Concurrent Asynchronous Byzantine Agreement in Expected-Constant Rounds, Revisited

Ran Cohen[1](✉), Pouyan Forghani[2], Juan Garay[2], Rutvik Patel[2], and Vassilis Zikas[3]

[1] Reichman University, Herzliya, Israel
cohenran@runi.ac.il
[2] Texas A&M University, College Station, USA
{pouyan.forghani,garay,rsp7}@tamu.edu
[3] Purdue University, West Lafayette, USA
vzikas@cs.purdue.edu

Abstract. It is well known that without randomization, Byzantine agreement (BA) requires a linear number of rounds in the synchronous setting, while it is flat out impossible in the asynchronous setting. The primitive which allows to bypass the above limitation is known as *oblivious common coin* (OCC). It allows parties to agree with constant probability on a random coin, where agreement is oblivious, i.e., players are not aware whether or not agreement has been achieved.

The starting point of our work is the observation that no known protocol exists for information-theoretic multi-valued OCC with optimal resiliency in the asynchronous setting (with eventual message delivery).

This apparent hole in the literature is particularly problematic, as multi-valued OCC is implicitly or explicitly used in several constructions.

In this paper, we present the first information-theoretic multi-valued OCC protocol in the asynchronous setting with optimal resiliency, i.e., tolerating $t < n/3$ corruptions, thereby filling this important gap. Further, our protocol efficiently implements OCC with an exponential-size domain, a property which is not even achieved by known constructions in the simpler, synchronous setting.

We then turn to the problem of round-preserving parallel composition of asynchronous BA. A protocol for this task was proposed by Ben-Or and El-Yaniv [Distributed Computing '03]. Their construction, however, is flawed in several ways. Thus, as a second contribution, we provide a simpler, more modular protocol for the above task. Finally, and as a contribution of independent interest, we provide proofs in Canetti's Universal Composability framework; this makes our work the first one offering composability guarantees, which are important as BA is a core building block of secure multi-party computation protocols.

The full version of this paper can be found at the *IACR Cryptology ePrint Archive*, report 2023/1003.

G. Rothblum and H. Wee (Eds.): TCC 2023, LNCS 14372, pp. 422–451, 2023.
https://doi.org/10.1007/978-3-031-48624-1_16

1 Introduction

Byzantine agreement (BA) [63,74] enables n parties to reach agreement on one of their inputs in an adversarial setting, facing up to t colluding and cheating parties. The core properties require all honest parties to eventually terminate with the same output (*agreement*), which equals their input value in case they all begin with the same common input (*validity*). BA and its closely related single-sender variant, *broadcast*, are fundamental building blocks in the construction of cryptographic protocols, in particular for *secure multi-party computation* (MPC) [13,28,56,82], in which parties wish to privately compute a joint function over their inputs.

We consider a complete network of authenticated and private point-to-point (P2P) channels, which enables every pair of parties to communicate directly. The central settings in which BA has been studied are:

The *synchronous* setting. Here the protocol proceeds in a round-by-round fashion, and messages sent in a given round are guaranteed to be delivered by the start of the next round. The round structure can be achieved given synchronized clocks and a known bound on message delivery, and enables the use of *timeouts*. This clean abstraction allows for simpler analyses of protocols, but comes at the cost that parties must wait for the worst-case delay in each round before they can proceed.

The *asynchronous* setting. Here no assumptions are made on the clocks or on the bound of message delivery (other than that each message will be eventually delivered), and messages may be adversarially delayed by an arbitrary (yet finite) amount of time. The main challenge is that timeouts cannot be used, implying the inability to distinguish a slow honest party from a silent, noncooperative corrupted party. On the positive side, asynchronous protocols can advance as fast as the network allows, irrespectively of the worst-case delay, and each party can proceed to the next step as soon as it gets sufficiently many messages.

The *feasibility* of BA is inherently related to the synchrony assumptions of the system. Synchronous BA with *perfect* security is achievable with deterministic protocols for $t < n/3$ [15,54,74]; this bound is tight in the plain model[1] even when considering weaker, computational security [17,47,74], but can be overcome with setup assumptions, yielding computationally secure BA [37], and even information-theoretically secure BA [75], for $t < n/2$.[2] On the other hand, deterministic asynchronous BA (A-BA) is impossible even facing a single crash failure [48], and randomized solutions are a necessity. Following Ben-Or [9] and Rabin [76], information-theoretic randomized A-BA has been achieved with $t < n/3$ corruptions [2,18,24], which, as opposed to the synchronous case, is a tight bound even given setup and cryptographic assumptions [19,39].

[1] That is, without setup assumptions and without imposing resource restrictions on the adversary.

[2] The bound $t < n/2$ is tight for BA [49]; however, under the same setup assumptions, broadcast can be solved for any number of corruptions.

Round Complexity of BA. In the synchronous setting, it is known that *deterministic* BA requires $t+1$ rounds [37,46], a bound that is matched by early feasibility results [37,54,74]. It is also known that t-secure randomized BA for $t \in \Theta(n)$ cannot terminate in a *strict* constant number of rounds [26,29,60]; yet, *expected* constant-round protocols have been constructed both for $t < n/3$ in the plain model [45] and for $t < n/2$ in the PKI model [50,61].[3] The latter protocols follow the approach of Rabin [76] and rely on an *oblivious common coin* (OCC), that is, a distributed coin-tossing protocol over a domain V such that the output of every honest party is a common random value $v \in V$ with constant probability p, but with probability $1-p$ is independently and adversarially chosen; the coin toss is "oblivious" since the parties cannot distinguish between a successful coin toss and an adversarial one.[4]

Loosely speaking, round complexity in the asynchronous setting can be defined based on the expected number of times an honest party has to alternate between sending and receiving messages that are "causally related" [36]. In an unpublished manuscript, Feldman [41] generalized the expected-constant-round synchronous protocol of [42] to asynchronous networks with $t < n/4$; at the core of the construction lies a **binary** (asynchronous) OCC protocol which actually has resiliency $\min(t', \lceil n/3 \rceil - 1)$, where t' is the corruption threshold of an *asynchronous verifiable secret sharing* (A-VSS) scheme.[5] Canetti and Rabin [24] constructed A-VSS for $t < n/3$, thereby obtaining A-BA in expected-constant rounds for the optimal threshold $t < n/3$.

Concurrent BA. All of the above constructions are for solving a *single* instance of BA. However, most applications, particularly MPC protocols, require composing multiple instances of BA: sequentially, in parallel, or concurrently. While the composition of synchronous, deterministic protocols is relatively simple (although care must be taken in the cryptographic setting [65]), composing expected-constant-round protocols with probabilistic termination is a much more challenging task.

Indeed, Ben-Or and El-Yaniv [11,12] observed that running m instances of a probabilistic-termination protocol in parallel may incur a blow-up in the expected number of rounds until they all terminate. The technical reason is that the expectation of the maximum of m independent, identically distributed random variables does not necessarily equal the maximum of their expectations. In particular, for expected-constant-round BA with a geometric termination probability (which is the case in all known protocols), the parallel composition of m instances terminates after expected $\Theta(\log m)$ rounds [31]. Further, even when all

[3] Fitzi and Garay [50] devised expected-constant-round BA for $t < n/2$ in the PKI model under number-theoretic assumptions. Katz and Koo [61] established a similar result from the minimal assumption of digital signatures, which yields an information-theoretic variant using pseudo-signatures [75].

[4] This primitive is sometimes known as a "weak" common coin in the literature.

[5] Feldman's A-VSS suffers from a negligible error probability. An *errorless* A-VSS scheme for $t < n/4$ is given in [10] and used to construct a *perfectly* secure asynchronous MPC protocol with resiliency $t < n/4$.

parties start the protocol together, simultaneous termination is not guaranteed as the adversary can force some honest parties to terminate after the others; although this gap can be reduced to a single round, inaccurate sequential re-synchronization of ℓ instances of BA may lead to an exponential blow up in ℓ if not done with care [64]. Following [12,61,64,65], recent works have shown how to compose synchronous BA in a round-preserving way with simulation-based security in Canetti's framework for universal composability (UC) [23], with optimal resiliency and *perfect* security in the plain model for $t < n/3$ [31], and with cryptographic security in the PKI model for $t < n/2$ [32].

In the asynchronous setting, the parties are not assumed to begin the protocol at the same time, so intuitively, sequential composition is not problematic. However, running m instances of expected-constant-round A-BA concurrently would yield a $\Theta(\log m)$ blowup as in the synchronous case. Ben-Or and El-Yaniv [12] further showed how to execute m instances of A-BA in expected-constant rounds and with optimal resiliency $t < n/3$; however, their solution is more complicated than the synchronous one. In addition, they only prove a property-based security definition of A-BA that does not necessarily address modern security requirements such as security under composition, or facing adaptive adversaries.

1.1 Concurrent A-BA in Expected-Constant Rounds: Cracks in the Concrete

The underlying idea behind the synchronous protocol of Ben-Or and El-Yaniv [12] for parallel BA is to execute each BA instance multiple times over the same inputs, but only for a constant number of rounds. For a suitable choice of parameters, this ensures that with high probability each party will have obtained at least one output value in each such batch. To coordinate these outputs, the parties then run an *oblivious leader election* (OLE) protocol, which guarantees that with constant probability, a random leader is elected. In the event that the leader is honest and the parties obtained an output in each batch (which, again, occurs with constant probability) the parties will terminate; otherwise they repeat the process.

The same general technique underlies Ben-Or and El-Yaniv's asynchronous protocol, but great care is needed to deal with the low message dispersion inherent in asynchronous networks while maintaining optimal resiliency $t < n/3$. A closer look at their security proof indeed raises a number of subtle issues. First, they point to Canetti and Rabin [24] for instantiating the (asynchronous) OLE primitive used in their construction (called *A-Election()*). (Recall that Canetti and Rabin construct an OCC to obtain A-BA in expected-constant rounds; an n-valued OCC would indeed imply OLE.) As it turns out, the OCC construction in [24, Sec. 8] (as well as the more detailed versions [22, Sec. 5.7] and [25, Sec. 8]) is only *binary* (i.e., it only works for $V = \{0, 1\}$), and it does not seem straightforward to generalize to larger (non-constant-sized) domains.

Further, running $\log n$ executions of a binary OCC in parallel to make it multi-valued yields only $1/\mathsf{poly}(n)$ probability of agreement, and as long as the coin is not *perfectly* fair (i.e., non-oblivious and unbiased, which is impossible in the asynchronous setting [77]), that would not imply OLE with constant success probability. We note that Patra *et al.* [73] claim to construct a $(t+1)$-bit asynchronous OCC, but their main focus is on communication complexity, and the agreement probability of their protocol is no better than would be obtained by running $t+1$ executions of a binary OCC protocol (i.e., exponentially small). Techniques used to get OLE in the synchronous setting [42,61] do not seem to extend in asynchrony for $t < n/3$ (we elaborate on this in Sect. 1.2). Thus, to the best of our knowledge, no existing OCC construction is *simultaneously* optimally resilient, multi-valued, and asynchronous, without relying on computational assumptions (in Sect. 1.3 we discuss solutions in the cryptographic setting).

Second, there is a subtle issue in the logic of one of the proofs in [12]. This issue raises concerns about the validity of the proof claiming an expected-constant round complexity of one of the main subroutines—namely, the Π_{select} subroutine, which handles the shortcomings of their message-distribution mechanism. Specifically, in the analysis of Π_{select} it is claimed that if the leader is chosen from a certain set, the protocol will terminate. However, further examination reveals that there are scenarios in which the protocol may not terminate for certain leaders from that set. As a result, this issue casts doubt on the promised expected-constant round complexity of their concurrent A-BA protocol.

Finally, the concurrent asynchronous (resp., synchronous) BA protocol in [12] relies on *multi-valued* asynchronous (resp., synchronous) BA in expected-constant rounds.[6] Recall that running the binary protocols in [45] or [24] $\omega(1)$ times in parallel would terminate in expected $\omega(1)$ rounds, so they cannot be naïvely used for this task. In the synchronous setting, Turpin and Coan [79] extended binary BA to multi-valued BA for $t < n/3$ with an overhead of just two rounds. Ben-Or and El-Yaniv claim that this technique can be adapted for asynchronous networks by using Bracha's "A-Cast" primitive [18] for message distribution. However, a closer look reveals that although the Turpin-Coan extension works (with appropriate modifications) in the asynchronous setting for $t < n/5$, it **provably does not work** when $t \geq n/5$, regardless of the specific choice of the underlying binary A-BA protocol and even when the adversary is limited to static corruptions (see further discussion in the full version of this paper [33]).

More recently, an optimally resilient multi-valued A-BA protocol with expected-constant round complexity was proposed by Patra [72], but it relies on the *Agreement on a Common Subset* (ACS) protocol in [14]; however, as we explain below, this ACS protocol does not achieve expected-constant round complexity without some modifications, which require either expected-constant-round concurrent A-BA (this would be circular), or information-theoretic asynchronous OLE with optimal resiliency. Fortunately, Mostéfaoui and Raynal [70] recently gave a black-box, constant-round reduction from multi-valued to binary

[6] This is true even if one is interested only in *binary* concurrent BA (i.e., when the input vectors consist of bits). Multi-valued BA is needed to agree on the leader's output vector.

A-BA for $t < n/3$,[7] using just one invocation of the underlying binary protocol as in [79].

We emphasize that the asynchronous protocol of Ben-Or and El-Yaniv [11,12] lies, either explicitly or implicitly, at the core of virtually every round-efficient asynchronous MPC construction [8,10,14,16,30,36,57,58,66]. The concerns raised above regarding the result of [12] render this extensive follow-up work unsound. In this paper, we revisit this seminal result and rectify these issues.

1.2 Overview of Our Results

We now present an overview of our results, which are three-fold, including a detailed exposition of our techniques.

Multi-valued and Asynchronous Oblivious Common Coin. As a starting point, we look at the *binary* asynchronous OCC protocol of Canetti and Rabin [24]. The idea (following the approach of [45] in the synchronous setting and [41] in the asynchronous setting) is that each party secret-shares a random vote for each party, using an optimally resilient A-VSS scheme. Each party accepts $t + 1$ of the votes cast for him (at least one of which must have come from an honest party), and once it is determined that enough secrets have been fixed (based on several rounds of message exchange, using Bracha's A-Cast primitive [18] for message distribution), the parties begin reconstructing the accepted votes. The sum or "tally" of these votes becomes the value associated with the corresponding party. After computing the values associated with an appropriate set of at least $n - t$ parties, an honest party outputs 0 if at least one of those tallies is 0, and outputs 1 otherwise. Note that each tally must be uniformly random, and the properties of A-Cast guarantee that no two honest parties can disagree on the value of any given tally (although up to t tallies may be "missing" in an honest party's local view); moreover, Canetti and Rabin show using a counting argument that at least $n/3$ of the tallies are known to all honest parties (i.e., common to their local views). This can be used to show that with probability at least 1/4 all honest parties output 0, and similarly for 1.

In the conference version of Feldman and Micali's paper [42], there is a brief remark suggesting that the synchronous version of the above protocol can be modified to obtain (oblivious) leader election.[8] Rather than outputting a bit, parties output the index of the party whose tally is minimum; when the domain of secrets is large enough, with constant probability the same *honest* party is chosen. This approach was fully materialized by Katz and Koo [61] for $t < n/3$ in the information-theoretic setting and $t < n/2$ in the computational setting.

[7] They are also concerned with obtaining $O(n^2)$ message complexity. The novelty of their result, even without this more stringent requirement, does not seem to be acknowledged in the paper.

[8] This claim no longer appears in the ICALP [43] or journal [45] versions of the paper, or in Feldman's thesis [44].

We note that what is obtained is not exactly OLE as we have defined it, as the adversary can of course bias the index of the elected party, but nevertheless, this is sufficient for the concurrent BA protocols of Ben-Or and El-Yaniv [12].

Unfortunately, the approach effective in the synchronous setting cannot be directly applied to the asynchronous setting while maintaining optimal resiliency. The challenge arises from the fact that the adversary can selectively remove t coordinates from the honest parties' local views of the tallies, ensuring that the leader is not chosen from the parties corresponding to those missing coordinates. This poses a significant obstacle as the original concurrent A-BA protocol by Ben-Or and El-Yaniv [12] terminates successfully when the leader is honest and selected from an adversarially chosen subset of $n - 2t$ parties. Consequently, when $t < n/3$, the size of this subset is only $t + 1$, allowing the adversary to reduce the probability of choosing an appropriate leader to as low as $\frac{1}{n-t}$. The same issue arises in our simplified concurrent A-BA protocol, where we also require the leader to be selected from an adversarially chosen subset of parties, but with size greater than $n/3$. Again, when $t < n/3$, this subset can be of size $t + 1$, leading to the same challenge. However, when $t < n/4$, the set of potential appropriate leaders becomes larger in both concurrent A-BA protocols, enabling us to circumvent this issue. Therefore, in addition to the inherent value in obtaining a true OCC (where a successful coin toss produces a uniform value), we specifically need such a primitive to obtain optimal resiliency for concurrent A-BA.

In conclusion, no OLE construction—and therefore no concurrent A-BA protocol in expected-constant rounds—exists with optimal resiliency $t < n/3$ in asynchronous networks. We now describe our solution to this problem, which is based on the following simple combinatorial observation. If we work over a field of size $N \in \Theta(n^2)$, then with *constant* probability, at least one value will be repeated in the global view of tallies, and, moreover, all repeats will occur within the subset of indices known to all honest parties. The intuition for this fact is that when $N \in O(n^2)$ there is, due to the birthday paradox, at least one repeat in any constant fraction of the indices with constant probability, and when $N \in \Omega(n^2)$, there are no repeats at all with constant probability. There is a "sweet spot" between these extremes that can be leveraged to extract shared randomness: Honest parties output the value that is repeated in their local views of the tallies and has the minimum index in the vector of tallies. With this modification of Canetti and Rabin's protocol [24], we obtain the first n^2-valued asynchronous OCC for $t < n/3$ in the information-theoretic setting. We remark that the above combinatorial observation at the heart of our construction may be of independent interest.

It is straightforward to extend our OCC protocol to accommodate arbitrary domains V. One approach is to work over a prime field of size at least $\text{lcm}(n^2, |V|)$. In this setting, we can still identify "repeats" by considering the tallies modulo n^2. Once the repeat with minimum index is determined, we can generate a common random output by reducing the corresponding (original) field element modulo $|V|$. In particular, when $|V| = n$, we get asynchronous OLE with optimal resiliency.

Proving the security of our multi-valued OCC protocol in a *simulation-based* manner is not without its own challenges. The issue is that the simulator must expose a view of the vector of tallies that both adheres to the distribution in the real world and is consistent with the random value chosen by the OCC functionality (in the case of a successful coin toss) in the ideal world. While the simulator can easily determine the exact set of indices known to all honest parties from its internal execution with the real-world adversary, properly sampling the "repeat pattern" according to these constraints is a delicate task; furthermore, since the functionality is only parameterized by a (constant) lower bound on the probability of a successful coin toss, the simulator must handle the complementary case carefully in order to avoid skewing the distribution. It is not immediately clear how to perform this inverse sampling efficiently.

A heavy-handed solution is to simply have the functionality sample the tallies itself, and then determine the output based on the location of repeats relative to the subset of indices (supplied by the simulator) that are known to all honest parties. This protocol-like functionality would certainly allow for simulation—and would in fact be sufficient for our purposes—but its guarantees are more difficult to reason about and, more importantly, it cannot be realized by other protocols! Instead, we construct a simulator that, given the exact probabilities of certain events in the protocol, can also efficiently sample from those events, potentially conditioned on the output of a successful coin toss. By selecting the appropriate sampling procedure, the simulator can derive a vector of tallies that preserves the (perfect) indistinguishability of the real and ideal worlds. Equipped with the means to carry out the inverse sampling, we can now realize a more abstract (and natural) OCC functionality, which is ready to be plugged into higher-level protocols.

Simplified Concurrent A-BA. Ben-Or and El-Yaniv [12] devised an expected-constant-round concurrent A-BA protocol. However, in addition to relying on unspecified building blocks, as mentioned above, it suffers from logical issues in its proof. Although our new OCC protocol can instantiate the missing primitive, doubts remain about the expected-constant round complexity due to a lingering issue in the proof. This issue stems from the steps taken to address low message dispersion, and the possibility of resolving it without changing the protocol is unclear. To tackle this, we redesign the message-distribution phase, avoiding the problem in the proof and obtaining stronger guarantees. These guarantees simplify the protocol structure, achieving a level of simplicity comparable to the synchronous solution.

In more detail, we build on Ben-Or and El-Yaniv's work, where parties initiate multiple executions for each A-BA instance that are "truncated" after a fixed number of iterations. However, we adopt a different approach to message distribution, allowing us to establish the rest of the protocol based on the simpler structure of their synchronous solution. Similarly to [12], we set parameters to ensure that each party receives at least one output from each batch of truncated A-BA executions (for each instance) with constant probability. Based on their local results from those truncated A-BAs, parties create suggestions for the final output. Subsequently, the message-distribution phase begins with parties running a binary A-BA to verify a precondition, ensuring the ability to validate each other's suggestions. If they choose to proceed, they A-Cast their suggestions and only accept those they can validate using their local results from the truncated A-BAs. This validation relies on the property that honest parties terminate within two consecutive iterations in each A-BA execution, ensuring the validity of suggestions even when provided by corrupted parties. Once parties receive $n - t$ suggestions, they proceed by A-Casting the set of the first $n - t$ suggestions they receive, along with the identity of their providers. The checked precondition guarantees that everyone can move on to the next step.

Parties then wait until they receive enough suggestions and enough sets, ensuring that at least $n - t$ sets are completely contained within the set of suggestions they received. By employing a counting argument similar to [24, 41], we can ensure the delivery of suggestions from at least $n/3$ parties to all honest participants. Moreover, the validation process applied to the suggestions guarantees that each honest party only accepts valid suggestions, even if they originate from corrupted sources. These robust guarantees in the distribution of suggested outputs establish that the $n/3$ suggestions commonly received by honest parties are legitimate outputs. This enables us to directly utilize the synchronous protocol in the asynchronous setting.

We employ our new asynchronous OCC protocol to elect a leader for party coordination. Every party adopts the leader's suggestion and runs a multi-valued A-BA to handle the obliviousness of the leader-election mechanism. If the leader is chosen from those $n/3$ commonly received suggestions, all honest parties initiate the A-BA protocol with the leader's suggestion and output the same value. If the leader is not among those $n/3$ parties, precautions are taken to ensure no malicious value is output. For this purpose, we utilize an "intrusion-tolerant" A-BA protocol that guarantees the output to be either a default value or one of the honest parties' inputs.[9] Finally, parties run a binary A-BA to determine if they have reached consensus on a non-default value and terminate.

By following the above approach, we overcome the issues in the proof and achieve a significantly simpler protocol structure for the asynchronous setting compared to the one presented in [12]. Somewhat surprisingly, the resulting protocol is conceptually as simple as its synchronous counterpart.

Applications to Asynchronous MPC. Asynchronous MPC crucially relies on ACS for determining the set of input providers [10]; this task commonly boils down to concurrently executing n instances of A-BA. Our expected-constant-round concurrent A-BA protocol can be directly plugged into the asynchronous MPC protocols in [8,10,30,57,58,66], preserving their (expected) round complexity.

[9] Ben-Or and El-Yaniv [12] introduced and used a strengthened property for (A-)BA without naming it, which was later called "non-intrusion" validity in [70]. Non-intrusion validity lies between standard validity and "strong" validity [50], as it requires that a value decided by an honest party is either an honest party's input or a special symbol $\perp$ (i.e., the adversary cannot intrude malicious values into the output).

However, despite folklore belief, concurrent A-BA cannot be used in a black-box way in the BKR protocol [14] to achieve asynchronous MPC with round complexity linear in the depth of the circuit. The issue (as pointed out in [1,83]) is that the ACS protocol outlined in [14] assigns input values for certain A-BA instances based on the outputs of other instances. This problem also affects other works like [16,36] that rely on [14].

Fortunately, this issue can be readily addressed by modifying the ACS protocol from BCG [10] (which is secure for $t < n/3$). Recall that this protocol involves a preprocessing step for distributing input shares before initiating concurrent A-BA, which necessitates $O(\log n)$ rounds. Replacing the $O(\log n)$-round preprocessing step with the constant-round "gather" protocol described in [4]-an enhanced version of Canetti and Rabin's counting argument [24]-results in an ACS protocol with constant round complexity. This modified ACS protocol can be seamlessly integrated into both [10] and [14], effectively rendering their round complexity independent of the number of parties. Alternatively, one can leverage the expected-constant-round ACS protocol recently proposed by Abraham *et al.* [1], or the one by Duan *et al.* [38] (instantiating all building blocks with information-theoretic realizations; in particular, the assumption of a "Rabin dealer" [76], used for leader election, can be replaced by our multi-valued OCC).

Composable Treatment of Expected-Constant-Round Concurrent A-BA. We choose to work in Canetti's Universal Composability (UC) framework [23], and as such, we prove the security of our protocols in a *simulation-based* manner. The UC framework provides strong composability guarantees when secure protocols are run as a subroutine in higher-level protocols (this is absolutely critical in our context given that expected-constant-round concurrent A-BA is a key building block in many round-efficient cryptographic protocols, as mentioned above), and even in *a priori* unknown or highly adversarial environments (such as asynchronous networks). Moreover, it enables us to provide a modular, bottom-up security analysis. However, obtaining a composable and round-preserving treatment of "probabilistic-termination" BA is non-trivial, as pointed out by Cohen *et al.* [31,32] in the synchronous setting. In the following, we discuss some unique issues in asynchrony and how we address them.

To model eventual message delivery, we follow [30,36,62] and require parties to repeatedly attempt fetching messages from the network. The first D such requests are ignored by the functionality, where D is a value provided by the adversary in unary so that it remains bounded by the adversary's running time (i.e., so that messages cannot be delayed *indefinitely*). It is straightforward then to derive a formal notion of asynchronous rounds in UC, based on this mechanism. We remark that unlike in the synchronous setting [62], (asynchronous) rounds cannot be used by the environment to distinguish the real and ideal worlds in the asynchronous setting. Thus, as opposed to [31,32], our functionalities are round-unaware. Similarly, we do not need to deal with standard issues in sequential composition, namely, non-simultaneous start/termination ("slack"), since asynchronous protocols are already robust to slack. Indeed, it is entirely possible that some parties receive output from a (secure) asynchronous protocol before other parties have even started the protocol!

On the other hand, the issue of *input incompleteness* is trickier to address. This refers to the problem that in the asynchronous setting, the inputs of up to t honest parties may not be considered in the result of the computation; the remaining $n-t$ parties form a "core set" of input providers. Note that in the worst case, the core set is adversarially chosen and includes all corrupted parties. Prior work [30,36] allowed the adversary to send an explicit core set to the functionality; however, this approach does not always accurately model what happens in the real world, and can cause difficulties in the simulation. Instead, our solution is to allow the adversary to define the core set *implicitly*, by delaying the submission of inputs to the functionality in the same way that it delays the release of outputs from the functionality. Using this novel modeling approach, we obtain updated functionalities for some standard asynchronous primitives that more accurately capture realizable security guarantees.

1.3 Additional Related Work

As mentioned earlier, the $t+1$ lower bounds for deterministic BA [37,46] were extended to rule out *strict*-constant-round t-secure randomized BA for $t = \Theta(n)$ [26,29,60]; these bounds show that any such r-round BA must fail with probability at least $(c \cdot r)^{-r}$ for a constant c, a result that is matched by the protocol of [55]. Cohen *et al.* [35] showed that for $t > n/3$, two-round BA are unlikely to reach agreement with constant probability, implying that the *expected* round complexity must be larger; this essentially matches Micali's BA [68] that terminates in three rounds with probability 1/3. Attiya and Censor-Hillel [6] extended the results on worst-case round complexity for $t = \Theta(n)$ from [29,60] to the asynchronous setting, showing that any r-round A-BA must fail with probability $1/c^r$ for some constant c.

In the dishonest-majority setting, expected-constant-round broadcast protocols were initially studied by Garay *et al.* [53], who established feasibility for $t = n/2 + O(1)$ as well as a negative result. A line of work [27,51,78,80,81] established expected-constant-round broadcast for any constant fraction of corruptions under cryptographic assumptions.

Synchronous and (binary) asynchronous OCC protocols in the information-theoretic setting were discussed earlier. Using the synchronous protocol in [12], Micali and Rabin [69] showed how to realize a *perfectly unbiased* common coin in expected-constant rounds for $t < n/3$ over secure channels (recall that this task is impossible in asynchronous networks [77]). In the cryptographic setting, both synchronous and asynchronous OCC protocols with optimal resiliency are known, relying on various computational assumptions; we mention a few here. Beaver and So [7] gave two protocols tolerating $t < n/2$ corruptions in synchronous networks, which are secure under the quadratic residuosity assumption and the hardness of factoring, respectively. Cachin *et al.* [21] presented two protocols for $t < n/3$ and asynchronous networks, which are secure in the random oracle model based on the RSA and Diffie-Hellman assumptions, respectively.

Nielsen [71] showed how to eliminate the random oracle and construct an asynchronous OCC protocol relying on standard assumptions alone (RSA and DDH). Although these constructions are for asynchronous networks, they can be readily extended to work in synchronous networks for $t < n/2$ (e.g., so that they can be used in the computationally secure BA protocol from [50]). We also note that while the resiliency bounds for asynchronous OCC protocols coincide in the information-theoretic and computational settings, working in the latter typically yields expected-constant-round A-BA protocols that are much more efficient in terms of communication complexity (i.e., *interaction*) than the unconditionally secure protocol of Canetti and Rabin [24].

Lastly, we discuss solutions to multi-valued A-BA in the computational setting. Cachin *et al.* [20] studied a more general version of this problem, in which the validity property is replaced with "external" validity, where the output domain can be arbitrarily large but the agreed-upon value must only satisfy an application-specific predicate. They gave a construction for multi-valued "validated" A-BA that runs in expected-constant rounds for $t < n/3$, assuming a PKI and a number of threshold cryptographic primitives (including an asynchronous OCC), and used it to obtain an efficient protocol for asynchronous *atomic* broadcast. Recently, Abraham *et al.* [5] (and follow-ups, e.g., [4,52,67]) have improved the communication complexity. The work of Fitzi and Garay [50] also considered *strong* (A-)BA. Here we require "strong" validity: the common output must have been one of the honest parties' inputs (note that this is equivalent to standard validity in the binary case). When the size of the input domain is $m > 2$, neither a Turpin-Coan-style reduction [79] nor the obvious approach of running $\log m$ parallel executions of a binary protocol would suffice to realize this stronger notion of agreement; indeed, Fitzi and Garay showed that strong A-BA is possible if and only if $t < n/(m+1)$. This bound holds in both the information-theoretic and computational settings. Their unconditionally secure asynchronous protocol actually involves oblivious coin flipping on the domain, but $m > 2$ forces $t < n/4$ and the binary asynchronous OCC protocols in [24,41] can be extended to multi-valued in this regime, as discussed in Sect. 1.2.

In the next section, we start with some preliminaries. Due to space constraints, we refer the reader to the full version [33] for proofs and other details.

2 Model and Preliminaries

For $m \in \mathbb{N}$, we use $[m]$ to denote the set $\{1, \ldots, m\}$.

We prove our constructions secure in the UC framework [23], with which we assume the reader has some familiarity. However, the base communication model in UC is completely unprotected. To capture asynchronous networks with eventual message delivery, we work in a hybrid model with access to the multi-use *asynchronous secure message transmission* functionality $\mathcal{F}_{\mathsf{a\text{-}smt}}$, which was introduced in [36] and is itself based on the (single-use) *eventual-delivery secure channel* functionality $\mathcal{F}_{\mathsf{ed\text{-}sec}}$ from [62]. In the full version [33], we include a formal specification of $\mathcal{F}_{\mathsf{a\text{-}smt}}$.

The functionality $\mathcal{F}_{\textsf{a-smt}}$ models a *secure* eventual-delivery channel between a sender P_s and receiver P_r.[10] To reflect the adversary's ability to delay the message by an arbitrary finite duration (even when P_s and P_r are not corrupted), the functionality operates in a "pull" mode, managing a message buffer M and a counter D that represents the current message delay. This counter is decremented every time P_r tries to fetch a message, which is ultimately sent once the counter hits 0. The adversary can at any time provide an additional integer delay T, and if it wishes to immediately release the messages, it needs only to submit a large negative value. It is important to note that T must be encoded in unary; this ensures that the delay, while arbitrary, remains bounded by the adversary's computational resources or running time. Also, note that $\mathcal{F}_{\textsf{a-smt}}$ guarantees eventual message delivery assuming that the environment gives sufficient resources to the protocol, i.e., activates P_r sufficiently many times.

Finally, we note that while the UC framework has no notion of time, and we are in the asynchronous setting (with eventual message delivery) where parties may proceed at different rates, one can still formally define a notion of asynchronous rounds along the lines of [36], which will be referred to in our statements. We defer an in-depth discussion to the full version [33], as there are some subtleties that need to be addressed.

3 Ideal Functionalities for a Few Standard Primitives

In this section, we present ideal functionalities for asynchronous primitives used in our constructions. This seemingly simple task requires careful consideration of certain aspects, as the adversary's ability to delay messages in the network has an upstream impact on the achievable security guarantees of distributed tasks. In particular, the adversary can obstruct the output release procedure and impede honest parties' participation. To model *delayed output release*, we extend the mechanism discussed in Sect. 2 (using per-party delay counters). However, to model *delayed participation*, we introduce a novel and more natural approach that addresses limitations of prior work and more closely captures the effects of asynchrony.

3.1 Modeling Delayed Participation

In the asynchronous setting, honest parties cannot distinguish whether uncooperative parties are corrupted and intentionally withholding messages, or if they are honest parties whose messages have been delayed. Consequently, when the number of corruptions is upper-bounded by t, waiting for the participation of the last t parties can result in an indefinite wait. In ideal functionalities, this translates to expecting participation and/or input only from a subset of adversarially chosen parties, known as the "core set," with a size of $n - t$. Observe

[10] Recall that while (concurrent) A-BA is not a private task, secure channels are needed to construct an OCC.

that the value of t is closely related to the behavior of the functionality. In this work, we specifically consider optimal resiliency, where $t = \lceil \frac{n}{3} \rceil - 1$. However, our functionalities can be easily adapted to accommodate other resiliency bounds.

In our modeling approach, the adversary implicitly determines the core set by strategically delaying participation (or input submission) to the functionality. If we instruct the ideal functionality to proceed once $n-t$ parties have participated, the adversary can precisely determine the core set by manipulating the order of participation (or input submissions). To accommodate arbitrary but finite delays for input submissions, we employ a technique similar to the one we use for the output-release mechanism. That is, in addition to a per-party *output delay counter*, there is an *input delay counter* (updatable by the adversary) which is decremented every time the party pings the functionality; once the counter reaches zero, the party is allowed to participate.

This approach contrasts with the work of Cohen [30] and Coretti *et al.* [36], wherein the adversary (simulator) **explicitly** sends a core set to the functionality. Obtaining the core set from the adversary all at once does not accurately mimic real-world executions and requires careful consideration to ensure the implementation works in all scenarios. For instance, a challenging case to model is when the simulator sets the core set, but the environment never activates some parties in that core set. If not handled properly, this situation can lead to either the ideal functionality stalling indefinitely while the real-world execution proceeds, or allowing for core sets of smaller sizes, neither of which is acceptable.

There are other aspects of modeling the core set that can potentially cause issues. If the simulator is required to set the core set early on, it may encounter issues during the simulation because in some real-world protocols, such as the asynchronous MPC protocol of Ben-Or *et al.* [14], the core set is not fixed in the early stages of the execution. Similarly, if the functionality allows late submission of the core set, then when using the functionality as a hybrid, the adversary can potentially stall the functionality unless appropriate preventive mechanisms are in place. In contrast, implicitly defining the core set by delaying parties' participation aligns more closely with real-world executions, reducing the probability of errors. Additionally, it can potentially simplify the simulation process, as the simulator can gradually define the core set as the protocol progresses.

3.2 Ideal Functionalities for a Few Standard Primitives

We now cast a few standard asynchronous primitives as UC functionalities, following our novel modeling approach. We also present security statements showing how classical protocols can be used to realize these primitives.

Asynchronous Broadcast (A-Cast). The first essential primitive used in both our OCC and concurrent A-BA protocols, which also finds numerous applications in other asynchronous protocols, is *Bracha's Asynchronous Broadcast* (A-Cast) [18]. A-Cast enables a distinguished sender to distribute its input, such that if an honest party outputs a value, then all honest parties must (eventually) output the same value. Moreover, if the sender is honest, then all honest parties must (eventually) output the sender's input. While these are essentially the agreement and validity properties required from regular (synchronous) broadcast, we stress that honest parties may not terminate when the sender is corrupted. We formulate A-Cast as the ideal functionality $\mathcal{F}_{\mathsf{a\text{-}cast}}$, presented in the full version [33].

Note that although A-Cast is a single-sender primitive, assuming it can proceed to the output generation phase without sufficient participation from other parties is too idealized. A realizable functionality should only proceed to the output generation phase when $n - t$ parties (which may include the sender) have participated. Parties demonstrate their participation by sending (dummy) input messages, followed by issuing fetch requests to the functionality. An important technicality here is that the participation of parties before the sender initiates the session should not contribute to the count. Therefore, $\mathcal{F}_{\mathsf{a\text{-}cast}}$ starts a session only once the input from the sender is received. Consequently, any efforts for participation before that point will not be taken into account. An implication of this design choice when using $\mathcal{F}_{\mathsf{a\text{-}cast}}$ as a hybrid is that parties other than the sender will not know when the session has started. As a result, they should constantly switch between sending input messages and fetch requests until they receive the output.

Bracha's asynchronous broadcast protocol [18] can be used to UC-realize $\mathcal{F}_{\mathsf{a\text{-}cast}}$ with perfect security:

Proposition 1. *$\mathcal{F}_{\mathsf{a\text{-}cast}}$ can be UC-realized with perfect security in the $\mathcal{F}_{\mathsf{a\text{-}smt}}$-hybrid model, in constant rounds and against an adaptive and malicious t-adversary, provided $t < \frac{n}{3}$.*

Asynchronous Verifiable Secret Sharing (A-VSS). Another crucial primitive we require, mainly for our OCC protocol, is *Asynchronous Verifiable Secret Sharing* (A-VSS). A-VSS allows a dealer to secret-share a value among all parties, ensuring that no unauthorized subset of colluding parties can learn any information about the secret. However, any authorized subset of parties should be able to efficiently reconstruct the secret using their shares. The term "verifiable" reflects that the dealer cannot cheat, for example by causing the reconstruction to fail or by inducing inconsistent output values from honest parties. In particular, whenever the sharing phase succeeds, any authorized subset of parties should be able to efficiently complete the reconstruction phase, and all honest parties doing so must recover the same secret. Moreover, if the dealer is honest, the sharing phase must always succeed, and everyone should recover the value originally shared by the dealer. In our context, we consider only the threshold access structure, where a subset of parties can recover the secret if and only if it contains at least $1/3$ of the parties. The formulation of A-VSS as an ideal functionality, $\mathcal{F}_{\mathsf{a\text{-}vss}}$, can be found in the full version [33].

A-VSS, being a single-sender primitive, also requires the participation of at least $n - t$ parties to be realizable. We adopt a similar approach to $\mathcal{F}_{\mathsf{a\text{-}cast}}$ in modeling this requirement. Parties other than the dealer demonstrate their participation by sending (dummy) input messages and issuing fetch requests, ensuring that the minimum participation threshold is met for the A-VSS protocol to proceed. Also, it is important to note that $\mathcal{F}_{\text{a-vss}}$ only initiates a session once it receives the first input from the dealer. Any participation efforts made before that point are not taken into account.

The A-VSS protocol given by Canetti and Rabin in [24] can be used to UC-realize $\mathcal{F}_{\text{a-vss}}$ with statistical security for $t < n/3$. This result is formally stated in the following proposition. It is worth noting that perfectly secure A-VSS is impossible for $t \geq n/4$ [3,14].

Proposition 2. *For any finite field $\mathbb{F}$ (with $|\mathbb{F}| > n$), $\mathcal{F}^{\mathbb{F}}_{\text{a-vss}}$ can be UC-realized with statistical security, in constant rounds and in the $\mathcal{F}_{\text{a-smt}}$-hybrid model against an adaptive and malicious t-adversary, provided $t < \frac{n}{3}$.*

Asynchronous Byzantine Agreement (A-BA). We use A-BA for both binary and multi-valued domains V in our revised concurrent A-BA protocol. In this primitive, each party P_i has an input $v_i \in V$. The goal is for all parties to output the same value, such that if $n - 2t$ input values are the same, that value is chosen as the output; otherwise, the adversary determines the output. We initially consider *corruption-unfair* A-BA, where the adversary learns the input of each party the moment it is provided. Corruption fairness, as introduced in [59] and later coined in [34], essentially ensures that the (adaptive) adversary cannot corrupt a party and subsequently influence the input value of that party based on the original input value. It is worth noting that corruption-fair A-BA can easily be defined by avoiding leaking honest parties' inputs before the output is generated. We formulate A-BA as the ideal functionality $\mathcal{F}_{\text{a-ba}}$, presented in the full version [33].

The functionality $\mathcal{F}_{\text{a-ba}}$ encompasses an additional property known as "non-intrusion" validity (see Sect. 1.2). In a nutshell, this property guarantees that no malicious value can be present in the output. In other words, the output must be either an honest party's input or a default value $\perp$. This stronger notion of A-BA is vital for the security of our concurrent A-BA protocol.

The expected-constant-round binary A-BA protocol of Canetti and Rabin [24] can be used to UC-realize $\mathcal{F}^{V}_{\text{a-ba}}$ with statistical security for binary domains ($|V| = 2$) and $t < n/3$. However, our concurrent A-BA protocol (and the one in [12]) require **multi-valued** A-BA, where $|V|$ is not constant (in fact, exponential), in expected-constant rounds. Ben-Or and El-Yaniv [12] claim that the constant-round reduction of multi-valued to binary BA proposed by Turpin and Coan [79], which works in the synchronous setting for $t < n/3$, can be extended to work in the asynchronous setting by using A-Cast for message distribution. However, in the full version of the paper [33], we demonstrate that the asynchronous version of this reduction works if and only if $t < n/6$, even when using A-Cast and considering a static adversary. Some additional modifications can improve this bound to $t < n/5$, but achieving optimal resiliency is not straightforward.

More recently, Mostéfaoui and Raynal [70] presented a constant-round transformation from binary to multi-valued A-BA that works for $t < n/3$. Furthermore, the resulting protocol satisfies the non-intrusion validity property mentioned above. By applying this transformation to the binary A-BA protocol of Canetti and Rabin [24], we can UC-realize $\mathcal{F}_{\mathsf{a\text{-}ba}}^{V}$ for arbitrary V with statistical security:

Proposition 3. *For any domain V, $\mathcal{F}_{\mathsf{a\text{-}ba}}^{V}$ can be UC-realized with statistical security in the $\mathcal{F}_{\mathsf{a\text{-}smt}}$-hybrid model, in expected-constant rounds and against an adaptive and malicious t-adversary, provided $t < \frac{n}{3}$.*

4 Asynchronous Oblivious Common Coin

As highlighted in the Introduction, none of the existing OCC proposals is simultaneously information-theoretic, asynchronous, multi-valued, and optimally resilient. Furthermore, no straightforward adaptation of the existing schemes yields an OCC with all of these properties. In this section, we propose our own OCC protocol that aims to satisfy all of these properties. Recall that we are primarily interested in the case where the output domain has size equal to the number of parties; this can be used for asynchronous OLE (defined in Sect. 5), which sets the stage for concurrent A-BA in expected-constant rounds.

At a high level, our protocol is based on the binary OCC of Feldman [41] and Canetti and Rabin [24], and incorporates a novel combinatorial technique derived from our observation stated in Lemma 1 below. By leveraging this lemma, we unveil interesting and powerful properties of the local views formed during the protocol's execution, leading to enhanced extraction capabilities. In fact, instead of extracting a single bit, by choosing appropriate parameters we can extract random values from any arbitrary domain still with a constant probability.

A-OCC Ideal Functionality. An oblivious common coin is parameterized by a set V and some constant probability p. Each party starts with an empty input λ, and every party P_i outputs a value from V where with probability at least $p > 0$ every party outputs the same uniformly random value $x \in V$ and with probability $1 - p$ the adversary chooses each party's output.[11]

The above goal can be translated to a UC functionality as follows: Initially, the ideal functionality samples a "fairness bit" $b \leftarrow \text{Bernoulli}(p)$ and a random value $y \overset{\text{R}}{\leftarrow} V$. Then, if $b = 1$ or no meaningful input is received from the adversary, it outputs y to every party. However, if $b = 0$ and meaningful input is received from the adversary, it assigns each party the value provided by the adversary. The functionality also informs the adversary about the fairness bit and the random value once they have been sampled. Asynchronous aspects, including delayed output release and participation, are handled as in Sect. 3. The resulting functionality is shown in Fig. 1.

[11] It is important to note that the term "oblivious" in this context refers to the fact that parties do not learn whether an agreement on a random coin value has been achieved or not, while the adversary does.

Functionality $\mathcal{F}_{\mathsf{a\text{-}occ}}^{V,p}$

The functionality is parameterized by a set V of possible outcomes and a fairness probability p, and it proceeds as follows. At the first activation, verify that $\mathsf{sid} = (\mathcal{P}, \mathsf{sid}')$, where $\mathcal{P}$ is a player set of size n. For each $P_i \in \mathcal{P}$, initialize y_i to a default value $\perp$, $\mathsf{participated}_i := 0$, and delay values $D_i^{\text{input}} = D_i^{\text{output}} := 1$. Also initialize a, y, and b to $\perp$, and $t := \lceil \frac{n}{3} \rceil - 1$.

– Upon receiving $(\mathtt{delay}, \mathsf{sid}, P_i, \mathsf{type}, D)$ from the adversary for $P_i \in \mathcal{P}$, $\mathsf{type} \in \{\text{input}, \text{output}\}$, and $D \in \mathbb{Z}$ represented in unary notation, update $D_i^{\mathsf{type}} := \max(1, D_i^{\mathsf{type}} + D)$ and send $(\mathtt{delay\text{-}set}, \mathsf{sid})$ to the adversary.
– Upon receiving $(\mathtt{input}, \mathsf{sid})$ from $P_i \in \mathcal{P}$ (or the adversary on behalf of corrupted P_i), run the Input Submission Procedure and send $(\mathtt{leakage}, \mathsf{sid}, P_i)$ and any other messages set by the Input Release Procedure to the adversary.
– Upon receiving $(\mathtt{replace}, \mathsf{sid}, v)$ from the adversary, record $a := v$.
– Upon receiving $(\mathtt{fetch}, \mathsf{sid})$ from $P_i \in \mathcal{P}$ (or the adversary on behalf of corrupted P_i) run the Input Submission and Output Release Procedures, and send $(\mathtt{fetched}, \mathsf{sid}, P_i)$ to the adversary and any messages set by the Output Release Procedure to P_i.

Input Submission Procedure: If $\mathsf{participated}_i = 0$ and $(\mathtt{input}, \mathsf{sid})$ was already received from P_i, then do:

1. Update $D_i^{\text{input}} := D_i^{\text{input}} - 1$.
2. If $D_i^{\text{input}} = 0$ then set $\mathsf{participated}_i := 1$.
3. If $b = \perp$ then sample a "fairness bit" $b \leftarrow \text{Bernoulli}(p)$ and a random value $y \overset{\text{R}}{\leftarrow} V$, and set $(\mathtt{reveal}, \mathsf{sid}, b, y)$ to be sent to the adversary.

Output Release Procedure: If $\sum_{j=1}^{n} \mathsf{participated}_j \geq n - t$ then do:

1. Update $D_i^{\text{output}} := D_i^{\text{output}} - 1$.
2. If $D_i^{\text{output}} = 0$, then do the following. If $y_i = \perp$:
   - If $b = 1$ or a cannot be parsed as $(a_1, \ldots, a_n) \in V^n$, set $y_k := y$ for each $P_k \in \mathcal{P}$.
   - If $b = 0$ and a can be parsed as $(a_1, \ldots, a_n) \in V^n$, set $y_k := a_k$ for each $P_k \in \mathcal{P}$.

   Additionally, set $(\mathtt{output}, \mathsf{sid}, y_i)$ to be sent to P_i.

Fig. 1. The asynchronous OCC functionality.

The A-OCC Protocol. We proceed to present our asynchronous and multi-valued OCC protocol. We begin by discussing all the essential building blocks employed in our protocol. Subsequently, we provide a high-level overview of the protocol, highlighting its key ideas. A detailed description appears in the full version [33].

The basic building blocks of our A-OCC protocol are A-VSS and A-Cast. A-VSS enables parties to contribute by privately providing their local randomness and only revealing this randomness when the contributions to the output are determined. Thus, A-VSS ensures the secrecy and verifiability of the shared secrets in an asynchronous setting. The A-VSS primitive is formally modeled as the ideal functionality $\mathcal{F}_{\textsf{a-vss}}$, described in Sect. 3.2. On the other hand, A-Cast facilitates communication among parties by providing stronger guarantees than simple message distribution. This is especially crucial in asynchronous settings where challenges such as low message dispersion can occur. A-Cast helps in overcoming these challenges and ensures reliable message dissemination among the parties. We use the ideal functionality $\mathcal{F}_{\textsf{a-cast}}$, described in Sect. 3.2, to model this primitive.

As previously mentioned, our multi-valued protocol is built upon existing binary A-OCC constructions [24,41] and introduces a novel combinatorial technique for extracting values from arbitrary domains. In both the binary protocol and our proposed protocol, each party secret-shares n random elements from a field. It can be observed that at some point during the protocol execution, a vector of length n consisting of random elements from the same field is established (with up to t missing values due to asynchrony). For each coordinate of this vector, random elements shared by $t+1$ parties are utilized to prevent the adversary, controlling up to t parties, from biasing any specific coordinate. Subsequently, each party starts reconstructing secrets shared by other parties to form the same vector locally. In the asynchronous setting, due to the low dispersion of messages, not all coordinates can be reconstructed by honest parties. This can result in different parties reconstructing different subsets of coordinates. However, by using mechanisms to improve message dispersion, as originally demonstrated by Feldman [41], it has been proven that when $t \leq n/3$, while the local vectors of honest parties may have up to t missing coordinates, they have an overlap of size at least $n/3$.[12] This is significant because without such mechanisms, and allowing for t missing components when $n = 3t + 1$, the overlap in the local vectors of just four parties could be empty.

Traditionally, existing protocols extract a single bit from the local views of the random vector by instructing parties to take all existing coordinates modulo n and output 0 if any coordinate is 0 or output 1 otherwise. In contrast, our protocol represents a significant improvement by going beyond the extraction of a single bit from the local views of the random vector. This enhanced randomness extraction is through a combinatorial observation regarding vectors of random values, as formulated in Lemma 1. This observation, allows the parties to agree non-interactively on certain coordinates of the random vector with a constant probability, while also ensuring that these agreed-upon coordinates lie within their overlap section. The minimum such coordinate is then used to select a common output value from the vector.

Another important observation regarding existing binary A-OCC protocols is the lack of a proper termination mechanism. This poses significant challenges as network delays can cause parties to operate out of sync. In such cases, some

[12] Feldman calculated the size of the overlap, denoted as x, based on the number of participants n and the maximum number of corruptions t. The general relation is $x \geq n - t - \frac{t^2}{n-2t}$, which yields $x \geq n/3$ and $x \geq 5n/8$ when $t \leq n/3$ and $t \leq n/4$, respectively. This argument was later used in [24] to achieve optimal resiliency.

parties may receive the output before completing their role in the execution, and if they stop, others may not be able to generate the output at all. This directly affects the simulator's ability to accurately simulate the protocol, especially in managing input and output delays in the ideal functionality. This is mainly because, unlike the protocol, the ideal functionality ensures that once a party receives the output, sufficient participation has occurred, and any other party, regardless of others' participation, can fetch the output if sufficiently activated. One potential solution could be invoking A-BA on the output at the end; however, this would create a circular dependency since A-OCC itself is used in A-BA. Instead, we choose to adopt a simpler approach inspired by Bracha's termination mechanism. This approach resolves the participation issue without causing deadlocks and ensures agreement once all parties initiate the procedure with the same value. The formal description of our asynchronous OCC protocol $\Pi_{\text{a-occ}}$ appears in the full version [33].

Having described our A-OCC protocol, we proceed to present the formal security statement that demonstrates how the protocol UC-realizes $\mathcal{F}_{\text{a-occ}}$. However, we first state a combinatorial observation regarding vectors of random values that facilitates the security proof. We formulate this observation separately in the following lemma, as it may be of independent interest.

Lemma 1. *Let V be a vector of n values chosen independently and uniformly at random from a set S of size $N \in \Theta(n^2)$, and let α be a constant satisfying $0 < \alpha \leq 1$. Then for any subset of indices $I \subseteq [n]$ such that $|I| \geq \alpha n$, with constant probability p there is at least one repeated value in V; moreover, all of the repeated values are constrained to the indices in I.*

With the groundwork laid out, we now state the security of protocol $\Pi_{\text{a-occ}}$:

Theorem 1. *There exists a probability $p \in \Theta(1)$ such that for any integer domain V, protocol $\Pi^{V}_{\text{a-occ}}$ UC-realizes $\mathcal{F}^{V;p}_{\text{a-occ}}$ with perfect security in the ($\mathcal{F}_{\text{a-cast}}$, $\mathcal{F}^{\mathbb{F}}_{\text{a-vss}}$)-hybrid model where $\mathbb{F}$ is the smallest prime field of size at least $\text{lcm}(|V|, n^2)$, in constant rounds and in the presence of an adaptive and malicious t-adversary, provided $t < \frac{n}{3}$.*

5 Concurrent A-BA in Expected-Constant Rounds

In this section, we dive into the important problem of achieving concurrent A-BA in an expected-constant number of rounds. As discussed in the Introduction, Ben-Or and El-Yaniv [12] highlighted the potential issue of running multiple executions of a probabilistic-termination protocol in parallel, which could lead to an increase in the expected number of rounds required for *all* executions to terminate. The concurrent A-BA protocol proposed in [12] relies on A-OLE and multi-valued A-BA, which can be instantiated using our A-OCC protocol from Sect. 4 and the extended A-BA protocol from [24,70], respectively. However, during our analysis, we discovered certain issues in their analysis that cast doubt on the expected-constant round complexity of one of their main building blocks and, consequently, their concurrent A-BA protocol. For a more comprehensive presentation of these issues, see the full version of the paper [33]. It is unclear how to address these issues without modifying the protocol itself.

To rectify these concerns, we modify the underlying message distribution mechanism and incorporate an additional layer of message validation. These changes not only resolve the identified issues, but also significantly simplify the protocol. It is worth emphasizing that our revised concurrent A-BA protocol achieves a level of simplicity that is comparable to the synchronous version proposed in [12]. This accomplishment is significant because when designing an asynchronous counterpart to a synchronous protocol, achieving a level of simplicity on par with the synchronous version is often considered the ideal outcome. In the following, we describe an ideal functionality for concurrent A-BA, our protocol, and its required building blocks, and provide a security statement.

Concurrent A-BA Ideal Functionality. Concurrent A-BA, as the name suggests, refers to a primitive that enables parties to solve N instances of A-BA concurrently. We are primarily interested in the case $N = n$, corresponding to the emulation of n ideal A-BA primitives, commonly used in asynchronous MPC protocols to form the core set and overcome low message dispersion. However, in our study, we consider a more general version that allows for a broader range of values for N. In this setting, each party P_i initiates the concurrent A-BA by providing N values, namely $v_{i,1}, \ldots, v_{i,N}$. Subsequently, all parties receive the same set of N output values, denoted as $y_1, \ldots, y_N$. Each individual output value y_j is computed based on the input values $v_{1,j}, \ldots, v_{n,j}$, following the prescribed procedure outlined in the standard A-BA primitive. Specifically, if $n - 2t$ input values are identical, that common value is selected as the output; otherwise, the output is determined by the adversary. We capture the task of concurrent A-BA using the ideal functionality $\mathcal{F}_{\mathsf{conc\text{-}a\text{-}ba}}$, defined in the full version [33]. Since we are able to achieve non-intrusion validity (i.e., for each instance, the corresponding output must be either $\perp$ or the corresponding input of an honest party), we consider an *intrusion-tolerant* concurrent A-BA functionality.

Building Blocks. Our concurrent A-BA protocol relies on A-Cast as the fundamental communication primitive due to its enhanced guarantees compared to basic message distribution mechanisms. The ideal functionality $\mathcal{F}_{\mathsf{a\text{-}cast}}$, which models the A-Cast primitive, was described in Sect. 3.2.

As another crucial building block, our protocol incorporates asynchronous oblivious leader election (A-OLE) as a coordination mechanism among the parties. A-OLE enables parties to randomly elect a leader from among themselves. The term "oblivious" indicates that parties are unaware of whether or not agreement on a random leader has been achieved. In our concurrent A-BA protocol, similar to the approach described in [12], there comes a point where all parties suggest outputs, and A-OLE assists them in reaching agreement on the output by adopting the suggestion of the elected leader. To capture the task of A-OLE, we parameterize the A-OCC functionality $\mathcal{F}_{\mathsf{a\text{-}occ}}$, given in Sect. 4, by a domain with size equal to the number of parties. This yields an ideal functionality for

A-OLE, denoted as $\mathcal{F}_{\mathsf{a\text{-}ole}}$, which is defined as $\mathcal{F}^{[n],p}_{\mathsf{a\text{-}occ}}$ for appropriate $p \in \Theta(1)$. Recall from Theorem 1 that $\mathcal{F}_{\mathsf{a\text{-}ole}}$ can be realized using protocol $\Pi^{[n]}_{\mathsf{a\text{-}occ}}$.

Our concurrent A-BA protocol leverages both binary and multi-valued A-BA. Binary A-BA is employed to achieve agreement on critical decisions within the protocol, such as whether to continue a particular iteration, or whether to terminate the protocol. On the other hand, the use of multi-valued A-BA addresses the inherent obliviousness of the A-OLE primitive by providing a means for parties to reach an agreement on the output. The ideal functionality for A-BA was described in Sect. 3.2.

As in [12], another essential component in our concurrent A-BA protocol is *truncated* executions of an A-BA protocol limited to a predefined number of iterations, consequently implying a fixed number of rounds. In the spirit of [31], we model those executions with the ideal functionality $\mathcal{F}_{\mathsf{trunc\text{-}a\text{-}ba}}$, defined in the full version [33]. $\mathcal{F}_{\mathsf{trunc\text{-}a\text{-}ba}}$ is parameterized with V, p, and itr, where V denotes the domain, p represents the termination probability in each iteration, and itr indicates the maximum number of iterations in the execution.

This ideal functionality also encapsulates a "1-shift" property for termination, meaning that all honest parties produce the output within two consecutive iterations. We note that the adversary has the discretion to determine which parties discover the output first. Unlike a traditional A-BA functionality that outputs a single value, $\mathcal{F}_{\mathsf{trunc\text{-}a\text{-}ba}}$ produces a vector of values that includes the output after each iteration of the execution. Modeling the 1-shift property and providing outputs for all iterations is crucial since our concurrent A-BA protocol relies on those properties of truncated executions of A-BA.

It is worth mentioning that $\mathcal{F}^{V,p,\mathsf{itr}}_{\mathsf{trunc\text{-}a\text{-}ba}}$ can be implemented by executing any intrusion-tolerant A-BA protocol with the 1-shift property and a termination probability of p in each iteration, precisely for itr iterations, and concatenating the output of all iterations to get the final output (with λ representing iterations without output). Canetti and Rabin's binary A-BA protocol [24] possesses the desired properties of intrusion tolerance and terminating with a constant probability in each iteration. However, using Bracha's technique for termination [18] in Canetti and Rabin's binary A-BA protocol does not admit the 1-shift property. The main reason is that parties may take the shortcut and use Bracha termination messages to generate the output in their very early iterations. Fortunately, Bracha's termination procedure is unnecessary in the truncated execution of Canetti and Rabin's binary A-BA protocol. This is primarily due to the fact that all parties will naturally terminate after a fixed number of iterations. With this adjustment, Canetti and Rabin's binary A-BA protocol also successfully attains the 1-shift property. Moreover, Mostéfaoui and Raynal's multi-valued A-BA protocol [70] offers intrusion tolerance, the 1-shift property, and terminating with a constant probability in each iteration if the underlying binary A-BA used in their construction also exhibits these characteristics. Thus, we can formulate the following proposition about realizing $\mathcal{F}^{V,p,\mathsf{itr}}_{\mathsf{trunc\text{-}a\text{-}ba}}$.

Proposition 4. *For some constant probability p, any domain V, and any integer itr, $\mathcal{F}^{V,p,\mathsf{itr}}_{\mathsf{trunc\text{-}a\text{-}ba}}$ can be UC-realized with statistical security in the $\mathcal{F}_{\mathsf{a\text{-}smt}}$-hybrid* *model, in constant rounds and in the presence of an adaptive and malicious t-adversary, provided $t < n/3$.*

The above statement guarantees statistical security since the A-BA protocols we consider rely on the A-VSS primitive, which can only be realized with statistical security when $t < n/3$. In fact, by working in the $\mathcal{F}_{\mathsf{a\text{-}vss}}$-hybrid model, we can achieve perfectly secure $\mathcal{F}_{\mathsf{trunc\text{-}a\text{-}ba}}$. We remark that the intrusion tolerance of $\mathcal{F}_{\mathsf{trunc\text{-}a\text{-}ba}}$ is not a requirement in our concurrent A-BA protocol; however, employing a non-intrusion-tolerant version of truncated A-BA will naturally lead to a non-intrusion-tolerant concurrent A-BA protocol.

The New Concurrent A-BA Protocol. Our protocol builds on the core ideas presented in [12]. In addition to instantiating the missing OLE building block using our OCC protocol from Sect. 4, we address the issue in the analysis by redesigning the message distribution phase. Our revised message distribution mechanism not only resolves the issue in the proof but also provides stronger guarantees, which in turn simplifies the final protocol design. In fact, apart from the message distribution phase, the overall structure of our protocol closely resembles the synchronous version of the protocol described in [12].

Before diving into the high-level description of our protocol, we highlight the choices we made in the message distribution mechanism and discuss some alternative approaches that fail to meet our requirements. In the eventual-delivery model, at least $n - t$ parties will receive each other's messages if they wait for a sufficient duration, as messages from honest parties will eventually be delivered to one another, but determining the exact waiting time required is not straightforward. One possible approach is to instruct parties to A-Cast the identities of the parties from which they have received messages. After constructing a graph with parties as vertices and adding edges between parties that have reported message receipts from each other, we can look for a clique of size $n - t$; however, finding a maximum clique is known to be NP-complete and also difficult to approximate [40].

To overcome this challenge, one possible approach is to investigate alternative structures that offer weaker guarantees regarding message dispersion but can be efficiently identified [10,14,22,72]. However, it is important to note that this approach does not guarantee the precise message dispersion required for our specific application. In our analysis, it is crucial that *all* honest parties receive messages from a linear fraction of other parties. Even finding a clique of size $n-t$ does not guarantee this level of message dispersion, rendering this approach unsuitable for our protocol. Thus, we adopt another approach that has been used in prior works [24,41], and extend it by incorporating a precondition check before the process and introducing a validation layer during the execution. These additions enhance the guarantees, making them more suitable and effective for our specific purpose. We proceed to explain our protocol.

Similar to the (synchronous) protocol described in [12], in our concurrent A-BA protocol, each party initiates for every A-BA instance a batch of m executions of the A-BA protocol (over the same inputs) for a fixed number of iterations, denoted as itr. If the A-BA protocol has a termination probability of at least a constant value p in each iteration, which is the case for most existing A-BA protocols, suitable values of m and itr can be determined so that each party obtains at least one output value for each batch. Each party then selects an output for each instance and forms a suggestion for the final output. The next paragraph, which explains the mechanism for distributing suggestions among parties, is our main modification to the protocol. Then, for the remaining part, we can use a similar structure as the synchronous version of the protocol.

Firstly, parties initiate a binary A-BA protocol to determine a specific condition that allows for choosing and validating suggested outputs later in the execution. Based on the outcome of this binary A-BA, parties decide whether to continue or start over. In the case of continuation, parties perform A-Cast operations to distribute their suggestions and wait to receive suggestions from other parties. Each party only accepts an A-Cast message containing a suggested output if the value is consistent with the outputs obtained from their own truncated A-BA executions. This validation step is crucial as it ensures that even corrupted parties provide acceptable (correct) suggestions. The validation is based on the 1-shift property of A-BA that ensures all honest parties terminate within two consecutive iterations. Parties wait to accept A-Casts of suggested outputs from at least $n - t$ parties and then A-Cast the set of all these $n - t$ suggestions along with the identities of the corresponding senders. They continue accepting A-Casts of suggestions and sets until they receive at least $n-t$ sets that are fully contained within their accepted suggestions. At this point, a sufficient number of messages have been exchanged, and using a counting argument similar to the one in [24,41], it can be deduced that at least $n/3$ parties have their suggested outputs received by all honest parties.

After the message distribution phase described above, our protocol proceeds similarly to the synchronous protocol presented in [12]. Specifically, parties execute OLE to elect a random leader to adopt its suggested output. Subsequently, a multi-valued A-BA is performed on the adopted output to address the oblivious nature of OLE. Finally, a binary A-BA is executed to determine whether an agreement on the output has been reached, resulting in the termination or restarting of the protocol. The intrusion-tolerance property of multi-valued A-BA is crucial to make sure that the result of the agreement on the output is not provided by a corrupted party. It is worth noting that the favorable scenario occurs when the leader is among the $n/3$ parties whose suggested outputs have been accepted by all honest parties. If the leader is elected randomly, this event happens with a probability of $1/3$. In this case, all parties adopt the same output, leading to termination in the subsequent A-BA calls.

It is worth noting that the set of $n/3$ parties whose suggested outputs have been accepted by all honest parties may only contain a single honest party. This single honest party can only be elected with a probability of $O(1/n)$ by the OLE. However, due to the validation step in the message-distribution phase, there is no longer a need to ensure that an honest leader is elected, as the suggested values from corrupted parties are considered valid outputs. Refer to the full version [33] for a formal description of protocol $\Pi_{\mathsf{conc\text{-}a\text{-}ba}}$.

We now consider the security of our concurrent A-BA protocol. Before stating the theorem, it is worth noting that the specific parameters of the hybrid model, which combine the different ideal functionalities, are not explicitly specified in the theorem statement. However, they can be determined from the protocol's parameters and are integral to the overall security guarantees of the protocol. Now, let us state the theorem formally:

Theorem 2. *For any domain V, integer N, constant $0 < p < 1$, and constant integer $\mathsf{itr} > 1$, setting $m := \log_{\frac{1}{1-p}} N$, the protocol $\Pi^{V,N,m,p,\mathsf{itr}}_{\mathsf{conc\text{-}a\text{-}ba}}$ UC-realizes $\mathcal{F}^{V,N}_{\mathsf{conc\text{-}a\text{-}ba}}$ with statistical security in the $(\mathcal{F}_{\mathsf{a\text{-}cast}}, \mathcal{F}_{\mathsf{a\text{-}ba}}, \mathcal{F}_{\mathsf{trunc\text{-}a\text{-}ba}}, \mathcal{F}_{\mathsf{a\text{-}ole}})$-hybrid model, in expected-constant rounds and in the presence of an adaptive and malicious t-adversary, provided $t < n/3$.*

Acknowledgements. Our original motivation for this project was to provide a simulation-based treatment of concurrent A-BA protocols, such as Ben-Or and El-Yaniv's [12], but the search for building blocks, in particular of an optimally resilient asynchronous OCC protocol became a bit of a "detective story," as many references pointed to an unpublished manuscript by Feldman [41], which was nowhere to be found. We thank Michael Ben-Or for providing it to us, which corroborated its in-existence.

Ran Cohen's research is supported in part by NSF grant no. 2055568. Juan Garay's research is supported in part by NSF grants no. 2001082 and 2055694. Vassilis Zikas's research is supported in part by NSF grant no. 2055599 and by Sunday Group. The authors were also supported by the Algorand Centres of Excellence programme managed by Algorand Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Algorand Foundation.

References

1. Abraham, I., Asharov, G., Patra, A., Stern, G.: Perfectly secure asynchronous agreement on a core set in constant expected time. IACR Cryptology ePrint Archive, Report 2023/1130 (2023). https://eprint.iacr.org/2023/1130
2. Abraham, I., Dolev, D., Halpern, J.Y.: An almost-surely terminating polynomial protocol for asynchronous byzantine agreement with optimal resilience. In: 27th ACM PODC, pp. 405–414. ACM (2008)
3. Abraham, I., Dolev, D., Stern, G.: Revisiting asynchronous fault tolerant computation with optimal resilience. Distributed Comput. **35**(4), 333–355 (2022)
4. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Reaching consensus for asynchronous distributed key generation. In: 40th ACM PODC, pp. 363–373. ACM (2021)

5. Abraham, I., Malkhi, D., Spiegelman, A.: Asymptotically optimal validated asynchronous byzantine agreement. In: 38th ACM PODC, pp. 337–346. ACM (2019)
6. Attiya, H., Censor-Hillel, K.: Lower bounds for randomized consensus under a weak adversary. SIAM J. Comput. **39**(8), 3885–3904 (2010)
7. Beaver, D., So, N.: Global, unpredictable bit generation without broadcast. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 424–434. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_36
8. Beerliová-Trubíniová, Z., Hirt, M.: Simple and efficient perfectly-secure asynchronous MPC. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 376–392. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_23
9. Ben-Or, M.: Another advantage of free choice: completely asynchronous agreement protocols (extended abstract). In: 2nd ACM PODC, pp. 27–30. ACM (1983)
10. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: 25th ACM STOC, pp. 52–61. ACM Press (1993)
11. Ben-Or, M., El-Yaniv, R.: Interactive consistency in constant expected time. Technical report, Inst. of Math. and Comp. Sci., Hebrew University, Jerusalem (1988)
12. Ben-Or, M., El-Yaniv, R.: Resilient-optimal interactive consistency in constant time. Distrib. Comput. **16**(4), 249–262 (2003)
13. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC, pp. 1–10. ACM Press (1988)
14. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience (extended abstract). In: 13th ACM PODC, pp. 183–192. ACM (1994)
15. Berman, P., Garay, J.A., Perry, K.J.: Towards optimal distributed consensus (extended abstract). In: 30th FOCS, pp. 410–415. IEEE Computer Society Press (1989)
16. Blum, E., Liu-Zhang, C.-D., Loss, J.: Always have a backup plan: fully secure synchronous MPC with asynchronous fallback. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 707–731. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_25
17. Borcherding, M.: Levels of authentication in distributed agreement. In: Babaoğlu, Ö., Marzullo, K. (eds.) WDAG 1996. LNCS, vol. 1151, pp. 40–55. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61769-8_4
18. Bracha, G.: Asynchronous byzantine agreement protocols. Inf. Comput. **75**(2), 130–143 (1987)
19. Bracha, G., Toueg, S.: Asynchronous consensus and broadcast protocols. J. ACM **32**(4), 824–840 (1985)
20. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 524–541. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_31
21. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantinople: practical asynchronous byzantine agreement using cryptography. J. Cryptol. **18**(3), 219–246 (2005)
22. Canetti, R.: Studies in secure multiparty computation and applications. Ph.D. thesis, Weizmann Institute of Science (1996)
23. Canetti, R.: Universally composable security. J. ACM **67**(5), 1–94 (2020)
24. Canetti, R., Rabin, T.: Fast asynchronous byzantine agreement with optimal resilience. In: 25th ACM STOC, pp. 42–51. ACM Press (1993)

25. Canetti, R., Rabin, T.: Fast asynchronous byzantine agreement with optimal resilience. Full version of [24] (1998). https://www.cs.tau.ac.il/~canetti/materials/cr93.ps
26. Chan, T.H., Pass, R., Shi, E.: Round complexity of Byzantine agreement, revisited. IACR Cryptology ePrint Archive, Report 2019/886 (2019). https://eprint.iacr.org/2019/886
27. Chan, T.-H.H., Pass, R., Shi, E.: Sublinear-round byzantine agreement under corrupt majority. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 246–265. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_9
28. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: 20th ACM STOC, pp. 11–19. ACM Press (1988)
29. Chor, B., Merritt, M., Shmoys, D.B.: Simple constant-time consensus protocols in realistic failure models. J. ACM **36**(3), 591–614 (1989)
30. Cohen, R.: Asynchronous secure multiparty computation in constant time. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016, Part II. LNCS, vol. 9615, pp. 183–207. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_8
31. Cohen, R., Coretti, S., Garay, J., Zikas, V.: Probabilistic termination and composability of cryptographic protocols. J. Cryptol. **32**(3), 690–741 (2019)
32. Cohen, R., Coretti, S., Garay, J.A., Zikas, V.: Round-preserving parallel composition of probabilistic-termination cryptographic protocols. J. Cryptol. **34**(2), 12 (2021)
33. Cohen, R., Forghani, P., Garay, J.A., Patel, R., Zikas, V.: Concurrent asynchronous byzantine agreement in expected-constant rounds, revisited. IACR Cryptology ePrint Archive, Report 2023/1003 (2023). https://eprint.iacr.org/2023/1003
34. Cohen, R., Garay, J., Zikas, V.: Completeness theorems for adaptively secure broadcast (2023), cRYPTO '23 (2023, to appear)
35. Cohen, R., Haitner, I., Makriyannis, N., Orland, M., Samorodnitsky, A.: On the round complexity of randomized byzantine agreement. J. Cryptol. **35**(2), 10 (2022)
36. Coretti, S., Garay, J., Hirt, M., Zikas, V.: Constant-round asynchronous multiparty computation based on one-way functions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 998–1021. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_33
37. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. SIAM J. Comput. **12**(4), 656–666 (1983)
38. Duan, S., Wang, X., Zhang, H.: Practical signature-free asynchronous common subset in constant time. Cryptology ePrint Archive (2023), cCS '23 (2023, to appear)
39. Dwork, C., Lynch, N.A., Stockmeyer, L.J.: Consensus in the presence of partial synchrony. J. ACM **35**(2), 288–323 (1988)
40. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Approximating clique is almost NP-complete (preliminary version). In: 32nd FOCS, pp. 2–12. IEEE Computer Society Press (1991)
41. Feldman, P.: Asynchronous byzantine agreement in constant expected time (1989), unpublished manuscript
42. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: 20th ACM STOC, pp. 148–161. ACM Press (1988)
43. Feldman, P., Micali, S.: An optimal probabilistic algorithm for synchronous Byzantine agreement. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S.R. (eds.) ICALP 1989. LNCS, vol. 372, pp. 341–378. Springer, Heidelberg (1989). https://doi.org/10.1007/BFb0035770

44. Feldman, P.N.: Optimal Algorithms for Byzantine Agreement. Ph.D. thesis, Massachusetts Institute of Technology (1988)
45. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997)
46. Fischer, M.J., Lynch, N.A.: A lower bound for the time to assure interactive consistency. Inf. Process. Lett. **14**(4), 183–186 (1982)
47. Fischer, M.J., Lynch, N.A., Merritt, M.: Easy impossibility proofs for distributed consensus problems. Distrib. Comput. **1**(1), 26–39 (1986)
48. Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
49. Fitzi, M.: Generalized communication and security models in Byzantine agreement. Ph.D. thesis, ETH Zurich, Zürich, Switzerland (2003)
50. Fitzi, M., Garay, J.A.: Efficient player-optimal protocols for strong and differential consensus. In: 22nd ACM PODC, pp. 211–220. ACM (2003)
51. Fitzi, M., Nielsen, J.B.: On the number of synchronous rounds sufficient for authenticated byzantine agreement. In: Keidar, I. (ed.) DISC 2009. LNCS, vol. 5805, pp. 449–463. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04355-0_46
52. Gao, Y., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Efficient asynchronous byzantine agreement without private setups. In: 42nd ICDCS, pp. 246–257. IEEE (2022)
53. Garay, J.A., Katz, J., Koo, C., Ostrovsky, R.: Round complexity of authenticated broadcast with a dishonest majority. In: 48th FOCS, pp. 658–668. IEEE Computer Society Press (2007)
54. Garay, J.A., Moses, Y.: Fully polynomial byzantine agreement for n > 3t processors in t + 1 rounds. SIAM J. Comput. **27**(1), 247–290 (1998)
55. Ghinea, D., Goyal, V., Liu-Zhang, C.: Round-optimal byzantine agreement. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 96–119. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-06944-4_4
56. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: 19th ACM STOC, pp. 218–229. ACM Press (1987)
57. Hirt, M., Nielsen, J.B., Przydatek, B.: Cryptographic asynchronous multi-party computation with optimal resilience. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 322–340. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_19
58. Hirt, M., Nielsen, J.B., Przydatek, B.: Asynchronous multi-party computation with quadratic communication. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 473–485. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_39
59. Hirt, M., Zikas, V.: Adaptively secure broadcast. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 466–485. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_24
60. Karlin, A.R., Yao, A.C.: Probabilistic lower bounds for Byzantine agreement and clock synchronization (1986). unpublished manuscript
61. Katz, J., Koo, C.: On expected constant-round protocols for byzantine agreement. J. Comput. Syst. Sci. **75**(2), 91–112 (2009)
62. Katz, J., Maurer, U., Tackmann, B., Zikas, V.: Universally composable synchronous computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 477–498. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_27

63. Lamport, L., Shostak, R.E., Pease, M.C.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. **4**(3), 382–401 (1982)
64. Lindell, Y., Lysyanskaya, A., Rabin, T.: Sequential composition of protocols without simultaneous termination. In: 21st ACM PODC, pp. 203–212. ACM (2002)
65. Lindell, Y., Lysyanskaya, A., Rabin, T.: On the composition of authenticated byzantine agreement. J. ACM **53**(6), 881–917 (2006)
66. Liu-Zhang, C.-D., Loss, J., Maurer, U., Moran, T., Tschudi, D.: MPC with synchronous security and asynchronous responsiveness. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 92–119. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_4
67. Lu, Y., Lu, Z., Tang, Q., Wang, G.: Dumbo-MVBA: optimal multi-valued validated asynchronous byzantine agreement, revisited. In: 39th ACM PODC, pp. 129–138. ACM (2020)
68. Micali, S.: Very simple and efficient byzantine agreement. In: ITCS 2017. LIPIcs, vol. 4266, pp. 6:1–6:1. Schloss Dagstuhl (2017)
69. Micali, S.: Very simple and efficient byzantine agreement. In: ITCS 2017. LIPIcs, vol. 4266, pp. 6:1–6:1. Schloss Dagstuhl (2017)
70. Mostéfaoui, A., Raynal, M.: Signature-free asynchronous byzantine systems: from multivalued to binary consensus with $\mathrm{t} < \mathrm{n}/3$, $\mathrm{O}(\mathrm{n}^2)$ messages, and constant time. Acta Informatica **54**(5), 501–520 (2017)
71. Nielsen, J.B.: A threshold pseudorandom function construction and its applications. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 401–416. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_26
72. Patra, A.: Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) OPODIS 2011. LNCS, vol. 7109, pp. 34–49. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25873-2_4
73. Patra, A., Choudhury, A., Rangan, C.P.: Asynchronous byzantine agreement with optimal resilience. Distrib. Comput. **27**(2), 111–146 (2014)
74. Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
75. Pfitzmann, B., Waidner, M.: Unconditional Byzantine agreement for any number of faulty processors. In: Finkel, A., Jantzen, M. (eds.) STACS 1992. LNCS, vol. 577, pp. 337–350. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55210-3_195
76. Rabin, M.O.: Randomized byzantine generals. In: 24th FOCS. pp. 403–409. IEEE Computer Society Press (1983)
77. de Souza, L.F., Kuznetsov, P., Tonkikh, A.: Distributed randomness from approximate agreement. In: 36th DISC. LIPIcs, vol. 246, pp. 24:1–24:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
78. Srinivasan, S., Loss, J., Malavolta, G., Nayak, K., Papamanthou, C., Thyagarajan, S.A.K.: Transparent batchable time-lock puzzles and applications to byzantine consensus. In: PKC 2023, Part I. LNCS, pp. 554–584. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-31368-4_20
79. Turpin, R., Coan, B.A.: Extending binary byzantine agreement to multivalued byzantine agreement. Inf. Process. Lett. **18**(2), 73–76 (1984)
80. Wan, J., Xiao, H., Devadas, S., Shi, E.: Round-efficient byzantine broadcast under strongly adaptive and majority corruptions. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 412–456. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_15

81. Wan, J., Xiao, H., Shi, E., Devadas, S.: Expected constant round byzantine broadcast under dishonest majority. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part I. LNCS, vol. 12550, pp. 381–411. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_14
82. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS, pp. 160–164. IEEE Computer Society Press (1982)
83. Zhang, H., Duan, S.: PACE: fully parallelizable BFT from reproposable byzantine agreement. In: ACM CCS 2022, pp. 3151–3164. ACM (2022)