Communication-Efficient and Error-Free Gradecast with Optimal Resilience

Jianjun Zhu, Fan Li, Jinyuan Chen
Louisiana Tech University
Department of Electrical Engineering, Ruston, LA 71272, USA
{jzh013, fanli, jinyuan}@latech.edu

***Abstract*—Gradecast is a variant of the Byzantine broadcast problem introduced by Feldman and Micali in 1988. In Gradecast, n processors would like to agree on a value sent from a leader, as well as a grade in $\{0, 1, 2\}$, such that the following three requirements are satisfied: 1) Every non-faulty processor outputs the leader's initial value and grade 2 if the leader is non-faulty; 2) For any two non-faulty processors, if their decided grades are greater than zero, then they output the same value; and 3) For any two non-faulty processors, the difference of their decided grades is less than 2. In this work, we present a new Gradecast protocol with a total communication complexity of $O(n\ell + n^2 \log n)$ bits, given $t < n/3$, where ℓ is the message size and t is the maximum number of faulty processors tolerated in n consensus processors. The proposed protocol is an error-free and deterministic Gradecast protocol that does not rely on the authentication techniques such as signatures and secret sharing. The proposed protocol is also information-theoretic secure, i.e., it satisfies the above three requirements even if the computation power of the adversary is unbounded.**

I. Introduction

Graded-broadcast or Gradecast is a relaxed version of the Byzantine broadcast problem [1], [2]. In the Byzantine broadcast problem, n processors would like to agree on a value sent from a leader who is potentially dishonest or faulty, such that: 1) All non-faulty processors output the leader's initial value if the leader is non-faulty (validity); 2) All non-faulty processors should have the same output (consistency); and 3) All non-faulty processors eventually terminate (termination) [3], [4]. In Gradecast, n processors would like to agree on a value sent from a leader, as well as a grade in $\{0, 1, 2\}$, such that the following three requirements are satisfied: 1) All non-faulty processors output the leader's initial value and grade 2 if the leader is non-faulty (validity); 2) For any two non-faulty processors, if their decided grades are greater than zero, then they output the same value (non-equivocation); and 3) For any two non-faulty processors, the difference of their decided grades is less than 2 (confidence). As we can see, Gradecast does not require strict conditions in termination and consistency like Byzantine broadcast.

Recently, Gradecast was studied in [5], [6] in terms of communication complexity. Specifically, the authors in [5] proposed a Gradecast protocol with a total communication complexity of $O(n\ell + n^3 \log n)$ bits, where ℓ is the size of the message. This communication complexity performance was improved in [6] to $O(n\ell + n^2 \log n)$ bits, however, at a cost of non-zero error probability.

TABLE I
Comparison of Different Gradecasts.

Reference	Resilience	Communication Complexity	Error-Free
[1], [2]	$n \geq 3t + 1$	$O(n^2\ell)$	Yes
[5]	$n \geq 3t + 1$	$O(n\ell + n^3 \log n)$	Yes
[6]	$n \geq 3t + 1$	$O(n\ell + n^2 \log n)$	No
This Paper	$n \geq 3t + 1$	$O(\max\{n\ell, nt \log t\})$	Yes

In this work we focus on the *error-free* Gradecast protocol. Specifically, we propose an error-free Gradecast protocol that achieves the communication complexity of $O(\max\{n\ell, nt \log t\})$ bits, given $t < n/3$, where t is the maximum number of faulty processors tolerated in the consensus network. The proposed Gradecast protocol is error-free, i.e., it guarantees that the three requirements of validity, non-equivocation, and confidence of Gradecast are satisfied in all executions. The proposed Gradecast protocol is also information-theoretic secure, i.e., it satisfies the above three requirements even if the computation power of the adversary is unbounded. The proposed protocol is built from the COOL protocol in [7], [8], which was originally designed for the Byzantine broadcast and agreement problems. The comparison of different Gradecast protocols is shown in Table I.

Gradecast is the fundamental building block in various protocols for secure multi-party computation and Byzantine agreement. In this work, we show that the proposed Gradecast protocol could also be applied to some other settings such as all-to-all Gradecast and approximate agreement.

Throughout this work, $[n]$ denotes the set of integers from 1 to n. Majority$(\bullet)$ is a function that returns the most frequent value of the inputs. Logarithms are in base 2. The rest of this paper is organized as follows. Section II describes the system models. The main results are provided in Section III.

This work was supported in part by the NSF EPSCoR-Louisiana Materials Design Alliance (LAMDA) Program under Grant OIA-1946231.

TABLE II
APPLICATIONS OF GRADECAST.

Applications	Reference	Resilience	Communication Complexity	Round Complexity
All-to-All Gradecast	[1], [2]	$n \geq 3t+1$	$O(n^3\ell)$	$O(1)$
	[9]	$n \geq 3t+1$	$O(n^2 t\ell)$	$O(1)$
	This Paper	$n \geq 3t+1$	$O(\max\{n^2\ell, n^2 t\log t\})$	$O(1)$
Approximate Agreement	[10]	$n \geq 3t+1$	$O(n^2\ell\tau)$	$O(\log n)$
	[11]	$n \geq 3t+1$	$O(n^3\ell\tau)$	$O(\log n/\log\log n)$
	[9]	$n \geq 3t+1$	$O(n^2 t\ell\tau)$	$O(\log n/\log\log n)$
	This Paper	$n \geq 3t+1$	$O(n^2\ell\tau)$	$O(\log n/\log\log n)$

The proposed Gradecast protocol is discussed in Section IV. Some Gradecast-based applications are discussed in Section V. Finally, this work is concluded in Section VI.

II. System models

We consider the Gradecast problem in a synchronous setting. Particularly, every pair of processors are connected through a reliable and private message-passing channel. One of the n processors, designated as the leader, sends its initial input value to all the processors in the network. In this network, up to t processors may be Byzantine processors who have complete knowledge of the state of the other processors. All of the non-faulty processors will perform according to the protocol design. We provide the formal definitions of Gradecast as follows:

Definition 1 (Gradecast)**.** *Let $\mathcal{P}$ be a protocol in which one of the n processors $\{P_1, ..., P_n\}$, publicly known as the leader, holds an initial input value $\boldsymbol{w}$ of ℓ bits, while every processor P_i outputs a pair $(\boldsymbol{w}_i, g_i)$, where $g_i \in \{0, 1, 2\}$ (see Fig. 1). We say $\mathcal{P}$ is a Gradecast protocol if the following conditions hold:*

- *Validity: If the leader is non-faulty, then all non-faulty processors output the leader's value $\boldsymbol{w}$ and grade 2.*
- *Non-equivocation: For any two non-faulty processors P_i and P_j, if $g_i > 0$ and $g_j > 0$, then $\boldsymbol{w}_i = \boldsymbol{w}_j$.*
- *Confidence: For any two non-faulty processors P_i and P_j, $|g_i - g_j| \leq 1$.*

Definition 2 (All-to-All Gradecast)**.** *All-to-all Gradecast consists of n parallel Gradecasts, where the ith Gradecast corresponds to the standard Gradecast defined in Definition 1, with the ith processor being the leader.*

Definition 3 (Approximate Agreement)**.** *Assume that each non-faulty processor starts with a real input value. For any preassigned $\epsilon > 0$ (as small as desired), an approximate agreement algorithm is said to be t-correct if the following two conditions hold:*

- *Agreement: All non-faulty processors eventually halt with output values that are within ϵ of each other.*
- *Validity: Every non-faulty processor's output must be within the range of the initial input values of non-faulty processors.*

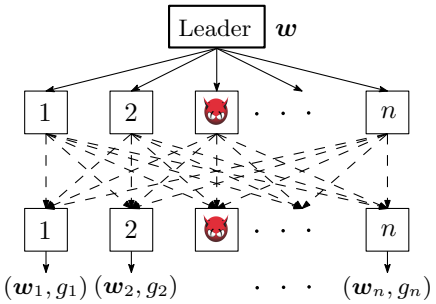

Fig. 1. One-to-All Gradecast

Remark 1. *Communication complexity refers to the total amount of communication in bits. Resilience is the maximum number of faulty processors that the protocol can tolerate while maintaining robustness. Round complexity refers to the expected number of rounds to exchange information.*

III. Main results

In this work, we propose an error-free Gradecast protocol called EFGradecast. The performance of EFGradecast is provided below.

Theorem 1 (Gradecast)**.** *The proposed EFGradecast is an error-free graded broadcast protocol with a total communication complexity of $O(max\{n\ell, nt\log t\})$ bits, given $n \geq 3t+1$.*

Proof. The proof of Theorem 1 is provided in Section IV (See Lemmas 1-5). Lemma 1 shows that the proposed EFGradecast has a communication complexity of $O(\max\{n\ell, nt\log t\})$ bits.

Algorithm 1 : EFGradecast protocol, code for P_i, $i \in [n]$

Phase 1:

The leader sends $\boldsymbol{w}$ to P_i. P_i sets $\boldsymbol{w}^{(i)} = \boldsymbol{w}$, $\forall i \in [n]$.

P_i encodes $\boldsymbol{w}^{(i)}$ into n symbols as $y_j^{(i)} \triangleq \boldsymbol{h}_j^{\intercal} \boldsymbol{w}^{(i)}, j \in [n]$.

P_i sends $(y_j^{(i)}, y_i^{(i)})$ to P_j, $j \in [n] \setminus i$.

P_i sets a binary link indicator $\mathrm{u}_i(j)$ as 1 if $(y_i^{(j)}, y_j^{(j)}) == (y_i^{(i)}, y_j^{(i)})$; otherwise, P_i sets $\mathrm{u}_i(j)$ as 0.

P_i sets a binary success indicator s_i as 1 if $\sum_{j=1}^{n} \mathrm{u}_i(j) \geq n - t$; otherwise, P_i sets s_i as 0 and $\boldsymbol{w}^{(i)}$ as $\perp$.

P_i sends s_i to P_j, $j \in [n] \setminus i$.

P_i creates two sets: $\mathcal{S}_1 = \{j : \mathrm{s}_j = 1, j \in [n]\}$, $\mathcal{S}_0 = \{j : \mathrm{s}_j = 0, j \in [n]\}$.

Phase 2: P_i **whose** s_i **is 1**

P_i updates $\mathrm{u}_i(j)$ to 0, $\forall j \in \mathcal{S}_0$.

if $\left(\sum_{j=1}^{n} \mathrm{u}_i(j) < n - t\right)$ **then**

    P_i sets $\mathrm{s}_i = 0$ and $\boldsymbol{w}^{(i)} = \perp$.

    P_i sends s_i to P_j for $j \in [n] \setminus i$ and updates $\mathcal{S}_1$ and $\mathcal{S}_0$.

Phase 3: P_i **whose** s_i **is 1**

P_i updates $\mathrm{u}_i(j)$ to 0, $\forall j \in \mathcal{S}_0$.

if $\left(\sum_{j=1}^{n} \mathrm{u}_i(j) < n - t\right)$ **then**

    P_i sets $\mathrm{s}_i = 0$ and $\boldsymbol{w}^{(i)} = \perp$.

    P_i sends s_i to P_j and updates $\mathcal{S}_1$ and $\mathcal{S}_0$.

P_i sets a binary voting indicator v_i as 1 if $\sum_{j=1}^{n} \mathrm{s}_j \geq 2t + 1$; otherwise, P_i sets v_i as 0.

P_i sends v_i to $P_j, j \in [n] \setminus i$.

Phase 4 and Output:

P_i sets a binary ready indicator r_i as 1 if $\sum_{j=1}^{n} \mathrm{v}_j \geq t + 1$; otherwise, P_i sets r_i as 0.

P_i sends r_i to $P_j, j \in [n] \setminus i$.

if $\left(\sum_{j=1}^{n} \mathrm{r}_j \geq 2t + 1\right)$ **then**

    if $\mathrm{s}_i == 0$ **then**

        P_i updates $y_i^{(i)} \leftarrow$ Majority$(\{y_i^{(j)} : j \in \mathcal{S}_1\})$.

        P_i sends updated $y_i^{(i)}$ to $P_j, \forall j \in \mathcal{S}_0$.

        P_i decodes the message $\boldsymbol{w}^{(i)}$ using $\{y_j^{(j)} : j \in [n]\}$.

    P_i outputs $(\boldsymbol{w}^{(i)}, 2)$ if $\sum_{j=1}^{n} \mathrm{v}_j \geq 2t + 1$; otherwise P_i outputs $(\boldsymbol{w}^{(i)}, 1)$.

else

    P_i outputs $(\perp, 0)$.

Lemmas 3-5 reveal that the proposed EFGradecast satisfies the validity, non-equivocation, and confidence conditions of Gradecast. □

The proposed EFGradecast can be applied to all-to-all Gradecast by invoking the proposed EFGradecast n times in a parallel manner (called A2A-EFGradecast). A2A-EFGradecast is an error-free all-to-all Gradecast protocol with a total communication complexity of $O(\max\{n^2\ell, n^2 t \log t\})$ bits. The comparison of different all-to-all Gradecast protocols is provided in Table II.

The proposed EFGradecast can also be extended to approximate agreement. The protocol is provided in Algorithm 2 (see Section V-B). This protocol has a total communication complexity of $O(n^2\ell\tau)$ bits, where τ is the number of rounds. Table II provides the comparison of different approximate agreement protocols.

IV. EFGradecast

The proposed EFGradecast protocol is shown in Algorithm 1. In this protocol, each processor receives an initial message from the leader. This message might be updated in each phase. The key idea is to ensure that at the end of Phase 3,

there exists at most 1 group of non-faulty processors that hold the same non-empty updated message (see Lemma 2). In the protocol design, the (n, k) Reed-Solomon error correction code will be used in order to reduce the communication complexity. The (n, k) Reed-Solomon error correction code can correct up to $\lfloor \frac{n-k}{2} \rfloor$ errors. In what follows, we will provide the proof of Theorem 1.

Lemma 1 (Communication Complexity)**.** *The total communication complexity of the proposed EFGradecast is* $O(max\{n\ell, nt \log t\})$ *bits.*

Proof. The communication complexity of each phase in Algorithm 1 is analyzed as follows:

- Phase 1: The leader sends an ℓ-bit message to all the processors in the network. Thus, the communication complexity of this step is $n\ell$ bits (see Line 1 in Algorithm 1). Since each encoded symbol has c bits, the communication complexity of exchanging a pair of symbols is $2cn(n-1)$ bits (see Line 3 in Algorithm 1), where c and k are designed as

$$c \triangleq \lceil \frac{\max\{\ell,\ (t/5+1) \cdot \log(n+1)\}}{k} \rceil, k \triangleq \lfloor \frac{t}{5} \rfloor + 1.$$

  Since the success indicator is binary, the communication complexity of sending success indicators is $n(n-1)$ bits (see Line 6 in Algorithm 1).
- Phase 2: Because the success indicator is binary, the communication complexity of sending success indicators is $n(n-1)$ bits (see Line 11 in Algorithm 1).
- Phase 3: Since the success indicator and the voting indicator are both binary, the total communication complexity of exchanging these two indicators is $2n(n-1)$ bits (see Lines 15 and 17 in Algorithm 1).
- Phase 4: Given that the ready indicator is binary, the communication complexity of sending ready indicators is $n(n-1)$ bits (see Line 19 in Algorithm 1). Each encoded symbol has c bits, then the communication complexity of sending a symbol is $cn(n-1)$ bits (see Line 23 in Algorithm 1).

Therefore, the total communication complexity of the proposed EFGradecast is $O(\max\{n\ell, nt \log t\})$ bits. □

Lemma 2. *[7, Lemma 3] Given $n \geq 3t+1$, there exists at most one group of non-faulty processors that have the same non-empty updated message at the end of Phase 3.*

Lemma 3 (Validity)**.** *If the leader is non-faulty, then all non-faulty processors output the leader's value $\boldsymbol{w}$ and grade 2.*

Proof. If the sender is non-faulty, then all of the non-faulty processors will receive the same message $\boldsymbol{w}$ in Phase 1. In this case, for any non-faulty processor P_i and P_j, we have

$$(y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)}) \tag{1}$$

where $y_j^{(i)} \triangleq \boldsymbol{h}_j^{\mathrm{T}} \boldsymbol{w}^{(i)}$; and $\boldsymbol{h}_j$ is the jth encoded vector that can be constructed by *Lagrange polynomial interpolation* [12], as shown below

$$\boldsymbol{h}_j \triangleq \left[h_{j,1}, h_{j,2}, \cdots, h_{j,k}\right]^T \tag{2}$$

$$\text{and} \quad h_{j,m} \triangleq \prod_{p=1, p\neq m}^{k} \frac{j-p}{m-p}, \quad j \in [n],\ m \in [k]. \tag{3}$$

$\boldsymbol{w}^{(i)}$ is the message received by P_i, for $i, j \in [n]$. After exchanging information, every non-faulty processor can receive at least $2t+1$ success indicators of 1s, that is,

$$\sum_{j=1}^{n} \mathrm{s}_j \geq 2t+1 \tag{4}$$

where s_j is the success indicator of P_j for $j \in [n]$ (see Line 5 in Algorithm 1). In this case, every non-faulty processor sets a binary voting indicator as 1. Then, every non-faulty processor in Phase 4 sends a ready indicator of 1. In this case, every non-faulty processor will go to Lines 20-25 of Algorithm 1, from which every non-faulty processor decodes the same message with grade 2. □

Lemma 4 (Non-equivocation)**.** *For any two non-faulty processors P_i and P_j, if $g_i > 0$ and $g_j > 0$, then $\boldsymbol{w}_i = \boldsymbol{w}_j$.*

Proof. For any two non-faulty processors decoding with non-zero grades, they must have received at least $2t+1$ ready indicators as ones (see Lines 20 and 25 in Algorithm 1), out of which at least $t+1$ ready indicators are sent from non-faulty processors. Therefore, at least one non-faulty processor sets and sends a voting indicator as one (see Lines 16 and 18 in Algorithm 1). This implies that at least $2t+1$ processors have sent out success indicators as ones, out of which at least $t+1$ non-faulty processors have sent out success indicators as ones (see Line 16 in Algorithm 1). Based on the result of Lemma 2, at the end of Phase 3, it is guaranteed that at most 1 group of non-faulty processors hold the same non-empty updated messages. This means that the size of non-faulty processors holding the same non-empty updated message is greater than or equal to $t+1$. All of the other non-faulty processors outside this group have the same empty updated messages. In this case, all of the non-faulty processors in the set of $\mathcal{S}_0$ can calibrate their coded symbols based on the majority rule (see Line 22 in Algorithm 1), and then output the same message as other non-faulty processors (see Line 24 in Algorithm 1).

□

Lemma 5 (Confidence)**.** *For any two non-faulty processors P_i and P_j, it is true that $|g_i - g_j| \leq 1$.*

Proof. Let us consider the case where one non-faulty processor outputs with a grade of 2 while the other non-faulty processor outputs with a grade of 0. This is the only case that violates the condition of $|g_i - g_j| \leq 1$, for $i, j \in [n]$. We will prove that this case is not existing. If a non-faulty processor outputs with a grade of 2, it implies that at least

$2t+1$ processors have sent out voting indicators as ones (see Line 25 in Algorithm 1), out of which at least $t+1$ non-faulty processors have sent out voting indicators as ones. In this case, all of the non-faulty processors will send ready indicators as ones (see Lines 18-19 in Algorithm 1). In this case, every non-faulty processor can receive at least $2t+1$ ready indicators as ones (see Line 20 in Algorithm 1). This means that all of the non-faulty processors will go to Lines 20-25 in Algorithm 1. Therefore, if a non-faulty processor outputs a grade 2, then none of the non-faulty processors will output grade as 0. □

V. Applications

Gradecast is the fundamental building block in various protocols for secure multi-party computation and Byzantine agreement. In this section, we will focus on two applications, i.e., all-to-all Gradecast and approximate agreement.

A. All-to-All Gradecast

All-to-All Gradecast is a parallel version of Gradecast, where each processor broadcasts its initial message to all of the other processors in the network. It can be used as a building block in distributed computing, where one step is required to gather the information from all the processors in order to make an intermediate output for the next step. The application examples include Byzantine broadcast and agreement, approximate agreement, interactive consistency, and clock synchronization. A method was proposed in [9] to utilize forward error correction codes to mask Byzantine errors. Since the protocol in [9] invokes n instances of the original Gradecast protocol, it has a total communication complexity of $O(n^3\ell)$ bits when $t = O(n)$. In this work, we propose a new all-to-all Gradecast protocol by invoking the proposed EFGradecast n times in a parallel manner, which achieves an improved communication complexity of $O(\max\{n^2\ell, n^2 t\log t\})$ bits.

B. Approximate Agreement

Approximate agreement, first introduced in [10], is a variant of Byzantine agreement that seeks to reach agreement among a set of processors in the presence of Byzantine faults. Instead of reaching an agreement on the exact same value, approximate agreement requires every non-faulty processor to approximately agree on a value within the range of the initial values of the non-faulty processors. This allows approximate agreement to provide a weaker but more efficient guarantee compared to the Byzantine agreement, especially in situations where achieving an exact agreement is difficult or impossible due to network delays, failures, or other factors. Approximate agreement protocols can be used in a variety of applications, such as distributed databases, distributed control systems, distributed sensor networks, and collaborative machine learning. In [11], a simple Gradecast-based approximate agreement algorithm was provided with a communication complexity of $O(n^3\ell\tau)$ bits. The communication complexity was improved to $O(n^2t\ell\tau)$ bits in [9]. In this work, we improve the protocols of [11] and [9] to achieve a communication complexity of $O(n^2\ell\tau)$ bits, by replacing the original Gradecast protocol with our proposed EFGradecast (see Algorithm 2).

Algorithm 2 : Gradecast-based Approximate Agreement protocol, code for P_i, $i \in [n]$

// Notations

- BAD: a set of processors that will be ignored.
- $(P_j, \boldsymbol{w}_i, g_i)$: P_j EFGradecasted $\boldsymbol{w}_i$ with grade g_i.
- $values$: a set of received values with grade ≥ 1, and add "0" to the set until the size of $values$ is n.
- $values'$: a set of received values with grade 2.
- AVG: a function that eliminates the t lowest and highest values from the set of $values$ and takes the average of the rest.

// Initialization

1: Set $BAD \triangleq \perp$.
2: **while** $True$ **do**
   $EFGraadecast(P_i, BAD)$ with input value $\boldsymbol{w}_i$.
   $\boldsymbol{w}_i = AVG(values)$.
   $BAD = BAD \cup \{P_j | \text{received}\,(P_j, *, g_i)\,\text{with}\, g_i \leq 1\}$.
3: **if** there are $n-t$ items in $values'$ that are at most ϵ
4: apart **then**
5: $break$.
6: Do one more iteration.
7: **return** $\boldsymbol{w}_i$

C. Other applications built on the variants of Gradecast

In addition to the two applications mentioned above, variants of Gradecast can be applied to Byzantine agreement [1], [2], [13]–[17], multidimensional graded consensus [18], [19], interactive consistency [3], [9], [20], and multi-consensus [11], [21]. The authors in [22] proposed a protocol with multiple grades, which was used for a randomized broadcast algorithm. The authors in [23] proposed a moderated Gradecast where a moderator re-gradecasts the values received from the sender. It is worth mentioning that Gradecast is also useful for establishing verifiable secret sharing (VSS) protocols [1], [2], [5]. For example, Gradecast was used to grade secrets in [1], [2] and to share more secrets in [5].

VI. Conclusion

In this work, we proposed an error-free and deterministic Gradecast protocol called EFGradecast. The proposed EFGradecast achieves a total communication complexity of $O(n\ell + n^2\log n)$ bits given $n \geq 3t+1$. EFGradecast is a simple protocol with a constant number of rounds. The proposed protocol can be applied to Byzantine agreement, secure multi-party computation, all-to-all Gradecast, and approximate agreement problems. One of the future work is to extend the proposed protocol from the synchronous setting to the asynchronous setting.

REFERENCES

[1] P. Feldman and S. Micali, "Optimal algorithms for Byzantine agreement," in *ACM symposium on Theory of computing (STOC)*, Jan. 1988.

[2] ——, "An optimal probabilistic protocol for synchronous Byzantine agreement," *SIAM Journal on Computing*, vol. 26, no. 4, pp. 873–933, Aug. 1997.

[3] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, Apr. 1980.

[4] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[5] I. Abraham, G. Asharov, S. Patil, and A. Patra, "Asymptotically free broadcast in constant expected time via packed VSS," in *Theory of Cryptography (TCC)*, Nov. 2022, pp. 384–414.

[6] I. Abraham and G. Asharov, "Gradecast in synchrony and reliable broadcast in asynchrony with optimal resilience, efficiency, and unconditional security," in *ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2022, pp. 392–398.

[7] J. Chen, "Fundamental limits of Byzantine agreement," 2020, available on ArXiv: https://arxiv.org/pdf/2009.10965.pdf.

[8] ——, "Optimal error-free multi-valued Byzantine agreement," in *International Symposium on Distributed Computing (DISC)*, Oct. 2021.

[9] J. Bridgman and V. Garg, "All-to-all gradecast using coding with Byzantine failures," in *Stabilization, Safety, and Security of Distributed Systems (SSS)*, vol. 7596, Oct. 2012, pp. 285–298.

[10] D. Dolev, N. Lynch, S. Pinterr, E. Stark, and W. Weihl, "Reaching approximate agreement in the presence of faults," *Journal of the ACM*, vol. 33, no. 3, pp. 499 – 516, May 1986.

[11] M. Ben-Or, D. Dolev, and E. Hoch, "Simple gradecast based algorithms," 2010, arXiv preprint arXiv:1007.1049.

[12] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, Jun. 1960.

[13] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, and A. Smith, "Detectable Byzantine agreement secure against faulty majorities," in *ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2002, pp. 118–126.

[14] S. Goldwasser, E. Pavlov, and V. Vaikuntanathan, "Fault-tolerant distributed computing in full-information networks," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, Oct. 2006, pp. 15–26.

[15] M. Ben-Or, E. Pavlov, and V. Vaikuntanathan, "Byzantine agreement in the full-information model in $o(logn)$ rounds," in *ACM Symposium on Theory of Computing (STOC)*, May 2006, pp. 179–186.

[16] J. Katz and C. Koo, "On expected constant-round protocols for Byzantine agreement," in *Annual International Cryptology Conference (CRYPTO)*, vol. 4117, Aug. 2006, pp. 445–462.

[17] G. Deligios, M. Hirt, and C. Liu-Zhang, "Round-efficient Byzantine agreement and multi-party computation with asynchronous fallback," in *Theory of Cryptography (TCC)*, Nov. 2021, pp. 623–653.

[18] J. Chen and S. Micali, "Algorand," 2016, arXiv preprint arXiv:1607.01341.

[19] A. Flamini, R. Longo, and A. Meneghetti, "Multidimensional Byzantine agreement in a synchronous setting," *Applicable Algebra in Engineering, Communication and Computing*, pp. 1–19, 2022.

[20] J. Helary, M. Hurfin, A. Mostefaoui, M. Raynal, and F. Tronel, "Computing global functions in asynchronous distributed systems prone to process crashes," in *IEEE International Conference on Distributed Computing Systems*, Apr. 2000, pp. 584–591.

[21] A. Barnoy, X. Deng, J. Garay, and T. Kameda, "Optimal amortized distributed consensus," *Information and Computation*, vol. 120, no. 1, pp. 93–100, 1995.

[22] J. Garay, J. Katz, C. Koo, and R. Ostrovsky, "Round complexity of authenticated broadcast with a dishonest majority," in *IEEE Symposium on Foundations of Computer Science (FOCS)*, Oct. 2007, pp. 658–668.

[23] A. Alexandru, J. Loss, C. Papamanthou, and G. Tsimos, "Sublinear-round broadcast without trusted setup against dishonest majority," *Cryptology ePrint Archive*, 2022.