Asynchronous Byzantine Agreement with Optimal Resilience and Linear Complexity

Cheng Wang
EPFL
cheng.wang@epfl.ch

Abstract

Given a system with $n > 3t+1$ processes, where t is the tolerated number of faulty ones, we present a fast asynchronous Byzantine agreement protocol that can reach agreement in $O(t)$ expected running time. This improves the $O(n^2)$ expected running time of Abraham, Dolev, and Halpern [1]. Furthermore, if $n = (3 + \varepsilon)t$ for any $\varepsilon > 0$, our protocol can reach agreement in $O(1/\varepsilon)$ expected running time. This improves the result of Feldman and Micali [7] (with constant expected running time when $n > 4t$).

1. Introduction

The *Byzantine Agreement* (BA) problem, first introduced by Pease, Shostak, and Lamport [11, 12], is a fundamental problem in distributed computing. Given n processes, t of which being faulty, the problem consists for all correct processes to agree on one of the input values. The faulty processes might deviate from the algorithm assigned to them arbitrarily, e.g., to prevent correct processes from agreeing on one of their input values.

A lot of work has been devoted to the problem in the last three decades. Despite the effort, the *Asynchronous Byzantine Agreement* (ABA) problem, where the communication between processes can take an arbitrary amount of time, is still not very well understood. Certain results are however known. For example, it is known that the problem is impossible to solve if $n \leqslant 3t$ [9, 12]. Any ABA protocol assuming $n > 3t$ is called *optimally resilient*. According to the seminal result of [8], any deterministic ABA protocol must have some non-terminating execution.

Faced with the impossibility result [8], a natural direction of research is to design efficient *randomized* Byzantine agreement protocol. This direction was started with the work of Ben-or [3], Rabin [13], and Bracha [4]. Remarkably, Canetti and Rabin [6] proposed an ABA protocol with constant expected running time and overwhelming probability to terminate. With a randomized ABA protocol the best that can be achieved is to have every execution terminate with probability one. Such protocols are said to be *almost-surely terminating* [1].

Several almost-surely terminating ABA protocols were proposed. In 1983, Ben-Or [3] proposed an almost-surely terminating ABA protocol for $n > 5t$, which runs in exponential expected time. One year later, Bracha [4] presented an almost-surely terminating ABA, which also runs in exponential expected time, but with optimal resilience, i.e., for $n > 3t$. In 1988, Feldman and Micali [7] presented an almost-surely terminating ABA protocol with constant expected time, assuming however $n > 4t$. Twenty years later, Abraham, Dolev, and Halpern [1] presented an almost-surely terminating optimally resilient ABA protocol with polynomial efficiency (the expected running time is $O(n^2)$). In some sense, state-of-the-art results for almost-surely terminating ABA are [7] and [1]: optimally resilience with polynomial efficiency on the one hand, or constant expected time, assuming however $n > 4t$, on the other hand.

We present in this paper a new almost-surely terminating ABA protocol that achieves a significant progress with respect to the state-of-the-art. For $n > 3t$, our protocol completes in $O(t)$ expected running time. If $n > (3+\varepsilon)t$ where ε is an arbitrary positive constant, our protocol has $O(1/\varepsilon)$ expected running time. Table 1 aggregates these results in the context of related work.

Most ABA protocols follow the idea of Ben-or [3], Rabin [13], and Bracha [4], namely a reduction of the ABA problem to the implementation of a *common coin* (namely, a source of common randomness with certain properties). Specially, the reduction of Bracha [4] is optimally resilient and runs in constant expected time. Thus, designing efficient ABA protocols could be solved by designing efficient common coins. The protocol of Feldman and Micali [7] includes a method to implement a common coin by making

Reference	Resilience	Expected Running Time
Ben-Or (1983) [3]	$n > 5t$	$O(2^n)$
Bracha (1984) [4]	$n > 3t$	$O(2^n)$
Feldman, Micali (1988) [7]	$n > 4t$	$O(1)$
Abraham, Dolev, Halpern (2008) [1]	$n > 3t$	$O(n^2)$
This paper	$n > 3t$	$O(t)$
This paper	$n > (3+\varepsilon)t, (\varepsilon > 0)$	$O(1/\varepsilon)$

Table 1. Results for almost-surely terminating ABA problem

use of a *verifiable secret sharing* (VSS) scheme. (For a complete description of the reduction from VSS to ABA, see [5].) Canetti and Rabin [6] have an implementation of *asynchronous verifiable secret sharing* (AVSS) with constant expected running time but overwhelming probability to terminate (the resulting ABA protocol is thus not almost-surely terminating). Recently, King and Saia [10] introduced a novel technique for implementing common coin via a spectral method.

This paper follows the reduction from (some form of) AVSS to ABA. We first recall the standard AVSS scheme [6]. Roughly speaking, an AVSS scheme consists of a *sharing* phase and a *reconstruction* phase, involving a process designated as the *dealer* which has a value (usually called *secret*) to share. In the sharing phase, the dealer shares its secret among all processes and each process locally verifies that a unique secret is being considered. In the reconstruction phase, the processes reconstruct the secret from the shares. The correctness of AVSS lies on two properties: (1) if the dealer is correct, then all correct processes will reconstruct the secret of the dealer, and (2) if the dealer is faulty, then all correct processes will reconstruct the same value that is fixed in the sharing phase.

We introduce in this paper a variant of AVSS called IVSS (standing for *inferable (asynchronous) verifiable secret sharing*). Our IVSS scheme has a weaker correctness property than AVSS, but provides strong fault-detection ability. Specifically, IVSS requires that if the correctness property of AVSS does not hold in an invocation of some round, then correct processes will ignore (or infer) at least $t(n-3t)$ faulty pairs from that round on. Here, by a *faulty pair*, we mean a pair of processes of which at least one is faulty. In our IVSS protocol, secrets are shared through symmetric bivariate polynomials. If processes reconstruct different secrets in the protocol, the symmetry of polynomials can be used to infer faulty pairs.

There are existing secret sharing protocols with fault-detection capacity, e.g., shunning verifiable secret sharing in [2] and secret sharing with dispute control in [1]. These protocols are composed of several levels of secret sharing subprotocols, while our protocol is very simple with only one-level secret sharing subprotocol. In all previous approaches, the Byzantine agreement algorithm proceeds round by round and, once a round is over, the correct processes forget it and never look back to it. In fact, if a correct process could look back at the history of invocations of the secret sharing protocol, it may infer more failures. We implement this history-based checking mechanism in a *certification* subprotocol. This subprotocol is invoked when the Byzantine agreement protocol is initialized and then runs concurrently with all invocations of our IVSS protocol. The main technique for inferring faults in our protocol is also different from [1, 2]. Our fault-detection mechanism is based on symmetric polynomials which enable our protocol to infer a linear number of faults when secret sharing does not succeed, while protocols in [1, 2] can generally infer only one fault.

The rest of this paper is organized as follows. In Section 2, we recall the asynchronous computing model and the Byzantine agreement problem. In Section 3, we state the properties of our IVSS scheme and describe an algorithm that implements it. In Section 4, we show how to obtain our fast ABA protocol from our IVSS scheme. For space limitations, some algorithms and proofs are given in the appendices.

2. Model and Definitions

The Model. We consider an asynchronous computing model in the classical sense, e.g., [1, 6]. We consider a complete network of n processes with identifiers $\{1, 2, \ldots, n\}$. The number n is always strictly greater than $3t$. The communication channels are private, i.e. no one can read or alter messages transmitted along it. Messages sent on a channel may have arbitrary (but finite) delay. A t-adversary can control at most t processes during the Byzantine agreement protocol. Once a process is controlled, it hands all its data over to the adversary and follows its instructions. We call all these controlled processes as *faulty* ones and other uncontrolled processes as *correct* ones. Note that the adversary cannot access messages transmitted between correct processes due to private communication channels.

We measure the running time of a protocol by the maximal expected number of communication rounds it takes to reach agreement [6, 10]. Consider a virtual 'global clock' measuring time in the network. This clock cannot be accessed by the processes. Let the *delay* of a message transmission denote the time elapsed from its sending to its reception. The *period* of a finite execution of a protocol is the longest delay of a message transmission during this execution. Let the *duration* of a finite execution denote the total time measured by the global clock divided by the period of this execution. The *expected running time* of a protocol, is the maximum over all inputs and applicable adversaries, of the average of the duration of executions of the protocol over the random inputs of the processes. In addition, each process divides its local time into *rounds* and execute a protocol round by round. The time of each round is less than or equal to a period of the execution of a protocol. The expected running time of a protocol can be computed by the expected rounds in execution.

Asynchronous Byzantine Agreement.

***Definition* 1** (ABA)**.** Let π be any asynchronous protocol in which each process has a binary input. We say that π is an almost-surely terminating, t-resilient ABA protocol if the following properties hold for every t-adversary and every input:

- ***Termination***: With probability one, every correct process terminates and outputs a value.
- ***Correctness***: All correct processes which have terminated have the same outputs. Moreover, if all correct processes have the same input, denoted v, then all correct processes output v.

Asynchronous Broadcast: A-Cast. We will often make use of this asynchronous broadcast primitive, introduced by Bracha [4] (for $n > 3t$). We follow the terminology in [5]. For completeness, the implementation is provided in Appendix I.

***Definition* 2** (A-Cast)**.** Let π be any asynchronous protocol initiated by a designated process (the sender) which has an input value u to be broadcast. We say that π is a t-resilient A-Cast protocol if the following properties hold for every t-adversary:

- ***Termination***:
  1. If the sender is correct and all correct processes participate in π, then every correct process eventually completes π.
  2. If some correct process completes π, then every correct process eventually completes π.
- ***Correctness***:
  1. All correct processes which complete π receive the same value v.
  2. If the sender is correct, then $v = u$.

3. Inferable Verifiable Secret Sharing

In this section, we first state the properties of our IVSS scheme. Then we provide an implementation of IVSS. We prove that our implementation satisfies all the IVSS properties and finally we analyze its fault-detection.

3.1. Definition

***Definition* 3** (Faulty Pair)**.** An unordered pair $\{i, j\}$ of processes is called a *faulty pair* if either i or j is faulty.

Our IVSS protocol consists of two subprotocols: $\mathcal{S}$ (*sharing* protocol) and $\mathcal{R}$ (*reconstruction* protocol). These two are invoked separately but $\mathcal{R}$ is never called unless $\mathcal{S}$ is completed, and $\mathcal{R}$ may not be called even if $\mathcal{S}$ is completed. If the correct processes do not reconstruct a same secret in $\mathcal{R}$, then a set of faulty pairs will be inferred. We assume that each IVSS invocation is unique for every correct process. This can be easily guaranteed, e.g. by associating with each IVSS invocation the identifier of the dealer and an invocation counter.

***Definition* 4** (IVSS)**.** Let $(\mathcal{S}, \mathcal{R})$ be any pair of sharing-reconstruction protocol with a dealer which has a secret s to share. We say that $(\mathcal{S}, \mathcal{R})$ is an IVSS protocol if the following properties (called *IVSS properties*) hold.

- ***Termination***:
  1. If the dealer is correct and all correct processes keep participating in protocol $\mathcal{S}$, then every correct process eventually completes protocol $\mathcal{S}$.
  2. If some correct process completes protocol $\mathcal{S}$, then every correct process that keeps participating in protocol $\mathcal{S}$ eventually completes protocol $\mathcal{S}$.
  3. If some correct process completes protocol $\mathcal{S}$ and all correct processes begin protocol $\mathcal{R}$ and keep participating in protocol $\mathcal{R}$, then every correct process eventually completes protocol $\mathcal{R}$.
  4. If some correct process completes protocol $\mathcal{R}$, then every correct process that keeps participating in protocol $\mathcal{R}$ eventually completes protocol $\mathcal{R}$.
- ***Correctness***: Once a correct process has completed protocol $\mathcal{S}$, then there is a unique value v such that the following holds.
  1. Either every correct process upon completing protocol $\mathcal{R}$ outputs v, or a set of new faulty pairs is eventually inferred by correct processes. (In our implementation, the size of the set of new faulty pairs is at least $t(n-3t)$.)
  2. If the dealer is correct, then $v = s$.
- ***Secrecy***: If the dealer is correct and no correct process invokes protocol $\mathcal{R}$, then the faulty processes have no information about secret s.

Note that, a correct process is said to keep participating in a protocol if it follows the protocol until completion. Another note is that we assume all secrets, random values, and polynomials to be over the integer ring.

3.2. Implementation

In our ABA protocol, the processes invoke a set of secret sharing instances in each round (starting from round 1). Every process records its invocations in each round r and A-Casts these invocations in the next round $r+1$ to let other processes know about its behavior in round r. We introduce a new component, which we call the *certification protocol*, to take care of the IVSS invocations from past rounds and infer faulty pairs. The certification protocol is invoked before round 1 and runs concurrently with all invocations of IVSS. Hence our IVSS protocol should be aware of the particular round it is involved in, and should make progress based on the data from past rounds. Therefore, we use the notion IVSS[r] with round number r as a parameter. In this section, we give a high-level description of our IVSS[r] and our certification protocols.

In the sharing phase, we assume that the dealer with secret s selects a random *degree-t symmetric bivariate polynomial* f such that $f(0,0) = s$. Let f_i denote the degree-t polynomial such that $f_i(y) = f(i, y)$ for $y \in \{1, \dots, n\}$. The dealer shares secret s by sending polynomial f_i to process i. By polynomial interpolation, if the dealer is correct, then any $t+1$ correct processes could reconstruct f. Since f is a symmetric polynomial, we should have $f_i(j) = f_j(i)$. Each process k that receives f_k sends $f_k(i)$ to process i. When k receives $f_i(k)$ from process i, k checks whether $f_i(k) = f_k(i)$. This equality may not be true since the dealer or process i could be faulty. If the equality is correct, then k A-Casts

Sharing protocol IVSS$[r]$-$\mathcal{S}$:

1. If the dealer wants to share secret s in round r, it selects a random degree-t symmetric bivariate polynomial $f(x, y)$ such that $f(0, 0) = s$. Let f_i denote the degree-t polynomial such that $f_i(y) = f(i, y)$ for $y \in \{1, \ldots, n\}$. The dealer sends f_i to process i.
2. If process k receives $\widehat{f_k}$ from the dealer, then k sends $\widehat{f_k}(i)$ to process i. (Note that $\widehat{f_k}$ is supposed to be f_k if the dealer is correct.)
3. If process k receives $\widehat{f_k}$ from the dealer and receives $\widehat{f_i(k)}$ from process i, and $\widehat{f_k}(i) = \widehat{f_i(k)}$, then k A-Casts "equal: (k, i)". (Note that $\widehat{f_i(k)}$ is supposed to be $\widehat{f_i}(k)$ if i is correct.)
4. If there is a set $\mathcal{M}$ of $n-t$ processes such that the following conditions are satisfied for the dealer:
   a) for every $i, j \in \mathcal{M}$, the dealer receives "equal: (i, j)";
   b) for every $i, j, p, q \in \mathcal{M}$, the dealer receives "checked$_r$: $p, q, \{i, j\}$" from p,

   then the dealer A-Casts $\mathcal{M}$. ($\mathcal{M}$ is called *candidate set*.)
5. If process k receives $\mathcal{M}$ from the dealer and the following conditions are satisfied:
   a) for every $i, j \in \mathcal{M}$, k receives "equal: (i, j)";
   b) for every $i, j, p, q \in \mathcal{M}$, k receives "checked$_r$: $p, q, \{i, j\}$" from p,

   then k completes the sharing protocol.

Reconstruction protocol IVSS$[r]$-$\mathcal{R}$:

1. If process $k \in \mathcal{M}$, then k A-Casts polynomial $\widehat{f_k}$.
2. If there is a set IS_k (standing for Interpolation Set) of $n - 2t$ processes such that
   a) k receives $\widetilde{f_i}$ from each process $i \in IS_k$; (Note that $\widetilde{f_i}$ is supposed to be $\widehat{f_i}$ if i is correct.)
   b) there is a symmetric bivariate degree-t polynomial $\bar{f}$ such that $\bar{f}(i, j) = \widetilde{f_i}(j)$ for all $i \in IS_k$ and $j \in \mathcal{M}$,

   then k sets $v = \bar{f}(0, 0)$, A-Casts "ready to complete" and adds this instance of IVSS$[r]$ to CoreInvocations$_r^k$.
3. If k completes Step 2 and receives "ready to complete" from $n - t$ processes, then k outputs v and completes the reconstruction protocol.

Certification protocol:

1. Process k initializes empty sets FP_k and CoreInvocations$_0^k$.
2. Process k sets CoreInvocations$_r^k = \varnothing$ and A-Casts CoreInvocations$_{r-1}^k$ in the beginning of round r ($r \geqslant 1$).
3. (*Infer faulty pairs*) If k receives CoreInvocations$_r^l$ from process l, then for any instance $\mathbb{I}$ in CoreInvocations$_r^l$, if k receives $\widetilde{f_i}$ and $\widetilde{f_j}$ from process i and j ($i, j \in \mathcal{M}$ of $\mathbb{I}$) in Step 1 of IVSS-$\mathcal{R}$ such that $\widetilde{f_i}(j) \neq \widetilde{f_j}(i)$, then k adds unordered pair $\{i, j\}$ to FP_k.
4. If k receives CoreInvocations$_r^l$ from process l, then for any invocation $\mathbb{I}$ in CoreInvocations$_r^l$, k completes the sharing protocol of $\mathbb{I}$ and Step 1 of IVSS$[r]$-$\mathcal{R}$ of $\mathbb{I}$. (Note that k does this because different process might complete different instances of IVSS$[r]$ in round r.)
5. If the following conditions are satisfied for process k (check in order a, b, c):
   a) k receives CoreInvocations$_{r'}^l$ from process l for all $r' < r$;
   b) for every IVSS invocation $\mathbb{I}$ in $\bigcup_{r'<r}$ CoreInvocations$_{r'}^l$, if i (j resp.) is included in the candidate set $\mathcal{M}$ of $\mathbb{I}$ then k should receive the polynomial A-Cast by i (j resp.) in Step 1 of IVSS-$\mathcal{R}$ of $\mathbb{I}$;
   c) $\{i, j\} \notin FP_k$ (Here FP_k has been updated after checking condition b, see Step 3),

   then k A-Casts "checked$_r; k, l, \{i, j\}$". (Intuitively, this means k has checked that $\{i, j\}$ is not a faulty pair according to the invocation history of l before round r.)

"equal: (k,i)". When the dealer receives "equal: i" from every process i in a set $\mathcal{M}$ that contains $n-t$ processes, and checks that $\mathcal{M}$ does not contains faulty pairs according to the IVSS invocations in the past rounds (see the description of the certification protocol below), the dealer A-Casts $\mathcal{M}$. Intuitively, $\mathcal{M}$ is a candidate set that processes could trust to reconstruct the secret. If process k receives set $\mathcal{M}$ from the dealer and checks the correctness of $\mathcal{M}$ as the dealer, then k completes the sharing protocol.

In the reconstruction phase, processes in $\mathcal{M}$ A-Cast their polynomials received from the dealer. When process k receives polynomials from $n-2t$ processes and these polynomials can be interpolated to a degree-t symmetric bivariate polynomial $\bar{f}$, k considers $\bar{f}(0,0)$ as the dealer's secret. If $\bar{f}$ is not equal to the polynomial f selected by the dealer in the sharing phase, we can show that a set of faulty pairs will be inferred. In order to get every secret sharing instance checked by the certification protocol in the next round, it is important that, when a correct process completes a secret sharing invocation, at least $t+1$ correct processes take this invocation as its history invocation. Therefore, after getting polynomial $\bar{f}$, k first A-Casts a message "ready to complete" and records the invocation. Then k completes the reconstruction phase if k receives $n-t$ "ready to complete".

Our certification protocol handles the history of invocations. Process k uses set FP_k to track the faulty pairs it inferred. In each round r, k records all invocations of IVSS[r] and adds them into a set called $\mathrm{CoreInvocations}_r^k$. Then, at the beginning of round $r+1$, k will A-Cast $\mathrm{CoreInvoations}_r^k$ to let other processes know its action in round r. Intuitively, this means that every correct process should know what the other processes have done in the past rounds. If a process k receives f_i from i and f_j from j but $f_i(j) \neq f_j(i)$ for some IVSS instance, then k knows that at least one of i, j is faulty and adds unordered pair $\{i,j\}$ into FP_k. The word "inferable" in IVSS means that correct pairs could infer faulty pairs during the execution. If k receives $\mathrm{CoreInvoations}_r^l$ from process l, then it checks for each invocation $\mathbb{I}$ in $\mathrm{CoreInvoations}_r^l$ that every correct process in $\mathcal{M}$ of $\mathbb{I}$ should A-Cast its polynomial in the beginning of the reconstruction phase, and no pair of correct processes should be considered as a faulty pair according to these invocations. If k has checked that an unordered pair $\{i,j\}$ is not a faulty pair according to the invocation history of l before round r, then k will A-Cast "$\mathsf{checked}_r : k, l, \{i,j\}$". In the sharing protocol, a correct process accepts a candidate set $\mathcal{M}$ only if every pair of processes in $\mathcal{M}$ are checked by every process in $\mathcal{M}$.

3.3. Proof of IVSS properties

Lemma 1. If i,j,k are correct processes, then unordered pair $\{i,j\}$ will not be added to FP_k.

Proof. The pair $\{i,j\}$ will be added to FP_k only if there is an invocation $\mathbb{I}$ of IVSS[r] such that $i,j \in \mathcal{M}$ and the polynomials $\widetilde{f_i}$ and $\widetilde{f_j}$ A-Casted by i and j in Step 1 of IVSS[r]-$\mathcal{R}$ satisfy $\widetilde{f_i}(j) \neq \widetilde{f_j}(i)$. However, if $i,j \in \mathcal{M}$ then i and j must have A-Casted "equal: (i,j)" and "equal: (j,i)" and hence must have checked that $\widetilde{f_i}(j) = \widetilde{f_j}(i)$ in IVSS[r]-$\mathcal{S}$. Thus $\{i,j\}$ will not be added to FP_k. □

Lemma 2. In round r ($r \geqslant 1$), if i, j, k, and l are correct processes, then k eventually A-Cast "$\mathsf{checked}_r : k, l, \{i,j\}$".

Proof. Since $\{i,j\}$ is not in FP_k by Lemma 1, we only need to check conditions ***a*** and ***b*** of Step 5 in the certification protocol.

Condition ***a***: Since l is correct, l will A-Cast $\mathrm{CoreInvocations}_r^l$ in the beginning of round r. Then k will receive these $\mathrm{CoreInvocations}_r^l$ by the correctness property of A-Cast.

Condition ***b***: Suppose that i is in the set $\mathcal{M}$ of an IVSS[r'] invocation $\mathbb{I}$ in $\bigcup_{r'<r} \mathrm{CoreInvocations}_{r'}^l$. Since l adds $\mathbb{I}$ into its CoreInvocations, l must have completed the sharing protocol of $\mathbb{I}$. Then i must have received polynomial $\widehat{f_i}$ from the dealer in invocation $\mathbb{I}$. According to Step 4 of the certification protocol, i will complete Step 1 of IVSS[r']-$\mathcal{R}$ of $\mathbb{I}$. So k will receive the polynomial A-Casted by i in Step 1 of IVSS[r]-$\mathcal{R}$ of $\mathbb{I}$.

Taking above together, k will A-Cast "$\textsf{checked}_r : k, l, \{i,j\}$". □

Lemma 3. Let N be a subset of $\{1,\dots,n\}$ and $|N| \geqslant t+1$. Let $\{f_i\}_{i\in N}$ be a set of degree-t univariate polynomials. If $f_i(j) = f_j(i)$ for all $i,j \in N$, then there is a unique symmetric bivariate degree-t polynomial f such that $f(i,j) = f_i(j)$ for all $i,j \in N$.

Proof. Select any subset N_0 of N such that $|N_0| = t+1$. Let

$$f_0(x,y) = \sum_{i\in N_0; j\in N_0} \frac{\prod\limits_{k\in N_0, k\neq i}(x-k) \prod\limits_{k\in N_0, k\neq j}(y-k)}{\prod\limits_{k\in N_0, k\neq i}(i-k) \prod\limits_{k\in N_0, k\neq j}(j-k)} f_i(j).$$

By Lagrange interpolation, $f_0(i,j) = f_i(j)$ for all $i,j \in N_0$. Since $f_i(j) = f_j(i)$, f_0 is a symmetric bivariate degree-t polynomial by definition. Now we prove that $f_0(i,j) = f_i(j)$ for all $i,j \in N$.

Consider any arbitrary i in N. We have $f_i(j) = f_j(i) = f_0(j,i)$ for all $j \in N_0$. Since f_0 is symmetric, we have $f_i(j) = f_0(i,j)$ for all $j \in N_0$. Since $|N_0| = t+1$, we have $f_i(y) = f_0(i,y)$ for any y. Especially, we have $f_i(j) = f_0(i,j)$ for all $j \in N$. Hence, f_0 satisfies $f_0(i,j) = f(i,j)$. The uniqueness follows easily from Lagrange interpolation. □

Theorem 1. Assume $n > 3t$. Then the pair (IVSS[r]-$\mathcal{S}$, IVSS[r]-$\mathcal{R}$) satisfies all the IVSS properties.

Proof. We check below the IVSS properties.

Termination (1): Suppose the dealer is correct and all correct processes keep participating in IVSS[r]-$\mathcal{S}$. Every correct process will receive correct messages from the dealer. Then for each pair (i,j) of correct processes, i will A-Cast "$\textsf{equal}$: (i,j)". By Lemma 2, for correct processes i,j,k,l, "$\textsf{checked}_r : k, l, \{i,j\}$" will be A-Cast by k. Thus the set of correct processes will satisfy the conditions in Step 4 of IVSS[r]-$\mathcal{S}$. Therefore, a correct dealer will A-Cast a set $\mathcal{M}$ with respect to Step 4 of IVSS[r]-$\mathcal{S}$. Since all messages that the dealer received in Step 4 are sent using A-Cast, it follows that all correct processes will receive $\mathcal{M}$ and check that $\mathcal{M}$ satisfies the conditions in Step 5 of IVSS[r]-$\mathcal{S}$. Hence, every correct process will complete IVSS[r]-$\mathcal{S}$.

Termination (2): If a correct process completes IVSS[r]-$\mathcal{S}$, then, since all messages required in Step 5 of IVSS[r]-$\mathcal{S}$ are sent by A-Casting, every correct process that keeps participating in IVSS[r]-$\mathcal{S}$ will receive these messages and complete IVSS[r]-$\mathcal{S}$.

Termination (3): If some correct process completes protocol IVSS[r]-$\mathcal{S}$ and all correct processes begin IVSS[r]-$\mathcal{R}$ and keep participating, we show that every correct process will complete IVSS[r]-$\mathcal{R}$. Let C be the set of correct processes in $\mathcal{M}$. Since $|\mathcal{M}| \geqslant n-t$, then $|C| \geqslant n-2t$. Let $\widehat{f_i}$ be the polynomial i ($i \in C$) received from the dealer. Since $C \subset \mathcal{M}$, we have $\widehat{f_i}(j) = \widehat{f_j}(i)$ for all $i,j \in C$. By Lemma 3, there is a symmetric bivariate degree-t polynomial $\bar{f}$ such that $\bar{f}(i,j) = \widehat{f_i}(j)$ for all $i,j \in C$. Thus C satisfies the conditions in Step 2 of IVSS[r]-$\mathcal{R}$. It follows that every correct process will complete Step 2 of IVSS[r]-$\mathcal{R}$ and A-Casts "$\textsf{ready to complete}$". Therefore, every correct process will receive at least $n-t$ "$\textsf{ready to complete}$" messages and complete IVSS[r]-$\mathcal{R}$.

Termination (4): If a correct process completes IVSS[r]-$\mathcal{R}$, then, since all messages required for completing IVSS[r]-$\mathcal{R}$ are sent by A-Casting, every correct process that keeps participating in IVSS[r]-$\mathcal{R}$ will receive these messages and complete IVSS[r]-$\mathcal{R}$.

We now turn to the correctness properties.

Suppose that a correct process has completed the sharing protocol. By Lemma 3, there is a symmetric bivariate degree-t polynomial $\bar{f}$ such that $\bar{f}(i,j) = \widehat{f_i}(j)$ for all $i,j \in C$ where C is the set of all correct processes in $\mathcal{M}$. We denote $\bar{f}(0,0)$ as v.

Correctness (1): If some correct process k completes IVSS[r]-$\mathcal{R}$ and outputs a value different from v, then IS_k must be different from C. And there must be some process $i \in IS_k$ and some process $j \in C$ such that $\bar{f}(i,j) \neq \widehat{f_i}(j)$, otherwise IS_k also interpolates $\bar{f}$ and output $\bar{f}(0,0)$. Since $\bar{f}(i,j) = \bar{f}(j,i) = \widehat{f_j}(i)$, we have $\widehat{f_i}(j) \neq \widehat{f_j}(i)$, which means some faulty pair will be inferred (we will analysis how many pairs could be inferred in the following section).

Correctness (2): If the dealer is correct, then $\widehat{f_i}(j) = f(i,j)$ for all $i, j \in C$ where f is the polynomial selected by the dealer. Thus $\bar{f} = f$ and $v = \bar{f}(0,0) = f(0,0) = s$.

Secrecy: By polynomial interpolation, the combined view of the t faulty processes is not enough to compute the initial random degree-t polynomial selected by the dealer. As long as no correct process invokes IVSS[r]-$\mathcal{R}$, the shared secret is independent of the information obtained by the faulty processes. Hence, the faulty processes have no information of the shared secret.

So all the IVSS properties hold for IVSS[r]. The theorem follows. □

3.4. Fault-Detection Analysis

We introduce the following convention for the analysis of Fault-Detection in the certification protocol. Consider an instance $\mathbb{R}$ of IVSS[r]-$\mathcal{R}$ in $\mathrm{CoreInvocations}_r^i$ for a correct process i. If faulty process l in $\mathcal{M}$ of $\mathbb{R}$ does not send its polynomial in Step 1 of $\mathbb{R}$, then in round r' (greater than r), no correct process will allow l to appear in $\mathcal{M}$ of IVSS[r'] (see condition (b) of Step 5 in IVSS[r']-$\mathcal{S}$ and Step 5 of the certification protocol). This is the best case for correct processes. Therefore without loss of generality, we use the following convention.

Convention. *In any instance of IVSS[r]-$\mathcal{R}$ and any round r, every faulty process in the corresponding set $\mathcal{M}$ eventually A-Casts a polynomial (can be arbitrary) according to Step 1 of IVSS[r]-$\mathcal{R}$.*

Consider an arbitrary instance of IVSS[r]. With the above convention, let $\widehat{f_i}$ be the polynomial eventually A-Casted by process $i \in \mathcal{M}$ in Step 1 of IVSS[r]-$\mathcal{R}$. We say that a set $S \subset \mathcal{M}$ of at least $n-2t$ processes is an *interpolation set* if there is a symmetric bivariate degree-t polynomial g such that $g(i,j) = \widehat{f_i}(j)$ for all $i \in S$. Two interpolation sets S and S' are different, if the corresponding bivariate polynomial are different, which implies $|S \cap S'| \leqslant t$ by Lemma 3.

For an instance $\mathbb{I}$ of IVSS[r], recall that by Lemma 3 the polynomials that processes in C received from the dealer actually define a unique symmetric bivariate degree-t polynomial, and therefore define a unique secret s. We say that s is the secret defined by $\mathbb{I}$.

***Definition* 5.** Let $\mathbb{E}$ be the event that at least one of the correct processes output a value s' in the reconstruction phase of $\mathbb{I}$ such that $s' \neq s$.

If $\mathbb{E}$ never occurs, then we could get a common coin with high probability (we will show this later in Section 4). Thus it is significant to analyze the situation when $\mathbb{E}$ occurs.

Lemma 4. $\mathbb{E}$ could only occur in some instance $\mathbb{I}$ of IVSS[r] when $n \leqslant 4t$.

Proof. If $\mathbb{E}$ occurs, then there are at least two different *interpolation sets*. One of these is the set C of correct processes in $\mathcal{M}$, the other one is the interpolation set IS causing some correct process to output a different secret. Since $|C| \geqslant n-2t$, $|IS| \geqslant n-2t$, $|C \cap IS| \leqslant t$, we have $|C \cup IS| = |C| + |IS| - |C \cap IS| \geqslant 2n-5t$. If $n > 4t$, then $|C \cup IS| > n-t$. This is impossible since $|C \cup IS| \leqslant |\mathcal{M}| = n-t$. Therefore, $\mathbb{E}$ could only occur when $n \leqslant 4t$. □

Lemma 5. If $\mathbb{E}$ occurs in some instance $\mathbb{I}$ of IVSS[r], then at least $t(n-3t)$ faulty pairs will be inferred by every correct process due to $\mathbb{I}$.

Proof. When $\mathbb{E}$ occurs, at least one correct process completes instance $\mathbb{I}$. According to Step 3 of IVSS[r]-$\mathcal{R}$, there are at least $n-2t$ correct processes that have A-Casted "ready to complete". According to Step 2 of IVSS[r]-$\mathcal{R}$, these correct processes must have added $\mathbb{I}$ into the set CoreInvocations. In the next round $r+1$, the candidate set $\mathcal{M}$ of any instance of IVSS[$r+1$] will contain at least one of these $n-2t$ processes since $\mathcal{M} = n-t$ and $n-2t > t$. Thus $\mathbb{I}$ will be checked by every correct process in the certification protocol. Since the faulty pairs are inferred from the polynomials A-Casted by processes in $\mathcal{M}$ of $\mathbb{I}$, all correct processes will infer the same faulty pairs. So we only need to prove the lemma for correct process k.

Let $\{S_1, S_2 \ldots, S_r\}$ be all maximal interpolation sets with respect to the inclusion relation of sets. Since $\mathbb{E}$ occurs, there must be at least two maximal interpolation sets, one of which implies the secret s defined by $\mathbb{I}$ and another of which implies the secret $s' \neq s$, i.e. $r \geqslant 2$. Suppose $i, j \in 1, \ldots, r$ and $i \neq j$. By the assumption of maximal interpolation sets, $|S_i \cap S_j| \leqslant t$. Let S_0 be the interpolation set in $\{S_1, S_2 \ldots, S_r\}$ with the smallest cardinal number. Since $|S_i \cup S_j| \leqslant |\mathcal{M}| = n-t$ and $|S_i \cap S_j| \leqslant t$, then $|S_i| + |S_j| = |S_i \cup S_j| + |S_i \cap S_j| \leqslant n$. Therefore $|S_0| \leqslant \frac{|S_i|+|S_j|}{2} \leqslant \frac{n}{2}$. Also from the definition of the interpolation set, we have $|S_0| \geqslant n-2t$.

Suppose the corresponding symmetric bivariate polynomial for S_0 is f^0. Let f_i^0 be the polynomial with $f_i^0(j) = f^0(i,j)$. Since f^0 is symmetric, $f_i^0(j) = f^0(i,j) = f^0(j,i) = \widehat{f_j}(i)$ for every $j \in S_0$. Recall that $\widehat{f_j}$ is the polynomial eventually A-Casted by process $j \in \mathcal{M}$ in Step 1 of IVSS[r]-$\mathcal{R}$ of instance $\mathbb{I}$. For any $i \in \mathcal{M}$ but $i \notin S_0$, we have $\widehat{f_i} \neq f_i^0$ because otherwise $S_0 \cup i$ is an interpolation set bigger than S_0, which contradicts the fact that S_0 is maximal. Since $\widehat{f_i} \neq f_i^0$, $\widehat{f_i} - f_i^0$ has at most t zero points. So there are at least $|S_0| - t$ processes j in S_0 such that $\widehat{f_i}(j) \neq f_i^0(j)$, i.e. $\widehat{f_i}(j) \neq \widehat{f_j}(i)$ (since $f_i^0(j) = \widehat{f_j}(i)$ for $j \in S_0$) which leads to the faulty pair $\{i,j\}$. Therefore, for each $i \in \mathcal{M}$ but $i \notin S_0$, k will infer at least $|S_0| - t$ faulty pairs. In total, k could infer at least $(|S_0| - t)(n - t - |S_0|)$ faulty pairs. Since $n - 2t \leqslant |S_0| \leqslant \frac{n}{2}$, then $(|S_0| - t)(n - t - |S_0|) \geqslant (n-3t)t$. The lemma is proved. $\square$

In the lemma above, we show that a set of faulty pairs will eventually be inferred if $\mathbb{E}$ occurs in an instance of IVSS[r]. However, "eventually" is not enough to improve running time. In the next lemma, we will show that the faulty pairs inferred from instance of IVSS[r] will not appear in candidate set $\mathcal{M}$ of IVSS[$r+1$] even though these faulty pairs might be inferred after the invocation of IVSS[$r+1$].

Lemma 6. If $\mathbb{E}$ occurs in some instance $\mathbb{I}^r$ of IVSS[r], and $\{i,j\}$ is eventually inferred as faulty pairs by the correct processes due to $\mathbb{I}^r$, then i and j could not appear simultaneously in the set $\mathcal{M}$ of any instance of IVSS[r'] with $r' > r$.

Proof. Since $\mathbb{E}$ occurs, there must be a correct process (say k) that completes instance $\mathbb{I}^r$. Then, by Step 3 of IVSS[r]-$\mathcal{R}$, k must have received "ready to complete" from $n-t$ processes. According to Step 2 of IVSS[r]-$\mathcal{R}$, these $n-t$ processes must have added instance $\mathbb{I}^r$ into $\mathrm{CoreInvocations}_r^*$. Then there are at least $n-2t$ correct processes (denoted by S) that have added instance $\mathbb{I}^r$ into $\mathrm{CoreInvocations}_{*,r}$.

Now consider round $r' > r$. In any instance $\mathbb{I}^{r'}$ of IVSS[r'], set $\mathcal{M}$, A-Casted by the dealer, contains at least one correct process (denoted by l) from S since $|\mathcal{M}| \geqslant n-t$ and $|S| \geqslant n-2t \geqslant t+1$. If i and j are both in the set $\mathcal{M}$ of $\mathbb{I}^{r'}$, then every correct process k' in $\mathcal{M}$ must A-Cast "$\mathsf{checked}_{r'}, k', l, \{i,j\}$" according to Step 5 of IVSS[r']-$\mathcal{S}$. By Step 5 of our certification protocol, k' must have received the corresponding polynomials of i and j A-Cast in Step 1 of IVSS[r']-$\mathcal{R}$. However, this would make k' add $\{i,j\}$ into $FP_{k'}$ and not A-Cast "$\mathsf{checked}_{r'} : k', l, \{i,j\}$". This is a contradiction. Therefore, i and j could not appear simultaneously in candidate set $\mathcal{M}$ of any instance of IVSS[r'] with $r' > r$. $\square$

Lemma 7. If $n = 3t + \delta$, then there are at most $\frac{3t}{\delta} + 1$ rounds where $\mathbb{E}$ occurs.

Proof. Suppose $\mathbb{E}$ occurs in round $r_1, r_2, \ldots, r_c$ and denote the faulty pairs that could be inferred for these rounds by $S_1, S_2, \ldots, S_c$. By Lemma 6, S_i is different from S_j for $1 \leqslant i, j \leqslant c$ and $i \neq j$. According to Lemma 5, there will be at least $c \cdot (n-3t)t$ different faulty pairs inferred. Since each faulty process can only appear in n faulty pairs, we have $t \cdot n \geqslant c \cdot (n-3t)t$. Thus, $c \leqslant \frac{tn}{(n-3t)t} = \frac{3t}{\delta} + 1$. $\square$

4. From IVSS to Asynchronous Byzantine Agreement

Using our IVSS[r] protocol, we now design an ABA protocol (following the reduction scheme of Canetti and Rabin [6]). The first step is to get a common coin. In the common coin protocol of [6], every process shares n random secrets using n different invocations of the AVSS protocol of [6]. Following Figure 5-9 of [5] and using our IVSS[r] protocol, we obtain an *Inferable Common Coin* (ICC) protocol which always terminates.

***Definition* 6** (ICC)**.** Let π be any protocol where every process has a random input and a binary output. We say that π is a terminating, t-resilient Inferable Common Coin protocol if the following properties (called *ICC properties*) hold for every t-adversary.

- ***Termination***.
  1. If all correct processes keep participating in π, then every correct process eventually completes.
  2. If some correct process completes π, then every other correct process that keeps participating in π eventually completes.
- ***Correctness***. For every invocation, either
  - for each $v \in \{0, 1\}$, with probability at least $1/4$, every correct process upon completing π outputs v; or
  - a set of faulty pairs is eventually inferred by correct processes.

Lemma 8. For $n > 3t$ and each round r, there is a terminating, t-resilient Inferable Common Coin protocol.

Proof. The protocol implementing ICC by using our IVSS[r] subprotocol is a slight variant of figure 5-9 of [5]. We call this protocol ICC[r]. The proof is in Appendix II. □

The second step is to use the common coin protocol to get an ABA protocol. In [6], Canetti and Rabin use their common coin protocol (that terminates with probability $1 - \varepsilon$) to get an ABA protocol (that terminates with probability $1 - \varepsilon$). We replace the common coin protocol of [6] by ICC[r] to obtain our almost-surely terminating ABA protocol.

Theorem 2 (Byzantine Agreement)**.** If $n = 3t + \delta$, then there is an almost-surely terminating ABA protocol with expected running time $O(\frac{t}{\delta})$.

Proof. By Lemma 7, we know there are at most $\frac{3t}{\delta} + 1$ rounds in which the adversary could break the correctness of secret sharing. In the rest of the rounds, all correct processes reconstruct the same value and this value is equal to the secret of the dealer if the dealer is correct, with which we can have a common coin that is sufficient for Byzantine agreement with constant expected running time. Therefore, the expected running time of our ABA protocol is $O(\frac{t}{\delta})$. we give the details in Appendix III. □

If we take $\delta = 1$ in the above theorem, we have the following corollary, which improves the result of Abraham, Dolev, and Halpern [1].

Corollary 1. If $n = 3t + 1$, then there is an almost-surely terminating, optimally resilient ABA protocol with expected running time $O(t)$.

If we take $\delta = \varepsilon t$ where $\varepsilon > 0$, we have the following corollary, which improves the result of Feldman and Micali [7].

Corollary 2. If $n = (3 + \varepsilon)t$ where $\varepsilon > 0$, then there is an almost-surely terminating ABA protocol with expected running time $O(1/\varepsilon)$.

References

[1] Abraham, I., Dolev, D., and Halpern, J. Y. An almost-surely terminating polynomial protocol for asynchronous Byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing* (2008), PODC '08, ACM, pp. 405–414.

[2] Beerliová-Trubíniová, Z., and Hirt, M. Efficient multi-party computation with dispute control. In *Proceedings of the Third Conference on Theory of Cryptography* (2006), TCC '06, Springer-Verlag, pp. 305–328.

[3] Ben-Or, M. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing* (1983), PODC '83, ACM, pp. 27–30.

[4] Bracha, G. An asynchronous [(n-1)/3]-resilient consensus protocol. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing* (1984), PODC '84, ACM, pp. 154–162.

[5] Canetti, R. *Studies in secure multiparty computation and applications*. PhD thesis, The Weizmann Institute of Science, 1996.

[6] Canetti, R., and Rabin, T. Fast asynchronous Byzantine agreement with optimal resilience. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing* (1993), STOC '93, ACM, pp. 42–51.

[7] Feldman, P., and Micali, S. Optimal algorithms for byzantine agreement. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (1988), STOC '88, ACM, pp. 148–161.

[8] Fischer, M. J., Lynch, N. A., and Paterson, M. S. Impossibility of distributed consensus with one faulty process. *J. ACM 32*, 2 (1985), 374–382.

[9] Karlin, A., and Yao, A. Probabilistic lower bounds for Byzantine agreement. *Unpublished document* (1986).

[10] King, V., and Saia, J. Faster agreement via a spectral method for detecting malicious behavior. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (2014), SODA '14, SIAM, pp. 785–800.

[11] Lamport, L., Shostak, R., and Pease, M. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst. 4*, 3 (1982), 382–401.

[12] Pease, M., Shostak, R., and Lamport, L. Reaching agreement in the presence of faults. *J. ACM 27*, 2 (1980), 228–234.

[13] Rabin, M. O. Randomized Byzantine generals. In *Proceedings of the Twenty-Fourth Annual Symposium on Foundations of Computer Science* (1983), FOCS '83, IEEE Computer Society, pp. 403–409.

Appendix I.
A-Cast Protocol

***Definition* 7** (A-Cast)**.** Let π be any asynchronous protocol initiated by a designated process (the sender) which has an input value u to be broadcast. We say that π is a t-resilient A-Cast protocol if the following properties hold for every t-adversary:

- ***Termination***:
  1. If the sender is correct and all correct processes participate in π, then every correct process eventually completes π.
  2. If some correct process completes π, then every correct process eventually completes π.
- ***Correctness***:
  1. All correct processes which complete π receive the same value v.
  2. If the sender is correct, then $v = u$.

A-Cast Protocol:

1. The sender with input u sends "msg: u" to all processes.
2. i waits until receiving "msg: u". Then i sends "echo: u" to all processes.
3. i waits until receiving $n - t$ "echo: u'" that agree on the value of u'. Then i sends "ready: u'" to all processes.
4. i waits until receiving $t + 1$ "ready: u'" that agree on the value of u'. Then i sends "ready: u'" to all processes.
5. i waits until receiving $2t+1$ "ready: u'" that agree on the value of u'. Then i outputs u' and completes the protocol.

Appendix II.
Inferable Common Coin Protocol

***Definition* 8** (ICC)**.** Let π be any protocol where every process has a random input and a binary output. We say that π is a terminating, t-resilient Inferable Common Coin protocol if the following properties (called *ICC properties*) hold for every t-adversary.

- ***Termination***.
  1. If all correct processes keep participating in π, then every correct process eventually completes.
  2. If some correct process completes π, then every other correct process that keeps participating in π eventually completes.
- ***Correctness***. For every invocation, either
  - for each $v \in \{0, 1\}$, with probability at least $1/4$, every correct process upon completing π outputs v; or
  - a set of faulty pairs is eventually inferred by correct processes.

Our implementation (called ICC[r]) of ICC follows [6]. Roughly speaking, the protocol consists of two phases. First, every process shares n random secrets using our IVSS[r]-$\mathcal{S}$ protocol. The ith secret shared by each process is assigned to process i. Once a process i completes $t + 1$ sharing protocols of secrets assigned to it, i A-Casts the identity of the dealers of these secrets. After this, by the correctness property of IVSS[r], a fixed value (yet unknown) is *attached* to i. The second phase is to select a subset of processes (say H) and reconstruct the attached values of H. Different processes may choose different H to reconstruct secrets. However, if an instance of IVSS[r]-$\mathcal{R}$ is invoked by a strict subset of correct processes, then there is no guarantee of termination. Hence, in ICC[r] we require every process to A-Cast its H before completion, so that each process could try to reconstruct values with different H.

ICC[r] protocol: code for process i

1. Choose a random value $x_{i,j}$ for all $1 \leqslant j \leqslant n$ and invoke IVSS[r]-$\mathcal{S}$ as a dealer for this value. Denote this execution by IVSS[r]-$\mathcal{S}(x_{i,j})$.
2. Participate in IVSS[r]-$\mathcal{S}(x_{j,k})$ for every $j, k \in \{1, \ldots, n\}$.
3. Define a set $\mathcal{T}_i$. Add process j to $\mathcal{T}_i$ if all IVSS[r]-$\mathcal{S}(x_{j,l})$ have been completed for all $1 \leqslant l \leqslant n$. Wait until $|\mathcal{T}_i| = t+1$, then assign $T_i = \mathcal{T}_i$ and A-Cast "attach T_i to i". (we say that the secrets $\{x_{j,i} | j \in T_i\}$ are attached to process i.)
4. Define a set $\mathcal{A}_i$. Add process j to $\mathcal{A}_i$ if the A-Cast "attach T_j to j" has been completed and $T_j \subseteq \mathcal{T}_i$. Wait until $|\mathcal{A}_i| = n - t$, then assign $A_i = \mathcal{A}_i$ and A-cast "i accepts A_i".
5. Define a set $\mathcal{S}_i$. Add process j to $\mathcal{S}_i$ if "j accepts A_j" is received from j and $A_j \subseteq \mathcal{A}_i$. Wait until $|\mathcal{S}_i| = n - t$, then A-Cast "Reconstruct Enabled". Let S_i denote the current content of $\mathcal{S}_i$ and H_i denote the current content of $\mathcal{A}_i$. Then A-Cast (H_i, S_i).
6. Participates in IVSS[r]-$\mathcal{R}(x_{k,j})$ for every $k \in T_j$ and $j \in \mathcal{A}_i$. Let $y_{k,j}$ be the corresponding output.
7. Let $u = \lceil 0.87n \rceil$. Every process $j \in \mathcal{A}_i$ is associated with a value, say v_j, which is computed as follows: $v_j = \left(\sum_{k \in T_j} y_{k,j}\right) \bmod u$.
8. Wait until receiving $(\widetilde{H_j}, \widetilde{S_j})$ from j with $\widetilde{H_j} \subseteq \mathcal{A}_i$ and $\widetilde{S_j} \subseteq \mathcal{S}_i$ and the values associated with all processes in $\widetilde{H_j}$ are computed. Now if there exists a process $k \in \widetilde{H_j}$ such that $v_k = 0$, then output 0. Otherwise output 1.

We now state and prove the following lemmas which are slight variants of lemmas 5.28-5.31 presented in [5].

Lemma 9. If some correct process completes ICC[r], then every other correct process that keeps participating in ICC[r] eventually completes.

Proof. If a correct process i completes ICC[r] with respect to $(\widetilde{H_j}, \widetilde{S_j})$, then, since all messages are sent by A-Casting, every correct process that keeps participating in ICC[r] will receive at least $t+1$ $(\widetilde{H_j}, \widetilde{S_j})$ as well. By the termination property (4) of IVSS[r], every correct process that keeps participating will also compute the values associated with all the processes in $\widetilde{H_j}$ and then complete the protocol. □

Lemma 10. If all correct processes keep participating in ICC[r], then all correct processes complete ICC[r] in constant time.

Proof. First we show that every correct process will A-Cast "Reconstruct Enabled". By termination property (1) of our IVSS[r] protocol, every correct process eventually completes IVSS-$\mathcal{S}(x_{j,k})$ for every $k \in \{1, \ldots, n\}$ and correct j. Since there are at least $n - t$ correct processes, for each correct process i, $\mathcal{T}_i$ will eventually contain at least $t+1$ (actually $n-t$) processes and thus i will eventually A-Cast "attach T_i to i". So eventually, correct process i will receive "attach T_j to j" from every correct process j. Now since every process k that is included in $\mathcal{T}_j$ will be eventually included in $\mathcal{T}_i$ (by termination property (2) of IVSS[r]), $T_j \subseteq \mathcal{T}_i$ will eventually hold. Therefore, every correct process j will eventually be included in $\mathcal{A}_i$. Thus for every correct process i, $\mathcal{A}_i$ will eventually be of size $n-t$ and hence i will A-Cast "i accepts A_i". Following the same argument, $\mathcal{S}_i$ will be of size $n-t$ and hence i will A-Cast "Reconstruct Enabled" and A-Cast (H_i, S_i).

We now show that all correct processes will complete ICC[r]. By the lemma above, we only need to show that at least one of the correct processes will complete ICC[r]. Suppose by contradiction that no correct process will complete ICC[r]. Let i be a correct process. If i receives "attach T_j to j"

from j and includes j in $\mathcal{A}_i$, then eventually every other correct process will do the same. Hence if i invokes IVSS[r]-$\mathcal{R}(x_{k,j})$ for $k \in T_j$ and $j \in \mathcal{A}_i$, then eventually every other correct process will also invoke IVSS[r]-$\mathcal{R}(x_{k,j})$. By termination property (3) of IVSS[r], all correct processes will complete this IVSS[r]-$\mathcal{R}(x_{k,j})$. Therefore, the values associated with all processes in H_i will be computed. So i will complete the protocol, in contradiction with the assumption that no correct process will complete. Therefore, all correct processes will complete.

In the certification protocol, each process need to check all the past invocations in past rounds. However, since every correct process does this in every round, it is equal to that each process in every round checks all the invocations in the previous round. Therefore, all "$\mathsf{checked}_r; k, l, \{i, j\}$" could be finished in constant time for correct process i, j, k, l. Thus all invocations of IVSS[r]-$\mathcal{S}$ and IVSS[r]-$\mathcal{R}$ in ICC[r] complete in constant time. Since all A-Casts also complete in constant time, our ICC[r] protocol completes in constant time as well. □

Lemma 11. In ICC[r], once some correct process j receives "attach T_i to i" from the A-Cast of i, a unique value v_i is fixed such that

1. Every correct process will associate v_i with i or a set of faulty pairs will eventually be inferred by correct processes.
2. Value v_i is distributed uniformly over $[0, \ldots, u-1]$ and is independent of the values associated with the other processes.

Proof. The correctness property of IVSS[r] ensures that for each $k \in T_i$ there is a fixed value $y_{k,i}$ such that all correct processes will output $y_{k,i}$ in IVSS[r]-$\mathcal{S}(x_{k,i})$ or a set of faulty pairs will be inferred. Let $v_i = \left(\sum_{k \in T_i} y_{k,i}\right) \bmod u$, then every correct process will associate v_i with i except that event $\mathbb{E}$ occurs in some instances of IVSS[r], i.e., a set of faulty pairs will eventually be inferred by correct processes.

It remains to show that v_i is uniformly distributed over $[0, \ldots, u-1]$, and is independent of the values associated with the other processes. A correct process starts reconstructing the secrets attached to process i only after it completes the "attach T_i to i" A-Cast. So the set T_i is fixed before any correct process invokes IVSS[r]-$\mathcal{R}(x_{k,i})$ for some process k. The secrecy property of IVSS[r] now ensures that, by the time the set T_i is fixed, the adversary view of the invocations of IVSS[r]-$\mathcal{S}(x_{k,i})$ where the dealers are correct is distributed independently of the shared values. Since T_i contains at least one correct process and every correct process's shared secrets are uniformly distributed and mutually independent, the sum v_i is uniformly and independently distributed over $[0, \ldots, u-1]$. □

Lemma 12. Once a correct process A-Casts "Reconstruct Enabled", there is a set M such that

1. For every process $j \in M$, some correct process has received "attach T_j to j" from the A-Cast of j.
2. If any correct process k receives $(\widetilde{H_j}, \widetilde{S_j})$ from j with $\widetilde{H_j} \subseteq \mathcal{A}_k$ and $\widetilde{S_j} \subseteq \mathcal{S}_k$ and the values associated with all processes in $\widetilde{H_j}$ are computed, then $M \subseteq \widetilde{H_j}$.
3. $|M| \geqslant \frac{n}{3}$.

Proof. Let i be the first correct process to A-Cast "Reconstruct Enabled". Let M be the set of processes, k, for which $k \in A_l$ for at least $t+1$ processes $l \in S_i$. We now show that all processes in M satisfy the properties of the lemma.

It is clear that $M \subseteq H_i$. Thus process i has received "attach T_j to j" for every $j \in M$. Since i is assumed to be correct, the first part of the lemma is proved.

We now prove the second part. First $\widetilde{S_j}$ contains $n - t \geqslant 2t + 1$ processes. Now if $k' \in M$ then k' belongs to A_l for at least $t+1$ processes $l \in S_i$. This ensures that there is at least one process l which belongs to $\widetilde{S_j}$ as well as S_i. Now $l \in \widetilde{S_j}$ implies that j has ensured that $A_l \subseteq \widetilde{H_j}$. Consequently, $k' \in \widetilde{H_j}$.

It remains to show that $|M| \geqslant \frac{n}{3}$. We use a counting argument for this purpose. Let $h = |H_i|$. We have $h \geqslant n - t$. Consider the $h \times n$ table Λ (relative to process i), where $\Lambda_{l,k} = \text{one}$ iff i has received

"l accepts A_l" from l before A-Casting "Reconstruct Enabled" and $k \in A_l$. Then M is the set of processes k such that the kth column in Λ has at least $t+1$ one entries. There are $n-t$ one entries in each row of Λ; thus there are $h(n-t)$ one entries in Λ.

Let m denote the minimum number of columns in Λ that contain at least $t+1$ one entries. We show that $m \geqslant \frac{n}{3}$. Clearly, the worst distribution of one entries in Λ is letting m columns be all one entries and letting each of the remaining $n-m$ columns have t one entries. This distribution requires the number of one entries to be no more than $mh + (n-m)t$. Thus, we must have:

$$mh + (n-m)t \geqslant h(n-t).$$

This gives $m \geqslant \frac{h(n-t)-\mathrm{nt}}{h-t}$. Since $h \geqslant n-t$ and $n \geqslant 3t+1$, we have

$$m \geqslant \frac{(n-t)^2 - nt}{n-2t} = n - 2t + \frac{nt - 3t^2}{n-2t} \geqslant n - 2t \geqslant \frac{n}{3}.$$

This shows that $|M| \geqslant \frac{n}{3}$.

□

Lemma 13. For every invocation of ICC[r], either
- For each $v \in \{0,1\}$, with probability at least $1/4$, all correct processes output v; or
- A set of faulty pairs will eventually be inferred by correct processes.

Proof. If $\mathbb{E}$ occurs in any instance of IVSS[r] while executing ICC[r], then a set of faulty pairs will be inferred by correct processes. We prove the first part of the lemma assuming $\mathbb{E}$ does not occur. Suppose correct process j completes ICC[r] with respect to $(\widetilde{H_k}, \widetilde{S_k})$. Since $\mathbb{E}$ does not occur, by Lemma 11, for every process i in $\mathcal{A}_j$, there is a fixed value v_i that is distributed uniformly and independently over $[0, \ldots, u-1]$. Now we consider two cases:
- Let M be the set of processes discussed in the lemma above. Clearly if $v_i = 0$ for some $i \in M$, then all correct processes associate 0 with j and output 0. The probability that at least one process $i \in M$ has $v_i = 0$ is $1 - \left(1 - \frac{1}{u}\right)^{|M|}$. Since $u = \lceil 0.87n \rceil$, $n \geqslant 4$, and $|M| \geqslant \frac{n}{3}$ by Lemma 12, we have $1 - \left(1 - \frac{1}{u}\right)^{|M|} \geqslant 1 - e^{-0.29} \geqslant 0.25$. This implies that all correct processes output 0 with probability at least $1/4$.
- If no process i has $v_i = 0$ (and all correct process associate v_i with i), then all correct processes output 1. The probability of this event is at least $\left(1 - \frac{1}{u}\right)^n \geqslant e^{-1.15} \geqslant 0.25$.

□

Hence we have the following theorem.

Theorem 3. Protocol ICC[r] is a terminating, t-resilient inferable common coin protocol.

Proof. The termination properties follow from Lemma 10. The correctness properties follow from Lemma 13. □

Appendix III. From Common Coin to Byzantine Agreement

First we recall a voting protocol called Vote from [5] which is a primitive required for the construction of our ABA protocol. Protocol Vote computes whether a detectable majority for some value among the (binary) inputs of all processes. The output of protocol Vote is a tuple with the following meanings.
- For $\sigma \in \{1,2\}$, output $(\sigma, 2)$ means that there is an overwhelming majority for σ.
- For $\sigma \in \{1,2\}$, output $(\sigma, 1)$ means that there is a distinct majority for σ.
- $(\perp, 0)$ means that there is no distinct majority.

```
Vote protocol: code for process $i$ with binary input $x_i$
1. A-Cast "$\textsf{input}, j, x_j$".
2. Define a set $\mathcal{A}_i$. Add $(j, x_j)$ to $\mathcal{A}_i$ if "$\textsf{input}, j, x_j$" is received from the A-Cast of process $j$.
3. Wait until $|\mathcal{A}_i| = n - t$. Then assign $A_i = \mathcal{A}_i$. Set $a_i$ to the majority bit among $\{x_j : (j, x_j) \in A_i\}$
   and A-Cast "$\textsf{vote}, i, A_i, a_i$".
4. Define a set $\mathcal{B}_i$. Add $(j, A_j, a_j)$ to $\mathcal{B}_i$ if "$\textsf{vote}, j, A_j, a_j$" is received from the A-Cast of process $j$,
   $A_j \subset \mathcal{A}_i$, and $a_j$ is the majority bit of $A_j$.
5. Wait until $|\mathcal{B}_i| = n - t$. Then assign $B_i = \mathcal{B}_i$. Set $b_i$ to the majority bit among $\{a_j : (j, A_j, a_j) \in B_i\}$
   and A-cast "$\textsf{revote}, i, B_i, b_i$".
6. Define a set $C_i$. Add $(j, B_j, b_j)$ to $C_i$ if "$\textsf{revote}, j, B_j, b_j$" is received from the A-cast of process $j$,
   $B_j \subset \mathcal{B}_i$, and $b_j$ is the majority bit of $B_j$.
7. Wait until $|C_i| \geqslant n - t$. If all processes $j \in B_i$ had the same vote $a_j = \sigma$, then output $(\sigma, 2)$ and
   terminate. Otherwise, if all processes $j \in C_i$ have the same revote $b_j = \sigma$, then output $(\sigma, 1)$ and
   terminate. Otherwise, output $(\perp, 0)$ and complete the protocol.
```

This voting protocol is identical to that of [5]. The readers may refer to lemmas 5.32-5.35 [5] for complete proofs.

Lemma 14. All correct processes complete the voting protocol in constant time.

Lemma 15. If all correct processes have input σ, then all correct processes output $(\sigma, 2)$.

Lemma 16. If some correct process outputs $(\sigma, 2)$, then every correct process outputs either $(\sigma, 2)$ or $(\sigma, 1)$.

Lemma 17. If some correct process outputs $(\sigma, 1)$, and no correct process outputs $(\sigma, 2)$, then every correct process outputs either $(\sigma, 1)$ or $(\perp, 0)$.

Given the voting protocol and our ICC[r] protocol, we can design our ABA protocol following [6].

```
ABA protocol: code for process $i$ with binary input $x_i$
1. Set $r = 0$ and $v_1 = x_i$. Start the certification protocol.
2. Repeat until completing: (each iteration is consider as a round)
   a) Set $r = r + 1$. Set $(y_r, m_r) = \mathrm{Vote}(v_r)$.
   b) Invoke ICC[$r$] and wait until completion. Let $c_r$ be the output of ICC[$r$].
   c) Consider the following cases:
        I. If $m_r = 2$, set $v_{r+1} = y_r$ and A-Cast "$\textsf{complete with } v_r$". Participate in only one more
           instance of the voting protocol and only one more ICC[$r$] protocol.
       II. If $m_r = 1$, set $v_{r+1} = y_r$.
      III. Otherwise, set $v_{r+1} = c_r$.
   d) Upon receiving $t + 1$ "$\textsf{complete with } \sigma$" A-Casts for some value $\sigma$, output $\sigma$ and complete the
      protocol.
```

We now state and prove the following lemmas which are slight variants of lemmas 5.36-5.39 presented in [5].

Lemma 18. If all correct processes are in rounds greater than or equal to r, then every correct process eventually completes ICC[r].

Proof. If some correct process is in a round greater than r, then it must have completed ICC[r]. Then by termination property (2) of ICC[r], every correct process eventually completes ICC[r].

If all correct processes are in round r, Suppose that no correct process will complete ICC[r]. Since no correct process completes ICC[r], all correct processes keep participating. Then by termination property (1) of ICC[r], every correct process eventually completes. This is a contradiction.

Therefore, the lemma is proved. □

Lemma 19. In our ABA protocol, if all correct processes have the same input σ, then all correct processes complete and output σ.

Proof. If all correct processes have the same input σ, then by Lemma 15 every correct process will output $(y_1, m_1) = (\sigma, 2)$ by the end of Step a. Therefore, every correct process A-Casts "complete with σ" in the first iteration. Therefore, every correct process will receive at least $n-t$ "complete with σ" A-Casts, and at most t "complete with σ'" A-Casts. Consequently, every correct process will output σ. □

Lemma 20. In our ABA protocol, if a correct process completes with output σ, then all correct processes will complete with output σ.

Proof. Let us first show that if a correct process A-Casts "complete with σ" for some value σ, then all correct processes will A-Cast "complete with σ". Let k be the first round when a correct process i A-Casts "complete with σ". By Lemma 16, every correct process i has $y_k = \sigma$ and either $m_k = 2$ or $m_k = 1$. Therefore, no correct process A-Casts "complete with σ'" at iteration k. Furthermore, all correct processes invoke the voting protocol in round $k+1$ with input σ. Lemma 15 now implies that, by the end of Step a of round $k+1$, every correct process has $(y_{k+1}, m_{k+1}) = (\sigma, 2)$. Thus, all correct processes A-Cast "complete with σ", either at round k or at round $k+1$.

Now assume a correct process completes with output σ. Thus, at least one correct process A-casted "complete with σ". Consequently, all correct processes A-Cast "complete with σ". Hence, every correct process will receive at least $n-t$ "complete with σ" A-Casts and at most t "complete with σ'" A-Casts. Therefore, every correct process will output σ. □

Lemma 21. If all correct processes have initiated and completed some round k, then with probability at least $1/4$, all correct processes have the same value for v_{k+1} or a set of faulty pairs will eventually be inferred by correct processes.

Proof. We have two cases here. If all correct processes execute Step III in round k, then all correct processes set their v_{k+1} to the output of ICC[r]. According to the correctness property of ICC[r], the lemma is true.

Otherwise, some correct process has set $v_{k+1} = \sigma$ for some $\sigma \in \{0, 1\}$, either in Step I or Step II of round k. By Lemma 17, no correct process will set its v_{k+1} to σ'. According to the correctness property of ICC[r], with probability at least $1/4$, all correct processes have output σ or a set of faulty pairs will eventually be inferred by correct processes. □

Lemma 22. Let $n = 3t + \delta$, then all correct processes complete the ABA protocol in expected running time $O(\frac{t}{\delta})$.

Proof. We first show that all correct processes complete protocol ABA within constant time after the first correct process initiates a "complete with σ" A-Cast in Step III of the protocol. Assume the first correct process initiates a "complete with σ" A-Cast in round k. Then all correct processes participate in the voting and common coin protocols of all the rounds up to round $k+1$. We have seen in the proof of Lemma 20 that all correct processes will A-Cast "complete with σ" in round $k+1$. All these A-Casts complete in constant time. Then every correct process completes the ABA protocol after completing

$t+1$ of these A-Casts. Consequently, once the first correct process A-Casts "complete with σ", the ABA protocol completes in constant time.

Let the random variable τ count the number of rounds until the first correct process A-Casts "complete with σ". We have

$$\mathrm{Prob}(\tau > k) = \mathrm{Prob}(\tau \neq 1) \cdot \mathrm{Prob}(\tau \neq 2 | \tau \neq 1) \ldots \cdot \mathrm{Prob}(\tau \neq k | \tau \neq 1 \cap \ldots \cap \tau \neq k-1).$$

If event $\mathbb{E}$ does not occur in round k, we have $\mathrm{Prob}(\tau \neq k | \tau \neq 1 \cap \ldots \cap \tau \neq k-1) \leqslant \frac{3}{4}$. Hence, by Lemma 7, $\mathrm{Prob}(\tau > k) \leqslant \left(\frac{3}{4}\right)^{k-3t/\delta-1}$. By a simple calculation, we have $E(\tau) \leqslant \frac{3t}{\delta} + 17$. Therefore, the expected running time is $O(\frac{t}{\delta})$. □

We have thus shown the following:

Theorem 4. If $n = 3t + \delta$, then there is an almost-surely terminating ABA protocol with expected running time $O(\frac{t}{\delta})$.