Gradecast in Synchrony and Reliable Broadcast in Asynchrony with Optimal Resilience, Efficiency, and Unconditional Security

Ittai Abraham* Gilad Asharov†

May 30, 2022

Abstract

We revisit Gradecast (Feldman and Micali, STOC'88) in Synchrony and Reliable Broadcast (Bracha, Information and Computation'87) in Asynchrony. For both tasks ,we provide new protocols that have three desirable properties: (1) *optimal resilience*, tolerating $t < n/3$ malicious parties; (2) are *communication-efficient*, where honest parties send just $O(nL)$ bits for a sender with a message of $L = \Omega(n \log n)$ bits; (3) and are *unconditionally secure*, without needing to rely on any computational or setup assumptions (while having a statistical error probability). To the best of our knowledge, no previous work obtains all three properties simultaneously.

1 Introduction

Feldman and Micali's Gradecast [FM88] (in Synchrony), and Bracha's Reliable Broadcast [Bra87] (in Asynchrony) are fundamental protocols from the 1980s. In this paper we provide new protocols for both Gradecast and Reliable Broadcast that obtain three desirable properties:

1. **Communication Efficient**: The total number of bits that honest parties send and receive is just $O(nL)$ bits for a sender that has a message of size $L = \Omega(n \log n)$ bits. We call this *communication efficient* because this is asymptotically optimal: just sending L bits requires each party to receive L bits for a total of $O(nL)$.

2. **Unconditionally secure**: There are no restrictions on the computational power of the adversary and there are no setup assumptions. The protocol achieves statistical security.

3. **Optimal Resilience**: Tolerating a malicious adversary controlling $t < n/3$ parties. This is the *optimal resilience* for statistical security.

The classic protocol of Bracha [Bra87] is optimally resilient but not communication efficient, broadcasting L bits requires $O(n^2 L)$ bits. The protocol of Cachin and Tessaro [CT05] is optimally resilient and nearly (up to polylog factors) communication efficient, but relies on a collision resistant

*VMWare Research. iabraham@vmware.com

†Department of Computer Science, Bar-Ilan University. Gilad.Asharov@biu.ac.il. Sponsored by the Israel Science Foundation (grant No. 2439/20), by JPM Faculty Research Award, by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234

hash function and hence is not unconditionally secure. The recent protocol of Das, Xiang, and Ren [DXR21] is optimally resilient and communication efficient, but still relies on a collision resistant hash function and hence is not unconditionally secure.

Concurrent and independent work of Alhaddad, Das, Duan, Ren, Varia, Xiang, and Zhang [ADD+22a, DXR22, DZ22, ADVZ21] obtains *balanced* Byzantine Reliable Broadcast with near-optimal communication and improved computation. Additional concurrent and independent work of Alhaddad, Das, Duan, Ren, Varia, Xiang, and Zhang [ADD+22b] use this to obtain Asynchronous Verifiable Information Dispersal with near-optimal communication.

Finally, concurrent and independent work of Das, Ren, Xiang [DXR22, ADD+22a] show how to use the technique of Chen [Che21] to obtain optimally resilient and communication efficient reliable broadcast with perfect security.

Our work shows that optimally resilient and communication efficient solutions can be unconditionally secure. A central contribution of this work is the observation that there is no need to make any *computational assumptions* or setup assumptions. Compared to the concurrent and independent work of Das, Ren, Xiang [DXR22], while both protocols are of constant number of rounds, our protocol even in its balanced version requires less rounds (6 vs 9 rounds). This poses an intriguing trade-off between the exact round complexity and the level of security (perfect vs. statistical security).

1.1 Gradecast in Synchrony

Feldman and Micali's Gradecast protocol [FM88], is a useful building block in synchronous protocols [BDH10b, BDH10a, FGH+02, KK09]. Its formal definition:

Definition 1.1. *In Gradecast a sender has an input and each party outputs a value and a grade* $\{0, 1, 2\}$ *such that:*

1. *(Validity) If the sender is honest then all honest parties output the sender's input and grade 2.*
2. *(Non-equivocation) If two honest parties output a grade* ≥ 1 *then they output the same value.*
3. *(Agreement) If an honest party outputs grade 2 then all honest parties output the same output and with grade* ≥ 1*.*

In is known that for unconditional security tolerating $t < n/3$ malicious parties is the best one can hope for [FLM85]. Hence such a protocol is said to have optimal resilience. If the number of bits of the sender's input is L, then the message complexity of the Feldman Micali's Gradecast protocol is $O(n^2L)$ bits. In this paper we ask the question: Can we reduce the message complexity at the cost of moving from perfect security to unconditional security[1]? We show:

Theorem 1.2. *Let* $\epsilon > 0$ *and* $t \in \mathbb{N}$ *and* $n \geq 3t + 1$*. There exists an n-party gradecast protocol that for input of length* $L > n/3 \cdot (\log(n^3/\epsilon))$ *bits achieves validity (with no probability of error), and achieves non-equivocation and agreement with probability* $1 - \epsilon$*. The protocol tolerates up to t malicious parties and requires the transmission of* $O(nL)$ *bits.*

In particular, if one is interested in $2^{-\lambda}$ probability of error, then our protocol achieves the optimal $O(nL)$ bits transmitted for $L > n/3 \cdot (3 \log n + \lambda)$.

[1] Perfect security requires 0 probability of error, whereas we relax the requirement and allow ϵ probability of error. In both cases, the security is unconditional and does not rely on any intractable assumption.

1.2 Reliable Broadcast in Asynchrony

Bracha's Reliable Broadcast [Bra87], is a useful building block in asynchronous protocols. The recent paper of Das, Xiang, and Ren [DXR21] has a detailed survey of the usefulness of communication efficient Reliable Broadcast. In particular, it implies Asynchronous Verifiable Secret Sharing and Asynchronous Distributed Key Generation [AJM+21].

Definition 1.3. *In Reliable Broadcast a sender has an input and each party that terminates outputs a value such that:*

1. *(Validity) If the sender is honest then all honest parties terminate and output the sender's input.*
2. *(Non-equivocation) If two honest parties terminate then their output is the same value.*
3. *(Termination) If an honest party terminates then all honest parties will eventually terminate.*

Here too, it is known that $t < n/3$ is the optimal resilience. If the number of bits of the sender's input is L, then the message complexity of Bracha's Reliable Broadcast protocol is $O(n^2L)$ bits. In this paper we ask the question: Can we reduce the message complexity at the cost of moving from perfect security to unconditional security?

Theorem 1.4. *Let $\epsilon > 0$ and $t \in \mathbb{N}$ and $n \geq 3t + 1$. There exists an n-party reliable broadcast protocol that for input of length $L > n/3 \cdot (\log(n^3/\epsilon))$ achieves validity (with no probability of error), and achieves non-equivocation and termination with probability $1-\epsilon$. The protocol tolerates t malicious parties and requires the transmission of $O(nL)$ bits.*

Just like our gradecast, if one is interested in $2^{-\lambda}$ probability of error, then our protocol achieves the optimal $O(nL)$ bits transmitted for $L > n/3 \cdot (3 \log n + \lambda)$.

Our result is an improvement of a recent result of Das, Xiang, and Ren [DXR21] in both security and asymptotic efficiently. In that paper a similar result was obtained in the Random Oracle model, under the assumption that the adversary is computationally bounded. Our result is unconditionally secure. Moreover, asymptotically, the use of a hash function requires a domain of size $\log(n/\epsilon^2)$ and hence at least $O(nL \log(n/\epsilon^2) + n^2)$ bits. Our result provides a $\log(1/\epsilon)$ better asymptotic communication complexity.

Prior to the recent result of Das, Xiang, and Ren, the best result was by Cachin and Tassero [CT05], which for optimal resilience, also required the Random Oracle model, computationally bounded adversaries, and $O(nL \log(n/\epsilon^2) \log n + n^2)$ bits.

2 Techniques

At a high level our work uses the uni-variant (trivial) Schwartz–Zippel lemma [Sch80, Zip79]. The Schwartz–Zippel lemma is a fundamental tool of unconditional randomized algorithm design and has numerous applications in many domains. In this work we show its applicability to fault tolerant distributed computing by showing how it can provide new trade-offs (better communication efficiency, with less assumptions) for core protocols that have been studied for over 30 years. For completeness we state the lemma:

Lemma 2.1 ([Sch80, Zip79]). *Let $P(x)$ be a non-zero univariate polynomial of degree t over a field $\mathbb{F}$. Let $\beta \in \mathbb{F}$ chosen uniformly at random. Then $\Pr[P(\beta) = 0] \leq t/|\mathbb{F}|$.*

Reed solomon codes [Ree60]. Another primitive that we rely on is Reed-Solomon code. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > n$ where n is the number of parties, and let $\alpha_1, \ldots, \alpha_n$ be distinct field elements. To encode a message $m = (m_0, \ldots, m_t)$ where each $m_i \in \mathbb{F}$, the encoding algorithm defines a polynomial $P(x) = m_0 + m_1 x + \ldots + m_t x^t$ and outputs $(P(\alpha_1), \ldots, P(\alpha_n))$. Since two polynomials of degree-t can agree on at most t points, the distance of two messages is at least $n - t$. Thus, we can correct up to $(n - t - 1)/2$ errors. When $t < n/3$, this means that we can correct up to t errors. Moreover, the decoding procedure is efficient. That is, given a possibly corrupted codeword $(\delta_1, \ldots, \delta_n) \in \mathbb{F}^n$ that is of distance at most t from a codeword $(\delta'_1, \ldots, \delta'_n)$, the output of the algorithm is a polynomial $P(x)$ such that for every α_i it holds that $\delta'_i = P(\alpha_i)$.

3 Gradecast Protocol (Synchrony)

Model. We assume pairwise channels, and a computationally-unbounded adversary. The channels are not necessarily private – the adversary can see messages transmitted between two honest parties, but it cannot delay or alter them. Moreover, it cannot also inject messages on behalf of honest parties.

We prove the following theorem:

Theorem 3.1. *Let $\epsilon > 0$ and $t \in \mathbb{N}$ and $n \geq 3t + 1$. Protocol 3.2 is an n-party gradecast protocol such that for input of length $L > n/3 \cdot (\log(n^3/\epsilon))$ bits tolerates up to t malicious parties and requires the transmission of $O(nL)$ bits. Security is unconditional, and the probability of error of non-equivocation and agreement is ϵ.*

Concrete instantiation. For statistical error probability of $\epsilon = 2^{-40}$, the table shows the minimum input length $L > n/3 \cdot (\log(n^3/\epsilon))$ for which Theorem 3.1 holds.

n	10^2	10^3	10^4	10^5	10^6
$L >$	0.253 KB	3.534 KB	45.34 KB	0.55 MB	6.53 MB

The protocol. The sender encodes its input using a finite field $\mathbb{F}$. We let $||\mathbb{F}|| = L/t$, i.e., $|\mathbb{F}| = 2^{L/t}$. Observe that $L > n/3 \cdot (\log(n^3/\epsilon))$ implies that $\frac{n^3}{|\mathbb{F}|} < \epsilon$ and that $|\mathbb{F}| > n$. The sender first encodes its message using a degree-t polynomial and sends that polynomial to all parties. The parties then exchange random points on the polynomial to check its validity. Once $t + 1$ honest parties receive $2t + 1$ "good points", the parties can reconstruct the polynomial from the points themselves, and essentially correct the polynomials that other honest parties hold. The protocol is as follows:

Protocol 3.2: Gradecast for $t < n/3$ (Synchrony)

- **Input:** The sender holds $t + 1$ field elements, $a_0, \ldots, a_t \in \mathbb{F}$.
- **Public parameters:** n distinct non-zero field elements $\alpha_1, \ldots, \alpha_n$.
- **The protocol:**

  Round 1 – the sender:

  1. Encode $F(x) = \sum_{i=0}^{t} a_i x^i$ and send $F(x)$ to each party P_i (i.e., send the $t+1$ coefficients to each party).

  Round 2 – each party P_i (each party sends a random share):

1. Denote by $F_i(x)$ the polynomial received by party P_i.
2. Choose $\beta_i \in \mathbb{F}$ uniformly at random.
3. Send $(\beta_i, F_i(\beta_i))$ to all parties.

Round 3 – each party P_i (send share if happy):

1. Let $\{(\beta_j, \gamma_j)\}_{j\in[n]}$ be the message received, where (β_j, γ_j) is the message received from party P_j. If $F_i(\beta_j) = \gamma_j$ for at least $2t+1$ indices $j \in [n]$, then P_i is happy.
2. If P_i is happy, then it sends to each other party P_j the message $(\texttt{YourPoint}, F_i(\alpha_j))$.

Round 4 – each party P_i (reconstruct the polynomial):

1. If $t+1$ messages $(\texttt{YourPoint}, \gamma_i)$ with the same value γ_i, then send to all parties the message $(\texttt{Reconstruct}, \gamma_i)$ to all parties.
2. Otherwise, forward $(\texttt{Reconstruct}, \perp)$ to everyone.

- **Output – each party P_i:** Let $(\texttt{Reconstruct}, \delta_k^i)$ be the message P_i receives from party P_k in the previous round (if a value was not received from some party P_k then let $\delta_k^i = \perp$). If received at least $2t+1$ values that are non-$\perp$, use Reed-Solomon decoding procedure to decode $(\delta_1^i, \ldots, \delta_n^i)$ and let $F_i'(x)$ be the reconstructed polynomial obtained from the robust interpolation.

  1. If there is no unique decoding (or did not receive at least $2t+1$ messages), output $\perp$ with grade 0.
  2. Otherwise, if (1) P_i is happy in round 3; and (2) $F_i(x) = F_i'(x)$; and (3) in round 4 its value has received $2t+1$ messages $(\texttt{YourPoint}, \gamma_i')$ with $\gamma_i' = F_i'(\alpha_i)$; then output $F_i'(x)$ with grade 2.
  3. Otherwise, output $F_i'(x)$ with grade 1.

Proof of Theorem 3.1: We start with analyzing the copmlexity of the protocol: in round 1, the sender sends $n \cdot (t+1)||\mathbb{F}|| = O(nL)$ bits where recall that $||\mathbb{F}|| = L/t$. In rounds 2, 3 and 4, each party sends at most constant number of points to each other party, i.e., a total of $n^2||\mathbb{F}||$ bits. Thus, we get that the total communication is $O(n^2)$ words, or $O(n^2||\mathbb{F}||) = O(nL)$ bits. We now show that validity, non-equivocation and agreement hold with all but an ϵ error probability. We will show agreement before non-equivocation.

Validity. We first show that if the sender is honest and holds the message $(a_0, \ldots, a_t)$, then all honest parties output the same message $(a_0, \ldots, a_t)$ with grade 2.

Clearly, the sender sends the same polynomial $F(x)$ in the first round. Moreover, each honest party evaluates the polynomial on a random point, and sends it to each other party. Since all honest parties hold the same polynomial $F(x)$, they are all happy in round 3. Thus, party P_j receives the point $(\texttt{YourPoint}, F(\alpha_j))$ from at least $2t+1$ parties in round 4 and sends $(\texttt{Reconstruct}, F(\alpha_j))$ to all other parties. That is, at least $2t+1$ parties sends points on the same polynomial of degree-t, $F(x)$. As a result, the robust interpolation succeeds to all parties, and results with $F(x)$. All honest parties interpolate $F(x)$ and output grade 2.

Agreement. We show that except for probability ϵ, if an honest party outputs grade 2 then all honest parties output the same output and with grade ≥ 1. An honest party outputs grade 2 if (see Output, Step 2 in Protocol 3.2):

1. It was happy at round 3, that is, it received from at least $2t+1$ parties random points that agree with $F_i(x)$.

2. There is a unique robust interpolation at the end of round 4 to a polynomial $F'_i(x)$ and $F_i(x) = F'_i(x)$.
3. It received $(\texttt{YourPoint}, \gamma_i)$ with the same γ_i from at least $2t+1$ other parties, and it holds that $\gamma_i = F'_i(\alpha_i)$, where $F'_i(x)$ is the polynomial it has reconstructed;

Since the honest party that outputs grade 2 has received at least $2t+1$ round 3 messages $(\texttt{YourPoint}, \gamma_i)$ with the same γ_i, it means that at least $t+1$ honest parties sent it a point and therefore are happy. We will show below (Lemma 3.3) that except for probability ϵ, all honest parties that are happy hold the same polynomial. Thus, all honest parties that are happy in round 3 hold the same polynomial, denoted as $F_i(x)$. Thus, each other honest party P_k in round 3 receives $(\texttt{YourPoint}, F_i(\alpha_k))$ with the same value $F_i(\alpha_k)$ from at least $t+1$ parties. Since all honest parties that are happy hold the same polynomial, and since the number of corrupted parties is at most t, each honest party P_k cannot receive any other message that has plurality $t+1$. This implies that P_k must send $(\texttt{Reconstruct}, F_i(\alpha_k))$ to all other parties. Since this holds for any honest party, we get that there all parties receive at least $2t+1$ points on the same polynomial $F_i(x)$, and therefore there is a unique decoding to that polynomial $F_i(x)$. We conclude that all honest parties output the same polynomial $F_i(x)$ with grade at least 1.

To conclude the proof (of agreement), we prove the following lemma:

Lemma 3.3. *All honest parties that are* happy *at round 3 have the same polynomial $F(x)$, except for a probability $\binom{n}{2}\frac{t}{|\mathbb{F}|} < \epsilon$.*

Proof: First, for every two distinct fixed polynomials $f(x), g(x)$ of degree-t it holds that:

$$\Pr_{\beta \leftarrow \mathbb{F}}[f(\beta) = g(\beta)] = \frac{t}{|\mathbb{F}|} \ .$$

To see that, consider the polynomial $p(x) = f(x) - g(x)$. This is a polynomial of degree at most t. If $f(\beta) = g(\beta)$ then $p(\beta) = 0$, i.e., β is a root of the polynomial $p(x)$. Since $p(x)$ is a polynomial of degree at most t, it has at most t roots, and the according to the Schwartz–Zippel lemma (Lemma 2.1), $p(\beta) = 0$ with probability at most $t/|\mathbb{F}|$.

Now, assume that not all honest parties are happy and hold the same polynomial. This implies that there exists two honest parties P_k, P_j that are happy but hold two distinct polynomials, say $F_k(x), F_j(x)$, respectively. Since each party is happy it must have received $2t+1$ points that agree with its polynomial. Let $\mathsf{Agree}_k \subseteq [n]$ (resp. $\mathsf{Agree}_j \subseteq [n]$) be the set of parties that sent points that were accepted by P_k (resp. P_j). Since $|\mathsf{Agree}_k| \geq 2t+1$ and $|\mathsf{Agree}_k| \geq 2t+1$, and $|\mathsf{Agree}_k \cup \mathsf{Agree}_j| \leq 3t+1$, we get that $|\mathsf{Agree}_k \cap \mathsf{Agree}_j| \geq t+1$. That is, there is at least one honest party in the intersection, i.e., at least one honest party that sent a random point that agreed with both $F_k(x)$ and $F_j(x)$. In any case, either P_k received a correct point from some honest party that does not hold $F_k(x)$, or P_j received a correct point from some honest party that does not hold $F_j(x)$. This occurs with probability at most $t/|\mathbb{F}|$. The claim then holds by a union bound over all pairs of parties P_k, P_j. That is:

$$\binom{n}{2}\frac{t}{|\mathbb{F}|} < \frac{n^3}{|\mathbb{F}|} = \frac{n^3}{2^{L/t}} < \epsilon \ ,$$

where the last step is true since $L > n/3 \cdot (\log(n^3/\epsilon))$. □

Non-equivocation. We show that if two honest parties output a grade ≥ 1, then they output the same value. A party P_i outputs grade 1 only if:

1. It received at least $2t+1$ non-$\perp$ round 4 messages $(\texttt{Reconstruct}, \cdot)$;
2. Those $2t+1$ points have a unique interpolation.

Let $F_i(x), F_j(x)$ be the output of honest parties P_i, P_j, respectively, that both have grade ≥ 1. We now show that $F_i(x) = F_j(x)$. To output grade ≥ 1, both must have received $2t+1$ non-$\perp$ round 4 messages, $(\texttt{Reconstruct}, \cdot)$. Thus, they must have received non-$\perp$ round 4 messages from at least $t+1$ honest parties. Each such honest party have received at least $t+1$ round 3 messages, $(\texttt{YourPoint}, \gamma)$ with the same value γ, which implies that there is at least one honest party that is happy at round 3. As we saw from Lemma 3.3, except for probability ϵ, all honest parties that are happy hold the same polynomial in round 3. Denote that polynomial as $F'(x)$. We now show that $F_i(x) = F_j(x) = F'(x)$.

In round 3, the happy honest parties send $(\texttt{YourPoint}, F'(\alpha_k))$ to each honest party P_k. Therefore, to reach $t+1$ values that are the same, (except for probability ϵ) the only message that can be sent by an honest P_k in round 4 is the message $(\texttt{Reconstruct}, F'(\alpha_k))$. Since P_i and P_j have a unique interpolation, i.e., there exists a unique polynomial of degree-t that is of distance at most t from the values they have received, then it must be that this decoded codeword is $F'(x)$. The only errors are obtained from the points of the corrupted parties, and those are of distance at most t. We remark that some honest parties might output $\perp$. It might be that some honest parties received $2t+1$ points on $F'(x)$ while others did not (i.e., corrupted parties might contribute correct points to only some subset of honest parties). Recall that non-equivocation requires that all honest parties that output some output with grade ≥ 1 output the same value, and there is no guarantee that all honest parties output the same output. □

Making the protocol balanced. In the above protocol, all parties (for large enough L) send or receive $O(L)$ bits, except for the sender, who sends $O(nL)$ bits. To avoid bottlenecks, it is preferable to construct a balanced protocol [DXR22] where all parties, including the sender, send or receive $O(L)$ bits.

To make the protocol balanced, we let the sender send each party P_i just the point $F(\alpha_i)$. Each party then forwards the point it received to all other parties. The parties then use the Reed-Solomon decoding procedure to reconstruct the polynomial, and treat this polynomial as the message that the sender has sent in the first message of Protocol 3.2. If there is no unique interpolation, then we treat it as the sender has sent $\perp$. Note that if the sender is honest, then all honest parties receive at least $2t+1$ correct points and therefore all reconstruct the exact same polynomial. In case of a corrupted sender, it is easy to map any outcome of the new protocol (after those two rounds) to an outcome in the original protocol, where the sender send different inputs to different parties.

4 Reliable Broadcast (Asynchrony)

Model. We assume pairwise channels, and a *computationally-unbounded* adversary. We first show a protocol where we assume *private channels* – the adversary cannot see messages transmitted between two honest parties, and it alter them or inject new messages. Nevertheless, we assume *asynchronous* network, and so the adversary can arbitrarily delay messages sent between honest parties. Yet, each message from an honest party to another honest party would eventually arrive. In Section 4.1 we extend the protocol and add one more round ti remove the private channel assumption.

We prove the following theorem:

Theorem 4.1. *Let $\epsilon > 0$ and $t \in \mathbb{N}$ and $n \geq 3t+1$. Protocol 4.2 is an n-party reliable broadcast protocol such that for input of length $L > n/3 \cdot (\log(n^3/\epsilon))$ bits tolerates t malicious parties and requires the transmission of $O(nL)$ bits. The security is unconditional, and the error probability of non-equivocation and termination is ϵ.*

Notations. Before proceeding to the protocol, we describe some writing conventions for asynchronous protocols, following [CR93]. The protocol is a sequence of instructions. Upon activation, the player scans all instruction in the specified order. Each instruction has one of the three following possible forms:

- ⟨instruction⟩. Here the instruction is carried out at the first activation of the protocol.
- `Wait until` ⟨condition⟩. `Then` ⟨instruction⟩. If the condition is satisfied, and this instruction was not executed in the previous activation, then the instruction is execution. Otherwise, the instruction is ignored.
- `If` ⟨condition⟩ `then` ⟨instruction⟩. Here, if the condition is satisfied, then the instruction is execution, even if it was already executed in a previous activation.

Protocol 4.2: Reliable Broadcast for $t < n/3$ (asynchrony)

- **Input:** The sender holds $t+1$ field elements $a_0, \ldots, a_t \in \mathbb{F}$.
- **Initialization:** Each P_i sets $F_i = F_i' = \perp$, $\mathsf{Agree_i} = \emptyset$.
- **The protocol:**
  1. **The sender:** Encode $F(x) = \sum_{i=0}^{t} a_i x^i$ and send $F(x)$ to each party P_i.
  2. **Each party P_i:** Wait until a polynomial is received from the sender, and set it as F_i. Then, send to each party P_j the point $(\beta_{i,j}, F_i(\beta_{i,j}))$ for a random $\beta_{i,j} \in \mathbb{F}$.
  3. **Each party P_i:**
     (a) Wait until the message $(\beta_{j,i}, \gamma_{j,i})$ is received from party P_j. If $F_i(\beta_{j,i}) = \gamma_{j,i}$ then add j to Agree_i.
     (b) Wait until $|\mathsf{Agree}_i| \geq 2t+1$. Then, P_i is set itself as `happy`, and sets $F_i'(x) = F_i(x)$. Moreover, it sends the messages to each P_j (1) $(\texttt{YourPoint}, F_i(\alpha_j))$; (2) $(\texttt{Reconstruct}, (\alpha_i, F_i(\alpha_i)))$. Then, it moves to Output step.
  4. **Each party P_i with $F_i' = \perp$:**
     (a) Wait until $t+1$ messages $(\texttt{YourPoint}, \gamma_i)$ with the same value γ_i are received. Then, send to each other party the message $(\texttt{Reconstruct}, (\alpha_i, \gamma_i))$.
     (b) If a message $(\texttt{Reconstruct}, (\alpha_i, \gamma_i))$ was sent, then upon each time a message of the form $(\texttt{Reconstruct}, (\cdot, \cdot))$ arrives:
     If a total of $2t+1$ messages $(\texttt{Reconstruct}, (\alpha_j, \gamma_j))$ arrived, then use Reed-Solomon decoding procedure to find the unique degree-t polynomial $G_i(x)$ for which $G_i(\alpha_j) = \gamma_j$ for at least $2t+1$ points. If such a polynomial exists, then set $F_i'(x) := G_i(x)$, and send $(\texttt{YourPoint}, F_i'(\alpha_k))$ to each party P_k.
     If no such polynomial exists, then wait to receive more $(\texttt{Reconstruct}, (\cdot, \cdot))$ messages.
- **Output – each party P_i:** If $F_i'(x) \neq \perp$ and $2t+1$ messages with $(\texttt{YourPoint}, (\alpha_i, \gamma_i))$ with $F_i'(\alpha_i) = \gamma_i$ were received, then output $F_i'(x)$.

Proof of Theorem 4.1:

Validity: We claim that if the sender is honest then all honest parties output the same polynomial $F(x)$ that the sender holds as input. We show:

Claim 4.3. *The following holds:*

1. *The first honest party that sends the message* `YourPoint` *to all other honest parties must be* `happy`*. It holds $F'_i(x) = F(x)$, where $F(x)$ is the polynomial the sender holds as input.*
2. *For every $k \geq 1$, the $(k+1)$th honest party that sends the message* `YourPoint` *to all parties must have set $F'_i(x) = F(x)$, and send to each party P_k the message* $(\texttt{YourPoint}, (\alpha_k, F(\alpha_k)))$.

Proof: There are two options that an honest party would send the `YourPoint` message:

1. If the party is `happy`, then it holds the same polynomial that it has originally received from the sender. When $2t+1$ `YourPoint` messages would arrive that agree with its point, the party would terminate and output that polynomial.
2. If the party is not `happy`, then it has received $2t+1$ messages of the form $(\texttt{Reconstruct}, (\cdot, \cdot))$ and there exists a unique decoding. The party then sets $F'_i(x)$ to be the unique decoded polynomial, and sends the message `YourPoint` to all other parties.

Clearly, the first honest party that sent `YourPoint` message must be `happy`. If all honest parties are not happy, then no honest party sends $(\texttt{YourPoint}, (j, \cdot))$. If there are messages of the form $(\texttt{YourPoint}, (j, \cdot))$ that are sent, the first message of this form that was sent must be from an honest party that is `happy`.

Now, assume (by induction) that the first k honest parties that sent `YourPoint` hold the same polynomial $F(x)$ as the sender holds as input. We claim that the $(k+1)$th party that sends `YourPoint` holds the same polynomial $F(x)$. If the party sends `YourPoint` because it is `happy`, then it sets $F'_i(x)$ according to the message it has received from the sender, i.e,. $F(x)$. If the party sends `YourPoint` while it is not `happy`, then it must have received $2t+1$ messages $(\texttt{Reconstruct}, (\cdot, \cdot))$. An honest party P_i might send $(\texttt{Reconstruct}, (\alpha_i, \gamma_i))$ only if it is `happy` and then $\gamma_i = F(\alpha_i)$, or, if it has received $t+1$ messages of the form $(\texttt{YourPoint}, (\alpha_i, \gamma_i))$ with the same γ_i. That is, it must have received it also from at least one honest party. According to our assumption, the first k honest parties that sent $(\texttt{YourPoint}, (\alpha_i, \gamma_i))$ must have sent it with $\gamma_i = F(\alpha_i)$. As a result, the only point that an honest party might send with its `Reconstruct` message lies on the polynomial $F(x)$, and thus the only polynomial of degree-t that can have a unique decoding is the polynomial $F(x)$. □

The above shows that all honest parties that sends `YourPoint` message hold the same polynomial $F(x)$ that the sender holds as input. We also claim that all honest parties would eventually terminate. For that we show that there is a path in which the execution terminates. All honest parties receive the same polynomial from the sender and send a random point to each other. The adversary might introduce delays, but if a party does not terminate earlier due to unique decoding, it would eventually receive $2t+1$ "good points" and would be `happy`. If not terminated earlier due to unique decoding, every honest party would (eventually) become `happy`. We are guaranteed therefore that all honest parties will send the `YourPoint` message, eventually, and therefore eventually all honest parties will terminate.

Termination and non-equivocation: For Termination we show that if an honest party P_j terminates with output $F'(x)$, then with probability at least $1-\epsilon$ all other honest parties must terminate with the same polynomial $F'(x)$.

Using Claim 4.4 lets assume the good event that all honest parties that are `happy` hold the same $F(x)'$.

P_j terminates only if it has received $2t+1$ messages of the form $(\texttt{YourPoint}, (\alpha_j, F'(\alpha_j)))$. This implies that there is a set S of honest parties of cardinality at least $t+1$ that sent $(\texttt{YourPoint}, (\alpha_j, F'(\alpha_j)))$ message to P_j.

From our assumption on the good event of Claim 4.4, all parties S (of cardinality at least $t+1$) hold the same polynomial $F'(x)$.

We now show that all other honest parties will eventually terminate with $F'(x)$. The honest parties in S and any honest party that becomes happy send the message $(\texttt{YourPoint}, (\alpha_k, F'(\alpha_k)))$ to each honest party P_k. Thus, P_k would receive at least $t+1$ points $F'(\alpha_k)$ (and at most t conflicting points), and hence would eventually send $(\texttt{Reconstruct}, (\alpha_k, F'(\alpha_k)))$. Since this holds for all honest parties, each party would receive at least $2t+1$ messages `Reconstruct` on the same polynomial $F'(x)$, and this is the only unique reconstruction possible. That is, each honest P_j party eventually sets $F'_j(x) = F'(x)$. Each party then would send $(\texttt{YourPoint}, (\alpha_k, F'(\alpha_k)))$ to each other party P_k, and so each party P_k would receive $2t+1$ messages of the form `YourPoint` and output $F'(x)$. That is, all honest parties terminate and with the same polynomial.

To claim that S hold the same polynomial $F'(x)$ except for probability ϵ, we claim that all honest parties that are happy must hold the same polynomial (except for some probability ϵ). We prove this below (Claim 4.4). Moreover, similarly to Claim 4.3, we claim that the first party that sent `YourPoint` must be happy. Moreover, all other parties that send message `YourPoint` must hold the same polynomial as the first k parties that sent `YourPoint`. This holds from similar arguments to that of Claim 4.3. To conclude, we show:

Claim 4.4. *If the sender is corrupted, then all honest parties that are* happy *in Round 3 hold the same polynomial $F(x)$, except for a probability $\binom{n}{2} \cdot \frac{t}{|\mathbb{F}|}$.*

Proof: As in the proof of Lemma 3.3, for every two distinct and fixed polynomials $f(x), g(x)$ of degree-t it holds that $\Pr_{\beta \leftarrow \mathbb{F}}[f(\beta) = g(\beta)] = \frac{t}{|\mathbb{F}|}$. If there are two honest parties that do not hold the same probability then it must be the some honest party received a point that is "correct" from a party that does not hold the same polynomial, which occurs with probability at most $t/|\mathbb{F}|$. The claim then holds by a union bound over all pairs of parties. We note that this assumes that the channel between the two honest parties is private. Otherwise, the adversary might choose the polynomial $F_j(x)$ for some party P_j only after it saw the point $\beta_{i,j}$ from P_i. □

This concludes the proof of Theorem 4.1. □

4.1 On Private Channels and Balance Communication

Protocol 4.2 works in the private channels model, where the adversary might choose the polynomials it sends to some honest party P_j based on the the choice of $\beta_{i,j}$ the party P_j receives from some honest P_i. Therefore, we rewrite the first steps of the protocol:

1. **The sender:** Encode $F(x) = \sum_{i=0}^{t} a_i x^i$ and send $F(x)$ to each party P_i.
2. **Each party P_i:** Wait until a polynomial is received from the sender, and set it as F_i. Then, send MessageRecieved to each party P_j.
3. **Each party P_i:** If already received a polynomial from the sender, and upon receiving the message MessageRecieved from P_j, send to P_j the point $(\beta_{i,j}, F_i(\beta_{i,j}))$ for a random $\beta_{i,j} \in \mathbb{F}$.
4. Continue Protocol 4.2 from Step 3.

Note that now for every pair of honest parties, the two parties check that the polynomials agree only after confirming receiving them from the sender. Thus, the two polynomials are fixed and therefore if they are not the same, they agree on a random point with probability at most $t/|\mathbb{F}|$ as in Claim 4.4.

Making the protocol balanced. Similarly to our gradecast protocol, we can make the protocol balanced by letting the sender send to each party P_i a single point $F(\alpha_i)$, and each party waits until it has a robust interpolation before proceeding with the other steps of the protocol (i.e., before sending MessageRecieved).

Exact round complexity. We count the round complexity of the above protocol. Our basic protocol (no balanced, private channels) requires 3 rounds of communication in the optimistic case, and up to 5 rounds in the pessimistic case. If there are no private channels, we have to add one more round to both cases, and to make the protocol balanced, we have to add one additional round to both cases. Overall, balanced protocol, pessimistic, and without assuming private channels requires 7 rounds of communication.

5 Conclusion

We show two simple protocols for gradecast in the synchronous model, and reliable broadcast in the asynchronous model. Both protocols achieve statistical security. Our work leaves two directions of future work: (1) Better understanding the trade-off in the exact number of rounds of perfectly secure and unconditionally secure protocols; (2) Finding asymptotically optimal gradecast and reliable broadcast results for the regime $n < |L| \leq n \log n$ remains an open question.

Acknowledgement

We thank Sourav Das, Xiang Zhuolun and Ling Ren for pointing out the need to address private and non-private channels.

References

[ADD+22a] Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. to appear in PODC 2022, 2022.

[ADD+22b] Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Brief announcement: Asynchronous verifiable information dispersal with near-optimal communication. to appear in PODC 2022, 2022.

[ADVZ21] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. Succinct erasure coding proof systems. Cryptology ePrint Archive, Report 2021/1500, 2021. https://ia.cr/2021/1500.

[AJM+21] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, page 363–373, New York, NY, USA, 2021. Association for Computing Machinery.

[BDH10a] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Brief announcement: Simple gradecast based algorithms. In *DISC*, volume 6343 of *Lecture Notes in Computer Science*, pages 194–197. Springer, 2010.

[BDH10b] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Simple gradecast based algorithms. *CoRR*, abs/1007.1049, 2010.

[Bra87] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, November 1987.

[Che21] Jinyuan Chen. Optimal Error-Free Multi-Valued Byzantine Agreement. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[CR93] Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 42–51. ACM, 1993.

[CT05] C. Cachin and S. Tessaro. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201, 2005.

[DXR21] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2705–2721, New York, NY, USA, 2021. Association for Computing Machinery.

[DXR22] Sourav Das, Zhuolun Xiang, and Ling Ren. Balanced quadratic reliable broadcast and improved asynchronous verifiable information dispersal. Cryptology ePrint Archive, Report 2022/052, 2022. https://ia.cr/2022/052.

[DZ22] Sisi Duan and Haibin Zhang. Byzantine reliable broadcast with $o(nl + kn + n^2 logn)$ communication. Cryptology ePrint Archive, Report 2022/554, 2022. https://ia.cr/2022/554.

[FGH+02] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam Smith. Detectable Byzantine Agreement secure against faulty majorities. In *Proc. 21st ACM Symposium on Principles of Distributed Computing — PODC 2002*, pages 118–126, 7 2002.

[FLM85] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing*, PODC '85, page 59–70, New York, NY, USA, 1985. Association for Computing Machinery.

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 148–161, New York, NY, USA, 1988. Association for Computing Machinery.

[KK09] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, February 2009.

[Ree60] Gustave Reed, Irving S.; Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.