect.3.1).

There are a few MVBA protocols [4,52,65] with the communication complexity of $\mathcal{O}(\lambda n^2)$ (particularly here the input size is $O(\lambda)$) and the round complexity of $O(1)$. In particular, [4,52] present

MVBA protocols that use provable broadcast (PB) and leader election as subroutines. Naturally, we can use PB in Algorithm 5 and $\mathsf{Election}$ in Algorithm 3 to instantiate these components, which yields a communication-optimal MVBA protocol with a silent setup. In particular, the setup is $\mathsf{WeakCoin.Setup}$, which subsumes the setups needed for both PB and $\mathsf{Election}$.

Since both PB and $\mathsf{Election}$ are not guaranteed to be universally composable, we need to check the security of the composed protocol with more care. In Lemma 20, we proved the composition guarantees of this MVBA scheme.

7.2 Achieving Strong Common Coin

Now, we are ready to present our asynchronous strong common coin. At a high level, we use a silent-setup MVBA to help all honest nodes agree on an aggregated ciphertext c^* with a valid proof π^* so that they can decrypt this ciphertext and use (the random oracle output of) the decryption as the common coin value.

The Construction. Our strong common coin protocol requires the following building blocks: An STHE scheme, a digital signature scheme DS, a NIZK system NIZK for the relation $\mathcal{R}_{\mathsf{enc}}$ (cf. Eq.2), and SNARK for the relation $\mathcal{R}_{\mathsf{agg}}$ (cf. Eq.3). Basically, they are the same schemes that we need for our weak coin protocol $\mathsf{WeakCoin}$, and so they can function under the same setup $\mathsf{WeakCoin.Setup}$.

In addition, we need the MVBA protocol outlined in Lemma 20, whose setup is also $\mathsf{WeakCoin.Setup}$. The external predicate $\mathsf{Predicate}$ of MVBA is parameterized by $(\mathtt{crs}_{\mathsf{agg}}, n, f, (vk_i)_{i\in[n]}, ek, \mathtt{sid}, \mathtt{crs}_{\mathsf{enc}})$ where $\mathtt{sid}$ is the identifier of the coin instance, and $\mathsf{Predicate}(c^*, \pi^*) = 1$, iff

$$\mathsf{SNARK.Vrfy}(\mathtt{crs}_{\mathsf{agg}}, (n, f, (vk_i)_{i\in[n]}, ek, \mathtt{sid}, \mathtt{crs}_{\mathsf{enc}}, c^*), \pi^*) = 1, \tag{5}$$

which is actually the same predicate used in the Align inside $\mathsf{WeakCoin}$ except with different identifiers.

With these building blocks, we present our strong common coin protocol in Algorithm 4. The protocol assumes a setup of $\mathsf{WeakCoin.Setup}$, and works as follows. First, a few steps are similar to that in our weak common protocol. After each node P_i has generated (c_i^*, π_i^*), all nodes jointly run an MVBA instance to agree on certain (c^*, π^*) which satisfies $\mathsf{Predicate}$. After that, all honest nodes can jointly decrypt c^*, and decide on $H(\mathtt{sid}, \mathsf{STHE.Dec}(ek, dk_{\mathtt{sid}}, c^*))$.

Theorem 4. *The protocol* Coin *in Algorithm 4 satisfies the termination, 1-fairness as per Def.1, and it has the communication complexity of $\mathcal{O}(\lambda n^2)$ and the round complexity of $\mathcal{O}(1)$ in expectation, respectively.*

It is easy to check the communication complexity and the round complexity. We prove the termination in Lemma 10. Regarding the 1-fairness, we prove the agreement in Lemma 11 and the unpredictability in Lemma 12, respectively.

Lemma 10 (Termination of coin). *The protocol* Coin *in Algorithm 4 satisfies the termination.*

Proof. We prove the termination by showing every honest node can execute the code of Line 19-22 in Algorithm 2. For Lines 1-5, every honest node can perform the task without issues. For 6-12, due to the completeness of the underlying NIZK and the correctness of the underlying DS, every honestly generated proof π_j and signature σ_i can be verified, so that $(j, c_j, \sigma_j, \pi_j)$ carried by the

Algorithm 4 The asynchronous strong common coin Coin protocol with identifier $\mathtt{sid}$ for P_i

```
1: upon activated with sid do
2:   Initialize: $\mathsf{Fresh}, \mathsf{PartK} \leftarrow \emptyset$
3:   $r_i \leftarrow\$ \mathsf{Msg}_{\mathsf{te}}, \gamma_i \leftarrow\$ \mathsf{Rand}_{\mathsf{te}}, c_i \leftarrow \mathsf{STHE.Enc}(ek, \mathtt{sid}, r_i; \gamma_i)$
4:   $\sigma_i \leftarrow \mathsf{DS.Sign}(sk_i, c_i)$ $\pi_i \leftarrow \mathsf{NIZK.P}(\mathsf{crs}_{\mathsf{enc}}, (ek, \mathtt{sid}, c_i), (r_i, \gamma_i))$
5:   Multicast $(\mathtt{sid}, \textsc{Fresh}, c_i, \sigma_i, \pi_i)$
6: upon receiving $(\mathtt{sid}, \textsc{Fresh}, c_j, \sigma_j, \pi_j)$ from node $P_j$ for the first time do
7:   if $\mathsf{DS.Vrfy}(vk_j, c_j, \sigma_1) = 1 \wedge \mathsf{NIZK.Vrfy}(\mathsf{crs}_{\mathsf{enc}}, (ek, \mathtt{sid}, c_j), \pi_j) = 1$ then
8:     $\mathsf{Fresh} \leftarrow \mathsf{Fresh} \cup \{(j, c_j, \sigma_j, \pi_j)\}$
9:   if $|\mathsf{Fresh}| = f + 1$ and has not invoked $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$ then
10:    $c_i^* \leftarrow \mathsf{STHE.Eval}(ek, \mathtt{sid}, \{c_j\}_{(j,\dots)\in \mathsf{Fresh}})$        ▷ Aggregate then prove
11:    $\pi_i^* \leftarrow \mathsf{SNARK.P}(\mathsf{crs}_{\mathsf{agg}}, (n, f+1, (vk_z)_{z\in[n]}, ek, \mathtt{sid}, \mathsf{crs}_{\mathsf{enc}}, c_i^*), \mathsf{Fresh})$
12:    Invoke $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$ with input $(c_i^*, \pi_i^*)$
13: upon $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$ returns $(c^*, \pi^*)$ do
14:   $\zeta_i \leftarrow \mathsf{STHE.PartDec}(ek, \mathtt{sid}, dk_i)$
15:   Multicast $(\mathtt{sid}, \mathsf{PartDec}, \zeta_i)$
16: upon receiving $(\mathtt{sid}, \textsc{PartDec}, \zeta_j)$ from $P_j$ for the first time do
17:   if $\mathsf{STHE.PartVrfy}(ek, \mathtt{sid}, pk_j, \zeta_j) = 1$ then
18:     $\mathsf{ParK} \leftarrow \mathsf{ParK} \cup \{(j, \zeta_j)\}$
19:   if $|\mathsf{ParK}| = n - f$ then
20:     $dk_{\mathtt{sid}} \leftarrow \mathsf{STHE.DkAgg}(ek, \mathtt{sid}, \{\zeta_k\}_{(k,)\in \mathsf{ParK}})$        ▷ decryption key
21:     $r^* \leftarrow \mathsf{STHE.Dec}(ek, \mathtt{sid}, dk_{\mathtt{sid}}, c^*)$, and $v^* \leftarrow H(\mathtt{sid}, r^*)$.
22:     return $v^*$
```

$\textsc{Fresh}$ message will be included in the set Fresh. Since all honest nodes have multicasted the $\textsc{Fresh}$ message, the condition in Line 9 can be satisfied, and every honest node can execute Line 9-12.

Due to the completeness of SNARK, π_i^* is a valid proof w.r.t. c_i^* and the public parameters, so (c_i^*, π_i^*) satisfies the predicate of $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$. Due to the termination of MVBA, every honest node can output (c^*, π^*), so the code of Line 13-15 can be executed by all honest nodes. Due to the partial verification correctness of STHE, it follows that all honest nodes can receive enough valid partial decryptions, so the code of Lines 16-22 can be executed by all honest nodes. $\square$

Lemma 11 (Agreement of coin). *The protocol* Coin *in Algorithm 4 satisfies the agreement with the probability* $1 - \mathsf{negl}(\lambda)$.

Proof. Ensured by the agreement of MVBA, all honest nodes receive the same (c^*, π^*). Due to the external validity of MVBA, (c^*, π^*) satisfies the predicate $\mathsf{Predicate}$ of MVBA defined in Eq.5, *i.e.*, π^* is a valid proof showing the well-formedness of c^* w.r.t. SNARK. The knowledge soundness of SNARK implies that c^* is an aggregation of $f + 1$ ciphertexts $\{c_j\}_{j\in S}$ for some $S \subset [n]$ and $|S| = f + 1$, and every c_j has a well-formedness proof π_j w.r.t. NIZK for $j \in S$. Due to the simulation extractability of NIZK, there are $\gamma_j \in \mathsf{Rand}_{\mathsf{te}}$ and $r_j \in \mathsf{Msg}_{\mathsf{te}}$, such that $c_j = \mathsf{STHE.Enc}(ek, \mathtt{sid}, r_j; \gamma_j)$. Then, the evaluation correctness of STHE ensures there exists r^* and γ^* such that $c^* = \mathsf{STHE.Enc}(ek, \mathtt{sid}, r^*; \gamma^*)$. Then, following the decryption correctness of STHE, all honest nodes shall decrypt c^* to r^*, and then they output the same $H(\mathtt{sid}, r^*)$. $\square$

Lemma 12 (Unpredictability of coin). *The protocol* Coin *in Algorithm 4 satisfies the unpredictability and bias-resistance with the probability* $1 - \mathsf{negl}(\lambda)$.

Proof. Let $\mathsf{State}'_{\mathcal{A}}$ be the state of $\mathcal{A}$ at the first time that an honest P_i outputs in $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba}\rangle]$, where there must have been $n-f-|\mathcal{C}|$ honest nodes participating the instance of $\mathsf{Coin}[\mathtt{sid}]$. Without loss of generality, we assume $\mathsf{State}^{\mathtt{before}}_{\mathcal{A}}$ has been encoded in $\mathsf{State}'_{\mathcal{A}}$. Then, it suffices to show that

$$|\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, H(\mathtt{sid}, r^*)) = 1] - \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1]| \le \mathsf{negl}(\lambda),$$

for any PPT algorithm $\mathcal{D}$, and $v \leftarrow\$ V$.

To show this, we prove that the random oracle H has not been queried with $(\mathtt{sid}, r^*)$ (defined in the proof of Lemma 11) at the first time that an honest node outputs in $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba}\rangle]$ in Lemma 13, except with a negligible probability. With this fact, $\mathsf{State}'_{\mathcal{A}}$ is independent of $H(\mathtt{sid}, r^*)$. Then, it follows trivially to see $|\Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, H(\mathtt{sid}, r^*)) = 1] - \Pr[\mathcal{D}(\mathsf{State}'_{\mathcal{A}}, v) = 1]| \le \mathsf{negl}(\lambda)$. □

Lemma 13. *At the first time that an honest node P_i outputs (c^*, π^*) in the $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba}\rangle]$ instance, the probability that H has been queried with $(\mathtt{sid}, r*)$ is negligibly small, where r^* is the decryption of c^*.*

Proof. This proof is similar to the proof for Lemma 1. In particular, we prove this lemma by contradiction. Assume that there exists a PPT $\mathcal{A}$, such that in the execution of $\mathsf{Coin}[\mathtt{sid}]$, there is a non-negligible probability that H has been queried with the specified $(\mathtt{sid}, r^*)$ at the first time that an honest node P_i outputs (c^*, π^*) $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba}\rangle]$ instance. Then, we can build a PPT adversary $\mathcal{B}$ which can win the raUNP security game in Fig.7, i.e., $\mathsf{Exp}^{\mathcal{B}}_{\mathsf{raUNP}}(1^\lambda, n, f) = 1$, with a non-negligible probability.

We outline the strategy of $\mathcal{B}$ as follows, which runs a copy of $\mathcal{A}$ and tries to simulate the execution of the coin until an honest node outputs in the MVBA subroutine.

- *Setup Phase:* Note that the setup of Coin is exactly the setup of our $\mathsf{WeakCoin}$. Therefore, $\mathcal{B}$ can finish the setup in the following way: $\mathcal{B}$ forwards the messages it received in the raUNP security game to $\mathcal{A}$, and forwards the corruption requests and public keys of corrupted parties from $\mathcal{A}$ to the security game. In addition, the setup for Align is run by $\mathcal{B}$. Initialize two sets $\mathcal{H} \subset [n]$ and $\mathcal{C} \subset [n]$ to track the identities of honest parties and corrupted parties, respectively.
- *Protocol Simulation:* Note that in the raUNP security game, $\mathcal{B}$ receives $(c_i, \pi_i, \sigma_i)_{i \in \mathcal{C}_{\mathsf{init}}}$. Then, $\mathcal{B}$ acts on the behalf of every so-far-honest node P_i to multicast $(\mathtt{sid}, \textsc{Fresh}, c_i, \sigma_i, \pi_i)$. Next, $\mathcal{B}$ honestly follows the protocol specification to act on behalf of all honest nodes. Random oracle queries are responded to in a standard way, and $\mathcal{B}$ maintains two tables Hist of the query records.
- *Handle Oracle Queries:* Note that $\mathcal{A}$ can query the additional oracles in $\mathbb{O}_{\mathsf{weakcoin}} = \{\mathcal{O}_{\widehat{\mathsf{PartD}}}, \mathcal{O}_{\widehat{\mathsf{Sign}}}\} \cup \mathbb{O}_{\mathsf{Align}}$. When $\mathcal{A}$ queries $\mathcal{O}_{\widehat{\mathsf{PartD}}}$ (resp. $\mathcal{O}_{\widetilde{\mathsf{Sign}}}$) with $(\mathtt{sid}', i)$ (resp. $(\mathtt{sid}', i, \mathtt{msg})$) such that $i \in \mathcal{H}$ and $\mathtt{sid}' \neq \mathtt{sid}$, $\mathcal{B}$ queries $\mathcal{O}_{\widetilde{\mathsf{PartD}}}$ (resp. $\mathcal{O}_{\widetilde{\mathsf{Sign}}}$)in the raUNP security game with $(\mathtt{sid}', i)$ (resp. (resp. $(i, (\mathtt{sid}', \mathtt{msg}))$)) and forwards the message from $\mathcal{O}_{\widetilde{\mathsf{PartD}}}$ to $\mathcal{A}$. Regarding queries to oracles in $\mathbb{O}_{\mathsf{Align}}$, $\mathcal{B}$ answers them honestly by using the secret states of honest parties w.r.t. $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$.
- *Handle Corruption:* When $\mathcal{A}$ corrupts P_i, $\mathcal{B}$ checks if the number of corrupted parties has reached the limit. If not, $\mathcal{B}$ queries $\mathcal{O}_{\widetilde{\mathsf{Corr}}}$ with i, which returns $(dk_i, sk_i, r_i, \gamma_i, \hat{\gamma}_i, \bar{\gamma}_i, \mathsf{State_i})$. Then, it returns the tuple along with other secrets and randomness for SNARK and Align to $\mathcal{A}$.
- *Output:* At the first time that an honest node outputs (c^*, π^*) in MVBA, $\mathcal{B}$ obtains (c^*, π^*). As ensured by the external validity, $\mathsf{SNARK.Vrfy}(\mathtt{crs}_{\mathsf{agg}}, (n, f, (vk_i)_{i \in [n]}, ek, tag^*, \mathtt{crs}_{\mathsf{enc}}, c^*), \pi^*) =$

1, as (c^*, π^*) satisfy the predicate. Then, $\mathcal{B}$ extracts all records in Hist with the input form $(\mathtt{sid}, r')$, and uniformly picks one record $(\mathtt{sid}, r^*)$ from them. It outputs (c^*, π^*, r^*) in the raUNP security game.

It is easy to verify, in $\mathcal{A}$'s view, that the execution simulated by $\mathcal{B}$ is identical to a real execution of $\mathsf{Coin}[\mathtt{sid}]$. Therefore, there is a non-negligible probability that a tuple $(\mathtt{sid}, r^*)$, where r^* is the decryption of c^*, has been issued to H. Moreover, since the number of query records must be polynomial in λ, the probability that $\mathcal{B}$ happens to pick this tuple is at least $\frac{1}{\mathsf{poly}(\lambda)}$. There would be a non-negligible probability that $\mathcal{B}$ wins in the raUNP security game, which contradicts Lemma 6. $\square$

References

1. Abraham, I., Chan, T.H., Dolev, D., Nayak, K., Pass, R., Ren, L., Shi, E.: Communication complexity of byzantine agreement, revisited. Distributed Comput. **36**(1), 3–28 (2023)
2. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G.: Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 14081, pp. 39–70. Springer (2023)
3. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Reaching consensus for asynchronous distributed key generation. In: PODC. pp. 363–373. ACM (2021)
4. Abraham, I., Malkhi, D., Spiegelman, A.: Asymptotically optimal validated asynchronous byzantine agreement. In: PODC. pp. 337–346. ACM (2019)
5. Attema, T., Cramer, R., Rambaud, M.: Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In: ASIACRYPT (4). Lecture Notes in Computer Science, vol. 13093, pp. 526–556. Springer (2021)
6. Avitabile, G., Döttling, N., Magri, B., Sakkas, C., Wohnig, S.: Signature-based witness encryption with compact ciphertext. IACR Cryptol. ePrint Arch. p. 1477 (2024)
7. Bacho, R., Lenzen, C., Loss, J., Ochsenreither, S., Papachristoudis, D.: Grandline: Adaptively secure DKG and randomness beacon with (almost) quadratic communication complexity. In: CCS. ACM (2024)
8. Bacho, R., Loss, J.: On the adaptive security of the threshold BLS signature scheme. In: CCS. pp. 193–207. ACM (2022)
9. Bacho, R., Loss, J.: Adaptively secure (aggregatable) pvss and application to distributed randomness beacons. In: CCS. ACM (2023)
10. Bandarupalli, A., Bhat, A., Bagchi, S., Kate, A., Reiter, M.K.: Random beacons in monte carlo: Efficient asynchronous random beacon *without* threshold cryptography. In: CCS. pp. 2621–2635. ACM (2024)
11. Bangalore, L., Choudhury, A., Patra, A.: Almost-surely terminating asynchronous byzantine agreement revisited. In: PODC. pp. 295–304. ACM (2018)
12. Bauer, B., Fuchsbauer, G., Plouviez, A.: The one-more discrete logarithm assumption in the generic group model. In: ASIACRYPT (4). Lecture Notes in Computer Science, vol. 13093, pp. 587–617. Springer (2021)
13. Blum, E., Katz, J., Liu-Zhang, C., Loss, J.: Asynchronous byzantine agreement with subquadratic communication. In: TCC (1). Lecture Notes in Computer Science, vol. 12550, pp. 353–380. Springer (2020)
14. Blum, E., Katz, J., Loss, J., Nayak, K., Ochsenreither, S.: Abraxas: Throughput-efficient hybrid asynchronous consensus. In: CCS. pp. 519–533. ACM (2023)
15. Blum, M.: Coin flipping by telephone. In: CRYPTO. pp. 11–15. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04 (1981)
16. Boneh, D., Boyen, X.: Short signatures without random oracles. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 3027, pp. 56–73. Springer (2004)
17. Boneh, D., Boyen, X.: Efficient selective identity-based encryption without random oracles. J. Cryptol. **24**(4), 659–693 (2011)
18. Bracha, G.: Asynchronous byzantine agreement protocols. Information and Computation **75**(2), 130–143 (1987)
19. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure asynchronous byzantine consensus with optimal resilience. In: Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC). pp. 33–42. ACM (2001)
20. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography (extended abstract). In: PODC. pp. 123–132. ACM (2000)
21. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145. IEEE Computer Society (2001)
22. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: CRYPTO. Lecture Notes in Computer Science, vol. 1666, pp. 98–115. Springer (1999)
23. Canetti, R., Rabin, T.: Fast asynchronous byzantine agreement with optimal resilience. In: STOC. pp. 42–51. ACM (1993)
24. Choi, K., Manoj, A., Bonneau, J.: Sok: Distributed randomness beacons. In: SP. pp. 75–92. IEEE (2023)
25. Choudhury, A., Patra, A.: On the communication efficiency of statistically secure asynchronous MPC with optimal resilience. J. Cryptol. **36**(2), 13 (2023)
26. Das, S., Camacho, P., Xiang, Z., Nieto, J., Bünz, B., Ren, L.: Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 356–370 (2023)

27. Das, S., Duan, S., Liu, S., Momose, A., Ren, L., Shoup, V.: Asynchronous consensus without trusted setup or public-key cryptography. In: CCS. pp. 3242–3256. ACM (2024)
28. Das, S., Ren, L.: Adaptively secure BLS threshold signatures from DDH and co-cdh. In: CRYPTO (7). Lecture Notes in Computer Science, vol. 14926, pp. 251–284. Springer (2024)
29. Das, S., Xiang, Z., Kokoris-Kogias, L., Ren, L.: Practical asynchronous high-threshold distributed key generation and distributed polynomial sampling. In: USENIX Security Symposium. pp. 5359–5376. USENIX Association (2023)
30. Das, S., Xiang, Z., Tomescu, A., Spiegelman, A., Pinkas, B., Ren, L.: Verifiable secret sharing simplified. Cryptology ePrint Archive, Paper 2023/1196 (2023), https://eprint.iacr.org/2023/1196
31. Das, S., Yurek, T., Xiang, Z., Miller, A., Kokoris-Kogias, L., Ren, L.: Practical asynchronous distributed key generation. In: SP. pp. 2518–2534. IEEE (2022)
32. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 10821, pp. 66–98. Springer (2018)
33. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: STOC. pp. 148–161. ACM (1988)
34. Feldman, P.N.: Optimal algorithms for Byzantine agreement. Ph.d. thesis, Massachusetts Institute of Technology, Cambridge, MA (1988)
35. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997)
36. Feng, H., Gao, Y., Lu, Y., Tang, Q., Xu, J.: Practical asynchronous distributed key reconfiguration and its applications (2024), unpublished manuscript
37. Feng, H., Lu, Z., Tang, Q.: Dragon: Decentralization at the cost of representation after arbitrary grouping and its applications to sub-cubic DKG and interactive consistency. In: PODC. pp. 469–479. ACM (2024)
38. Feng, H., Mai, T., Tang, Q.: Scalable and adaptively secure any-trust distributed key generation and all-hands checkpointing. In: ACM CCS (2024)
39. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology – CRYPTO 2018. Lecture Notes in Computer Science, vol. 10992, pp. 33–62. Springer (2018)
40. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 10992, pp. 33–62. Springer (2018)
41. Ganesh, C., Kondi, Y., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Witness-succinct universally-composable snarks. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 14005, pp. 315–346. Springer (2023)
42. Gao, Y., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Efficient asynchronous byzantine agreement without private setups. In: ICDCS. pp. 246–257. IEEE (2022)
43. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A.: Registration-based encryption: Removing private-key generator from IBE. In: TCC (1). Lecture Notes in Computer Science, vol. 11239, pp. 689–718. Springer (2018)
44. Garg, S., Jain, A., Mukherjee, P., Sinha, R., Wang, M., Zhang, Y.: hints: Threshold signatures with silent setup. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 3034–3052. IEEE (2024)
45. Garg, S., Kolonelos, D., Policharla, G., Wang, M.: Threshold encryption with silent setup. In: CRYPTO (7). Lecture Notes in Computer Science, vol. 14926, pp. 352–386. Springer (2024)
46. Gelashvili, R., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A., Xiang, Z.: Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. In: Financial Cryptography. Lecture Notes in Computer Science, vol. 13411, pp. 296–315. Springer (2022)
47. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007)
48. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: SOSP. pp. 51–68. ACM (2017)
49. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer (2016)
50. Groth, J.: Non-interactive distributed key generation and key resharing. IACR Cryptol. ePrint Arch. p. 339 (2021)
51. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer (2008)
52. Guo, B., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Speeding dumbo: Pushing asynchronous BFT closer to practice. In: NDSS. The Internet Society (2022)

53. Guo, B., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Dumbo: Faster asynchronous BFT protocols. In: CCS. pp. 803–818. ACM (2020)
54. Gurkan, K., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Aggregatable distributed key generation. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 147–176. Springer (2021)
55. Huang, S., Pettie, S., Zhu, L.: Byzantine agreement with optimal resilience via statistical fraud detection. J. ACM **71**(2), 12:1–12:37 (2024)
56. Jaeger, J., Mohan, D.I.: Generic and algebraic computation models: When AGM proofs transfer to the GGM. In: CRYPTO (5). Lecture Notes in Computer Science, vol. 14924, pp. 14–45. Springer (2024)
57. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (2010)
58. Katz, J.: Round-optimal, fully secure distributed key generation. In: CRYPTO (7). Lecture Notes in Computer Science, vol. 14926, pp. 285–316. Springer (2024)
59. Katz, J., Koo, C.: On expected constant-round protocols for byzantine agreement. In: CRYPTO. Lecture Notes in Computer Science, vol. 4117, pp. 445–462. Springer (2006)
60. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: Omniledger: A secure, scale-out, decentralized ledger via sharding. In: IEEE Symposium on Security and Privacy. pp. 583–598. IEEE Computer Society (2018)
61. Kokoris-Kogias, E., Malkhi, D., Spiegelman, A.: Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In: CCS. pp. 1751–1767. ACM (2020)
62. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., abhi shelat, Shi, E.: C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Paper 2015/1093 (2015), https://eprint.iacr.org/2015/1093
63. Libert, B., Joye, M., Yung, M.: Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. In: Proceedings of the 2014 ACM symposium on Principles of distributed computing. pp. 303–312 (2014)
64. Lindell, Y., Lysyanskaya, A., Rabin, T.: On the composition of authenticated byzantine agreement. J. ACM **53**(6), 881–917 (2006)
65. Lu, Y., Lu, Z., Tang, Q., Wang, G.: Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In: PODC. pp. 129–138. ACM (2020)
66. Maurer, U.M.: Abstract models of computation in cryptography. In: IMACC. Lecture Notes in Computer Science, vol. 3796, pp. 1–12. Springer (2005)
67. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of BFT protocols. In: CCS. pp. 31–42. ACM (2016)
68. Momose, A., Ren, L.: Optimal communication complexity of authenticated byzantine agreement. In: DISC. LIPIcs, vol. 209, pp. 32:1–32:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
69. Mostéfaoui, A., Moumen, H., Raynal, M.: Signature-free asynchronous binary byzantine consensus with t < n/3, o(n2) messages, and O(1) expected time. J. ACM **62**(4), 31:1–31:21 (2015)
70. Mostéfaoui, A., Raynal, M.: Signature-free asynchronous byzantine systems: from multivalued to binary consensus with t< n/3, o(n^2) messages, and constant time. Acta Informatica **54**(5), 501–520 (2017)
71. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer (1991)
72. Qiu, T., Tang, Q.: Predicate aggregate signatures and applications. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 279–312. Springer (2023)
73. Rabin, M.O.: Randomized byzantine generals. In: FOCS. pp. 403–409. IEEE Computer Society (1983)
74. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 1233, pp. 256–266. Springer (1997)
75. de Souza, L.F., Kuznetsov, P., Tonkikh, A.: Distributed randomness from approximate agreement. In: DISC. LIPIcs, vol. 246, pp. 24:1–24:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)

A More Related Work

Preprocessing Coins. A few classical asynchronous coin protocols assume a preprocessing phase where the participants already share correlated secret states. The simplest solution is due to M.Rabin's pioneering work [73], in which many random values have been secretly shared among all participants, such that in the online phase, each honest node multicasts a secret share and then can reconstruct a random value based the shares it received. The online phase incurs $O(\ell n^2)$-bit communication for each coin of ℓ bits and only requires exactly 1 round. An obvious drawback is that the network has to share $O(N)$ random values in the preprocessing phase to supply $O(N)$ common coins in the online phase. This issue was resolved by Cachin, Kursawe, and Shoup [20] in the computational setting by leveraging cryptographic tools, including a random oracle and a non-interactive unique threshold signature scheme [28,63]. In their scheme, the group only needs to share one random value in the preprocessing phase, which is a secret key of the threshold signature scheme. Later on, in the online phase, a pseudorandom value is recovered as a common coin at the cost of $O(\lambda n^2)$ communication and exactly one round. In addition, Cachin et al. [20] can tolerate up to $\lceil \frac{n}{3} \rceil - 1$ Byzantine nodes, and if the underlying unique threshold signature scheme is adaptively secure (for instance, using [8]), it is strongly adaptive secure [1][8]. Due to its superior performance and security, Cachin et al. [20] has been widely adopted by recent practical asynchronous consensus protocols [4,52,53,67]. E.Blum et al. [13] presented an asynchronous common coin protocol with subquadratic communication complexity in this model, which, however, is not strongly adaptive secure and only offers sub-optimal resilience.

Essentially, in the preprocessing-model coin protocols, the parties can obtain a common coin in the online phase by somehow "consuming" the coin that is "prepared" in the preprocessing phase. Therefore, if there is no trusted dealer to share a random value with the group in the preprocessing phase (for example, in blockchain-related applications), the group still needs to generate a coin from scratch (at least as a setup for emulating the dealer), which motivates a long line of research [3,33,35,42,61] dedicating to common coin generation without preprocessing.

Coins without preprocessing. A setup phase is considered to be "silent" if all participants do not communicate with each other, except posting their own public information to a public bulletin board only once during the setup phase. The family of silent setups covers a variety of different setups, including that for establishing the global set of participants' identities, the standard PKI setup, and the common reference string (CRS) setup.

Information-theoretical Coins. The seminal work by Feldman and Micali [33,35] initiated the study of the common coin without preprocessing, with a focus on the information theoretical setting, under a setup for establishing global identities and pairwise private and authenticated channels. They presented a *weak common coin* protocol, which, on rough terms, satisfies the *agreement* property of a (strong) common coin with a constant probability. While Feldman and Micali's result assumes a *synchronous* network, Feldman [34] and Canetti and T. Rabin [23] extended it to asynchrony. In particular, Canetti and T. Rabin's protocol tolerates up to $\lceil \frac{n}{3} \rceil - 1$ Byzantine nodes, while the communication cost for generating a binary common coin is as large as $\Omega(n^8 \log n)$. Bangalore, Choudhury, and Patra [11] considered an even weaker form of common coin, which has the communication complexity of $O(n^6)$. These weak forms of common coins are sufficient for certain asynchronous consensus protocols such as asynchronous binary agreement [11,23,69]. In addition, there is a generic information-theoretical complier, sketched by Choudhury and Patra in [25], that lifts a weak common coin protocol to a strong common coin protocol at the cost of introducing another multiplicative factor of n in the communication complexity. Recently, Huang, Pettie, and Zhu [55] presented a strong common coin protocol that does not assume private channels

[8] It is secure against adaptive adversaries who can retract yet-delivered messages sent by a newly corrupted node.

and tolerates up to $(\frac{n}{3+\epsilon})$ Byzantine nodes, with communication complexity of $\Omega(n^7)$ and $\mathsf{poly}(n)$ rounds.

Computational Coins. On the other hand, cryptography has been extensively employed to improve the performance of common coin protocols. Katz and Koo [59] extend Feldman and Micali's protocol to the computational setting with a PKI setup, which yields a computationally secure weak common coin protocol with $O(\lambda n^4)$ communication complexity and $O(1)$ rounds. In recent years, motivated by replacing the trusted setup needed by Cachin et al.'s common coin [20] with an asynchronous distributed protocol, many efforts have been devoted to improving the performance of asynchronous distributed key generation (ADKG), which has a close relation with asynchronous common coin. Namely, the consensus protocols inside ADKG will rely on certain weak coin protocols, while an ADKG protocol directly implies an asynchronous strong common coin protocol with the same complexity.

Kokoris-Kogias et al. [61] gave the first ADKG protocol, which assumes a PKI setup and has the communication complexity of $O(\lambda n^4)$ and the round complexity of $O(n)$. Later, Abraham et al. [3] and Gao et al. [42] presented ADKG protocols with $O(\lambda n^3)$ communication complexity and $O(1)$ rounds. Both works assume a setup for PKI and the CRS and are proven statically secure in the random oracle model, though with an adaptively secure component from [9], they are adaptively secure in principle in the algebraic group model (AGM) [40]. Das et al. [29, 31] focused on the practical performance and presented statically secure ADKG with $O(\lambda n^3)$ communication complexity and $O(\log n)$ round complexity. Abraham et al. [2] gave a strongly adaptively secure DKG under a CRS setup and in the AGM model, with $O(\lambda n^3)$ communication complexity and $O(1)$ round complexity. Compared with [3, 42], [2] generates the standard secret key for DLog-based cryptosystems.

Other than research on ADKG, Freitas, Kuznetsov, and Tonkikh studied Bandarupalli et al. [10] adapted [75] to the hash-based setting and presetned a asycnhronous randomness beacon that countinosuly emitting common coins, which can be proved adaptively secure in the ROM. Though its amortized communication cost for each coin is subcubic, a single-shot execution incurs $O(\lambda n^3)$ communication complexity. Very recently, Das et al. [27] presented a hash-based weak coin protocol in the random oracle model, with the $O(\lambda n^3)$ communication complexity and $O(1)$ round complexity.

B Other Preliminaries

B.1 Binary Weak Common Coin

In the definition of [35], the termination is the same as ours, and the ρ-fairness is defined as follows:

- ϕ-fairness: let v_i be the output of P_i. Then for every $b \in \{0,1\}$, it holds that

$$\Pr[\forall i \in \mathcal{H} : v_i = b] \geq \phi - \mathsf{negl}(\lambda).$$

We have the following lemma regarding the relation between our definition and their definition.

Lemma 14. *A protocol with $V = \{0,1\}$ and satisfying φ-fairness in our definition directly satisfies $\varphi/2$-fairness in the definition of [35].*

Algorithm 5 The provable broadcast protocol PB with identifier $\mathtt{sid}$, the sender P_s,and the predicate $\mathsf{Predicate}$

```
    // Protocol for the sender $P_s$
    Initialize local variable $S = \emptyset$
 1: upon receiving input $v_s$ satisfying $\mathsf{Predicate}(v) = 1$ do
 2:   Multicast $(\mathtt{sid}, \textsc{input}, v_s)$
 3:   wait until $|S| = n - f$
 4:     $\varsigma \leftarrow \mathsf{STS.Combine}(vk, S, (\mathtt{sid}, v_s))$
 5:     return $\mathtt{b\text{-}proof} = \varsigma$.
 6: upon receiving $(\mathtt{sid}, \textsc{EchoInput}, \varsigma_{s,j})$ from $P_j$ for the first time do
 7:   if $\mathsf{TSig.PartVrfy}(pvk, pvk_j, \varsigma_{s,j}, (\mathtt{sid}, v_s)) = 1$ then
 8:     $S \leftarrow S \cup \{(j, \varsigma_{s,j})\}$
    // Protocol for other $P_i$
    Initialize local variable $v[\mathtt{sid}] = \perp$ and $\mathtt{stop} = 0$
 9: upon receiving $(\mathtt{sid}, \textsc{Input}, v_s)$ from $P_s$ for the first time do
10:   if $\mathsf{Predicate}(v_s) = 1$ and $\mathtt{stop} = 0$ then
11:     $v[\mathtt{sid}] \leftarrow v_s$
12:     $\varsigma_{s,i} \leftarrow \mathsf{TSig.PartS}(tvk_i, tsk_i, (\mathtt{sid}, v_s))$
13:     Send $(\mathtt{sid}, \textsc{EchoInput}, \varsigma_{s,i})$ to $P_s$
14:     Return $v[\mathtt{sid}]$
15:
    procedure $Abandon(\mathtt{sid})$
16:   $\mathtt{stop} \leftarrow 1$
    $\mathsf{Vrfy}(\mathtt{sid}, \mathtt{b\text{-}proof}, v)$
17:   return $\mathsf{TSig.Vrfy}(pvk, (\mathtt{sid}, v), \mathtt{bproof})$
```

Proof. According to our definition, there is an event $\mathtt{Fair}$ such that $\Pr[\mathtt{Fair}] \geq \varphi - \mathsf{negl}(\lambda)$, and $\Pr[\exists b, v_i = b \forall i \in \mathcal{H} | \mathtt{Fair}] = 1$. So $\Pr[\exists b, v_i = b \forall i \in \mathcal{H}] \geq \varphi - \mathsf{negl}(\lambda)$. Then, if there exists $b_0 \in \{0, 1\}$ s.t. $\Pr[v_i = b_0 \forall i \in \mathcal{H}] < \varphi/2 - \epsilon$, for some non-negligibel function ϵ. It follows that $\Pr[v_i = 1 - b_0 \forall i \in \mathcal{H}] > \varphi/2 + \epsilon - \mathsf{negl}(\lambda)$. So, the adversary can trivially distinguish the output b from a uniform bit. □

B.2 Provable Broadcast

For completeness, we include an instantiation available in Algorithm 5, which follows the classical construction of [4, 20].

B.3 Silent-Setup Threshold Signature

A threshold signature scheme allows a group of nodes to jointly sign a message w.r.t. a group public key, while an adversary controlling below a threshold number of nodes cannot forge a signature. It is a silent threshold signature scheme if it has a silent setup phase[9].

Formally, an (n, t) silent threshold signature (STS) scheme can be described using the following algorithms.

- $\mathsf{SilSetup}(1^\lambda, n, t) \to (\mathtt{crs}, vk, (vk_i), (sk_i))$. It is the silent setup phase of STS. We denote the public verification key and the secret signing key by vk and sk, respectively.

[9] We remark that existing silent threshold signature schemes lack uniqueness and therefore do not imply a common coin under Cachin et al.'s framework [20].

- $\mathsf{PartSign}(vk, vk_i, sk_i, \mathsf{msg}) \to \sigma_i$. It signs a message msg using the signing secret key sk_i of the user i and produces a partial signature σ_i.
- $\mathsf{PartVrfy}(vk, vk_i, \sigma_i, \mathsf{msg}) \to b \in \{0,1\}$. It validates a partial signature σ_i on the message msg under the public key vk_i of the user i.
- $\mathsf{Combine}(vk, \{(vk_i, \sigma_i)\}_{i\in S}, \mathsf{msg}) \to \sigma$. It combines $t+1$ valid signatures for the same message msg under distinct public keys into one signature σ.
- $\mathsf{Vrfy}(vk, \sigma, \mathsf{msg}) \to b \in \{0,1\}$. It validates a combined signature σ for a message msg under the group public key vk.

An STS scheme satisfies the correctness and the unforgeability.

Correctness. An (n,t)-STS scheme is *correct*, if for every $\lambda \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, under an adversary-involved setup $\mathsf{SilSetup}^{\mathcal{A}}(1^\lambda, n, t) \to (\mathsf{crs}, vk,$
$(vk_i)_{i\in[n]}; \mathsf{state}_{\mathcal{A}}; (sk_i)_{i\in[n]\setminus\mathcal{C}_{\mathsf{init}}})$, we have the following properties.

- *Partial Verification Correctness.* For any m and any $i \notin \mathcal{C}_{\mathsf{init}}$, it holds that

$$\Pr[\mathsf{PartVrfy}(vk, vk_i, \mathsf{PartSign}(vk, vk_i, sk_i, \mathsf{msg}), \mathsf{msg}) = 1] = 1.$$

- *Combination Correctness.* For any subset $S \in [n]$ with $|S| = t+1$, and $\{(vk_i, \sigma_i)\}_{i\in S}$ such that $\mathsf{PartVrfy}(vk, vk_i, \sigma_i, \mathsf{msg})$ for the same message msg, it holds that

$$\Pr[\mathsf{Vrfy}(vk, \mathsf{Combine}(vk, (vk_i, \sigma)_i)_{i\in S}, \mathsf{msg}) = 1] = 1.$$

Unforgeability. At a high level, an (n,t)-STS scheme satisfies the unforgeability if any PPT adversary $\mathcal{A}$ controlling up to t nodes cannot forge a valid signature on a message that has not been signed by any honest nodes. Furthermore, we allow the adversary to perform the chosen message attacks (modeled by a signing oracle $\mathcal{O}_{\mathsf{PartS}}$ in Fig.). In addition, an adaptive adversary may corrupt participants at any point; the capability of adaptive corruption is modeled by $\mathcal{O}_{\mathsf{Corr}}$.

Formally, an (n,t)-STS scheme is unforgeable, if for any PPT adversary $\mathcal{A}$ and $\lambda \in \mathbb{N}$, it holds that

Formally, we say an (n,t)-STS scheme satisfies the unforgeability if for any PPT adversary $\mathcal{A}$, it holds that, for every $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{UNF}^{\mathcal{A}}(1^\lambda, n, t) = 1] \le \mathsf{negl}(\lambda). \tag{6}$$

An (n,f)-STHE scheme satisfies the adaptive unforgeability, if

$$\Pr[\mathsf{Ada\text{-}UNF}^{\mathcal{A}}(1^\lambda, n, t) = 1] \le \mathsf{negl}(\lambda). \tag{7}$$

Instantiation. Silent threshold signatures have various instantiations [5,26,44,72]. In this work, we consider the scheme from [26] as the instantiation, as it offers the constant signature size and the constant verification cost while ensuring the adaptive unforgeability under the q-SDH assumption in the random oracle model. We remark that while the corruption oracle in the security of [26] does not leak the randomness w.r.t. partial signing, the partial signing algorithm in [26] (and most other existing schemes) is deterministic, so there is no randomness to leak.

```
$\boxed{\mathsf{Ada}}\text{-}\mathsf{UNF}^{\mathcal{A}}(1^\lambda, n, t)$
$\mathcal{C}, \mathsf{State}_1, \ldots, \mathsf{State}_n \leftarrow \emptyset, \mathsf{Msgs} \leftarrow \perp$
$(\mathsf{crs}, vk, (vk_i)_{i\in[n]}; \mathsf{state}_{\mathcal{A}}; (sk_i)_{i\in[n]\setminus\mathcal{C}_{\mathsf{init}}}) \leftarrow \mathsf{SilSetup}^{\mathcal{A}}(1^\lambda, n, t), \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\mathsf{init}}$
$(\sigma^*, \mathsf{msg}^*) \leftarrow \mathcal{A}^{\boxed{\mathcal{O}_{\mathsf{Corr}}(\cdot),}\, \mathcal{O}_{\mathsf{PartS}}(\cdot)}(\mathsf{state}_{\mathcal{A}})$
if $\mathsf{msg}^* \notin \mathsf{Msgs} \wedge \mathsf{Vrfy}(vk, \sigma^*, \mathsf{msg}^*) = 1$, then return $1$; else return $0$

$\mathcal{O}_{\mathsf{Corr}}(i)$
if $i \in [n] \setminus \mathcal{C} \wedge |\mathcal{C}| < t$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$
  return $(sk_i, \mathsf{State}_i)$

$\mathcal{O}_{\mathsf{PartS}}(\mathsf{msg}, i)$
if $i \notin \mathcal{C}$
  $\mathsf{Msgs} \leftarrow \mathsf{Msgs} \cup \{\mathsf{msg}\}$
  $\gamma \leftarrow \mathsf{Rand}_{\mathsf{PartS}}, \sigma_i \leftarrow \mathsf{PartSign}(vk, vk_i, sk_i, \mathsf{msg}; \gamma)$
  $\mathsf{State}_i \leftarrow \mathsf{State}_i \cup \{(\mathsf{msg}, i, \gamma, \sigma_i)\}$   return $\sigma_i$
```

Fig. 8. Security Game of STS. $\boxed{\mathcal{O}_{\mathsf{Corr}}}$ is only included in Ada-UNF

C The Details of STHE Scheme

Our STHE scheme represents a minimal modification from the STE scheme in [45]. To put it simply, the encryption algorithm in the STE scheme involves uniformly sampling a group element Γ, which will also be included in the ciphertext as a public element. To enable the desired tag-homomorphism, we need to replace the random element Γ with $H(tag)$, where H is a hash function modeled as a random oracle.

The STE scheme [45] can be considered as signature-based witness encryption [6] w.r.t. the STS scheme [44], where the plaintext is encrypted w.r.t. a "message" and a public key, and it can be decrypted by a valid signature for the "message" under the public key. The group element Γ in the ciphertext is the "message" from this perspective. As a message $\mathtt{m}$ in the STS scheme is encoded as $H(\mathtt{m})$, a "natural" version of STE, which is discussed in the introduction of [45], indeed uses $H(tag)$ in place of a random Γ. The final scheme of STE in [45] uses a random element Γ for the purpose of avoiding the additional random oracle assumption.

For completeness, we include the description of the STHE scheme in this section, which is basically the natural" version of STE scheme [45], while we add the homomorphic evaluation algorithm Eval to make it an STHE.

Notations and Bilinear Groups. We introduce the notations for polynomial and group operations. The following content is directly from [45] and [44].

We use the following notations for polynomials over finite fields. Let $\mathbb{H} \subset \mathbb{F}$ be a multiplicative subgroup of a finite field $\mathbb{F}$. Let ω be the generator of $\mathbb{H} = \{\omega, \omega^2, \ldots, \omega^{|\mathbb{H}|} = 1\}$.

Let $L_1(x), L_2(x), \ldots, L_{|\mathbb{H}|}(x)$ be the Lagrange basis polynomial. That is, L_i is the unique degree $|\mathbb{H}| - 1$ polynomial defined by:

$$L_i(\omega^j) = \begin{cases} 1 & \text{when } i = j, \\ 0 & \text{when } i \neq j. \end{cases}$$

Let $Z(x) = \prod_{i=1}^{|\mathbb{H}|}(x-\omega^i)$ be the vanishing polynomial on $\mathbb{H}$. Since $\mathbb{H}$ is a multiplicative subgroup, $Z(x) = x^{|\mathbb{H}|} - 1$ and

$$L_i(x) = \frac{\omega^i}{|\mathbb{H}|} \cdot \frac{x^{|\mathbb{H}|} - 1}{x - \omega^i}.$$

Note that $L_i(0) = |\mathbb{H}|^{-1}$. In the construction, We use $|\mathbb{H}| = n+1$, where n is the number of parties.

A Bilinear Group $\mathcal{BG}$, generated as $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{BG}(1^\lambda)$, is specified by three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ (the first two we call 'source groups' and the third 'target group') of prime order $p = 2^{\Theta(\lambda)}$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that we call 'pairing' and one random generator g_1, g_2 for each group. We use the *implicit notation*, i.e., $[x]_s := x \cdot g_s$ and more generally $[A]_s$ represents a matrix of the corresponding group elements, for $s \in \{1, 2, t\}$.

Also, we denote the group operation additively, $[x]_s + [y]_s = [x + y]_s$, for $s \in \{1, 2, t\}$. By $[x]_1 \circ [y]_2 = [y]_2 \circ [x]_1 = [x \cdot y]_T$, we denote the pairing $e([x]_1, [y]_2)$. We note that the way it is defined '$\circ$' is commutative, for instance $([a]_1, [b]_2)^\top \circ ([c]_2, [d]_1)$ is well-defined and gives the outcome $[ac + bd]_T$.

All the algorithms of our constructions implicitly take as input a Bilinear Group generated by $\mathcal{BG}(\lambda)$, even if it is not explicitly stated.

The construction. Most construction descriptions are from [45]. We simplified the interfaces by merging a few algorithms and variables. For example, the user key generation algorithm and the "hint" generation algorithm are included in a single key generation algorithm; the group encryption key and aggregation public key are treated as a single group public key. We also decouple the decryption algorithm into a decryption key aggregation algorithm and a decryption algorithm using this key, which enables the use of a single key to decrypt all ciphertexts under the same tag. We highlighted the changes for tag-homomorphism in green.

- $\mathsf{SilSetup}(1^\lambda, n, t)$: It can be described by a CRS generation algorithm CRS, a key generation algorithm KeyGen, and a deterministic group key aggregation algroithm $\mathsf{GroupPkGen}$.
  - $\mathsf{CRS}(1^\lambda, n)$: Sample $\tau \leftarrow \mathbb{Z}_p$ and output:

$$\mathtt{crs} = \left([\tau^1]_1, \ldots, [\tau^{n+1}]_1, [\tau^1]_2, \ldots, [\tau^{n+1}]_2\right).$$

  - $\mathsf{KeyGen}(\mathtt{CRS}, i, n)$: Sample $x_i \leftarrow \mathbb{Z}_p^*$, and set $\mathsf{dk}_i = x_i$, $\mathsf{enck}_i = [x_i]_1$. Then, compute

$$\mathsf{hint}_i = \left(\begin{array}{c} [\mathsf{dk}_i L_i(\tau)]_1, [\mathsf{dk}_i(L_i(\tau) - L_i(0))]_1, \left[\mathsf{dk}_i \dfrac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\right]_1, \\ \left[\mathsf{dk}_i \dfrac{L_i(\tau) - L_i(0)}{\tau}\right]_1, \left\{\left[\mathsf{dk}_i \dfrac{L_i(\tau)L_j(\tau)}{Z(\tau)}\right]_1\right\}_{j \in [0,n], j \neq i} \end{array}\right).$$

  Output $ek_i = (\mathsf{enck}_i, \mathsf{hint}_i)$, and $\mathsf{dk}_i = x_i$.
  - $\mathsf{GroupPKGen}(\mathtt{CRS}, \{\mathsf{hint}_i, \mathsf{ek}_i\}_{i \in [n]})$: Verify the validity of each ek_i: for each $i \in [n]$, run $\mathsf{isValid}(\mathtt{CRS}, \mathsf{ek}_i)$ (isValid is defined below) and let $V \subseteq [n]$ be the set of the indices with valid public keys. Set $\mathsf{dk}_0 = 1$ and set $\mathsf{dk}_i = 0$ for each $i \notin V$. Compute

$$\mathsf{ak} = \left(\begin{array}{c} V, \{\mathsf{ek}_i\}_{i \in V \cup \{0\}}, \{[\mathsf{dk}_i(L_i(\tau) - L_i(0))]_1\}_{i \in V \cup \{0\}}, \left\{\left[\mathsf{dk}_i \dfrac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\right]_1\right\}_{i \in V \cup \{0\}}, \\ \left\{\left[\mathsf{dk}_i \dfrac{L_i(\tau) - L_i(0)}{\tau}\right]_1\right\}_{i \in V \cup \{0\}}, \left\{\left[\sum_{j \in S, j \neq i} \mathsf{sk}_j \dfrac{L_j(\tau)L_i(\tau)}{Z(\tau)}\right]_1\right\}_{i \in V \cup \{0\}} \end{array}\right).$$

– enck:

$$\mathsf{enck} = \left(\left[\sum_{i \in V} \mathsf{dk}_i L_i(\tau) \right]_1, [Z(\tau)]_2 \right) := (C, Z).$$

Output $ek = (ak, \mathsf{enck})$.

- $\mathsf{Enc}(\mathsf{ek}, tag, \mathsf{msg})$: Set $[\gamma]_2 = H(tag) \in \mathbb{G}_2$ (rather than sample $[\gamma]_2 \leftarrow\$ \mathbb{G}_2$ in [45]), parse $\mathsf{ek} := (\mathsf{ak}, C, Z)$, set $Z_0 = [\tau - \omega^0]_2$ and set

$$A_{tag} = \begin{pmatrix} C & [1]_2 & Z & [\tau]_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & [\tau]_2 & [1]_2 & 0 & 0 & 0 \\ 0 & [\gamma]_2 & 0 & 0 & 0 & [1]_1 & 0 & 0 \\ [\tau^t]_1 & 0 & 0 & 0 & 0 & 0 & [1]_2 & 0 \\ [1]_1 & 0 & 0 & 0 & 0 & 0 & 0 & Z_0 \end{pmatrix}.$$

$$\mathbf{b} = ([0]_T, [0]_T, [0]_T, [0]_T, [1]_T)^\top.$$

Notice that each column of A contains elements from the same source group (looking ahead, this is so that the pairing $A \circ \omega$ can be properly performed.)

Sample a vector $\mathbf{s} \leftarrow (\mathbb{Z}_p^*)^5$ and output:

$$c = \left(\mathbf{s}^\top \cdot A_{tag}, \mathbf{s}^\top \cdot \mathbf{b} + \mathsf{msg} \right).$$

- $\mathsf{Eval}(ek, tag, \{c_i\}_{i \in S})$: Parse $c_i = (\mathsf{ct}_{i,1}, \mathbf{ct}_{i,2})$. Output $c^* = (\sum_{i \in S} \mathsf{ct}_{i,1}, \sum_{i \in S} \mathsf{ct}_{i,2})$.
- $\mathsf{PartDec}(ek, tag, \mathsf{dk}_i)$: Output $\zeta_i = \mathsf{dk}_i \cdot H(tag)$.
- $\mathsf{PartVrfy}(ek, tag, ek_i, \zeta_i)$: Parse $ek_i := ek_i = (\mathsf{enck}_i, \mathsf{hint}_i)$, and output 1 if and only if $\mathsf{enck}_i \circ H(tag) = [1]_1 \circ \zeta_i$.
- $\mathsf{DecAgg}(ek, tag, \{\zeta_i\}_{i \in S})$: Parse $ek := (ak, \mathsf{enck} = (C, Z))$, and compute the subset of indices with valid ek_i's, $S_v = S \cap V$. *Then proceed as follows:*
  1. Compute a polynomial $B(X)$ by interpolating 0 on all ω_i with $i \notin S_v$ and 1 on ω^0, i.e., interpolate B as $\{(\omega^0, 1), (\omega^i, 0)_{i \notin S_v}\}$ and set

  $$B = \left[\sum_{i=0}^{n} B(\omega^i) L_i(\tau) \right]_2.$$

  2. Set

  $$\mathsf{aPK} = \frac{1}{n+1} \left(\sum_{i \in S_v} B(\omega^i) \mathsf{enck}_i + [1]_1 \right).$$

  3. Compute polynomials $Q_x(X)$ and $Q_z(X)$ such that

  $$\mathsf{SK}(X) B(X) = \frac{\mathsf{aSK}}{n+1} + Q_z(X) Z(X) + Q_x(X) X$$

  where

  $$\mathsf{aSK} = \sum_{i=0}^{M} \mathsf{SK}(\omega^i) B(\omega^i) := \sum_{i \in S_v} \mathsf{dk}_i B(\omega^i) + 1.$$

  By definition, B evaluates to 0 outside S_v and to 1 on ω^0.

According to Lemma **1**, they can be computed as:

$$Q_z(X) = \sum_{i=0}^{n+1} B(\omega^i)\left(dk_i \frac{L_i^2(X) - L_i(X)}{Z(X)}\right) + \left(\sum_{i=0}^{n+1} B(\omega^i) \sum_{\substack{j=0\\ j\neq i}}^{n+1} \mathsf{dk}_j \frac{L_i(X)L_j(X)}{Z(X)}\right),$$

$$Q_x(X) = \sum_{i=0}^{n+1} B(\omega^i)\left(\mathsf{dk}_i \frac{L_i(X) - L_i(0)}{X}\right).$$

Then using ak, compute $Q_z = [Q_z(\tau)]_1$.

4. Compute $Q_x = [Q_x(\tau)]_1$.
5. Compute $\hat{Q}_x = [Q_x(\tau) \cdot \tau]_1$.
6. Set $\zeta^* = \frac{1}{n+1}\left(\sum_{i \in S_v} B(\omega^i)\zeta_i + H(tag)\right)$.
7. Compute $\hat{B} = [\tau B(\tau)]_1$.
8. Compute a KZG evaluation at ω^0, i.e., compute $Q_0(X)$ such that $B(X) - 1 = Q_0(X)(X - \omega^0)$ and compute $Q_0 = [Q_0(\tau)]_1$.
9. Output:

$$dk_{tag} = \left(B, -\mathsf{aPK}, -Q_z, -Q_x, \hat{Q}_x, \zeta^*, -\hat{B}, -Q_0\right)^\top,$$

- $\mathsf{Dec}(ek, tag, c, dk_{tag})$: Parse $c = (\mathsf{ct}_1, \mathbf{ct}_2)$ and output:

$$\mathsf{msg}^* = \mathsf{ct}_2 - \mathbf{ct}_1 \circ dk_{tag}.$$

In order to confirm that hint_i is well-formed in the Preprocess algorithm, we use an $\mathsf{isValid}$ algorithm which we define as follows:

$\mathsf{isValid}(\mathtt{CRS}, ek_i) \to \{0,1\}$: parses $\mathsf{ek}_i := (\mathsf{enck}_i, \mathsf{hint}_i)$, and $\mathsf{hint}_i := \left(h_1, h_2, h_3, h_4, \{h_{5,j}\}_{j\in[0,M], j\neq i}\right)$

And outputs 1 iff it holds that:

1. $h_1 \circ [1]_2 = \mathsf{enck}_i \circ [L_i(\tau)]_2$,
2. $h_2 \circ [1]_2 = \mathsf{enck}_i \circ [(L_i(\tau) - L_i(0))]_2$,
3. $h_3 \circ [1]_2 = \mathsf{enck}_i \circ \left[\frac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\right]_2$,
4. $h_4 \circ [1]_2 = \mathsf{enck}_i \circ \left[\frac{L_i(\tau) - L_i(0)}{\tau}\right]_2$,
5. $h_5 \circ [1]_2 = \mathsf{enck}_i \circ \left[\frac{L_i(\tau)L_j(\tau)}{Z(\tau)}\right]_2$, for each $j \in [0, M], j \neq i$.

Efficiency. It is almost identical to the STE scheme in [45], except that the ciphertext size is slightly reduced since there is no random element $\Gamma \in \mathbb{G}_2$ included in the ciphertext. In particular, the CRS contains $n+1$ elements in $\mathbb{G}_1$ and $n+1$ elements in $\mathbb{G}_2$, whose size is $O(\lambda n)$. The public key ek_i of each node P_i contains $n+4$ elements in $\mathbb{G}_1$, whose size is also $O(\lambda n)$. The group public key ek contains $5n+6$ elements in $\mathbb{G}_1$ and one element in $\mathbb{G}_2$, whose size is also $O(\lambda n)$. The size of dk_i is $O(\lambda)$. Notice that each node obtains this information in the setup phase.

A ciphertext contains 2 elements in $\mathbb{G}_1$, 6 elements in $\mathbb{G}_2$, and 1 element in $\mathbb{G}_T$, so its size is $O(\lambda)$. A partial decryption key consists of 1 element in $\mathbb{G}_2$, which is $O(\lambda)$ bits.

Correctness. The *partial verification correctness* and the *decryption correctness* trivially follow the correctness of the STE scheme. The evaluation correctness is easy to verify. In particular, for every $c_i = \mathsf{Enc}(ek, tag, \mathtt{msg}_i; \gamma')$, it has the following form:

$$c_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}) = \left(\mathbf{s}_i^\top \cdot A_{tag}, \mathbf{s}_i^\top \cdot b + \mathtt{msg}_i\right),$$

where $\mathbf{s}_i$ is the internal randomness (γ' in the notation) for each ciphertext, while A_{tag} is uniquely determined by tag, so every c_i has the same A_{tag} and b. Thus, $c^* = \mathsf{Eval}(ek, tag, \{c_i\}_{i\in S})$ has the following form:

$$(\sum_{i\in S} \mathsf{ct}_{i,1}, \sum_{i\in S} \mathsf{ct}_{i,2}) := \left((\sum_{i\in S} \mathbf{s}_i^\top) \cdot A_{tag}, (\sum_{i\in S} \mathbf{s}_i^\top) \cdot b + \sum_{i\in S} \mathtt{msg}_i\right),$$

which is a valid encryption for $\sum_{i\in S} \mathtt{msg}_i$ w.r.t. the randomness $\mathbf{s}^* = \sum_{i\in S} \mathbf{s}_i^\top$.

C.1 STHE Security

While the STHE scheme described above represents a minor modification from the STE scheme in [45], the security we defined in Fig.5 is significantly stronger than the security definition of STE [45]. Specifically, in our definition, (1) the adversary is allowed to query the partial decryption oracle $\mathcal{O}_{\mathsf{PartD}}$ to obtain the partial decryption key w.r.t. a tag $tag \neq tag^*$ from any uncorrupted party, and (2) it can adaptively corrupt parties. In contrast, in [45], the corruption is static, and the adversary cannot obtain the partial decryptions of any uncorrupted parties.

In this section, we argue that the STHE scheme indeed satisfies the stronger security definition. Our arguments are based on the original security proof for STE in [45]. At a high level, for handling the queries to $\mathcal{O}_{\mathsf{PartD}}$, we use a standard trick for random oracles so that a partial decryption key $\zeta_i = \mathsf{dk}_i \cdot H(tag)$ can be simulated as $\hat{r}_{tag} \cdot \mathsf{enck}_i = \mathsf{dk}_i \cdot \hat{r}_{tag} \cdot [1]_2$ when $H(tag)$ is programmed as $\hat{r}_{tag} \cdot [1]_2$. In this way, the simulator can respond to queries to $\mathcal{O}_{\mathsf{PartD}}$ without using any information about dk_i. In addition, for handling the adaptive corruption, we can apply the simulation techniques from [12] in a black box manner, which ensures that in the GGM, even under adaptive corruption, the adversary still does not know any non-trivial information about the secret keys of uncorrupted parties.

In the following, we first recall the basic background of the generic group model and some useful results from [12], and then we discuss why STHE satisfies the stronger security definition.

Generic Group Model. In GGM, the adversary cannot see the concrete presentations of group elements and can only use generic group operations. In particular, the adversary makes oracle queries of each group operation it wishes to perform and receives a handle for the resulting group elements. Generally, the adversary is initially provided by the handle of the element $[1]_1, [1]_2$ and $[1]_T$, and it can query the oracle $\mathsf{GC}_{\mathsf{MP}}$ for group operations, which takes as input two strings ξ and ξ' and a bit b, which indicates whether to compute the addition or the subtraction of the group elements. In groups supporting pairing operations, there is an oracle $\mathsf{GC}_{\mathsf{Pair}}$, which takes as input two strings ξ and ξ' and returns $\hat{\xi}$, handling for an element in $\mathbb{G}_T$ as the result of this pairing. Note that the "scalar multiplication" can be done trivially with the oracle $\mathsf{GC}_{\mathsf{MP}}$.

Useful Results from [12]. Now we recall some useful results from [12], which demonstrates the one-more discrete logarithm problem in unconditionally hard in GGM. In particular, the adversary is given (the handlings of) n random group elements $(g_1, \dots g_n)$, and can query the oracle DLog, which returns the scalar (i.e., discrete logarithm) of the queried (handling of) group element, for at most $n-1$ times. The adversary's goal is to produce the scalars of all the n elements. In the security proof in [12], the authors show a simulator (in [12, Fig.6]) that can simulate the security game without ever computing the scalars of g_i, such that the adversaries cannot output those scalars except with a negligible probability. We summarize some useful results from [12] in the following lemma.

Lemma 15 ([12]). *Consider an adversary $\mathcal{A}$ which is given the handlings of $(g_1, \dots g_n)$ and only allowed to query the oracle DLog with those handlings. There is a simulator $\mathcal{S}_{\mathsf{GGM}}$, which can simulate the handlings $(\xi_1, \dots, \xi_n)$ of $(g_1, \dots g_n)$ and responses for DLog and $\mathsf{GC}_{\mathsf{MP}}$, without assigning a scalar k_i to a handling ξ_i (such that ξ_i is the hanlding of $k_1 \cdot [1]_1$) until DLog is queried with ξ_i. In particular, $\mathcal{S}_{\mathsf{GGM}}$ uses random variables $K_1, \dots, K_n$ to represent the scalars $(k_1, \dots, k_n)$, and ensures the invariance in simulating group operation oracles by recording each query as a polynomial in $(K_1, \dots, K_n)$. Except with a negligible probability that $\mathcal{S}_{\mathsf{GGM}}$ may abort, the simulation, at the point of $\mathcal{A}$'s view, is indistinguishable from a "real" security game where $(k_1, \dots k_n)$ are created for $(g_1, \dots g_n)$ and all the oracle queries are responded honestly.*

Security analysis of STHE. The security proof for STHE can be done by going through the following hybrid security games.

Hybrid 0. This is the standard security game of STHE as we defined in Fig.5. Since the scheme is in the generic group model and the random oracle model, the adversary $\mathcal{A}$ also has access to the random oracle H and oracles w.r.t. GGM

Hybrid 1. This hybrid falls back to the "tag-free" setting, where the encryption algorithm $\mathsf{STHE.Enc}$ does not take *tag* as an input, and the random oracle H and the partial decryption oracle $\mathcal{O}_{\mathsf{PartD}}$ are totally removed from the security game. Instead, the encryption algorithm uses a freshly sampled random group element $[\gamma]_2$ in place of $H(tag)$, just like the original STE scheme in [45], while the random element $[\gamma^*]_2$ used in the challenge oracle $\mathcal{O}_b$ is sampled in advance and available to $\mathcal{A}$. For clarity, we describe this hybrid in Fig.9.

```
$\mathsf{Hybrid1}_b^{\mathcal{A}}(1^\lambda, n, f)$
$\mathcal{C} \leftarrow \emptyset, [\gamma^*]_2 \leftarrow\!\!\$\ \mathbb{G}_2$
$(\mathsf{crs}, ek, (ek_i)_{i\in[n]}; \mathsf{state}_{\mathcal{A}}; (dk_i)_{i\in[n]\setminus\mathcal{C}_{\mathsf{init}}})$
    $\leftarrow \mathsf{SilSetup}^{\mathcal{A}([\gamma^*]_2)}(1^\lambda, n, t), \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\mathsf{init}}$
$1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Corr}}(\cdot)\mathcal{O}_b(\cdot)}(\mathsf{state}_{\mathcal{A}})$

$\mathcal{O}_b(m_0, m_1)$
return $c_b \leftarrow \mathsf{Enc}(ek, m_b; [\gamma^*]_2)$

$\mathcal{O}_{\mathsf{Corr}}(i)$
if $i \in [n] \setminus \mathcal{C} \wedge |\mathcal{C}| < t$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$, return $(dk_i)$
```

Fig. 9. Hybrid 1

For the two hybrids, we have the following results.

Proposition 5. *If there is a PPT adversary $\mathcal{A}_0$ that can win the security game in Hybrid 0 with a non-negligible probability, then there exists a PPT adversary $\mathcal{A}_1$ that can win the security game in Hybrid 1 with a non-negligible probability.*

Proof. $\mathcal{A}_1$ can invoke $\mathcal{A}_0$ as a subroutine by simulating Hybrid 0 based on the information from Hybrid 1. In particular, $\mathcal{A}_1$ can program the random oracle H such that $H(tag^*) := [\gamma^*]_2$. For other $tag \neq tag^*$, it samples $\hat{r}_{tag} \leftarrow\$ \mathbb{Z}_p$, and returns $\hat{r}_{tag} \cdot [1]_2$ as the output of $H(tag)$. Then, $\mathcal{A}_1$ can respond a query (tag, i) to $\mathcal{O}_{\mathsf{PartrD}}$ as $\hat{r}_{tag} \cdot \mathsf{enck}_i$. When $\mathcal{A}_0$ queries $\mathcal{O}_b$ or $\mathcal{O}_{\mathsf{Corr}}$, $\mathcal{A}_1$ just forwards the query to the oracles in Hybrid 1. Note that the simulation provided by $\mathcal{A}_1$ is perfect at the point of $\mathcal{A}_0$'s view. So if $\mathcal{A}_0$ can distinguish $\mathcal{O}_0$ and $\mathcal{O}_1$ in Hybrid 1, it can also distinguish them in the simulation; So can $\mathcal{A}_1$. □

Hybrid 2. In this hybrid, the public key of each participant P_i is generated as $(\mathsf{enck}_i, ([\mathsf{dk}_i\tau]_1, \ldots, [\mathsf{dk}_i\tau^n]_1))$. Note that the standard public key with hint_i can be deterministically computed from this simplified one. This hybrid is only for simplifying notations, as used in [45].

Hybrid 3. In this hybrid, the adversary $\mathcal{A}$ is interacting with a simulator $\mathcal{S}_3$, which simulates the setup phase with $\mathcal{A}$ and simulates the oracles $\mathcal{O}_{\mathsf{Corr}}$ and $\mathcal{O}_b$. In particular, $\mathcal{S}_3$ runs a copy of $\mathcal{S}_{\mathsf{GGM}}$ (c.f. Lemma. 15), and interacts with $\mathcal{A}$ in the following manner.

- **Setup**: $\mathcal{S}_{\mathsf{GGM}}$ provides the handlings of $[1]_1, [1]_2, [1]_T$ as well as $\xi_1, \ldots, \xi_n$ to $\mathcal{S}_3$. $\mathcal{S}_3$ samples $\tau \in \mathbb{Z}_p$, and queries the group operation oracle to create the CRS and the public keys for all uncorrupted parties. In particular, the public key for an uncorrupted party P_i is generated as follows:
  - $\mathsf{enck}_i \leftarrow \xi_i$, while the handlings for each $[\mathsf{dk}_i \cdot \tau^j]_1$ is obtained by querying the oracle $\mathsf{GC}_{\mathsf{MP}}$ to realize the scalar multiplication between ξ_i and τ^j.

  All queries for generic group operations are realized by querying the corresponding oracle provided by $\mathcal{S}_{\mathsf{GGM}}$.
- **Queries**: $\mathcal{O}_b$ is responded by following the Fig.9. When $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Corr}}$ with i, $\mathcal{S}_3$ queries DLog with ξ_i provided by $\mathcal{S}_{\mathsf{GGM}}$. When DLog returns $k_i \in \mathbb{Z}_p$, $\mathcal{S}_3$ returns $\mathsf{dk}_i := k_i$ to $\mathcal{A}$.

Note that under the condition that $\mathcal{S}_{\mathsf{GGM}}$ does not abort, Hybrid 3 is identical to Hybrid 2 at the point of $\mathcal{A}$'s view. Following Lemma 15, the probability that $\mathcal{S}_{\mathsf{GGM}}$ aborts is negligible.

Hybrid 4. This hybrid is almost identical to Hybrid 3, except that in the challenge oracle $\mathcal{O}_b$, the encryption algorithm $\mathsf{Enc}(ek, m_b; \Gamma^*)$ is realized as follows:

$$c = \left(\mathbf{s}^\top \cdot A, R^* + m_b\right),$$

where R^* is a uniformly sampled element in $\mathbb{G}_T$, and A is the matrix A_{tag} while $[\gamma]_2 = H(tag)$ is now replaced with Γ^*.

It is easy to see that in Hybrid 4, the adversary has zero advantage to win this security game. So, it remains to show that any PPT adversary cannot distinguish Hybrid 4 and Hybrid 3.

At this point, the remaining part has been exactly done in [45]. The original proof in [45] focuses on a set of secret variables that are unknown to the adversary, which includes X (the random variable of the CRS trapdoor τ), $\{K_i\}_{i\in\mathcal{H}}$ (the random variables for the decryption keys of honest participants), $(S_1, \ldots, S_5)$ (the random variables for $\mathsf{s} \in \mathbb{Z}_p^5$ used in encryption algorithm), and Γ^* (the random variable for the scalar γ^* of $[\gamma^*]_2$). It is proved that, assuming all secret random variables are uniformly distributed, $[S_5]_T$ (which is $\mathsf{s}^\top \cdot \mathbf{b}$) is *indistinguishable* with R^*, even when the adversary is given the handlings of $[f_1(Y)]_1$ for all $f_1(Y) \in L_1(Y)$, $[f_2(Y)]_2$ for all $f_2(Y) \in L_2(Y)$, and $[1]_T$, where

$$L_1(Y) := \left(1, X, \dots, X^{n+1}, \{K_i, K_iX, \dots, K_iX^n\}_{i\in\mathcal{H}^*}, \underbrace{S_3}_{S^\top a_6}, \sum_{i\in\mathcal{H}} K_iL_i(X) + \sum_{i\in\mathcal{C}} \mathsf{dk}_iL_i(X) + S_4X^t + S_5\right),$$

$$L_2(Y) := \left(1, X, \dots, X^{n+1}, \varGamma^*, \underbrace{S_1 + \varGamma S_3}_{S^\top a_2}, \underbrace{S_1X^{n+1} - S_1}_{S^\top a_3}, \underbrace{S_1X + S_2X}_{S^\top a_4}, \underbrace{S_2}_{S^\top a_5}, \underbrace{S_4}_{S^\top a_7}, \underbrace{S_5X - S_5}_{S^\top a_8}\right).$$

In Hybrid 3 and Hybrid 4, note that the secret random variables are exactly $(X, \{K_i\}_{i\in\mathcal{H}}, \varGamma^*, (S_1, \dots, S_5))$, and all information available to the adversary that is related to those secret random variables can be expressed as $[f_1(Y)]_1$ or $[f_2(Y)]_2$ for some $f_1(Y) \in L_1(Y)$ or $f_2(Y) \in L_2(Y)$. Note that due to Lemma 15, the secret keys of all uncorrupted parties are random variables that have not been assigned concrete scalars in $\mathbb{Z}_p$. Therefore, their distributions are uniform, which cannot be biased by the adversary due to adaptive corruption. The other secret information, which includes $\tau \in \mathbb{Z}_p$, γ^* (the scalar of $\varGamma^*$), and the scalars of $(S_1, \dots, S_5)$, are all uniformly sampled, and the adversary never has access to them. Therefore, the remaining proofs can be done by following [45].

D Analysis of Composition Security

Composition of Align We demonstrate a few facts that are useful for proving security when composing our Align protocol with other protocol instances.

Recall that the PB protocol uses a silent-setup threshold signature scheme STS as a building block. The security game of STS in Fig.8 enables the adversary to query partial signatures on any messages except the challenge message via an oracle $\mathcal{O}_{\mathsf{PartS}}$. We show that exposing this partial signing oracle to the adversary $\mathcal{A}$ against Align will not undermine the security of Align.

Lemma 16. Align[$\mathtt{sid}$] *is secure even when the adversary can access* $\mathbb{O}_{\mathsf{Align}}$*, which consists of the following oralces:*

- $\mathcal{O}_{\widehat{\mathsf{PartS}}}$ *which on input* $(\mathtt{sid}, i, \mathtt{msg})$ *returns* $\mathcal{O}_{\mathsf{PartS}}((\mathtt{sid}', \mathtt{msg}), i)$.
- $\mathcal{O}_{\widehat{\mathsf{Corr}}}$, *which on input i returns the secrets and randomness used to respond to the previous queries with the form of* $(\cdot, i, \cdot)$.

Note that the adversary attacking Align[$\mathtt{sid}$] *can only query* $\mathcal{O}_{\widehat{\mathsf{PartS}}}$ *with* $(\mathtt{sid}', \cdot, \cdot)$ *s.t.* $\mathtt{sid}' \neq \mathtt{sid}$

Proof. Note that the security of Align solely relies on the security of each PB instance, which, in turn, relies on the unforgeability of the underlying STS scheme. Specifically, PB[sid] is secure when the adversary cannot forge an STS signature on a message in the form of $(\mathtt{sid}, \cdot)$. On the other hand, the security game of STS enables the adversary to query $\mathcal{O}_{\mathsf{PartS}}$ on any messages except the challenge message, which, in this case, must be in the form of $(\mathtt{sid}, \cdot)$. Therefore, allowing the adversary to query $\mathcal{O}_{\widehat{\mathsf{PartS}}}$ with messages not in the form of $(\mathtt{sid}, \cdot)$ as well as $\mathcal{O}_{\widehat{\mathsf{Corr}}}$ will not undermine the security of the instance PB[sid]. □

Since the protocol Align is deterministic after the setup phase, it is easy to see that, once the inputs are known, all information an adversary can access in the instance Align[sid] can be generated by itself with the help of $\mathcal{O}_{\widehat{\mathsf{PartS}}}$ and $\mathcal{O}_{\widehat{\mathsf{Corr}}}$. We summarize this observation in the following lemma.

Lemma 17. *Let* $\mathsf{Align.Setup}[sid]$ *be an instance of the setup phase of* Align*. Let* $\mathsf{Align}[\langle sid, sid_0\rangle]$ *be an instance under the setup. Then* $\mathsf{Align}[\langle sid, sid_0\rangle]$ *is* $\mathcal{O}_{\mathsf{Align}}$ *under* $\mathsf{Align.Setup}[sid]$*.*

Next, we show Align remains secure when it is used as a subroutine in $\mathsf{WeakCoin}$.

Proposition 6. $\mathsf{Align}[\langle sid, ali\rangle]$ *remains secure in* $\mathsf{WeakCoin}[sid]$*.*

Proof. It naturally follows Lemma 1. In particular, $\mathsf{WeakCoin}[\mathtt{sid}]$ has no inputs, and all messages and states due to the code outside $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$ are generated without using any information related to $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$, except the outputs of $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$, which, however, are public to the adversary $\mathcal{A}$ against $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{ali}\rangle]$. Therefore, there is a PPT algorithm $\mathcal{S}$, which on input of the view $\mathsf{VIEW}_{\mathcal{A}_{\mathsf{Align}}}$ of $\mathcal{A}_{\mathsf{Align}}$ can output VIEW', such that $(\mathsf{VIEW}_{\mathcal{A}_{\mathsf{Align}}}, \mathsf{VIEW}')$ distributes identically to the entire view of $\mathcal{A}$ in the composition. □

Composition of the weak coin. We now demonstrate a few facts that are useful for proving security when composing our $\mathsf{WeakCoin}$ protocol with other protocol instances. We consider the following oracles:

- $\mathcal{O}_{\widehat{\mathsf{PartD}}}$: On input $(\mathtt{sid}', i)$, if $\mathtt{sid}' \neq \mathtt{sid}$, returns $\mathsf{STHE.PartD}(ek, \mathtt{sid}', dk_i)$
- $\mathcal{O}_{\widehat{\mathsf{Sign}}}$: On input $(\mathtt{sid}', i, \mathtt{msg})$, if $\mathtt{sid}' \neq \mathtt{sid}$, returns $\mathsf{DS.Sign}(sk_i, (\mathtt{sid}', \mathtt{msg}))$.
- Oracles in $\mathbb{O}_{\mathsf{Align}}$, defined in Lemma.16 for the underlying Align.

We denote the set of the above oracles as well as the corruption oracle by $\mathbb{O}_{\mathsf{WeakCoin}}$.

We have proved that our $\mathsf{WeakCoin}$ protocol is secure even when the adversary has access to oracles in $\mathbb{O}_{\mathsf{WeakCoin}}$ in Theorem 2. Now we show an instance of $\mathsf{WeakCoin}$ is $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable under its setup.

Lemma 18. *Let* $\mathsf{WeakCoin.Setup}[sid]$ *be an instance of the setup. Let* $\mathsf{WeakCoin}[\langle sid, sid_0\rangle]$ *be an instance under the setup of* $\mathsf{WeakCoin.Setup}[sid]$*.Then,* $\mathsf{WeakCoin}[\langle sid, sid_0\rangle]$ *is* $\mathbb{O}_{\mathsf{WeakCoin}}$*-emulatable under* $\mathsf{WeakCoin.Setup}[sid]$*.*

Proof. Given the public information from the $\mathsf{WeakCoin.Setup}[\mathtt{sid}]$, in $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$, the Fresh message sent by an honest node P_i can be generated by querying $\mathcal{O}_{\widehat{\mathsf{PartD}}}$ and $\mathcal{O}_{\widehat{\mathsf{Sign}}}$ with messages in the form of $(\langle \mathtt{sid}, \mathtt{sid}_0\rangle, i, \cdot)$. Then, the input of P_i to $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{sid}_0, \mathtt{ali}\rangle]$ is based on the Fresh messages it received, which in turn can be generated by querying $\mathcal{O}_{\widehat{\mathsf{PartD}}}$ and $\mathcal{O}_{\widehat{\mathsf{Sign}}}$ with messages in the form of $(\langle \mathtt{sid}, \mathtt{sid}_0\rangle, i, \cdot)$. Then, as summarized in Lemma 17, the messages due to $\mathsf{Align}[\langle \mathtt{sid}, \mathtt{sid}_0, \mathtt{ali}\rangle]$ can be generated with the help of oracles in $\mathbb{O}_{\mathsf{Align}}$. Similarly, all subsequent messages sent by P_i do not need secret keys of P_i, so they can be generated by a PPT algorithm on the input of previous messages. The internal states leaked to $\mathcal{A}$ due to corruption can also be generated with the help of the corruption oracle in $\mathbb{O}_{\mathsf{WeakCoin}}$. Therefore, $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$ is $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable under $\mathsf{WeakCoin.Setup}[\mathtt{sid}]$.

Composition of MBA. Applying Theorem 2 and Lemma 18, we can prove that $\mathsf{WeakCoin}$ remains secure when it is used as a subroutine in MBA. In particular, we have the following lemma for the security of MBA.

Lemma 19. *Let* $\mathsf{MBA}[\langle sid, ba\rangle]$ *be an instance under the setup* $\mathsf{WeakCoin.Setup}[sid]$*, while the* $\{v_i\}_{i\in\mathcal{H}}$ *is the input set of the initial honest parties. Then, if any instance of* $\mathsf{WeakCoin}$ *under* *the setup of* $\mathsf{WeakCoin.Setup}[\mathit{sid}]$ *is secure even when the adversary has access to* $\{v_i\}_{i\in\mathcal{H}}$*, then* $\mathsf{MBA}[\langle \mathit{sid}, \mathit{ba}\rangle]$ *is secure.*

In addition, the security of $\mathsf{MBA}[\langle \mathit{sid}, \mathit{ba}\rangle]$ *preserved when the adversary has access to oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$ *under the restriction that queries must not start with* $\langle \mathit{sid}, \mathit{ba}\rangle$*, and* $\mathsf{MBA}[\langle \mathit{sid}, \mathit{ba}\rangle]$ *is* $\mathbb{O}_{\mathsf{WeakCoin}}$*-emulatable under* $\mathsf{WeakCoin.Setup}[\mathit{sid}]$*.*

Proof. Let $\mathsf{WeakCoin}[\langle \mathtt{sid}, , \mathtt{sid}_0, \mathtt{sid}'\rangle]$ be an instance inside $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$. We can view $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$ and its setup as a composition of

$$(\mathsf{WeakCoin.Setup}[\mathtt{sid}], \mathsf{WeakCoin}[\langle \mathtt{sid}, , \mathtt{sid}_0, \mathtt{sid}'\rangle])$$

and the "remaining part" that includes the other instances of $\mathsf{WeakCoin}$ inside $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$ and the coin-aided part of $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$. The other instances of $\mathsf{WeakCoin}$ are $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable under $\mathsf{WeakCoin.Setup}[\mathtt{sid}]$, according to Lemma 18. Since the coin-aided part of $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$ is information-theoretical and deterministic, all messages and states w.r.t. this part can be efficiently generated based on the inputs $\{v_i\}_{i\in\mathcal{H}}$ and the output of coin instances. Then, it is easy to see that the "remaining part" is $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable under $\mathsf{WeakCoin.Setup}[\mathtt{sid}]$. On the other hand, under the assumption that $\mathsf{WeakCoin}[\langle \mathtt{sid}, , \mathtt{sid}_0, \mathtt{sid}'\rangle]$ is secure in the stand-alone setting even if the adversary has access to $\{v_i\}_{i\in\mathcal{H}}$ and $\mathbb{O}_{\mathsf{WeakCoin}}$, following Lemma 2, $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{sid}_0\rangle]$ is secure. □

Composition of leader election. Now, we show that all components in our leader election protocol are secure.

Proposition 7. *All instances of* $\mathsf{WeakCoin}$ *and* MBA *inside* $\mathsf{Election}[\mathit{sid}]$ *remain secure.*

Proof. It follows the fact that all $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$ and $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{ba}, k\rangle]$ instances for $k = 1, \ldots$ are $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable, and all honest inputs to each $\mathsf{MBA}[\langle \mathtt{sid}, \mathtt{ba}, k\rangle]$ are outputs of $\mathsf{WeakCoin}[\langle \mathtt{sid}, \mathtt{wc}, k\rangle]$ that are also $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable. Then, as ensured by Theorem 2 and Lemma 19, they all remain secure in the composition. □

The following proposition states useful facts when using $\mathsf{Election}$ as a subroutine.

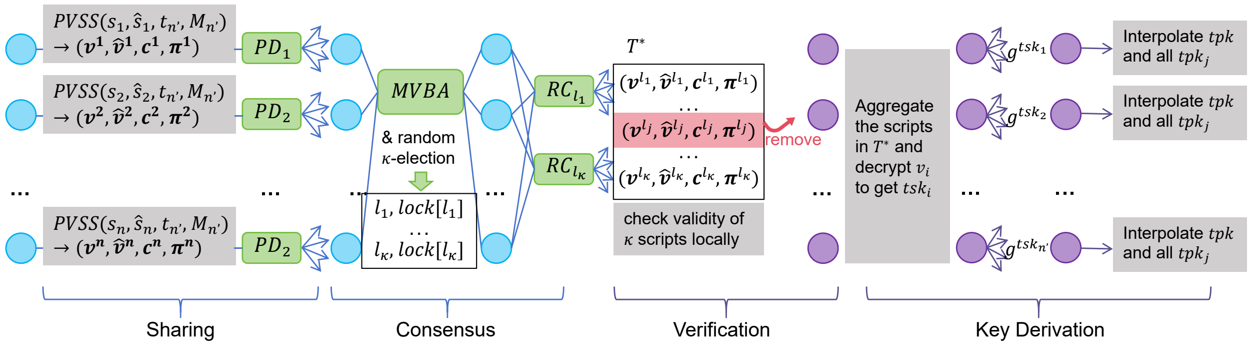
Proposition 8. *The security of* $\mathsf{Election}[\mathit{sid}]$ *preserves even when the adversary has access to oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$ *under the restriction that queries must not start with* sid*, and* $\mathsf{Election}[\mathit{sid}]$ *is* $\mathbb{O}_{\mathsf{WeakCoin}}$*-emulatable under its setup.*

Proof. It trivially follows the security of the underlying components.

Composition of MVBA.

Lemma 20 (MVBA with Silent Setup). *Let* MVBA *be the protocol obtained by instantiating the PB and leader election subroutines in the MVBA protocol of [52] with instances of* PB *(Algorithm 5) and* $\mathsf{Election}$ *(Algorithm 3), respectively. Let* $\mathsf{WeakCoin.Setup}[\mathit{sid}]$ *be an instance of the setup, and let* $\mathsf{MVBA}[\langle \mathit{sid}, \mathit{mvba}\rangle]$ *be an instance under the setup with* $\{v_i\}_{i\in\mathcal{H}}$ *as the inputs of honest nodes. Then, if any instance of* PB *and any instance of* $\mathsf{Election}$ *under the setup of* $\mathsf{WeakCoin.Setup}[\mathit{sid}]$ *are secure even when the adversary has access to* $\{v_i\}_{i\in\mathcal{H}}$ *and oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$*,* $\mathsf{MVBA}[\langle \mathit{sid}, \mathit{mvba}\rangle]$ *is secure.*

In addition, the security of $\mathsf{MVBA}[\langle \mathit{sid}, \mathit{mvba}\rangle]$ *is preserved when the adversary has access to oracles in* $\mathbb{O}_{\mathsf{WeakCoin}}$ *under the restriction that queries must not start with* $\langle \mathit{sid}, \mathit{mvba}\rangle$*, and* $\mathsf{MVBA}[\langle \mathit{sid}, \mathit{mvba}\rangle]$ *is* $\mathbb{O}_{\mathsf{WeakCoin}}$*-emulatable under* $\mathsf{WeakCoin.Setup}[\mathit{sid}]$*.*



$PVSS(s_1, \hat{s}_1, t_{n'}, M_{n'}) \to (v^1, \hat{v}^1, c^1, \pi^1)$
$PVSS(s_2, \hat{s}_2, t_{n'}, M_{n'}) \to (v^2, \hat{v}^2, c^2, \pi^2)$
$PVSS(s_n, \hat{s}_n, t_{n'}, M_{n'}) \to (v^n, \hat{v}^n, c^n, \pi^n)$
PD_1
PD_2
MVBA
& random κ-election
$l_1, lock[l_1]$
$l_\kappa, lock[l_\kappa]$
RC_{l_1}
RC_{l_κ}
T^*
$(v^{l_1}, \hat{v}^{l_1}, c^{l_1}, \pi^{l_1})$
$(v^{l_j}, \hat{v}^{l_j}, c^{l_j}, \pi^{l_j})$
$(v^{l_\kappa}, \hat{v}^{l_\kappa}, c^{l_\kappa}, \pi^{l_\kappa})$
remove
check validity of κ scripts locally
Aggregate the scripts in T^* and decrypt v_i to get tsk_i
g^{tsk_1}
g^{tsk_2}
$g^{tsk_{n'}}$
Interpolate tpk and all tpk_j
Sharing
Consensus
Verification
Key Derivation

Fig. 10. Coin-aided ADKG [36]

Proof. For any $\mathsf{PB}[\langle \mathtt{sid}, \mathtt{mvba}, \mathtt{pb}_0 \rangle]$ instance inside $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$, we can view $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$ as a composition of

$$(\mathsf{WeakCoin.Setup}[\mathtt{sid}], \mathsf{PB}[\langle \mathtt{sid}, \mathtt{mvba}, \mathtt{pb}_0 \rangle])$$

and the remaining part, including other instances of PB and $\mathsf{ELection}$ and others, that are $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable. So, $\mathsf{PB}[\langle \mathtt{sid}, \mathtt{mvba}, \mathtt{pb}_0 \rangle]$ remains secure in the composition, if it is secure when the adversary has access to $\mathbb{O}_{\mathsf{WeakCoin}}$ and $\{v_i\}_{i \in \mathcal{H}}$. Similar arguments apply to any instance of $\mathsf{ELection}$ inside $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$. Hence, $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$ is secure under the assumption. It follows trivially that $\mathsf{MVBA}[\langle \mathtt{sid}, \mathtt{mvba} \rangle]$ is secure even when the adversary has access to $\mathbb{O}_{\mathsf{WeakCoin}}$ and that it is $\mathbb{O}_{\mathsf{WeakCoin}}$-emulatable. □

E Sketch of ADKG Protocol in [36]

The execution flow of the coin-aided ADKG protocol in [36] is in Fig.10.