A New Way to Achieve Round-Efficient Byzantine Agreement

Matthias Fitzi[1], Chen-Da Liu-Zhang[2], and Julian Loss[3]

[1] IOHK. matthias.fitzi@iohk.io.
[2] ETH Zürich. lichen@inf.ethz.ch.
[3] University of Maryland. lossjulian@gmail.com.

Abstract. Minimizing the round complexity of Byzantine Agreement (BA) protocols is a fundamental problem in distributed computing. The typical approach to achieve round efficient (randomized) BA is to have a weak form of BA, called graded consensus (GC), followed by a distributed coin, and to repeat this process until some termination condition is met—as introduced by Feldman and Micali (STOC '88).

In this work, we revisit the question of building BA from GC, or, more precisely, from generalizations of GC. Concretely, for 'Monte Carlo' style BA, where the protocol is run for a fixed number of rounds in function of the security parameter (in contrast to protocols with probabilistic termination), we demonstrate that this generalization helps to considerably reduce the round complexity of BA.

In particular, assuming a setup for threshold signatures among the parties and corruption threshold $t < n/3$, we improve over the round complexity of the best known protocol by a factor of $1/2$, asymptotically; this is achieved by applying *one single* Feldman-Micali iteration consisting of one (generalized) GC instance and one round of coin tossing.

Our technique also applies to the dishonest-minority case ($t < n/2$), yielding an improvement by a factor of $1/4$ (asymptotically) over the round complexity of the best known fixed-round protocol.

1 Introduction

In the problem of Byzantine Agreement (BA), a set of n parties want to agree on a common output y as a function of their inputs x_i by means of a distributed protocol. The protocol must remain secure even when up to some t out of the n parties are corrupted by arbitrarily deviating from the protocol. More concretely, a BA protocol must satisfy *consistency*—all honest parties output the same value y; and, *validity*—if all parties start with identical inputs x_i then $y = x_i$.

First formalized in the seminal work of Lamport et al. [15], BA is one of the most fundamental problems in cryptography and distributed computing.

A key efficiency metric for BA protocols is their *round efficiency*: *how many synchronous rounds of communication are needed to reach agreement?* As proven by Dolev and Strong [9], no *deterministic* BA protocol can run in less than $t + 1$ rounds. As first demonstrated by Ben-Or [2], and Rabin [20], the lower bound of Dolev and Strong does not apply to *randomized* protocols. Feldman and Micali [11] (FM) gave the first expected-constant-round protocol with the optimal resilience $t < n/3$ [15] and unconditional security.

Termination flavors. Constant-round BA can be achieved in 'Las Vegas' style—with probabilistic, constant-expected runtime and guaranteed correctness, or, 'Monte Carlo' style—with fixed runtime in function of the security parameter but allowing to fail with a probability negligible in the security parameter. In the sequel, we will refer to the former as protocols with *probabilistic termination* and the latter as *fixed-round* protocols.

BA protocols with probabilistic termination are more round efficient (in expectation) than their fixed-round counterparts, but they cannot achieve *simultaneous termination*, i.e., that all parties terminate the protocol during the same communication round—as proven by Dwork and Moses [10], and Moses and Tuttle [19]. This can make such protocols unwieldy when used as building blocks in larger protocol contexts—as was for instance exposed by Lindell et al. [17] or Cohen et al [6]. Fixed-round protocols are thus often preferable—a prominent example for this choice being the BA protocol of Algorand [5].

The FM paradigm. We give a short recapitulation of the FM protocol construction to help the exposition of our generalization. As we focus on fixed-round protocols in this paper, we directly describe their protocol as such, although their initial protocol achieves probabilistic termination. Furthermore, for simplicity, we restrict our view to a binary input domain.[4]

FM achieves BA from a weaker type of agreement called *graded consensus* (GC).[5] In GC, each party P_i holds an input x_i, and an outputs y_i together with a *grade* $g_i \in \{0, 1\}$, where $g_i = 1$ indicates that the parties' outputs y_j are (sufficiently) consistent.

The BA protocol consists of a fixed number of iterations (the number depending on the security parameter), an iteration consisting of one execution of GC followed by a distributed coin toss. After each iteration, if a party computes $g_i = 1$, it sticks to y_i, whereas, otherwise, it replaces y_i by the output of the coin toss. A party's output serves as its input for the next iteration.

The effect of one FM iteration is as follows. If the parties start an iteration in agreement then their values remains unchanged, $y_i = x_i$, and they compute grade $g_i = 1$, thus ignore the coin, and thus remain in agreement on y_i until the end of the protocol. On the other hand, if the parties end an iteration in disagreement then the coin reunites the parties' outputs with some constant probability—contributing to an exponential decrease in the protocol error in the number of iterations.

[4] Note that BA for binary inputs can be extended to any finite input domain at the expense of at most three additional communication rounds as demonstrated by Turpin and Coan [21].

[5] More precisely, the weak form of BA required by the fixed-round construction is Crusader Agreement [8], which is a special case of our GC generalization.

Generalizing the FM paradigm: Expand and extract. The expand-and-extract pattern is already present in the original: at the beginning of the FM iteration, the parties hold a binary value x_i, which is expanded, by GC, to a domain of cardinality $s = 3$, $(y_i, g_i) \in \{(0,1), (\cdot,0), (1,1)\}$ from which, by help of the coin, a binary value y_i is extracted to serve as the input of the next iteration, $x'_i = y_i$.

We generalize this pattern by replacing the expansion step by applying a generalization of GC that expands to an arbitrary finite domain size $s \geq 3$ (by means of a larger than binary grade range), which we call *s-slot Proxcensus*. The extraction step is then implemented by a different randomization method (still based on a distributed-coin protocol) that increases the per-iteration success probability of putting the honest parties into a agreement.

Concrete results. Our generalization helps to obtain improvements for probabilistic protocols in various models, but, to avoid too many case distinctions, we focus on the most interesting cases in this paper. These cases assume a setup for threshold signatures among the parties, and security is proven in the random-oracle model.

- For $t < n/3$, we demonstrate a (perfectly secure) protocol for s-slot Proxcensus requiring $O(\log s)$ communication rounds (for the 'expand' step). We then show how to achieve BA (with overwhelming probability) from a single instance of Proxcensus (with a grade range exponential in the security parameter) and a (single) multivalued coin toss (via the 'extract' step)—in contrast to the traditional FM approach in which *several instances* of GC (and coin tosses) are iterated. This yields a binary BA protocol involving $\kappa + 1$ rounds in order to achieve a target error probability of (at most) $2^{-\kappa}$. The best known fixed-round binary BA protocol [11,18] for $t < n/3$ requires 2κ rounds to achieve the same target error probability. Both protocols can be extended to any finite input domain at the expense of 2 additional communication rounds.
- For $t < n/2$, we demonstrate two (computationally secure) protocols for s-slot Proxcensus: a simple one requiring $\lceil (s+1)/2 \rceil$ rounds, and a more involved one requiring (roughly) $\sqrt{s}$ rounds. We then show how to achieve BA from 5-slot Proxcensus, applying the same 'extract' step as above. Note, however, that for this case, we have to reiterate this process as in the original FM protocol. This yields a fixed-round BA protocol requiring $3\kappa/2$ rounds to achieve a target error probability of $2^{-\kappa}$ while the best known previous protocol [18] for $t < n/2$ requires 2κ rounds. Both protocols can be extended to any finite input domain at the expense of 3 additional communication rounds.

For completeness, as this may be of independent interest, in the appendix, we also give protocols for s-slot proxcast (the respective single-sender version) computationally secure against $t < n$ requiring $s - 1$ communication rounds—improving over the M-gradecast protocol by Garay et al. [13]. Further, we demonstrate how to adapt this protocol to $t < n/2$ in the player-replaceable setting of [4], with the side effect of improving over the communication complexity of the player-replaceable BA protocol in [18].

More on previous work. The original FM protocol [11] achieves unconditional security against $t < n/3$. The complexity of their solution lies in the implementation of a distributed coin protocol with unconditional guarantees. Recently, Chen and Micali [4] demonstrated a simple and elegant way of implementing such a coin by means of verifiable random functions—at the price of downgrading to computational security against an adversary that is not strongly rushing (i.e., cannot drop already sent round messages in the very round it corrupts a given party). Interestingly, under idealized assumptions (in particular, assuming an ideal coin), the fixed-round variant of the original FM construction is still the most round-efficient solution, requiring 2κ rounds to achieve an error probability of (at most) $2^{-\kappa}$.

The first constant-round protocols with computational security for $t < n/2$ were given in [12,14]. Micali and Vaikuntanathan [18] simplified over those constructions using the FM paradigm, by applying a 2-round GC protocol for $t < n/2$, and running its second round in parallel to their coin. Assuming an ideal coin, this protocol thus matches the round complexity of the FM protocol for $t < n/3$, e.g., 2κ rounds for target error $2^{-\kappa}$.

In [1], Abraham et al. gave the first constant-round BA protocol with expected $O(n^2)$ communication complexity improving over the previous $O(n^3)$ bound from [14]. Their gain in communication complexity implies the use of threshold signatures (in contrast to the protocol in [14]) which requires special setup assumptions. The construction of [1] is based on PBFT [3] which is a deterministic iteration-based state-machine replication protocol where each iteration consists of 5 non-uniform rounds of communication. The solution of [1] is essentially achieved by randomizing PBFT by the random election of a leader during each iteration, and by applying threshold signatures.

The concept of graded broadcast was generalized to larger grade ranges in [22,7], called *proxcast* – achieving stronger security properties with growing grade ranges. Our BA protocols make use of this generalization. In [13], a solution for a subclass of proxcast (called gradecast with multiple grades therein) was given that is secure against $t < n$.

2 Model and Preliminaries

Notation. We denote $\mathcal{P} = \{P_1, \ldots, P_n\}$ the set of parties participating in the protocol. We denote the set of integers $\{1, ..., n\}$ as $[n]$. We write $x \leftarrow \mathcal{S}$ to denote that x is sampled uniformly from set $\mathcal{S}$. Throughout the paper, we denote the security parameter by κ.

2.1 Communication and adversary model

We consider a synchronous communication network with authenticated point-to-point channels. We describe the protocols as proceeding in a series of *rounds*. A message sent by an honest party P_i at the beginning of a round is guaranteed to be delivered by the end of that round. We consider an adversary who can corrupt up to t parties in a malicious (a.k.a. Byzantine) way. That is, a corrupted party may deviate from the protocol arbitrarily.

We consider a *strongly rushing, adaptive adversary* who can corrupt parties at any given point of the protocol execution. In every round of the protocol, it can observe the messages that the honest parties sent before choosing its own messages for that round. It has the following additional capability: when it observes that an honest party P sends a message m during some round i, it can immediately corrupt that party and replace m with a message m' of its choice (in particular, it can decide to drop m).

2.2 Cryptographic primitives

Threshold signatures and coin-flip. We assume a t-out-of-n threshold signature scheme consisting of a tuple $(\mathsf{SignShare}, \mathsf{VerShare}, \mathsf{Ver}, \mathsf{Combine})$ of four algorithms that behave as follows.

– In a setup phase, parties run a distributed protocol after which each party P_i holds a secret key $\mathtt{sk}_i$ and all parties agree on a public key $\mathtt{pk}$. We remark that during the setup phase, all existing protocols assume the existence of either a broadcast channel or a trusted dealer.
– Given a message m and a secret key $\mathtt{sk}_i$, $\mathsf{SignShare}_{\mathtt{sk}_i}(\cdot) := \mathsf{SignShare}(\mathtt{sk}_i, \cdot) : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ computes a signature share $\sigma_i = \mathsf{SignShare}_{\mathtt{sk}_i}(m)$.
– Given the public key $\mathtt{pk}$, $\mathsf{VerShare}_{\mathtt{pk}}(\cdot,\cdot) := \mathsf{VerShare}(\mathtt{pk},\cdot,\cdot) : \{0,1\}^* \times \{0,1\}^\kappa \rightarrow \{0,1\}$ verifies the validity of a signature share σ_i; we say that a signature share σ_i for a message m is valid if and only if $\mathsf{VerShare}(\mathtt{pk}, \sigma_i, m) = 1$.
– Given $t+1$ valid signature shares, it is possible to compute a signature $\Sigma = \mathsf{Combine}(\{\sigma_i\}_{1 \leq i \leq t+1})$.
– Given the public key $\mathtt{pk}$, $\mathsf{Ver}_{\mathtt{pk}}(\cdot,\cdot) := \mathsf{Ver}(\mathtt{pk},\cdot,\cdot) : \{0,1\}^* \times \{0,1\}^\kappa \rightarrow \{0,1\}$ verifies the validity of a signature Σ; we say that a signature Σ for a message m is valid if and only if $\mathsf{Ver}(\mathtt{pk}, \Sigma, m) = 1$.

As is common in the literature, we treat (threshold) signatures as idealized objects: we require that for any given threshold t, signatures remain perfectly unforgeable for a message m, given t signature shares on m. In reality, one can instantiate the scheme accordingly using any scheme which is unforgeable under chosen-message attack and use a standard hybrid argument to achieve security against a computationally-bounded adversary. Moreover, we assume that all parties start the protocol after the setup phase has been completed, i.e., they agree on a public key $\mathtt{pk}$ and all hold secret keys with the properties described above. In our protocols, we assume the availability of an atomic primitive $\mathsf{CoinFlip}$ which, on input k, returns a uniform value Coin_k (uniform in some range depending on the protocol of choice). Moreover, the value of Coin_k remains uniform from the view of the adversary until the first honest party has queried $\mathsf{CoinFlip}$ on input k. Such a primitive can easily be constructed from a threshold signature scheme with threshold $t+1$ and unique signatures per message and public key and assuming random oracle, such as the ones in [16]. To obtain a uniform value on input k, parties simply sign the value k and send their so obtained signature share to all parties. Parties can then hash the reconstructed signature on the value k into a suitable domain to obtain a random value. Unforgeability of the scheme ensures that until at least one honest party sends its share, the value of Coin_k remains uniform from the adversary's view. Uniqueness ensures that all honest parties obtain the same coin.

For convenience, we measure the communication complexity in the number of signatures exchanged between the parties. When each signature has λ bits, multiplying the communication complexity by λ leads to the communication complexity in bits.

2.3 Byzantine Agreement and Proxcensus

We first recall the definition of Byzantine agreement (BA).

Definition 1. *A protocol among parties $\mathcal{P}$ where every party P_i inputs a value $x_i \in \mathcal{D}$ from some finite domain $\mathcal{D}$, and, upon termination, every party $P_i \in \mathcal{P}$ outputs a value $y_i \in \mathcal{D}$, achieves* Byzantine agreement *iff the following conditions hold with overwhelming probability in κ:*

Validity. *If all honest parties P_i input the same value $x_i = x$ then they all output $y_i = x$.*
Consistency. *Any two honest parties P_i and P_j compute the same output, $y_i = y_j$.*
Termination. *All honest parties eventually terminate the protocol.*

Degraded versions of BA that require weaker conditions than Definition 1 were introduced in [8,11] as building-blocks for more powerful protocols. These versions can both be seen as accompanying the output value with an additional grade value from a small domain to express the degree of agreement achieved after the protocol execution. In [7], this approach was generalized to arbitrary finite domains along the following lines (yielding their definition of *Proxcast*, the single-sender version of the following definition):

Definition 2. *Let $s \in \mathbb{N}$ and $G \triangleq \lfloor \frac{s-1}{2} \rfloor$. A protocol among parties $\mathcal{P}$ where every party $P_i \in \mathcal{P}$ inputs a value $x_i \in \mathcal{D}$ from some finite domain $\mathcal{D}$, and, upon termination, every party $P_i \in \mathcal{P}$ outputs a value $y_i \in \mathcal{D}$ and a grade $g_i \in [0, G]$, achieves s-*slot Proxcensus*, or* Prox_s *iff the following conditions hold:*

Validity. *If all honest parties input the same value $x \in \mathcal{D}$ then every honest party P_i outputs $y_i = x$ and $g_i = G$.*
Consistency. *For any two honest parties P_i and P_j:*
- $|g_i - g_j| \leq 1$.
- $\min(g_i, g_j) \geq 1 \quad \Rightarrow \quad y_i = y_j$.
- *if $s = 2k$ ($k \in \mathbb{N}$) and $g_i > 0$ then $y_i = y_j$.*

Termination. *All honest parties eventually terminate the protocol.*

Note that, for even (odd) s, a grade above 0 (1, respectively) implies agreement detection with respect to the value y, i.e. all parties are guaranteed to obtain the same value y.

Prox_s (see Fig. 1) can be seen as a functionality wherein all parties output on one of the s slots such that all honest parties end up in two adjacent slots, and all honest parties decide on an extremal slot for their input value in case of pre-agreement in case of pre-agreement on z').

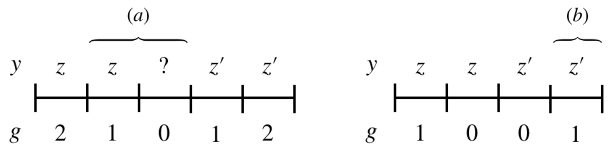

Fig. 1. Proxcensus for odd number of slots (left) and even (right). y indicates the output value and g the output grade. Brace (a) depicts the consistency requirement, which ensures that all honest parties end up in two adjacent slots. Brace (b) depicts the validity condition in case honest parties have pre-agreement on a value z'.

In particular, the well-known special cases mentioned above are Prox_3 known as *Crusader Agreement* [8] and Prox_5 known as *graded consensus* [11] (originally defined in its single-sender broadcast version).

3 A Generalized Iteration Paradigm for Byzantine Agreement

3.1 Revisiting the Feldman-Micali Construction

We give a high-level review of their fixed-round protocol variant with respect to a binary input domain; assuming GC and the common coin as black boxes. Note that, in contrast of GC (Prox_5) for the expected-round case, Prox_3 is sufficient as a substitute for GC in the fixed-round case.

Each party P_i starts every iteration with their initial input value $x_i \in \{0,1\}$, and at the end of the iteration, overwrites x_i for use during the next iteration. Each protocol iteration consists of one execution of Prox_3 on the inputs x_i, yielding outputs $y_i, g_i \in \{0,1\}$; followed by a common coin $c \in \{0,1\}$. At the end of the iteration, a party P_i overwrites their input $x_i := y_i$ if $g_i = 1$ (keep the output of Prox_3), and $x := c$ if $g_i = 0$ (adopt the value of the coin). At the end of the k protocol iterations, each party P_i outputs x_i.

Validity: If all honest parties start an iteration (the protocol) with the same bit $x_i \equiv b$ then, by the validity of Prox_3, they all hold $x_i = y_i \equiv b$ and $g_i \equiv 1$ at the end of the iteration (the protocol).

Consistency: An iteration where an honest party outputs $g_i = 1$ and the coin yields $c = y_i$ (or all honest parties output $g_i = 0$) puts the honest parties into agreement. This event happens with probability at least 1/2. Thus, by the validity argument above, the protocol fails to achieve BA with a probability of at most 2^{-k}.

3.2 Generalization

We propose a generalization of the Feldman-Micali iteration paradigm and show how it can be applied to achieve faster Byzantine agreement protocols. A generalized iteration with input x is composed of three components:

> **Expansion.** An invocation to an s-slot Proxcensus: $(z, g) \leftarrow \mathsf{Prox}_s(x)$
> **Coin-Flip.** An invocation to a multivalued coin-flip: $\mathsf{Coin} \leftarrow \mathsf{CoinFlip}$
> **Extraction.** A function f that takes as input (z, g, Coin) and outputs a value y, which is the output of the iteration.

Using this approach, we are able to substantially increase the probability of agreement per iteration round. For simplicity, we focus on protocols using ideal coins, meaning that the coin-flip component is an ideal 1-round multivalued coin-toss, which returns a uniform value with probability 1. Such a coin can be instantiated (tolerating a negligible failure probability) using unique threshold signatures in the random-oracle model [16]. However, our techniques can similarly be applied with other coins.

In the following, we show two binary fixed-round Byzantine agreement protocols achieving a target error of $2^{-\kappa}$. The first tolerates $t < n/3$ corruptions, runs in $\kappa + 1$ rounds and uses a *single* coin-flip. The second protocol tolerates $t < n/2$ and runs in $\frac{3}{2}\kappa$ rounds. The protocols can be made multivalued with an additional cost of 2 (resp. 3) rounds for the case where $t < n/3$ (resp. $t < n/2$) [21].

3.3 Expansion

We show the two Proxcensus protocols used in the BA protocols. First, we show a Proxcensus that tolerates up to $t < n/3$ corruptions and achieves $2^r + 1$ slots in r rounds. Second, we show a protocol for $t < n/2$ that achieves $2r - 1$ slots in r rounds. Further protocols for Proxcensus up to $t < n/2$ (with quadratic number of slots w.r.t the number of rounds) and for proxcast up to $t < n$ (with linear number of slots) are shown in Section A and B for completeness. The Proxcensus protocols for $t < n/2$ and proxcast for $t < n$ are described using threshold signatures. One can similarly describe the protocols using a PKI infrastructure (by appending plain signatures instead of combining signature shares) at the cost of a factor of n in the communication complexity.

Proxcensus for $t < n/3$ We show an expansion technique with unconditional security, which allows to expand a Proxcensus with s slots to a Proxcensus with $2s - 1$ slots in one additional round. Applying the expansion technique iteratively, we obtain a Proxcensus protocol with exponential slots in the number of rounds. The general idea is to run the Proxcensus protocol with s slots, $\mathsf{Prox_s}$, and echo the result. We know that all honest parties lie in two consecutive slots out of the s slots after $\mathsf{Prox_s}$. This implies that, after the echo, there will be $n-t$ values within two consecutive slots s_0 and s_1. Then, two consecutive slots accumulating $n-t$ votes constitute two consecutive slots in the new range, where the particular new slot is determined by which slot had $n-2t$ values (in case of a tie, the upper slot is chosen). The parties then output the highest possible slot. See Figure 2 for an illustrative example.

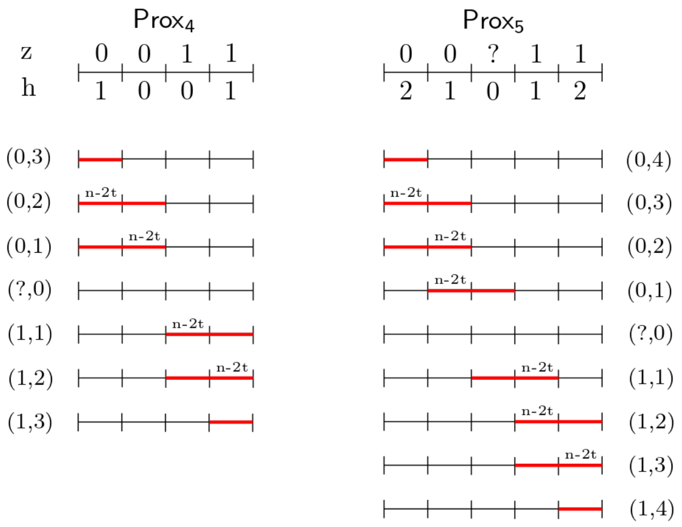

Fig. 2. Example of the Proxcensus expansion starting from $\mathsf{Prox_4}$ and $\mathsf{Prox_5}$, respectively, for binary values. The expanded Proxcensus have 7 and 9 slots respectively. Each row indicates the condition to achieve each output written on the side. The red line indicates the region where it is required that $n-t$ echoed pairs are received, and the upper text "$n-2t$" indicates that $n-2t$ are received on that specific slot.

Protocol $\mathsf{Prox}_{2s-1}(P_i)$

Let $G := \lfloor \frac{s-1}{2} \rfloor$, and $s := 2k+b$ for $k>0$ and $b \in \{0,1\}$. We describe the protocol from the point of view of party P_i with input x.

```
1: Run $\mathsf{Prox}_s(x)$. Let $(z,h)$ denote the output value.
2: Send $(z,h)$ to all parties. Denote as $(z_j,h_j)$ the message received from party
   $P_j$.
3: **Output Determination:**
4: $y_i := 0; g_i := 0$
5: $S_0 := \{j : h_j = 0\}$
6: $S_{z,g} := \{j : z_j = z \wedge h_j = g\}$
```

```
 7: if $b = 1 \land \exists z\colon |S_0 \cup S_{z,1}| \geq n - t \land |S_{z,1}| \geq n - 2t$ then
 8:     Set $y_i \coloneqq z, g_i \coloneqq 1$
 9: end if
10: for $g = b$ to $G - 1$ do
11:     if $\exists z\colon |S_{z,g} \cup S_{z,g+1}| \geq n - t \land |S_{z,g+1}| \geq n - 2t$ then
12:         Set $y_i \coloneqq z, g_i \coloneqq 2g + 2 - b$
13:     else if $\exists z\colon |S_{z,g} \cup S_{z,g+1}| \geq n - t \land |S_{z,g}| \geq n - 2t$ then
14:         Set $y_i \coloneqq z, g_i \coloneqq 2g + 1 - b$
15:     end if
16: end for
17: if $\exists z\colon |S_{z,G}| \geq n - t$ then
18:     Set $y_i \coloneqq z, g_i \coloneqq 2G + 1 - b = \lfloor \frac{2s-1}{2} \rfloor$
19: end if
20: Output $(y_i, g_i)$
```

Lemma 1. *Let $s \geq 2$. Protocol $\mathsf{Prox}_{\mathsf{2s-1}}$ satisfies validity.*

Proof. Suppose that all honest parties start with input v. Then, every honest party obtains (v, G) as output of $\mathsf{Prox}_{\mathsf{s}}$ and send it to every party. As a result, every honest party has $|S_{v,G}| \geq n - t$ and sets $y_i = v$ and the maximal grade $g_i = 2G + 1 - b = \lfloor \frac{2s-1}{2} \rfloor$.

Lemma 2. *Let $s \geq 2$. Protocol $\mathsf{Prox}_{\mathsf{2s-1}}$ satisfies consistency.*

Proof. Let P_i be an honest party that outputs y_i with the maximal grade g_i among all honest parties.

We prove that $|g_i - g_j| \leq 1$. Consider the case where $g_i > 1$, as otherwise it is trivial. We divide three cases for P_i:

- $\exists z\colon |S_{z,g} \cup S_{z,g+1}| \geq n - t \land |S_{z,g}| \geq n - 2t$, where $g_i = 2g + 1 - b$. If $S_{z,g+1}$ contains an honest party, since g_i is maximal, all honest parties sent (z, g) or $(z, g+1)$ after $\mathsf{Prox}_{\mathsf{s}}$. In this case, any honest P_j has $g_j = g_i$ because g_i is maximal. If all parties in $S_{z,g+1}$ are corrupted, this implies that there are at least $n - 2t$ honest parties in $S_{z,g}$ and all honest parties are in $S_{z,g-1} \cup S_{z,g}$. Hence, any honest P_j has $|S_{z,g-1} \cup S_{z,g}| \geq n - t \land |S_{z,g}| \geq n - 2t$, so $g_j \geq 2(g-1) + 2 - b = 2g - b \geq g_i - 1$.
- $\exists z\colon |S_{z,g} \cup S_{z,g+1}| \geq n - t \land |S_{z,g+1}| \geq n - 2t$, where $g_i = 2g + 2 - b$. Here, any honest P_j also received $|S_{z,g} \cup S_{z,g+1}| \geq n - t$ as g_i is maximal, and either $|S_{z,g}| \geq n - 2t$ or $|S_{z,g+1}| \geq n - 2t$. Hence, $g_j \geq 2g + 1 - b = g_i - 1$.
- $\exists z\colon |S_{z,G}| \geq n - t$. Here, any honest P_j received $|S_{z,G-1} \cup S_{z,G}| \geq n - t$, as honest parties obtain adjacent grades from $\mathsf{Prox}_{\mathsf{s}}$, and moreover $|S_{z,G}| \geq n - 2t$. Hence, $g_j \geq 2(G-1) + 2 - b = 2G - b \geq g_i - 1$.

We prove that $\min(g_i, g_j) \geq 1 \Rightarrow y_i = y_j$. Note that the above argument also covers this statement if $\max(g_i, g_j) > 1$. Hence, we consider the case where $g_i = g_j = 1$. Towards a contradiction, assume that $y_i \neq y_j$. We divide two cases, depending on b:

$b = 1$**:** In this case, from P_i we have that : $|S_0 \cup S_{y_i,1}| \geq n - t \wedge |S_{y_i,1}| \geq n - 2t$. This means that there is an honest party P that had $(y_i, 1)$ as output of $\mathsf{Prox_s}$. Symmetrically for P_j, there is also an honest party P' that had $(y_j, 1)$ as output of $\mathsf{Prox_s}$. This immediately contradicts the consistency of $\mathsf{Prox_s}$.

$b = 0$**:** In this case, from P_i we have $|S_{y_i,0} \cup S_{y_i,1}| \geq n - t \wedge |S_{y_i,0}| \geq n - 2t > t$. Symmetrically, from P_j we have $|S_{y_j,0} \cup S_{y_j,1}| \geq n - t \wedge |S_{y_j,0}| \geq n - 2t > t$. Since honest parties lie in adjacent slots, we know that all honest parties lie in $S_{y_i,0} \cup S_{y_j,0}$. Moreover, we know that there are $n - 2t$ honest parties in $S_{y_i,0}$ and another $n - 2t$ honest parties in $S_{y_j,0}$. This leads to a contradiction, since $S_{y_i,0} \cup S_{y_i,1}$ contains $n - t$ parties and in addition $S_{y_j,0}$ contains $n - 2t$ additional parties, which amount to a total of $n - t + n - 2t = 2n - 3t > n$ parties.

As a result, we obtain a Proxcensus protocol for the case of $t < n/3$ corruptions that, given $\mathsf{Prox_s}$, in r additional rounds it achieves $\mathsf{Prox_{2^r(s-1)+1}}$. Interpreting the parties' input configuration as the base case $\mathsf{Prox_2}$ (setting $g_i \equiv 0$), we obtain the following corollary. The communication complexity follows from inspection.

Corollary 1. *Let $t < n/3$. For any $r \geq 0$, $\mathsf{Prox_{2^r+1}}$ achieves Proxcensus with $2^r + 1$ slots and perfect security. The protocol runs in r rounds and has $O(rn^2)$ communication complexity.*

Proxcensus for $t < n/2$ We introduce a Proxcensus protocol that runs in r rounds and achieves $2r - 1$ slots. The protocol is similar to the proxcast in Section A (adapted to the agreement case): each party signs its input and sends it to all parties. Then, each party tries to collect a threshold signature on a value, and upon receiving such a threshold signature, it forwards it to all parties. However, parties now send in addition an extra message ω at the beginning of round two indicating whether a threshold signature Σ was reconstructed in round one. At the end of round two, if $n - t$ such ω are received, one computes a threshold signature Ω that proves that there was an honest party which reconstructed Σ. By propagating Ω, we are able to increase the number of slots to $2r - 1$. The way to determine the output and grade is different: a party P_i sets its output to (y, g) if it has a threshold signature Σ on y at round $r - g$, does not have any threshold signature on any $y' \neq y$ by round $g + 1$, and obtained the proof Ω at round $r - g + 1$. See Table 1 for an example.

	$1 \mid 0$	$? \mid 0$	$? \mid ?$	$0 \mid ?$	$0 \mid 1$
	Ω $1 \mid 0$	$1 \mid 0$	$? \mid ?$	$0 \mid 1$	$0 \mid 1$ Ω
	$1 \mid 0$	Ω $1 \mid ?$	$? \mid ?$	$? \mid 1$ Ω	$0 \mid 1$
(v,g):	(0,2)	(0,1)	($\perp$,0)	(1,1)	(1,2)

Table 1. Conditions for each slot in $\mathsf{Prox_5}$ for binary values. Row i indicates the condition to be satisfied at the end of round i. $b_0|b_1$, $b_0, b_1 \in \{0, 1\}$, indicates whether a threshold signature Σ on 0 or 1 was received, and ? indicates that anything could happen. Ω indicates that a proof Ω was received.

Protocol $\texttt{Prox}_{2r-1}(P)$

Setup: Parties make use of a unique $(n-t)$-out-n threshold signature scheme.

Party P starts with input v. Let $\mathcal{I}_k \subset [n]$ denote the parties that send correctly formed messages m in round k, i.e., where m is of the form $\{(x, \Sigma) | \texttt{Verify}(\texttt{pk}, \Sigma, x) = 1\}$. Initialize $\Omega^2 := \bot$.

```
1: **Round** 1:
2: $\sigma \leftarrow \mathsf{SignShare}(sk, v)$.
3: Send $(v, \sigma)$ to all parties. Denote as $(v_i^1, \sigma_i^1)$ the message received from party $P_i$.
4: Set $S^1 := \{(v, \Sigma) | \exists k_1, ..., k_{n-t}\colon \Sigma^1 \leftarrow \mathsf{Combine}(\sigma^1_{k_1}, \dots, \sigma^1_{k_{n-t}}) \wedge \mathsf{Ver}(pk, \Sigma^1, v) = 1\}$.
5: if $S_i^1 = \{(v, \Sigma)\}$ then
6:     $\omega \leftarrow \mathsf{SignShare}(sk, v)$
7: end if
8: Round 2:
9: Send $S^1$ and $\omega$ to all parties. Denote as $S_i^1, \omega_i$ the values received from party $P_i$
10: Set $S^2 := \bigcup_{i \in \mathcal{I}_2} S_i^1$
11: if $\exists k_1, \dots, k_{n-t}\colon \forall j\colon \mathsf{VerShare}(pk, \omega_{k_j}, v) = 1$ then
12:     $\Omega^2 \leftarrow \mathsf{Combine}(\omega_{k_1}, \dots, \omega_{k_{n-t}})$
13: end if
14: Rounds $j = 3$ to $s$:
15: Send $S^{j-1}, \Omega^{j-1}$ to all parties. Denote as $S_i^{j-1}, \Omega_i^{j-1}$ the values received from party $P_i$
16: Set $S^j := \bigcup_{i \in \mathcal{I}_j} S_i^{j-1}$; $\Omega^j := \bigcup_{i \in \mathcal{I}_j} \Omega_i^{j-1}$
17: Output Determination:
18: $y := 0; g := 0$
19: for $j = 1$ to $s - 1$ do
20:     if $\exists z, j\colon (z, \cdot) \in S^{s-j} \wedge \exists \Omega \in \Omega^{s-j+1}\colon \texttt{Verify}(pk, \Omega, z) = 1 \wedge \forall z' \neq z\colon (z', \cdot) \notin S^{j+1}$ then
21:         Set $y := z, g := j$
22:     end if
23: end for
24: Output $(y, g)$
```

Lemma 3. *Let $t < n/2$ and $r \geq 3$. Assuming unique threshold signatures,* $\texttt{Prox}_{\texttt{2r-1}}$ *achieves a $(2r-1)$-slot Proxcensus in r rounds and $O(rn^2)$ communication complexity.*

We prove validity and consistency in the following lemmas.

Lemma 4. *Let $r \geq 3$. Protocol* $\texttt{Prox}_{\texttt{2r-1}}$ *satisfies validity.*

Proof. Suppose that all honest parties start with input v. Observe that, there is no threshold signature computed on any value $v' \neq v$. In the first round, all honest parties send (v, σ^0) and so all honest parties hold $S^1 = \{(v, \Sigma)\}$ after the first round. Now, all honest parties compute a signature share ω on v and send it to all parties, together with S^1 in the second round. Therefore, honest parties will all hold $S^2 = \{(v, \Sigma)\}$ in round 2 and moreover are able to compute a threshold signature Ω^2 in that round. In each following round $j = 3$ to r, honest parties all send S^{j-1} and Ω, and so $S^j = \{(v, \Sigma)\}$. Therefore, all honest parties hold a threshold signature Ω^2 on v (that was computed in round two) and for all honest parties, $S^1 = S^2 = \cdots = S^r = \{(v, \cdot)\}$. Thus, all honest parties output v, as required.

Lemma 5. *Let $r \geq 3$. Protocol $\texttt{Prox}_{\texttt{2r-1}}$ satisfies consistency.*

Proof. Let P_i be the honest party that outputs y_i the maximal grade g_i among all honest parties.

We prove that $|g_i - g_j| \leq 1$. Consider the case where $g_i > 1$, as otherwise the statement is trivial. From P_i, we know:

- $(y_i, \cdot) \in S^{r-g_i}$. This implies that any honest P_j has $(y_i, \cdot) \in S^{r-g_i+1}$.
- There is $\Omega \in \Omega^{r-g_i+1}$ such that $\texttt{Verify}(pk, \Omega, y_i) = 1$. This implies that any honest P_j received Ω and hence has stored $\Omega \in \Omega^{r-g_i+2}$.
- $\forall z' \neq y_i \colon (z', \cdot) \notin S^{g_i+1}$. This implies that any honest P_j has $\forall z' \neq y_i \colon (z', \cdot) \notin S^{g_i}$.

With the above facts, we see that any P_j has grade $g_j \geq g_i - 1$.
Now we prove that $\min(g_i, g_j) \geq 1 \Rightarrow y_i = y_j$. Toward contradiction, assume that $y_i \neq y_j$. Since $g_i \geq 1$ (resp. $g_j \geq 1$), P_i (resp. P_j) obtained a threshold signature Ω on y_i (resp. y_j) at round $r - g_i + 1$ (resp. $r - g_j + 1$). This implies that there must be an honest party P that has sent a signature share ω in round 2. This implies, that for P, $y_i \in S^1$, which implies that also y_i in S^2 for P_j, which contradicts the requirement that $\forall z' \neq y_j \colon (z', \cdot) \notin S^2$.

3.4 Extraction

The extraction function can be interpreted pictorially as a *cut* that splits the slots in Proxcensus at the position indicated by the coin into two sides. If a party is placed at a position on the right (resp. left) side of the coin, it will decide on the output value 1 (resp. 0) (see Figure 3).

More formally, let s be the number of slots in Proxcensus, $G := \lfloor \frac{s-1}{2} \rfloor$ be the maximal grade, and $r := (s \mod 2)$ be the remainder modulo 2 of s. The extraction function takes as input a binary value $b \in \{0, 1\}$, a grade $g \in [0, G]$ and a coin value $c \in [1, s]$, and it outputs a binary value $f(b, g, c) \in \{0, 1\}$, defined as follows:

$$f(b, g, c) = \begin{cases} 1, & \text{if } (b = 1 \wedge c \leq g + G + 1 - r) \vee (b = 0 \wedge c \leq G - g) \\ 0, & \text{otherwise} \end{cases}$$

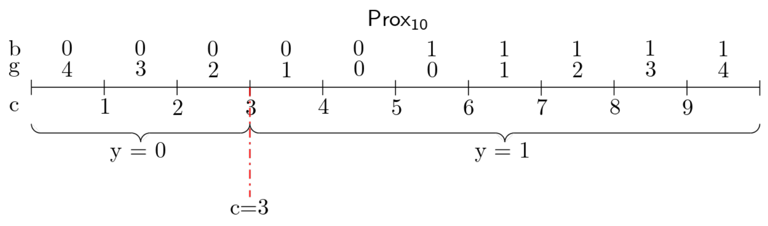

Fig. 3. Extraction function applied to Prox_{10}. The coin takes values from $[1, 9]$. If the slot (value $b \in \{0, 1\}$ and grade $g \in \{0, 4\}$) lies in the left (resp. right) side of the coin value c, the function $f(b, g, c)$ outputs the value $y = 0$ (resp. $y = 1$).

3.5 Efficient Fixed κ-Round Byzantine Agreement

We put the pieces together and show an efficient binary BA protocol. Using standard techniques [21], one can achieve a multivalued Byzantine agreement protocol with an additional cost of 2 (resp. 3) rounds when $t < n/3$ (resp. $t < n/2$).

The idea is to run the expansion component with one of the Proxcensus protocols Prox_s from Section 3.3, followed by a $(s-1)$-valued coin-flip, and the extraction function described in Section 3.4.

It is not hard to see that our approach allows to have an error per-iteration which is inversely proportional to the number of slots in Proxcensus. More precisely, since honest parties lie in two adjacent slots after the invocation of Prox_s, there is only one possible coin value (out of $s-1$) that lead to parties having different inputs.

The protocol is described from the point of view of party P_i and for a general Proxcensus protocol with s slots.

Protocol $\Pi^s_{\mathtt{iter}}(P_i)$

Let $G \coloneqq \lfloor \frac{s-1}{2} \rfloor$, and s be a positive number. Let f be the extraction function from Section 3.4. Let b denote the input bit.

```
1: $(b', g) \leftarrow \mathsf{Prox}_s(b)$
2: $c \leftarrow \mathsf{CoinFlip}$ // CoinFlip returns a uniform value in $[1, s-1]$
3: Output $f(b', g, c)$
```

Theorem 1. *Let $t < n$. $\Pi^s_{\mathtt{iter}}$ achieves binary Byzantine Agreement against an adaptive, strongly rushing adversary with probability $1 - \frac{1}{s-1}$. The protocol makes a single invocation to $\mathsf{Prox_s}$ and to a $(s-1)$-valued ideal Coin-Flip protocol.*

Proof. **Validity.** If all honest parties P_i input the same value b, then $b' = b$ and $g = G$ by validity of $\mathsf{Prox_s}$. Parties then output b because $c \in [1, s-1]$. More concretely, if $b' = 1$, then $c \leq 2G + 1 - j = s - 1$, so all parties output 1. And if $b' = 0$, then all parties output 0 because $c > 0$.

Consistency. Consistency of $\mathsf{Prox_s}$ guarantees that any two honest parties P_i and P_j lie on two consecutive slots. Parties only output different bits if the coin fails or the coin splits the two slots into different sides. Conversely, if the coin does not split the parties into different sides (which happens with probability $\frac{1}{s-1}$), then parties reach agreement. More concretely, we consider the following cases where honest parties lie on two consecutive slots (if all honest parties lie on the same slot, agreement is reached regardless of the coin value):

– s even: If there are honest parties that obtain $(0,0)$ and $(1,0)$, respectively, as output of $\mathsf{Prox_s}$, then parties output different bits only if $c = G+1$. Otherwise, assume that there are honest parties that output $(b', g-1)$ and (b', g), $1 \le g \le G$, respectively. If $b' = 1$ (resp. $b' = 0$), then parties output different bits only if $c = g + G + 1$ (resp. $c = G - g + 1$).
– s odd: We only need to consider the case where honest parties output $(b', g-1)$ and (b', g), $1 \le g \le G$, respectively, since we assume that not all honest parties lie on the same slot. If $b' = 1$ (resp. $b' = 0$), then parties output different bits only if $c = g + G$ (resp. $c = G - g + 1$).

Termination. Obvious.

We obtain the following corollary:

Corollary 2. *Assuming unique threshold signatures and a* 1*-round ideal Coin-Flip protocol, there are protocols that achieve binary Byzantine Agreement with probability* $1-2^{-\kappa}$ *secure against a strongly rushing adaptive adversary corrupting up to* t *parties, achieving the following:*

– *For* $t < n/3$*, it runs in* $\kappa+1$ *rounds and has* $O(\kappa n^2)$ *communication complexity. The protocol makes a single multivalued coin-flip invocation.*
– *For* $t < n/2$*, it runs in* $\frac{3}{2}\kappa$ *rounds, and has* $O(\kappa n^2)$ *communication complexity.*

Proof. **Case** $t < n/3$**:** The statement follows from Theorem 1 and the use of a 1-round 1-fair ideal Coin-Flip protocol, setting $s = 2^\kappa + 1$, and using the Proxcensus protocol that achieves s slots in κ rounds and $O(n^2)$ communication complexity from Corollary 1.

Case $t < n/2$**:** Security follows from Theorem 1 and the use of a 1-round 1-fair ideal Coin-Flip protocol, setting the number of slots to $s = 5$ and running the protocol $\Pi^s_{\mathtt{iter}}$ sequentially $\frac{\kappa}{2}$ times.[6] Each invocation to $\Pi^s_{\mathtt{iter}}$ takes 3 rounds, where we run the 3-round Prox_5 protocol from Section 3.3, and the coin-flip in parallel to the third round of Prox_5. Note that after round 2 of Prox_5, the slot-pair where the honest parties lie is already fixed. The probability of not reaching agreement in each invocation to $\Pi^s_{\mathtt{iter}}$ is $\frac{1}{4} = 2^{-2}$. Running the protocol sequentially $\frac{\kappa}{2}$ times therefore allows to achieve agreement except with probability

[6] One can check that other choices of number of slots will not lead to efficiency improvements.

$2^{-2\cdot\frac{\kappa}{2}} = 2^{-\kappa}$. The total number of rounds is $\frac{3}{2}\kappa$. The claim on communication complexity is inherited from the communication complexity of the Proxcensus sub-protocol from Section 3.3.

Efficiency comparison with previous protocols. We give a brief comparison with the most efficient, fixed-round protocol that we are aware of in the $n/3$ and $n/2$ regime, assuming an ideal 1-round coin flip. To the best of our knowledge, these are the fixed-round variant of the original Feldman-Micali (FM) [11] construction and the protocol by Micali and Vaikuntanathan (MV) [18]. The binary versions of these protocols both require 2κ rounds to achieve a termination error of $2^{-\kappa}$.

Our protocol for $t < n/3$ requires only $\kappa + 1$ rounds.This means that we achieve the same error probability within roughly half the number of rounds, with respect to the FM variant.

Our protocol for $t < n/2$ regime requires $\frac{3}{2}\kappa$ rounds, which gives an improvement of about 1/4 in the round complexity. Further note that we save a factor of n in the communication complexity with respect to MV, which incurs a communication complexity of $O(\kappa n^3)$, even assuming threshold signatures.

All protocols can be extended to arbitrary finite domains with an additional cost of 2 (resp. 3) rounds when $t < n/3$ (resp. $t < n/2$) by applying the construction of Turpin and Coan [21].

Finally, in context of MV and the Turpin-Coan construction, we observe an additional advantage of carefully adjusting the slot range of Proxcensus. In their original model (standard signatures, player replaceability), the communication complexity of the MV protocol (for $t < n/2$) can be reduced by a factor of n by substituting their 3-round $\{0, 1, 2\}$-gradecast protocol by 3-round Prox_s^4, the single-sender version of Prox_4—see Appendix A.

4 Acknowledgements

The authors would like to thank Ran Cohen for valuable comments on an early version of this paper.

References

1. Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous Byzantine agreement with expected $o(1)$ rounds, expected $o(n^2)$ communication, and optimal resilience. In *International Conference on Financial Cryptography and Data Security*, pages 320–334. Springer, 2019.
2. Michael Ben-Or. Another advantage of free choice (extended abstract) completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30, 1983.
3. Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.
4. Jing Chen and Silvio Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2016.

5. Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.
6. Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, Heidelberg, August 2016.
7. Jeffrey Considine, Matthias Fitzi, Matthew K. Franklin, Leonid A. Levin, Ueli M. Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, July 2005.
8. Danny Dolev. The Byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.
9. Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
10. Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.
11. Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.
12. Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 211–220. ACM, July 2003.
13. Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, October 2007.
14. Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, August 2006.
15. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
16. Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014.
17. Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. Sequential composition of protocols without simultaneous termination. In Aleta Ricciardi, editor, *21st ACM PODC*, pages 203–212. ACM, July 2002.
18. Silvio Micali and Vinod Vaikuntanathan. Optimal and player-replaceable consensus with an honest majority. 2017.
19. Yoram Moses and Mark R Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3(1):121–169, 1988.
20. Michael O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409. IEEE Computer Society Press, November 1983.
21. Russell Turpin and Brian A Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.
22. Oliver von Rotz. Reduktion von informationstheoretisch sicheren konsistenzprimitiven. Master's thesis, ETH Zürich, 2000.

A Efficient Generic Proxcast for $t < n$

In [13], under the notion *M-gradecast*, it was demonstrated how to achieve s-round s-slot Proxcast for *odd* s secure against $t < n$. We extend their result to achieving $(s-1)$-round s-slot Proxcast for general $s \geq 2$, secure against $t < n$, using essentially the same construction.

Definition 3. *Let $s \in \mathbb{N}$ and $G \triangleq \lfloor \frac{s-1}{2} \rfloor$. A protocol among parties $\mathcal{P}$ where a distinguished dealer (or sender) $P_d \in \mathcal{P}$ inputs a value $x_d \in \mathcal{D}$ from some finite domain $\mathcal{D}$, and, upon termination, every party $P_i \in \mathcal{P}$ outputs a value $y_i \in \mathcal{D}$ and a grade $g_i \in [0, G]$, achieves* s-slot proxcast, or Prox_s^d *(or* Prox_s*, in generic use) if and only if the following conditions hold:*

Validity. *If the dealer P_d is honest then every honest party P_i outputs $y_i = x_d$ and $g_j = G$.*
Consistency. *For any two honest parties P_i and P_j:*
- $|g_i - g_j| \leq 1$.
- $\min(g_i, g_j) \geq 1 \quad \Rightarrow \quad y_i = y_j$.
- *if $s = 2k$ ($k \in \mathbb{N}$) and $g_i > 0$ then $y_i = y_j$.*

Proxcensus

Let $s = 2k + b$, $b \in \{0, 1\}$. The protocol is similar to Dolev-Strong broadcast with the difference that parties do not add their signatures. In the first round, the dealer signs his input and sends the signed message to every other player. For the next $k - 1$ rounds, the parties collect all validating message/signature pairs originating from the dealer. If, during any one of these rounds, a "new" valid message/signature pair is received then this pair is sent to all parties (but only up to the second time as the existence of two contradicting signed messages by the dealer is sufficient to detect the dealer's misbehavior). At the end, a player accepts a message with grade $g \in [0, G]$ if, at the end of any $2g+1-b$ consecutive rounds, the same unique message/signature pair from the dealer was seen; and on grade $g = 0$, otherwise.

Protocol $\Pi_{\texttt{Proxcast}}(P_d, P_i)$

Setup: Parties know the dealer's public key pk, and the dealer has the secret key sk as well.
Let $G := \lfloor \frac{s-1}{2} \rfloor$, and $s := 2k + b$ for $b \in \{0, 1\}$.
The dealer P_d starts with input x.

```
1: Round 1: Dealer $P_d$ sends $(x, \sigma)$, $\sigma = \mathsf{Sign}_{\mathtt{sk}}(x)$, to all $P_j$. Each party $P_i$ sets
   $S_i^1 = \{(z, \sigma) \mid \mathsf{Ver}_{\mathtt{pk}}(z, \sigma) = 1\}$.
2: for $r = 2$ to $s - 1$ do
3:     Round $r$: Party $P_i$ sends $S_i^{r-1}$. Receive $S_j^{r-1}$ from each party $P_j$, and let
       $S_i^r = \bigcup_j S_j^{r-1}$.
4: end for
5: $y_i := 0$; $g_i := 0$;
6: for $g = 0$ to $G$ do
7:     if $\exists z, r : S_i^r = \cdots = S_i^{r+2g-b} = \{(z, \sigma)\}$ then
```

```
 8:         Set $y_i := z$, $g_i = g$.
 9:     **end if**
10: **end for**
11: $P_i$ outputs $(y_i, g_i)$.
```

Lemma 6. *Let $t < n$. Assuming that the dealer has a public-key setup, $\Pi_{\texttt{Proxcast}}$ achieves a s-slot Proxcensus in $s-1$ rounds and $O(sn^2)$ message complexity.*

Proof. **Validity.** If the dealer is honest, then each honest party P_i collects the same set $S_i^r = \{(x, \sigma)\}$ at the end of every round r.

Consistency. The cases $s \leq 3$ are trivial — thus consider $s > 3$.

- $|g_i - g_j| \leq 1$: Consider a party P_i with a maximal grade $g_i > 1$ among all honest parties (the case $g_i \leq 1$ is trivial). This means, there are $L = 2g + 1 - b > 2$ consecutive rounds such that $S_i^r = \cdots = S_i^{r+2g-b} = \{(z, \sigma)\}$. We claim that every honest party P_j sees at least $L - 2 > 0$ such rounds, namely rounds $r+1, \ldots, r+2g-b-1$:
  - As P_i sees a unique (z, σ)-pair at round $r + 2g - b$, P_j cannot have seen a conflicting pair in any round before as, otherwise, he would have sent it to P_i.
  - As P_i sees an (z, σ)-pair at round r, P_j sees it at round $r + 1$ as P_i sent it to P_j.
- $\min(g_i, g_j) \geq 1 \;\Rightarrow\; y_i = y_j$: Assume that $g_i > 0$ for an honest party P_i. This implies a sequence of at least two rounds such that $S_i^r = S_i^{r+1} = \{(z, \sigma)\}$. As $S_j^r \subseteq S_i^{r+1}$, and the sets grow monotonically, it follows that there is no round r' such that $S_j^{r'} = \{(z', \sigma')\}$ with $z' \neq z$. Hence, $y_j = y_i$ or $g_j = 0$.
- If $s = 2k$ ($k \in \mathbb{N}$) and $g_i > 0$ then $y_i = y_j$: Assume an honest party P_i with $g_i = 1$ implying that there are $L = 2g + 1 - b = 2g + 1 \geq 3$ consecutive rounds such that $S_i^r = \cdots = S_i^{r+2g-b} = \{(z, \sigma)\}$. Thus, an honest party P_j sees at least one such round, and $y_j = y_i$ due to the monotone growth of the sets.

A player-replaceable variant for $t < n/2$. The above proxcast protocol for $t < n$ relies on the fact that a player seeing a signature relays it during the next round in order to make it public. With player replacement, this is not guaranteed anymore since the participating player set is now different during every round. However, this can be compensated for by lowering the threshold to $t < n/2$, and strengthening the grade-determination condition

$$\exists z, r : S_i^r = \cdots = S_i^{r+2g-b} = \{(z, \sigma)\}$$

with the additional requirement that each such S_i^r ($r > 1$) must have been forwarded by at least $n - t$ parties during round r; implicitly guaranteeing the global forwarding of such a signature already during the same round as at least one of these $n - t$ forwarding parties must be honest.

B Quadratic Proxcensus for $t < n/2$

We introduce an improved version (for large r) of Proxcensus that runs in r rounds and achieves $3 + (r-3)(r-2)$ slots. The protocol develops on the ideas from the previous Proxcensus protocols in Section 3.3, with some changes: instead of forwarding a signature only after the first round and propagating it, parties repeatedly create and send an additional signature ω_j at each round $j > 1$ indicating whether a threshold signature was reconstructed in the previous round. More precisely, the protocol proceeds as follows. Each party P_i sends a signature share at round 1 on their input value. If P_i collects $n-t$ signature shares on the same value v, P_i forms a threshold signature Ω_1 for v at the end of round 1. At round 2, if Ω_1 was formed only for v, P_i echoes Ω_1 and also sends a signature share ω_2 indicating that Ω_1 was formed only for v. If $n-t$ such ω_2 are received at the end of round 2, P_i computes a threshold signature Ω_2. In general, P_i sends (resp. echoes) each formed (resp. received) threshold signature, and in addition sends a signature share ω_j for v at round j if P_i formed a threshold signature Ω_{j-1} for v at the end of round $j-1$, and was not able to form any threshold signature Ω_k, $k \in [1, j-1]$, for any value $v' \neq v$.

By propagating all these additional signatures ω_j, we are able to increase the number of slots to $3 + (r-3)(r-2)$, for $r \geq 3$. At the end of the protocol, P_i determines the output and grade checking a sequence of condition predicates. P_i evaluates a sequence of predicates, each indicating whether P_i received a certain threshold signature at a specific round. We denote $\mathsf{Condition}_{y,g,j}$ the predicate checking that a certain threshold signature needs to be formed or received at round j to output a value y with grade g. Moreover, we denote $\mathsf{Condition}_{y,g}$ the set of all conditions that need to be satisfied to output value y with grade g, over all rounds.

The condition predicates are defined inductively, starting from the highest grade (see Table 2 for a concrete example):

- Let $G = 1 + \frac{(j-3)(j-2)}{2}$. Then, $\mathsf{Condition}_{y,G,j}$ indicates whether P_i formed the threshold signature Ω_j for value y at round j.
- $\mathsf{Condition}_{y,g,j}$, $0 < g < G$, is inductively derived as follows: P_i formed or received a threshold signature Σ for value y by the end of round j, where $\Sigma = \Omega_{j-1}$ if there is a predicate $\mathsf{Condition}_{y,g+1,j'}$, $j' > r$, indicating that Ω_j is obtained for value y by round j'. Otherwise, Σ is the threshold signature that is obtained according to $\mathsf{Condition}_{y,g+1,j-1}$.
- $\mathsf{Condition}_{y,0,j}$ is always true.

Intuitively, if an honest party satisfies $\mathsf{Condition}_{y,g}$, $g \geq 1$, then every honest party satisfies $\mathsf{Condition}_{y,g-1}$ for two reasons: 1) honest parties forward each threshold signature that they receive or form, and 2) the existence of a threshold signature Ω_j, $j > 1$, implies that an honest party P_k sent ω_j at the beginning of round j, meaning that P_k obtained Ω_{j-1} at the end of round $j-1$. This P_k therefore sent Ω_{j-1} at the beginning of round j, and every honest party received Ω_{j-1} by the end of round j.

Moreover, the conditions are designed such that any $\mathsf{Condition}_{y,g}$, $g \geq 1$, requires that the threshold signature Ω_3 is obtained in some round. This guarantees that the conditions $\mathsf{Condition}_{y,1}$ and $\mathsf{Condition}_{y',1}$ are mutually disjoint, for $y \neq y'$. To see this, suppose P_i outputs $(y, 1)$, and thereby received Ω_3 at the last round. Note that this condition implies that there is an honest P_k that obtained Ω_2 for y and did not receive Ω_1 for any other value y' by round 2. This implies that no honest party received Ω_1 for y' by round 1, and therefore no honest party can output $(y', 1)$.

	Ω_1	?	?	?	?	?	?	?	?	?	?	?	?	?	Ω_1
	Ω_2	Ω_1	Ω_1	Ω_1	Ω_1	Ω_1	Ω_1	?	Ω_1	Ω_1	Ω_1	Ω_1	Ω_1	Ω_1	Ω_2
	Ω_3	Ω_2	Ω_2	Ω_2	Ω_2	Ω_2	Ω_2	?	Ω_2	Ω_2	Ω_2	Ω_2	Ω_2	Ω_2	Ω_3
	Ω_4	Ω_3	Ω_3	Ω_3	Ω_3	Ω_2	Ω_2	?	Ω_2	Ω_2	Ω_3	Ω_3	Ω_3	Ω_3	Ω_4
	Ω_5	Ω_4	Ω_4	Ω_3	Ω_3	Ω_3	Ω_2	?	Ω_2	Ω_3	Ω_3	Ω_3	Ω_4	Ω_4	Ω_5
	Ω_6	Ω_5	Ω_4	Ω_4	Ω_3	Ω_3	Ω_3	?	Ω_3	Ω_3	Ω_3	Ω_4	Ω_4	Ω_5	Ω_6
(v,g):	(0,7)	(0,6)	(0,5)	(0,4)	(0,3)	(0,2)	(0,1)	($\perp$,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)

Table 2. Conditions for each slot in Prox_{15} for binary values. Row i indicates the condition to be satisfied at the end of round i. To output the pair (v, g), all conditions in the column for that pair need to be satisfied, where Ω_r at row i indicates that a threshold signature Ω_r was received at round i, and ? indicates that there is no requirement. Note that Ω_3 is required at some position for every grade $g > 0$.

Protocol $\texttt{Prox}_{3+(r-3)(r-2)}(P_i)$

Setup: Parties make use of a unique $(n-t)$-out-n threshold signature scheme.

Party P_i starts with input v.

```
1: **Round** 1:
2: $\sigma \leftarrow \mathsf{SignShare}(sk, v)$.
3: Send $(v, \sigma)$ to all parties. Denote as $(v^i, \sigma^i)$ the message received from party $P_i$.
4: Set $S^1 := \{(v, \Sigma) | \exists k_1, ..., k_{n-t} \colon \Sigma \leftarrow \mathsf{Combine}(\sigma^{k_1}, \ldots, \sigma^{k_{n-t}}) \wedge \mathsf{Ver}(pk, \Sigma, v) = 1\}$.
5: Set $T := \varnothing$;
6: **Rounds** $j = 2$ **to** $r$:
7: **if** $S^{j-1} = \{(v, \Omega_{j-1})\} \wedge \forall v' \neq v \; \forall \ell < j \; (v', \Omega_\ell) \notin T$ **then**
8:     $\omega_j \leftarrow \mathsf{SignShare}(sk, (v, j-1))$
9: **end if**
10: Forward all new pairs of value and threshold signature received or formed in the previous round. Moreover, if a signature share $\omega_j$ was computed on a value $v$, also send $(v, \omega_j)$ to all parties. Denote as $(v^i, \omega_j^i)$ the message received from party $P_i$.
11: Set $S^j := \{(v, \Sigma) | \exists k_1, ..., k_{n-t} \colon \Sigma \leftarrow \mathsf{Combine}(\omega_j^{k_1}, \ldots, \omega_j^{k_{n-t}}) \wedge \mathsf{Ver}(pk, \Sigma, v) = 1\}$.
12: Add to $T$ all newly formed or received threshold signature schemes (with the corresponding value).
13: **Output Determination:**
```

```
14: Output $(y, g)$ with the highest grade such that $\mathsf{Condition}_{y,g}$ is satisfied.
```

Lemma 7. *Let $t < n/2$ and $r \geq 3$. Assuming unique threshold signatures, $\mathtt{Prox}_{\mathtt{3+(r-3)(r-2)}}$ achieves a $(3 + (r-3)(r-2))$-slot Proxcensus in r rounds and $O(rn^2)$ message complexity.*

We prove validity and consistency in the following lemmas.

Lemma 8. *Let $r \geq 3$. Protocol $\mathtt{Prox}_{\mathtt{3+(r-3)(r-2)}}$ satisfies validity.*

Proof. Suppose that all honest parties start with input v. Thus, all honest parties send a signature share on v in the first round, and so all honest parties hold a $S^1 = \{v, \Omega_1\}$ after the first round. Note that since no honest party ever signs a signature share on any other value $v' \neq v$, at each round $j \in [2, r]$, all honest parties compute a signature share ω_j on v and send it to all parties, and all honest parties compute a threshold signature Ω_j by the end of round j. Thus, $\mathsf{Condition}_{y,G}$ is satisfied and all honest parties output v.

Lemma 9. *Let $r \geq 3$. Protocol $\mathtt{Prox}_{\mathtt{3+(r-3)(r-2)}}$ satisfies consistency.*

Proof. We first prove that any two honest parties P_i and P_ℓ output grades g_i and g_ℓ with $|g_i - g_\ell| \leq 1$.

Let P_i be the honest party that outputs the maximal grade g_i among all honest parties. If $g_i \geq 1$, then trivially $|g_i - g_\ell| \leq 1$. Hence, suppose that P_i outputs (v, g_i), $g_i > 1$. This implies that P_i satisfies $\mathsf{Condition}_{y,g_i}$. We show that any honest party P_j satisfies $\mathsf{Condition}_{y,g_i-1}$. Let $j \in [2, r]$. We show that P_ℓ satisfies $\mathsf{Condition}_{y,g_i-1,j}$. There are two cases: 1) P_i obtained Ω_j at some round $j' > j$, then there is an honest party P_k that sent ω_j at the beginning of round j. This means that P_k obtained Ω_{j-1} at the end of round $j-1$, and therefore sent Ω_{j-1} at the beginning of round j, and every honest party received Ω_{j-1} by the end of round j; 2) P_i did not obtain such Ω_j, in which case every honest party satisfies $\mathsf{Condition}_{y,g_i-1,j}$ by the fact that P_i echoes all formed threshold signatures.

Now we prove that if $g_i \geq 1$ and $g_\ell \geq 1$, then the parties output the same value, i.e. $y_i = y_\ell$.

This follows from the fact that the conditions are designed in such a way that any condition $\mathsf{Condition}_{y,g}$, $g \geq 1$, requires that the threshold signature Ω_3 is obtained in some round for the corresponding value. That is, $\mathsf{Condition}_{y_i,g_i}$ (resp. $\mathsf{Condition}_{y_\ell,g_\ell}$) requires obtaining Ω_3 for value y_i (resp. y_ℓ). We show that both conditions cannot be simultaneously satisfied. From $\mathsf{Condition}_{y_i,g_i}$, we know that there is an honest P_k that obtained Ω_2 for y_i and did not receive Ω_1 for y_ℓ by round 2. This implies that no honest party received Ω_1 for y_ℓ by round 1, and therefore no honest party created a signature share ω_2 for $y_j\ell$. As a result, Ω_2 (and hence also Ω_3) cannot be computed for y_ℓ, and no honest party can satisfy $\mathsf{Condition}_{y_\ell,g_\ell}$.