Juggernaut: Efficient Crypto-Agnostic Byzantine Agreement

Daniel Collins[1*], Yuval Efron[2], and Jovan Komatovic[3†]

[1] Texas A&M University
[2] Columbia University
[3] École Polytechnique Fédérale de Lausanne (EPFL)
danielpatcollins@gmail.com, ye2210@columbia.edu, jovan.komatovic@epfl.ch

Abstract. It is well known that a trusted setup allows one to solve the Byzantine agreement problem in the presence of $t < n/2$ corruptions, bypassing the setup-free $t < n/3$ barrier. Alas, the overwhelming majority of protocols in the literature have the caveat that their security crucially hinges on the security of the cryptography and setup, to the point where if the cryptography is broken, even a single corrupted party can violate the security of the protocol. Thus these protocols provide higher corruption resilience ($n/2$ instead of $n/3$) for the price of increased assumptions. Is this trade-off necessary?

We further the study of *crypto-agnostic* Byzantine agreement among n parties that answers this question in the negative. Specifically, let t_s and t_i denote two parameters such that (1) $2t_i + t_s < n$, and (2) $t_i \leq t_s < n/2$. Crypto-agnostic Byzantine agreement ensures agreement among honest parties if (1) the adversary is computationally bounded and corrupts up to t_s parties, or (2) the adversary is computationally unbounded and corrupts up to t_i parties, and is moreover given all secrets of all parties established during the setup. We propose a compiler that transforms any pair of resilience-optimal Byzantine agreement protocols in the authenticated and information-theoretic setting into one that is crypto-agnostic. Our compiler has several attractive qualities, including using only $O(\lambda n^2)$ bits over the two underlying Byzantine agreement protocols, and preserving round and communication complexity in the authenticated setting. In particular, our results improve the state-of-the-art in bit complexity by at least two factors of n and provide either early stopping (deterministic) or expected constant round complexity (randomized). We therefore provide fallback security for authenticated Byzantine agreement *for free* for $t_i \leq n/4$.

1 Introduction

Byzantine agreement (BA), the problem of reaching agreement between n parties, of which at most t are *corrupt*, and controlled by an *adversary*, is arguably the core problem in fault-tolerant distributed computing, with research spanning more than four decades [PSL80,GK20]. In this paper, we focus on the synchronous case, in which messages are delivered to parties at most Δ time after being sent. Cryptographic tools and assumptions are central in the design of BA protocols, both for improved efficiency in various regimes and as well as to circumvent lower bounds [DS83,GK20,GKO+20,BKLZL20]. Perhaps the most eminent of those cryptographic tools are digital signatures, typically instantiated alongside a public key infrastructure (PKI) assumption, in which it is assumed that on top of knowing a list of identifiers of all parties participating in the protocol, each identifier has a corresponding public-secret key pair (pk, sk) with pk being known to all parties. By leveraging PKI, it is well known that BA can be solved in the presence of $t < \frac{n}{2}$ corrupt parties [DS83,KK06], while setup-free protocols must assume $t < \frac{n}{3}$ (even assuming cryptography like signatures) [PSL80,LSP82].

The reliance on PKI mandates two highly crucial assumptions. First, that any underlying cryptography remains secure.[4] Second, that the secrets established at setup remain secure. The vast majority of literature,

*Daniel worked on this project while at Purdue University and Georgia Institute of Technology, and was supported in part by AnalytiXIN and Sunday Group, Inc.

†Most of this work was completed while Jovan was at a16z crypto.

[4]This does not apply to protocols based on primitives like pseudosignatures [PW96] that are information theoretically-secure but require setup; these protocols generally have a high cost and are not deployed at present.

and all practical work on BA that assumes PKI, suffers from the following shortcoming: the security of the protocol hinges on the security of the employed cryptographic primitives, to the point where even a *single* corrupt party can violate security, if the cryptography used turned out to be broken. This precarious state of affairs is not only a theoretical concern, with perhaps the most notorious example being the transaction malleability attack in Bitcoin which resulted in losses of hundreds of millions of dollars [DW14]. Reliance on computational assumptions is more generally risky as they may, at any time, be publicly (let alone discreetly) broken, either classically or due to the looming threat of quantum computation. In some sense, despite the weaker corruption resilience that information-theoretic, setup-free protocols offer, they have the benefit of having no other potential weak spots in their security.

Can we get the best of both worlds? That is, a BA protocol that has optimal resilience given PKI and secure cryptography (the *authenticated* setting), that still maintains high security against a computationally unbounded adversary that can nullify any setup (the *sabotaged* setting)? Note that the specific setting in which the protocol executes is chosen by the adversary at the beginning of the protocol, and in particular *honest* (i.e., non-corrupt) parties are in general *oblivious* to the actual setting in which the protocol executes. Designing such protocols is precisely the question we address in this paper.

This question was in fact studied two decades ago by Fitzi, Holenstein and Wullschleger [FHW04] in the broader context of secure multi-party computation (MPC), in which they design an MPC (and thus a BA) protocol that has what they call *hybrid* security. In particular, it can tolerate up t_s corrupt parties against a computationally bounded adversary and secure cryptography, and up to t_i corrupt parties under no computational or setup assumptions, for any $t_i \leq t_s < \frac{n}{2}$ such that $t_s + 2t_i < n$.[5] They also prove that this bound is tight, even for BA. In particular, a protocol can support up to $t_i \leq \frac{n}{4}$ faults with *no loss of resilience* given a computationally-bounded adversary, i.e., with optimal $t_s < \frac{n}{2}$ fault tolerance! While an impressive feasibility result for general MPC, when one focuses on BA, existing protocols, namely from [FHW04] and subsequent work [FR09], suffer from several drawbacks hindering their usability.

Communication Complexity. Fitzi et al. [FHW04] propose a hybrid broadcast protocol with $O(\lambda n^4)$ communication complexity (in bits) that they use as a subroutine to solve MPC. Subsequent work [FR09] also builds broadcast with complexity $O(\lambda n^4)$ as written and using state-of-the-art sub-routines $O(\lambda n^3 \log n)$ or $O(\lambda n^3 + n^3 \log^2 n)$ bits [ANS23,CDG$^+$24b,AC24] (ignoring problems with composition due to non-simultaneous termination [CCGZ19]). Building crypto-agnostic BA from parallel broadcast generically [CGG$^+$23] would therefore require at least $O(\lambda n^4)$ bits. This leaves a large gap from the classic $\Omega(n^2)$ lower bound on the communication complexity of BA [DR85].

Round Complexity. It is well known that any deterministic BA protocol resilient against up to at most t Byzantine parties must take at least $t+1$ rounds in the worst case [DS83]. This can be circumvented if the protocol is *early stopping*, thereby using just $O(f)$ rounds when $f < t$ corruptions actually occur [PT84,DRS90], or if it is *randomized*, where expected constant-round protocols are known [KK06,ADD$^+$19]. Existing hybrid protocols as written however are deterministic and are not early stopping, requiring $O(n)$ rounds of communication in all cases, and it is not clear that the protocol of [FR09] can immediately lead to constant-round BA since parallel composition of expected constant-round protocols generically results in expected $O(\log n)$ round complexity [BOEY03], let alone the high communication this would incur even if it did work.

General Compiler. Existing hybrid protocols are either directly concrete [FHW04] or make use of a linear number of instances of underlying building blocks like broadcast and are thus not amenable to efficient implementation [FR09]. Ideally, one would want to be able to construct an efficient protocol Π with hybrid security in a *black box* manner from given protocols $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$ for the authenticated and sabotaged settings, respectively, without having to solve BA from scratch.

[5] The model of [FHW04] (and [FR09]) as stated here does not consider passive compromise of the setup as we do, and additionally considers inconsistent PKI which imposes different resilience bounds; we discuss this further later.

Our Contributions. Our main contribution is a compiler that enjoys all of the above properties. Our compiler transforms any two given protocols $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$ in the authenticated and sabotaged settings, respectively, into a protocol Juggernaut with crypto-agnostic security with optimal resilience $t_s + 2t_i < n$, $t_i \leq t_s < \frac{n}{2}$. Furthermore, Juggernaut uses $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$ in a black-box manner, Juggernaut has an additive factor of just $O(\lambda n^2)$ bits of communication over $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$. Our protocol optimizes for the practical authenticated case: if $\mathsf{BA}_{\mathsf{Auth}}$ is early stopping, then so is Juggernaut in the authenticated setting. Moreover, if $\mathsf{BA}_{\mathsf{Auth}}$ is a randomized protocol with expected round complexity R, then Juggernaut has expected round complexity $O(R)$ in the authenticated setting. Therefore, our protocol effectively provides crypto-agnostic security to an authenticated protocol *for free*.

Along the way, we propose two new graded consensus gadgets with $O(\lambda n^2)$ bit complexity and constant (worst-case) round complexity that provide partial security guarantees in one world (authenticated resp. sabotaged) and full security in the other (sabotaged resp. authenticated) that may be of independent interest.

Using our compiler, we propose two concrete protocols, one deterministic and one randomized. Our deterministic protocol has $O(\lambda n^2)$ bit complexity in all cases, has $O(f)$ round complexity for f actual failures in the authenticated case and uses $O(n)$ rounds in the sabotaged case. Our randomized protocol has $O(\lambda n^2)$ expected bit complexity and constant expected round complexity in the authenticated case, and uses $O(\lambda^2 n^2)$ bits and $O(\lambda + f)$ rounds in the sabotaged case.

1.1 Technical Overview

We first would like to stress the complexity of the problem by examining state-of-the-art authenticated BA protocols achieving optimal corruption resilience. Intuitively, without setup and given $t < \frac{n}{3}$, a *quorum* consisting of at least $\frac{2}{3}$ of parties suffices to convince an honest party to adopt a value, as a counting argument shows that no quorum for a different value can exist. This is no longer the case when one demands $t < \frac{n}{2}$, and the overwhelming majority of protocols make use of signature based *equivocation checks* to assert that only one value will be adopted by honest parties during the protocol. Any attempt to increase the size of a quorum can be met with silence from corrupt parties, resulting an unhalting executions due to $t < \frac{n}{2}$. On the other hand, any attempt to relax equivocation checks can be met with agreement violation attacks by corrupt parties. This forces one to rethink the problem from a first principles approach.

A Strawman Solution: Black Boxes and Graded Consensus. A natural approach is to use protocols secure in each setting as black boxes. Let $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$ be protocols solving BA in the authenticated setting and sabotaged (i.e., setup-free and information-theoretic) setting, respectively. Intuitively, we would like to run $\mathsf{BA}_{\mathsf{Auth}}$ first, check if agreement was reached, and if not, run $\mathsf{BA}_{\mathsf{Sab}}$. A typical tool in the literature for detecting pre-existing agreement is the *graded consensus* (GC) primitive, which allows parties to output, along with their value, a grade indicating their level of confidence in the output. The literature luckily contains efficient implementations of GC in both the authenticated setting [MR21a] and the sabotaged setting [AW23]. Alas, we are faced with a trickier scenario. Recall that the specific setting in which the protocol runs is chosen by the adversary at the beginning of protocol execution, and in particular they can choose a setting that renders any existing GC useless and provides no guarantees.

Building Our Juggernaut Protocol. An observation we make, inspired by similar techniques from *network-agnostic* protocols [BKL19,BZL20] that provide security under synchrony *or* asynchrony, is that we can design GCs that work as usual in one of the settings and provide partial guarantees in the other setting in order to build a crypto-agnostic protocol. Designing such GCs with optimal corruption resilience ($t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$), $O(\lambda n^2)$ bit complexity and constant round complexity, as well as appropriately combining everything together (which brings up technical challenges, as explained below), are the main technical contributions of this paper.

Recall that Byzantine agreement provides *consistency* (all parties output the same value), *termination* (all parties output a value and halt), and some *validity* property (in this work, namely that if all parties input the same value, that value is decided). We consider graded consensus with *two* grades, either 0 or 1:

graded validity then requires all honest parties output input v and grade 1, and *graded consistency* requires that if any honest party outputs $(v, 1)$, then all honest parties output $(v, 0)$ or $(v, 1)$.

As sketched above, our high level approach is to run $\mathsf{BA}_{\mathsf{Auth}}$, check if agreement was reached, and run $\mathsf{BA}_{\mathsf{Sab}}$ if not. However, if we are in the sabotaged setting, $\mathsf{BA}_{\mathsf{Auth}}$ can behave arbitrarily, and if we are in the authenticated setting, $\mathsf{BA}_{\mathsf{Sab}}$ can behave arbitrarily. We introduce two graded consensus protocols to deal with this. Our first, $\mathsf{GC}^*_{\mathsf{Auth}}$, provides *full* security in the authenticated case for up to t_s corruptions, and ensures *validity and termination* for up to t_i corruptions in the sabotaged case. Our second, $\mathsf{GC}^*_{\mathsf{Sab}}$, provides the *opposite* guarantees: full security in the sabotaged case with t_i corruptions, and validity and termination in the authenticated case with t_s corruptions.

At a high level, our protocol, which we call $\mathsf{Juggernaut}$, proceeds as follows. First, each party p_i runs $\mathsf{GC}^*_{\mathsf{Auth}}$ using their input v_i, which outputs pair (v_1, g_1). v_1 is then fed to $\mathsf{BA}_{\mathsf{Auth}}$, which outputs v_2. In the authenticated case, $\mathsf{GC}^*_{\mathsf{Auth}}$ and $\mathsf{BA}_{\mathsf{Auth}}$ provide full security, and therefore all honest parties output the same v_2 from $\mathsf{BA}_{\mathsf{Auth}}$. In the sabotaged case, however, we only have *validity* of $\mathsf{GC}^*_{\mathsf{Auth}}$. To preserve validity in the sabotaged case, parties then input v_1 to $\mathsf{GC}^*_{\mathsf{Sab}}$ if $g_1 = 1$, and otherwise input v_2 to $\mathsf{GC}^*_{\mathsf{Sab}}$; let the output of $\mathsf{GC}^*_{\mathsf{Sab}}$ be (v_3, g_3). Since $\mathsf{GC}^*_{\mathsf{Sab}}$ provides validity in the authenticated case, all honest parties will output the same $(v_2, 1)$, where $v_1 = v_2$ in 'valid' runs of $\mathsf{Juggernaut}$ (so validity is preserved from $\mathsf{GC}^*_{\mathsf{Auth}}$ up to this point).

At this point we do not yet have consistency in the sabotaged case, only validity. Therefore, all parties run $\mathsf{BA}_{\mathsf{Sab}}$ with input v_3, which provides full security in the sabotaged case. However, $\mathsf{BA}_{\mathsf{Sab}}$ provides *no security* in the authenticated case. To rectify this, as well as to provide early stopping in the authenticated case, parties multicast their output value v_3 as (decide, v_3) *only if* $g_3 = 1$ and wait for Δ time. Then, on receipt of $n - t_s$ (decide, v_3) messages (which is always guaranteed in the authenticated case), parties output v_3 and can safely halt. In the sabotaged case, however, some but not all parties may terminate at this stage. Thus, to ensure all other parties terminate, parties terminate if they receive (decide, v) from $n - t_s - t_i$ parties *after* running $\mathsf{BA}_{\mathsf{Sab}}$ (if they have not yet halted). Therefore, if an honest party halts due to receiving $n - t_s$ (decide, v_3) messages, all honest parties will receive $n - t_s - t_i$ (decide, v) messages and halt. Otherwise, all honest parties will not halt before terminating from $\mathsf{BA}_{\mathsf{Sab}}$. In this case, if an honest party receives $n - t_s - t_i$ (decide, v) messages, then since $n - t_s - t_i > t_i$, one honest party must have output $(v, 1)$ from $\mathsf{GC}^*_{\mathsf{Sab}}$, and by graded consistency, all honest parties output v, and by consistency in the sabotaged case of $\mathsf{BA}_{\mathsf{Sab}}$, all honest parties will output v from that, and thus agree on v. Otherwise, consistency of $\mathsf{BA}_{\mathsf{Sab}}$ ensures the consistency of $\mathsf{BA}_{\mathsf{Sab}}$.

Dealing with Non-Simultaneous Termination. The above works well if $\mathsf{BA}_{\mathsf{Auth}}$ is such that all parties terminate at the same time. However, if $\mathsf{BA}_{\mathsf{Auth}}$ is early-stopping or randomized, parties will not in general output from $\mathsf{BA}_{\mathsf{Auth}}$ at the same time. For instance, the adversary can force one honest party to produce an output significantly earlier than any other honest party. To rectify this, we utilize the *synchronizer* primitive, which ensures that honest parties "move on" from $\mathsf{BA}_{\mathsf{Auth}}$ at roughly the same time. Concretely, the synchronizer guarantees that honest parties quit executing $\mathsf{BA}_{\mathsf{Auth}}$ within at most one round of each other, regardless of whether we are in the authenticated or sabotaged setting. Furthermore, in the authenticated setting, the synchronizer ensures that honest parties progress from $\mathsf{BA}_{\mathsf{Auth}}$ with (asymptotically) no additional round overhead: (1) if $\mathsf{BA}_{\mathsf{Auth}}$ is early stopping with complexity $O(f)$, then honest parties progress in $O(f)$ rounds, and (2) if $\mathsf{BA}_{\mathsf{Auth}}$ is randomized with expected round complexity R, then honest parties progress in $O(R)$ rounds. This is essential to guarantee that, in the authenticated case, $\mathsf{Juggernaut}$ introduces no asymptotic round complexity overhead.

Constructing Crypto-Agnostic Graded Consensus. We provide efficient compilers that use a single instance of graded consensus protocol secure in a given setting that provide validity and termination in the other setting. Namely, our two compilers each incur *three* additional rounds and $O(\lambda n^2)$ communication overhead over the black boxes we use, for example the graded consensus protocols of Momose-Ren [MR21a] and Attiya-Welch [AW23] which themselves have constant round complexity and quadratic communication complexity.

Our first protocol $\mathsf{GC}^*_{\mathsf{Auth}}$ provides full authenticated security and validity and termination in the sabotaged case. Recall that the underlying authenticated graded consensus protocol, say Π_{MR}, in general provides

no security in the sabotaged case. Our goal is thus to augment Π_{MR} with a procedure to ensure sabotaged t_i-validity and termination. The main challenge is ensuring that this procedure does not interfere with the authenticated security of Π_{MR}.

The key observation lies in the fact that if the parties aren't in the validity case, then parties can change their inputs to Π_{MR} arbitrarily without violating security. Therefore, parties in our protocol cast votes in search of a sufficiently large quorum (of $n - t_i$ parties) to output a value with grade 1. If such a quorum is found, a certificate of the quorum is made and broadcast to the rest of the parties. Upon receiving a unique certificate of this kind, a party replaces its input with the received value, and enters Π_{MR} with the new input. Validity in the sabotaged case then becomes immediate, and careful analysis is required to ensure that this added part of the protocol can not violate the t_s-security of the protocol in the authenticated setting.

Our second protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ provides the opposite: sabotaged security and validity and termination in the authenticated case. The main challenge stems from the fact that honest parties might not initiate the protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ in the same round, due to possibly different exit times from $\mathsf{BA}_{\mathsf{Auth}}$. Thanks to the Synchronizer (see Section 4), which leverages the synchrony assumption of the network, we know that honest parties commence $\mathsf{GC}^*_{\mathsf{Sab}}$ at most 1 round apart from one another. This allows us to design $\mathsf{GC}^*_{\mathsf{Sab}}$ with that in mind, and not deal with the general case of asynchrony.

Similarly to $\mathsf{GC}^*_{\mathsf{Auth}}$, our augmenting of Π_{AW} with sabotaged t_s-validity relies on the observation that when not in the validity case, parties may change their inputs arbitrarily without harming the security of the protocol. A first attempt at augmenting Π_{AW} with authenticated t_s-validity might look as follows: Echo the inputs, and look for a quorum of $n - t_s$ echos for a value v, broadcast a certificate $\mathcal{C}(v)$ of this quorum (if found) using threshold signatures, and if no conflicting certificates were received, cast a vote for v, deciding it with grade 1 if a quorum of $n - t_s$ of votes was received. Validity in the authenticated case clearly holds, but alas, in the sabotaged setting, for reasons that become clear in the analysis, this approach fails.

The key observation here is that in the authenticated setting, we only care about validity, and so we can impose stricter conditions on deciding a value prior to running Π_{AW}. Specifically, our solution stipulates that witnessing a unique certificate for a value is no longer sufficient for party to decide a value, it must have also *created* a certificate itself. This saves us from consistency violations in the *sabotaged* setting by making sure that if an honest party decides a value before Π_{AW}, then all honest parties have seen a certificate for that value.

1.2 Related Work

Hybrid Security. Two previous works that we are aware of consider fallback security w.r.t. an unbounded adversary for an authenticated protocol [FHW04,FR09] (both cited above), both focused on the feasibility of MPC. These works additionally allow the adversary to completely compromise the PKI, given the adversary corrupts up to t_p parties - they call the resulting model *hybrid security*. They show to provide security for $t_p > 0$ that $2t_s + t_p < n$ is necessary and sufficient.[6] Thus, to guarantee any security in this case, one must sacrifice resilience in t_s. Our model further differs from previous work in that in the sabotaged case, we additionally allow the adversary to passively compromise the setup even under $t_s + 2t_i < n$, whereas the adversary cannot not do so in [FHW04,FR09] unless t_p or less corruptions are made. This is of particular note for information-theoretic authenticated BA with fallback since one can set $t_s = \frac{n}{2} - 1$ and still achieve fallback security for $t_i \leq \frac{n}{4}$ under passively compromised setup.

As noted above, [FHW04] build hybrid broadcast with $O(\lambda n^4)$ bit and $O(n)$ round complexity. They first build 'weak broadcast' which provides security in their hybrid setting, then generically build graded consensus with $O(n)$ instances of weak broadcast, followed by $O(n)$ instances of graded consensus (one per round) for the final broadcast protocol. We do not see how to easily reduce this complexity without starting from scratch, which we indeed do in this work.

[FR09] also build hybrid broadcast using $O(\lambda n^4)$ bits and $O(n)$ rounds as written; we now consider its most expensive components. First, they run Dolev-Strong broadcast w.r.t. the sender (or as we note, any

[6] In [FR09], the authors show for $t_p = 0$ that $t_s + 2t_i < n$ is enough, and so $t_s \leq n/2$ and $t_i > 0$ (in their model) is possible for functionalities like broadcast and MPC with (in their case unanimous) abort.

$t < n$ broadcast protocol). Then, they run so-called broadcast with extended validity, which they build from n instances of perfectly-secure broadcast [BGP+89] with a message-signature pair as input. Finally, they run parallel broadcast where each honest party may input $O(n)$ signatures on a message. Perfectly secure parallel broadcast with $O(\lambda)$-sized input can be built using $O(\lambda n^3 \log n)$ bits and $O(f)$ rounds for f actual corruptions [CDG+24b], and otherwise $O(\lambda n^2 + n^3 \log^2 n)$ bits and constant rounds [AC24]. Authenticated parallel broadcast under honest majority with $O(\lambda+n)$-sized inputs (using multisignatures) can be built using $O(n^3 + \lambda n^2)$ bits and $O(n)$ rounds [CDG+24a], and otherwise $O(\lambda n^3)$ bits and constant rounds [ANS23]. Ultimately, we cannot escape using at least $O(\lambda n^3)$ bits using the approach of [FR09] to build constant-round broadcast, let alone BA with constant round complexity.

Fallback Security. Many works consider providing additional security guarantees to primitives like BA or MPC on top of or in exchange for some security in the 'base' setting (in our work the authenticated setting) going back at least to Chaum [Cha90] in MPC; we survey some below. [GKKY10] considers a model where the secrets of t_c parties can be exposed to the adversary *and* t_a additional parties can be corrupted, showing in particular $2t_a + \min\{t_a, t_c\} < n$ for (fixed) $t_a, t_c > 0$ is sufficient and necessary; observe that their model is incomparable to ours. An *accountable* BA protocol [CGG+22] provides security given t corruptions, and given $t' > t$ corruptions, parties can generate a proof that some parties must have behaved maliciously. This resembles MPC with identifiable abort [IOZ14], namely MPC under corrupt majority that ensures a corrupt party is unanimously detected if the protocol aborts, but is not confined to the synchronous setting. [LZLM+20] considers synchronous MPC under t_s corruptions with *responsiveness* (like asynchronous protocols) under t_r corruptions, and achieve a comparable bound as us, namely $t_s + 2t_r < n$. [LRM10] considers trade-offs between information-theoretic robustness (preventing adversarial abortion) and computational privacy assuming broadcast and secure channels.

A line of work initiated in [BKL19] considers *network-agnostic* security, that is, providing security for up to t_a corruptions if the network is asynchronous and t_s if it is synchronous. Some works consider feasibility results, including [BKL19] for BA, which is possible if and only if $2t_s + t_a < n$, among others [BZL20,BKL21,DLZ23], as well as performance [DHLZ21,BCLZL23]. The recent work of [DE24] has a similar motivation to ours in that the authors show that network-agnostic BA can be built 'for free' in the *synchronous* case, namely with $O(\lambda n^2)$ bit and constant round overhead.

Byzantine agreement. There is a rich history of work on the Byzantine agreement problem in each our considered settings (when considered separately). In the authenticated setting, the state-of-the-art protocol for BA in terms of communication complexity and latency is [CDG+24a] in the deterministic case, in which they showcase a protocol with resilience $t < \frac{n}{2}$ with optimal $O(f)$ round complexity when $f \leq t$ corruptions actually occur, and $O(\lambda n^2)$ bit complexity. In the randomized case, [ADD+19] presents a protocol with $O(1)$ expected latency, and $O(\lambda n^2)$ expected bit complexity, with resilience of $t < \frac{n}{2}$. For the sabotaged setting, the protocol of [MMR15] presents a protocol with optimal $t < \frac{n}{3}$ resilience, $O(1)$ expected latency, and $O(n^2)$ expected bit complexity. Alas their protocol assumes the existence of a common coin. [BGP92] and [CW92] were the first to solve BA with $O(n^2)$ bits and linear rounds; later [LS22] built such a protocol that is additionally early stopping. In a breakthrough result, Chen [Che21] solved BA with strong unanimity ($\perp$ can be decided when not all honest parties propose the same value) with $O(nL + n^2 \log n)$ for messages of length L. [CDG+24b] achieve external validity [CKPS01] (decided values satisfy a given predicate) with $O((nL + n^2) \log n)$ bit complexity; note `Juggernaut` can be modified to support external validity by adding appropriate predicate checks.

2 Preliminaries and Definitions

Throughout the paper, we consider a fully connected network of n parties $p_1, \ldots, p_n$ that communicate over point-to-point authenticated channels. Some fraction of these parties are controlled by an adversary and may deviate arbitrarily from the protocol. We call these parties *corrupt* and the other parties *honest*. When we say that a party *multicasts* a message, we mean that it sends it to all n parties in the network. We denote the security parameter by λ. Throughout the paper, we assume a universe of values V.

Public Key Infrastructure. We assume that the parties have established a public key infrastructure before the protocol execution, which is a bulletin board or plain PKI. Namely, each party p_i has a secret-public key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ for the use of cryptography. In this paper, we assume that those keys are used to instantiate a secure digital signature scheme and all messages in our protocols (but not necessarily building blocks) *are implicitly signed.*

Threshold Signatures. On top of a PKI, a trusted setup allows the parties to map any vector of f valid signatures of the same message m by different parties (henceforth referred to as an f-certificate of m) into a single message π of length $O(\lambda)$, called a *threshold signature*, denoted $\mathcal{C}(m)$, with the property that the signature certification algorithm passes on π iff π is the image of a valid t-certificate on m.

Communication Model. We assume a synchronous network, where all parties begin the protocol at the same time, the clocks of the parties progress at the same rate, and all messages are delivered within some known finite time $\Delta > 0$ (called the network delay) after being sent. In particular, messages of honest parties cannot be dropped from the network and are always delivered. Thus, we can consider protocols that execute in rounds of length Δ where parties start executing round r at time $(r-1)\Delta$. We further assume that Δ is public information and is known to all parties and the adversary, and any action carried out by any party can depend on Δ. With that in mind, and to avoid notation encumbrance, we omit Δ from the list of inputs to algorithms and protocols in our definitions.

Adversarial Model. The adversary model we consider in the paper is an amalgamation of two common adversaries in the literature. Formally, given two parameters $t_i \leq t_s < n/2$ such that $2t_i + t_s < n$, the adversary $\mathcal{A}$ can be described as a tuple $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ such that

- $\mathcal{A}_0(\Pi, r, Tr_r) = \mathcal{F}_r$ where $\mathcal{F}_r$ denotes the set of corrupt parties at round r. I.e. $\mathcal{A}_0$ is an algorithm that chooses for every round the set of corrupt parties, based on the description of the protocol Π, the round r, and the transcript Tr_r of the protocol up to round t. We distinguish between two types of adversaries in this context. A *static* adversary satisfies that $\mathcal{A}_0(\Pi, r, Tr_r) = \mathcal{A}_0(\Pi, 0, Tr_0)$ for all rounds r. An *adaptive* adversary satisfies $\mathcal{A}_0(\Pi, r, Tr_r) \subseteq \mathcal{A}_0(\Pi, r+1, Tr_{r+1})$ for all rounds t. Unless otherwise stated, we assume an adaptive adversary For a given adversary $\mathcal{A}$, we say that a party p is *forever honest* if $p \notin \mathcal{F}_r$ for all rounds r.
- $\mathcal{A}_1(\Pi, r, Tr_r, \mathcal{F}_r)$ describes the algorithm run by corrupt parties throughout the execution of the protocol: it may depend on the description of the protocol Π, the round r, the transcript Tr_r of the protocol up to round r, and the internal state of all corrupt parties at round r. In this context, we distinguish between two settings, characterized by the capabilities of the adversary.
  - **Sabotaged.** $\mathcal{A}_1$ (and $\mathcal{A}_0$) are computationally unbounded, and in particular can break the security of any cryptographic primitive used in the protocol via the PKI. Furthermore, the adversary has complete access to all the information, secret and public of any setup protocol carried out by the parties prior to receiving their inputs. Equivalently, in the ideal setup world, the adversary receives from the trusted dealer of the setup all communication sent to any party.
  - **Authenticated.** $\mathcal{A}_1$ (and $\mathcal{A}_0$) are computationally bounded, and there is a trusted PKI setup. In this case, we assume in our security proofs that the cryptographic primitives used in the protocol provide perfect security, which, by a standard hybrid argument, does not affect the generality of our result and serves to simplify the exposition.
- $\mathcal{A}_2(\Pi) \to \{0, 1\}$. The adversary, at round $r = 0$ can view the protocol description Π and choose a bit b that indicates the *setting* of the current execution of the protocol. This choice is not revealed to honest parties. The following holds.
  - If $b = 1$, then $\mathcal{A}$ chose the sabotaged setting. Furthermore, $|\mathcal{F}_r| \leq t_i$ for all rounds r.
  - If $b = 0$, then $\mathcal{A}$ chose the authenticated setting. Furthermore $|\mathcal{F}_r| \leq t_s$ for all rounds r.

We say that the adversary $\mathcal{A}$ is t-bounded if $|\mathcal{F}_r| \leq t$ holds for all rounds r
Definitions and properties that we introduce hereafter are only required to hold with probability $1 - \mathsf{negl}(\lambda)$.

2.1 Distributed Primitives

When relevant, our primitives take input from a value set V with $|V| \geq 2$; we assume that default value $\perp \notin V$. Note that $\perp$ is considered as a valid output in each protocol.

Definition 1 (Byzantine Agreement). *Let Π be a protocol executed by parties $p_1, \ldots, p_n$, where each party p_i begins by calling* propose *with input $v_i \in V$. The* BA *problem pertains to the following properties.*

- **Validity:** *Π is sabotaged (authenticated) t-*valid *if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: If every honest party's input is equal to the same value v, then every honest party outputs v.*
- **Consistency:** *Π is sabotaged (authenticated) t-*consistent *if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: Every honest party that outputs a value outputs the same value v.*
- **Termination:** *Π is sabotaged (authenticated) t-*terminating *if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: Every honest party produces an output and terminates.*

*If Π is sabotaged (authenticated) t-valid, t-consistent, and t-terminating, we say it is sabotaged (authenticated) t-*secure.

Definition 2 (Graded Consensus). *In the graded consensus (*GC*) problem, each honest party invokes* propose *with input $v_i \in V$ and outputs a tuple $(y_i, g_i) \in V \times \{0, 1\}$. Let Π be a protocol executed by parties $p_1, ..., p_n$. The relevant properties attributable to Π are as follows.*

- ***Validity:*** *We say that Π is sabotaged (*authenticated*) t-valid if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: If every honest party's input is equal to the same value v, then every honest party outputs $(v, 1)$.*
- ***Consistency:*** *We say that Π is sabotaged (*authenticated*) t-consistent if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: If an honest party outputs $(v, 1)$ for some value v, then all honest parties output either $(v, 1)$ or $(v, 0)$.*
- ***Termination:*** *We say that Π is sabotaged (*authenticated*) t-terminating if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: There exists a round r such that all honest parties produce an output and terminate by round r.*

*If a protocol Π for GC is sabotaged (*authenticated*) t-valid, t-consistent, and t-terminating, we say that Π is sabotaged (*authenticated*) t-secure.*

Definition 3 (Synchronizer). *In the Synchronizer problem, we expose the following interface, that any party can engage with in a round of their choice.*

- start_synchronization*($v \in V$): a party starts synchronization with a value $v \in V$.*
- *output* synchronization_completed*($v' \in V$): a party completes synchronization with a value $v' \in V$.*

We make the assumption that each honest party starts synchronization at most once. Importantly, we do not *assume that all honest parties start synchronization, i.e., it could be the case that no honest party starts synchronization.*

We consider the following properties w.r.t. to a protocol Π in the context of the Synchronization primitive.

- ***Justification:*** *We say that Π has sabotaged (authenticated) t-justification if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: If the an honest party p completes synchronization with a value v' at round r, then there exists an honest party q that started synchronization with value v' at a round $r' < r$.*
- ***Totality:*** *We say that Π has sabotaged (authenticated) t-totality if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: Let ρ be the first round in which an honest party completes synchronization for some value v. Then, every honest party p_i completes synchronization at some round $\rho_i \leq \rho + 1$.*

– ***Liveness:*** *We say that Π has sabotaged (authenticated) t-liveness if the following holds in the sabotaged (authenticated) setting when at most t parties are corrupted: Suppose there exists a value $v \in V$ and a round ρ such that all honest parties start synchronization with value v by round ρ. Then, every honest party p_i completes synchronization with value v at some round $\rho_i \leq \rho + 1$.*

Generic Compiler. As mentioned in the introduction, we make use of black-box access to given protocols solving BA. One in the PKI setting with authenticated t_s-security, and one in the information theoretic setting with t_i-security. We further make the assumption that along with Π, we are also given a parameter T_Π indication the amount of rounds one must run the protocol to ensure that all honest nodes have produced an output except for with negligible probability We here formalize the notion of black-box access to a protocol in the context of our work.

Definition 4. *For a given tuple (Π, T_Π), where Π is a sabotaged (authenticated) t-secure* BA *protocol and T_Π is a parameter, black-box access implies the following guarantees for any adversary $\mathcal{A}$.*

– *If $\mathcal{A}$ chose the sabotaged (authenticated) setting and furthermore there exists a round r' s.t. all honest parties initiate $\Pi.\mathsf{propose}(\mathsf{v} \in \mathsf{V})$ at round r', then except for with $\mathsf{negl}(\lambda)$ probability, by round $r' + T_\Pi$, all honest parties have produced an output from Π, furthermore, these outputs satisfy the* BA *conditions so long as $|\mathcal{F}_r| \leq t$ for all rounds r.*

For a given protocol Π and an adversary $\mathcal{A}$, we denote its expected communication (in bits) complexity under $\mathcal{A}$ by $\mathsf{CC}_\mathcal{A}(\Pi)$. Note that this is well defined since in all protocols discussed in this work, there is a predetermined upper bound on the number of rounds for which each party is active before halting. We assume that values in V are of size $O(\lambda)$ when CC is calculated; this is without loss of generality (see Section 6).

3 Juggernaut

This section presents Juggernaut, our main protocol. We start by introducing Juggernaut's building blocks (Section 3.1). Then, we show how these building blocks are composed into Juggernaut (Section 3.2). Finally, we prove Juggernaut's security and complexity, captured in the following theorem.

Theorem 1. *Let t_s, t_i such $2t_i + t_s < n, t_i \leq t_s < \frac{n}{2}$. Assuming black-box (Definition 4) access to $(\mathsf{BA}_{\mathsf{Auth}}, T_{\mathsf{Auth}})$ and $(\mathsf{BA}_{\mathsf{Sab}}, T_{\mathsf{Sab}})$, where $\mathsf{BA}_{\mathsf{Auth}}$ is an authenticated t_s-secure protocol for* BA*, and $\mathsf{BA}_{\mathsf{Sab}}$ is a sabotaged t_i-secure protocol. Then, there exists a protocol* Juggernaut *which is authenticated t_s-secure* and *sabotaged t_i-secure. Furthermore, for any adversary $\mathcal{A}$ the following holds:*

1. *If $\mathsf{BA}_{\mathsf{Auth}}$ and $\mathsf{BA}_{\mathsf{Sab}}$ are secure against an adaptive adversary, then so is* Juggernaut*.*
2. *If $\mathcal{A}$ chose the sabotaged setting, and the adversary is t_i-bounded, the expected communication (in bits) complexity of* Juggernaut *is $O(\mathsf{CC}_\mathcal{A}(\mathsf{BA}_{\mathsf{Auth}}) + \mathsf{CC}_\mathcal{A}(\mathsf{BA}_{\mathsf{Sab}}) + \lambda n^2)$.*
3. *If $\mathcal{A}$ chose the authenticated setting, and the adversary is t_s-bounded, then the expected communication (in bits) complexity of* Juggernaut *is $O(\mathsf{CC}_\mathcal{A}(\mathsf{BA}_{\mathsf{Auth}}) + \lambda n^2)$.*
4. *If $\mathcal{A}$ chose the authenticated setting, and the adversary is t_s-bounded, then if r is the the first round such that all parties honest at round r have produced an output from $\mathsf{BA}_{\mathsf{Auth}}$ and terminated $\mathsf{BA}_{\mathsf{Auth}}$, then all forever honest parties produce and output from* Juggernaut *and terminate* Juggernaut *after $r + O(1)$ rounds.*

3.1 Building Blocks: Overview

In this subsection, we summarise the building blocks we use to build Juggernaut on top of Byzantine agreement.

Authenticated Graded Consensus with Sabotaged Validity. The authenticated graded consensus with fallback validity primitive exposes the following interface:

- Input $\mathsf{propose}(v \in V)$: A party proposes a value $v \in V$.
- Output: $(v' \in V, g' \in \{0,1\})$: A party outputs a value v' with a binary grade g'. Usually indicated by a left arrow $\leftarrow$.

We assume that all honest parties propose exactly once and they all do that simultaneously (i.e., in the same round). In this setting, we design a protocol $\mathsf{GC}^*_{\mathsf{Auth}}$, that satisfies the following properties w.r.t. the graded consensus primitive (See Definition 2) for any $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$: Authenticated t_s-secure, sabotaged t_i-valid, and sabotaged t_i-terminating.

Complexity. Juggernaut utilizes an implementation of the primitive that exchanges $O(\lambda n^2)$ bits and terminates in $T_1 = O(1)$ rounds. We relegate our implementation of the primitive to Section 5.1.

Sabotaged Graded Consensus with Authenticated Validity. The sabotaged graded consensus with authenticated validity primitive exposes the following interface:

- Input $\mathsf{propose}(v \in V)$: A party proposes a value $v \in V$.
- Output $(v' \in V, g' \in \{0,1\})$: A party outputs a value v' with a binary grade g'. Usually indicated with a left arrow $\leftarrow$.

All honest parties propose exactly once and they do so within one round of each other. Therefore, we do *not* assume that all honest parties propose in the same round. For this setting, we design a protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ with the following properties w.r.t. to the GC primitive (See Definition 2) for any $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$: Sabotaged t_i-secure and authenticated t_s-valid and t_s-terminating.

Complexity. In Juggernaut, we employ an implementation of the primitive that exchanges $O(\lambda n^2)$ bits and terminates in $T_2 = O(1)$ rounds. The implementation can be found in Section 5.2.

Synchronizer. The primitive exposes the following interface:

- input $\mathsf{start_synchronization}(v \in V)$: A party starts synchronization with a value $v \in V$.
- output $\mathsf{synchronization_completed}(v' \in V)$: A party completes synchronization with a value $v' \in V$.

We design a protocol Sync, that has the following properties in the context of the Synchronizer primitive (See Definition 3) for any $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$.

- Sabotaged t_i-totality.
- Authenticated t_s-justification, t_s-totality, t_s-liveness.

Complexity. We implement the synchronizer primitive with $O(\lambda n^2)$ exchanged bits. See Section 4 for more details.

3.2 Juggernaut's Implementation & Proof

Juggernaut's implementation is provided in Figure 1, whereas its visual dedication can be found in Figure 2. For clarity, we proceed with a written exposition on the stages of the protocol. On a high level, as seen in Figure 2, the protocol divided into five steps.

First, all parties propose their input into $\mathsf{GC}^*_{\mathsf{Auth}}$ – recall that this is authenticated t_s-secure and sabotaged t_i-valid. The output of $\mathsf{GC}^*_{\mathsf{Auth}}$ is then fed as input into $\mathsf{BA}_{\mathsf{Auth}}$. Note that here is where the synchronizer Sync comes into play, since in the the sabotaged case, or an early stopping/randomized protocol $\mathsf{BA}_{\mathsf{Auth}}$ in the authenticated setting, the adversary can cause significant gaps between the rounds in which honest parties

produce an output and move on from $\mathsf{BA}_{\mathsf{Auth}}$. Sync maintains that honest parties exit $\mathsf{BA}_{\mathsf{Auth}}$ at most 1 round apart from one another. The input to $\mathsf{GC}^*_{\mathsf{Sab}}$ is then decided depending on whether $g_1 = 1$, as depicted in the figure.

To maintain early stopping in the authenticated case, each party that has $g_2 = 1$ participates in an early stopping phase in order to detect pre-existing agreement in $\mathsf{GC}^*_{\mathsf{Sab}}$. During that phase, parties with $g_2 = 1$ multicast $\mathsf{decide}(v_4)$ messages for their output from $\mathsf{GC}^*_{\mathsf{Sab}}$. If a quorum of $n - t_s$ $\mathsf{decide}(v)$ messages is received for some value, then a party decides v and halts. Otherwise, after sufficient time has passed, all honest parties that haven't halted start executing the sabotaged BA protocol $\mathsf{BA}_{\mathsf{Sab}}$ in the same round. At its conclusion, parties produce an output based on conditions C2 and C3, as seen in Figure 1.

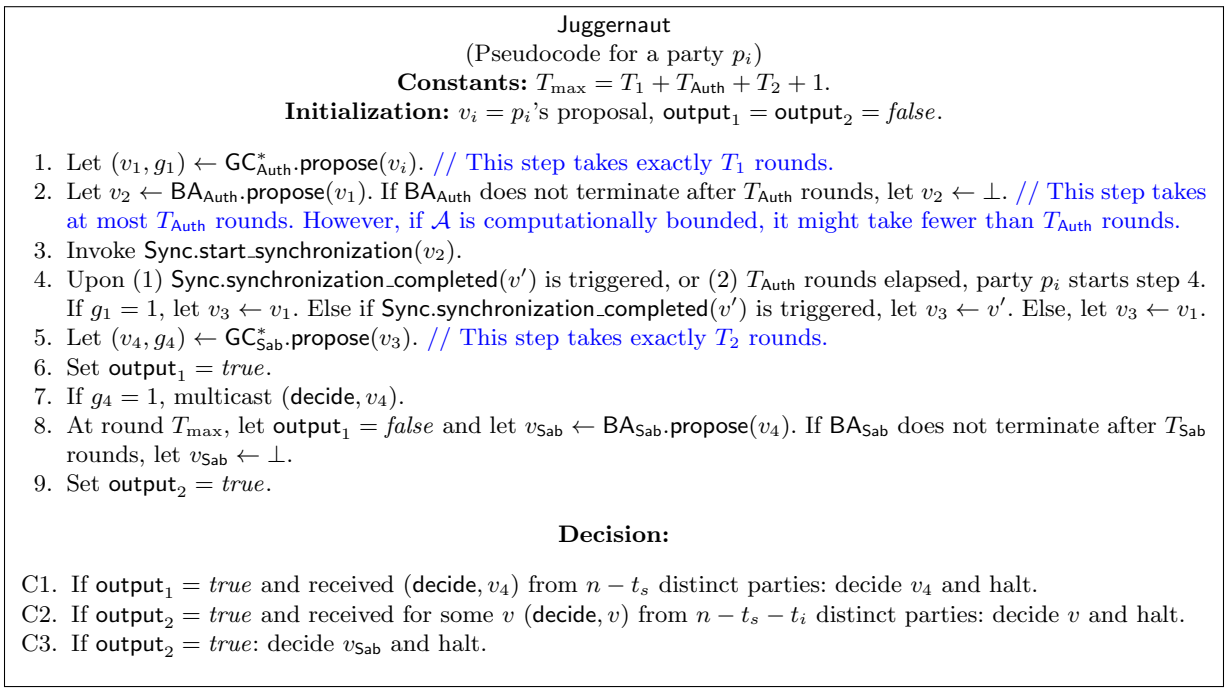

Fig. 1: BA with crypto-agnostic security for $2t_i + t_s < n$ given 1) an authenticated t_s-secure BA protocol $\mathsf{BA}_{\mathsf{Auth}}$; 2) a sabotaged t_i-secure BA protocol $\mathsf{BA}_{\mathsf{Sab}}$; 3) an authenticated t_s-secure and sabotaged t_i-valid graded consensus protocol $\mathsf{GC}^*_{\mathsf{Auth}}$; and 4) a sabotaged t_i-secure and authenticated t_s-valid graded consensus protocol $\mathsf{GC}^*_{\mathsf{Sab}}$.

Proof (of Theorem 1). First, we present a simple proposition showing that all honest parties propose to $\mathsf{GC}^*_{\mathsf{Sab}}$ and they do so within one round of each other. Recall that $\mathsf{GC}^*_{\mathsf{Sab}}$ operates under this assumption.

Proposition 1. *All honest parties propose to* $\mathsf{GC}^*_{\mathsf{Sab}}$ *and they do so within one round of each other.*

Proof. In the sabotaged (resp., authenticated) setting, the proposition follows from (1) the sabotaged t_i-termination (resp., authenticated t_s-termination) property of $\mathsf{GC}^*_{\mathsf{Auth}}$, (2) the sabotaged t_i-totality (resp., authenticated t_s-totality) property of Sync, and (3) the fact that all honest parties propose to $\mathsf{GC}^*_{\mathsf{Sab}}$ in the $(T_1 + T_{\mathsf{Auth}})$-th synchronous round at the latest.

Proposition 1 establishes that $\mathsf{GC}^*_{\mathsf{Sab}}$ is utilized in an appropriate way in $\mathsf{Juggernaut}$, which implies that $\mathsf{GC}^*_{\mathsf{Sab}}$ indeed satisfies its specification. To keep the proof uncluttered, we will not explicitly refer to Proposition 1 in its remainder.

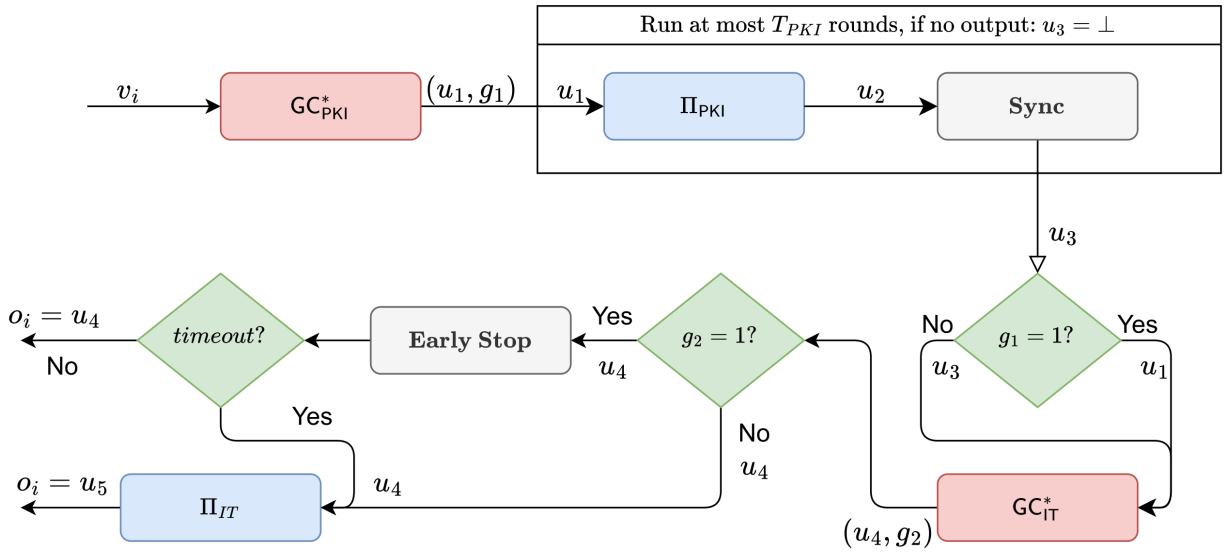

Fig. 2: Overview of the structure of the main protocol for each party. Beginning with input insertion at the top left with v_i, and ending with output production at the bottom left. The values next to arrows indicate the outputs and inputs produced and delivered into procedures.

Juggernaut's security in the sabotaged setting. We start by proving that Juggernaut is t_i-valid in the sabotaged setting.

Lemma 1 (t_i**-validity).** *Juggernaut (Figure 1) is* t_i-valid *in the sabotaged setting.*

Proof. Suppose all honest parties propose the same value v. Due to the validity property of $\mathsf{GC}^*_{\mathsf{Auth}}$ in the sabotaged setting, all honest parties decide $(v, 1)$ from $\mathsf{GC}^*_{\mathsf{Auth}}$. Hence, all honest parties propose v to $\mathsf{GC}^*_{\mathsf{Sab}}$. The fact that $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies validity in the sabotaged setting proves that all honest parties decide $(v, 1)$ from $\mathsf{GC}^*_{\mathsf{Sab}}$. Thus, all honest parties multicast a (decide, v) message. Hence, all honest parties receive $n - t_i$ such messages. (Note that no honest party receives $n - t_s$ decide message for any value $v' \neq v$ since that would imply that an honest party sends a $\mathsf{decide}(v')$ message, which cannot occur.) As $n - t_i \geq n - t_s$ (since $t_i \leq t_s$), all honest parties decide v according to decision condition C1. □

Next, we prove that Juggernaut is t_i-consistent in the sabotaged setting.

Lemma 2 (t_i**-consistency).** *Juggernaut (Figure 1) is* t_i-consistent *in the sabotaged setting.*

Proof. Suppose that at least one honest party outputs due to C1, i.e., receives (decide, v_4) from $n - t_s$ distinct parties. Since $n - t_s - t_i > t_i$, at least one honest party must have output $(v_4, 1)$ from $\mathsf{GC}^*_{\mathsf{Sab}}$, and thus by t_i-consistency of $\mathsf{GC}^*_{\mathsf{Sab}}$, all honest parties output the same value v_4, and in particular no honest party multicasts (decide, v'_4) for $v'_4 \neq v_4$. In addition, at least $n - t_s - t_i$ honest parties must have multicast (decide, v_4). Thus all honest parties will receive (decide, v_4) from at least $n - t_s - t_i$ distinct parties, and not for any other value $v'_4 \neq v_4$ since $n - t_s - t_i > t_i$. Therefore all honest parties will output v_4 either due to C1 or C2.

Suppose that no honest party outputs due to C1 and at least one outputs due to C2, i.e., due to receiving (decide, v) for some v from $n - t_s - t_i$ distinct parties. Then at least one honest party must have multicast (decide, v), since $n - t_s - t_i > t_i$. By the t_i-consistency of $\mathsf{GC}^*_{\mathsf{Sab}}$, all honest parties will propose the same v to $\mathsf{BA}_{\mathsf{Sab}}$. By the t_i-validity of $\mathsf{BA}_{\mathsf{Sab}}$, all honest parties will output the same $v_{\mathsf{Sab}} = v$. Thus all honest parties will output either due to C2 or C3 with the same value v.

Finally, supposing no honest party outputs due to C1 or C2, all honest parties output the same value v from $\mathsf{BA}_{\mathsf{Sab}}$ due to the t_i-consistency of $\mathsf{BA}_{\mathsf{Sab}}$ and then decide it due to C3. □

Finally, we prove that Juggernaut is t_i-terminating in the sabotaged setting.

Lemma 3 (t_i-termination). *Juggernaut (Figure 1) is t_i-terminating in the sabotaged setting.*

Proof. Juggernaut trivially terminates as all honest parties decide in round $T_{\max} + T_{\mathsf{Sab}}$ (at the latest). □

Juggernaut's security in the authenticated setting. First, we prove that Juggernaut is t_s-valid in the authenticated setting.

Lemma 4 (t_s-validity). *Juggernaut (Figure 1) is t_s-valid in the authenticated setting. Furthermore, all honest parties output according to C1.*

Proof. Let all honest parties propose the same value v. By the t_s-security of $\mathsf{GC}^*_{\mathsf{Auth}}$, all honest parties output $(v, 1)$. Therefore, all honest parties set $v_3 = v$. By the t_s-validity with termination of $\mathsf{GC}^*_{\mathsf{Sab}}$, all parties output $(v, 1)$. Validity and all parties outputting according to C1 then follows as in the proof of Lemma 1 □

Next, we prove that Juggernaut is t_s-consistent in the authenticated setting.

Lemma 5 (t_s-consistency). *Juggernaut (Figure 1) is t_s-consistent in the authenticated setting. Furthermore, all honest parties output according to C1.*

Proof. We first argue that all honest parties input the same value v_3 to $\mathsf{GC}^*_{\mathsf{Sab}}$.

- Suppose first that at least one honest party outputs $g_1 = 1$ from $\mathsf{GC}^*_{\mathsf{Auth}}$. Then by the t_s-consistency of $\mathsf{GC}^*_{\mathsf{Auth}}$, all honest parties will output the same v_1. By the t_s-validity of $\mathsf{BA}_{\mathsf{Auth}}$, all honest parties output $v_2 = v_1$ from $\mathsf{BA}_{\mathsf{Auth}}$, and thus no honest party can advance from Sync with any value $v' \neq v_1$ (by t_s-justification of Sync). Thus $v_3 = v_1$ is the same for all honest parties.
- Otherwise, no party outputs $g_1 = 1$ from $\mathsf{GC}^*_{\mathsf{Auth}}$. By t_s-consistency of $\mathsf{BA}_{\mathsf{Auth}}$, all honest parties output the same v_2 (and t_s-justification of Sync ensures that no honest party can advance from Sync with any value $v' \neq v_2$), and by construction all set $v_3 = v_2$ (i.e., do not override v_3 with v_1).

Then, by the t_s-validity of $\mathsf{GC}^*_{\mathsf{Sab}}$, all honest parties output $(v_4, 1)$ from $\mathsf{GC}^*_{\mathsf{Sab}}$, multicast (decide, v_4), and by the same reasoning as for the proof of Lemma 4, all honest parties will output due to C1 (and in particular never execute $\mathsf{BA}_{\mathsf{Sab}}$). □

Lastly, we observe that Juggernaut is t_s-terminating in the authenticated setting as all parties terminate without running $\mathsf{BA}_{\mathsf{Sab}}$.

Complexity. We now attend to the complexity analysis of Juggernaut. Lemma 4 and Lemma 5 implies that in the authenticated setting, all honest parties produce an output and halt after $\mathsf{GC}^*_{\mathsf{Sab}}$, without running $\mathsf{BA}_{\mathsf{Sab}}$. Since $T_2 = O(1)$, and by the Totality and Liveness properties of Sync, we get that if r is first the round by which all honest parties produced an output from $\mathsf{BA}_{\mathsf{Auth}}$, then all forever honest parties produce an output and halt after at most $r + O(1)$ rounds, as required. Furthermore, since no honest party executes $\mathsf{BA}_{\mathsf{Sab}}$, the expected communication (in bits) complexity of Juggernaut is $O(\mathsf{CC}_{\mathcal{A}}(\mathsf{BA}_{\mathsf{Auth}}) + \lambda n^2)$, as required. In the sabotaged setting, notice that all communication aside from $\mathsf{BA}_{\mathsf{Auth}}$ and $\mathsf{BA}_{\mathsf{Sab}}$ is bounded by $O(\lambda n^2)$ bits. Thus in the sabotaged setting, the expected communication (in bits) complexity of Juggernaut is $O(\mathsf{CC}_{\mathcal{A}}(\mathsf{BA}_{\mathsf{Auth}}) + \mathsf{CC}_{A}(\mathsf{BA}_{\mathsf{Sab}}) + \lambda n^2)$, as required.
Lastly, notice that aside from (potentially) $\mathsf{BA}_{\mathsf{Auth}}$ and $\mathsf{BA}_{\mathsf{Sab}}$, the protocol Juggernaut is deterministic, and thus if $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$ are secure against an adaptive adversary, then so is Juggernaut. This concludes the proof of Theorem 1. □

3.3 Corollaries

Now that we have proven Theorem 1, we can instantiate $\mathsf{BA}_{\mathsf{Auth}}$ and $\mathsf{BA}_{\mathsf{Sab}}$ with concrete protocols in order to obtain concrete crypto-agnostic protocols. Before we state those corollaries, one point needs to be addressed.

Bit complexity in the sabotaged setting. In order to maximize the generality of our results, the only assumption we made about $\mathsf{BA}_{\mathsf{Auth}}, \mathsf{BA}_{\mathsf{Sab}}$, as explained in Definition 4, is that we are provided with round complexity bounds for these protocols. In particular, we are given no such guarantee for bit complexity. As such, in the *sabotaged* setting, when the parties run $\mathsf{BA}_{\mathsf{Auth}}$, which is an *authenticated* t_s-secure BA protocol, we have no a-priori upper bound for the amount of bits sent by honest parties during the execution, hence why we define the bit complexity of a protocol w.r.t. a particular adversary. When considering concrete protocols, however, each party can infer a bound from the description of the protocol on the number of messages it has to send, and if is exceeded, an honest party can simply halt $\mathsf{BA}_{\mathsf{Auth}}$ and move onto Sync with input $\perp$. The corollaries we state assume that such a modification was made to the final protocol.

$\mathit{Juggernaut}_{\mathsf{det}}$. For a deterministic protocol, we instantiate $\mathsf{BA}_{\mathsf{Auth}}$ with the protocol of [MR21b] modified using the techniques of [LS22] to achieve $O(f)$ round complexity, and $\mathsf{BA}_{\mathsf{Sab}}$ with the protocol of [BGP92], to obtain the following.

Corollary 1. *Let t_s, t_i s.t. $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$. There exists a deterministic authenticated t_s-secure, sabotaged t_i-secure protocol $\mathit{Juggernaut}_{\mathsf{det}}$ solving the BA problem with the following properties.*

- *In the authenticated setting, $\mathit{Juggernaut}_{\mathsf{det}}$ has $O(\lambda n^2)$ bit complexity and $O(f)$ round complexity, where $f \leq t_s$ is number of* actual *corruptions.*
- *In the sabotaged setting, $\mathit{Juggernaut}_{\mathsf{det}}$ has $O(\lambda n^2)$ bit complexity and $O(n)$ round complexity.*

$\mathit{Juggernaut}_{\mathsf{ran}}$. For a randomized protocol, we instantiate $\mathsf{BA}_{\mathsf{Auth}}$ with the protocol of [ADD+19], setting $T_{\mathsf{Auth}} = O(\lambda)$, and $\mathsf{BA}_{\mathsf{Sab}}$ with the protocol of [LS22] (run bit-by-bit in parallel), to obtain the following.

Corollary 2. *Let t_s, t_i s.t. $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$. There exists a randomized authenticated t_s-secure, sabotaged t_i-secure protocol $\mathit{Juggernaut}_{\mathsf{ran}}$ solving the BA problem with the following properties.*

- *In the authenticated setting, $\mathit{Juggernaut}_{\mathsf{ran}}$ has $O(\lambda n^2)$ expected bit complexity and $O(1)$ expected round complexity.*
- *In the sabotaged setting, $\mathit{Juggernaut}_{\mathsf{ran}}$ has $O(\lambda^2 n^2)$ bit complexity and $O(\lambda + f)$ round complexity, where $f \leq t_i$ is the number of actual corruptions.*

4 Building a Synchronizer

In this section we construct Sync, that fills the role of the synchronizer as discussed in Section 3. Formally, we prove the following.

Theorem 2. *There exists a deterministic protocol* Sync *that satisfies the following in the context of the Synchronizer primitive (See Definition 3), for any $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$.*

- *Sabotaged t_i-totality.*
- *Authenticated t_s-totality, t_s-justification, and t_s-liveness.*

Furthermore, Sync *has communication (in bits) complexity of $O(\lambda n^2)$.*

The implementation of Sync can be found in Figure 3. On calling $\mathsf{start_synchronization}$, parties multicast $\mathsf{finish}(v_i)$. On receiving $n - t_s$ $\mathsf{finish}(v)$ for some v, parties form a certificate. Then, at this point or on receipt of such a certificate, the caller multicasts it and calls $\mathsf{synchronization_completed}(v)$.

Proof (of Theorem 2).

Synchronizer Sync
(Pseudocode for a party p_i)

- When start_synchronization($v_i \in V$) is invoked, multicast finish(v_i).
- If $\exists v \in V$ such that at least $n - t_s$ finish(v) messages are received, perform the following steps: (1) use TS to create a certificate $\mathcal{C}(v)$, (2) multicast $\mathcal{C}(v)$, and (3) trigger synchronization_completed(v).
- If $\exists v \in V$ such that a certificate $\mathcal{C}(v)$ is received, perform the following steps: (1) multicast $\mathcal{C}(v)$, and (2) trigger synchronization_completed(v).

Fig. 3: Synchronizer Sync with authenticated t_s-justification, totality and liveness, and sabotaged t_i-totality.

Sync*'s correctness in the authenticated setting.* First, prove the justification property.

Lemma 6 (t_s-justification). Sync *(Figure 3) satisfies t_s-justification in the authenticated setting.*

Proof. If an honest party completes synchronization with a value v', there is at least one honest party that multicasts a finish(v') message (as $n - t_s > t_s$ given $t_s < n/2$). Hence, the lemma holds. □

Next, we prove the totality property.

Lemma 7 (t_s-totality). Sync *(Figure 3) satisfies t_s-totality in the authenticated setting.*

Proof. The lemma trivially holds as each honest party disseminates a certificate once it completes synchronization. □

Finally, we prove the liveness property.

Lemma 8 (t_s-liveness). Sync *(Figure 3) satisfies t_s-liveness in the authenticated setting.*

Proof. By the beginning of the round $\rho + 1$, each honest party receives $n - t_s$ finish(v) messages. Therefore, each honest party completes synchronization with value v by round $\rho + 1$. □

Sync*'s correctness in the sabotaged setting.* We prove the totality property.

Lemma 9 (t_i-totality). Sync *(Figure 3) satisfies t_i-totality in the sabotaged setting.*

Proof. The lemma trivially holds as each honest party disseminates a certificate once it completes synchronization (as in the authenticated case). □

Complexity. Finally, we argue that honest parties exchange $O(\lambda n^2)$ bits in Sync. Observe that each honest party multicasts a single finish message, thus sending $O(n)$ bits (assuming that the values are of constant size). Moreover, each honest party multicasts one certificate of size $O(\lambda)$ bits, thus sending $O(\lambda n)$ bits. Hence, honest parties send $n \cdot O(n + \lambda n) = O(\lambda n^2)$ bits. □

5 Building Graded Consensus

In this section, we construct our two GC protocols that we use in Juggernaut – First, our authenticated protocol with sabotaged validity and termination, and then, our sabotaged protocol with authenticated validity and termination.

5.1 Authenticated Graded Consensus with Sabotaged Validity

We first construct graded consensus for the authenticated setting, augmented with validity for the sabotaged setting. We denote the protocol by $\mathsf{GC}^*_{\mathsf{Auth}}$. Formally, we prove the following.

Theorem 3. *Let t_s, t_i such that $t_s + 2t_i < n, t_i \leq t_s < \frac{n}{2}$. There exists a $T_1 = O(1)$ round deterministic protocol $\mathsf{GC}^*_{\mathsf{Auth}}$ that satisfies the following properties in the context of the GC primitive (see Definition 2) in a Δ-synchronous network when all parties commence the protocol in the same round.*

- *Authenticated t_s-secure.*
- *Sabotaged t_i-valid.*
- *Sabotaged t_i-terminating.*

Furthermore, given Π_{MR} has at most $O(\lambda n^2)$ bit complexity and constant round complexity, so too does $\mathsf{GC}^*_{\mathsf{Auth}}$.

Our protocol $\mathsf{GC}^*_{\mathsf{Auth}}$, described in Figure 4, assumes an authenticated graded consensus protocol (see Definition 2), like the constant-round, authenticated t_s-secure, and sabotaged t_i-terminating GC of Momose-Ren [MR21a]. In $\mathsf{GC}^*_{\mathsf{Auth}}$, parties run three rounds of filtering to additionally ensure sabotaged t_i-validity before executing Π_{MR}. First, each party multicasts its input (recall all values are signed by assumption). Then, if $n - t_i$ signatures from different parties on some v are received, a certificate on v is formed, the output of the protocol is locked to $(v, 1)$, p_i's input to $\Pi_{\mathsf{MR}}$is overwritten with v, and the certificate is multicast. In the third round, parties overwrite their input if they receive non-conflicting certificates for a single value v. Finally, parties run Π_{MR}, and output the result of that if they did not lock their output.

$\mathsf{GC}^*_{\mathsf{Auth}}(v_i)$
(Pseudocode for a party p_i)
Initialization: $y_i = v_i, g_i = 0$

- Round 1: Multicast $\mathsf{init}(v)$.
- Round 2: If there exists $v \in V$ such that at least $n - t_i$ $\mathsf{init}(v)$ messages are received, then perform the following steps: (1) use threshold signatures to create an $(n - t_i)$-certificate $\mathcal{C}(v)$ for $\mathsf{init}(v)$, (2) multicast $\mathcal{C}(v)$, and (3) **lock** $y_i = v, g_i = 1$.
- Round 3: If (1) there exists $v \in V$ such that $\mathcal{C}(v)$ is received, and (2) there does not exist $v' \neq v$ such that $\mathcal{C}(v')$ is received, then $y_i \leftarrow v$.
- Round 4+: Run $(y_i, g_i) \leftarrow \Pi_{\mathsf{MR}}(y_i)$.
- **Decision:** Output (y_i, g_i).

Fig. 4: Authenticated t_s-secure and sabotaged t_i-valid graded consensus protocol $\mathsf{GC}^*_{\mathsf{Auth}}$ given an authenticated t_s-scure graded consensus Π_{MR}.

Proof (of Theorem 3). Termination clearly holds in both settings by the behaviour of the protocol. We now move on to the other properties.

Validity in the sabotaged setting. First, we prove that $\mathsf{GC}^*_{\mathsf{Auth}}$ satisfies validity in the sabotaged setting.

Lemma 10. *If any honest party receives $n - t_i$ $\mathsf{init}(v)$ messages, then v is the proposal of an honest party.*

Proof. As $n > t_s + 2t_i$, we have that $n - t_i > t_i$. Therefore, the lemma holds. □

Lemma 11 (t_i-validity). $\mathsf{GC}^*_{\mathsf{Auth}}$ *satisfies sabotaged t_i-validity.*

Proof. Suppose all honest parties propose the same value $v \in V$. As there are at least $n - t_i$ honest parties and they all propose value v, every honest party eventually receives $n - t_i$ values for v (and, by Lemma 10, only v). Therefore, every honest party decides $(v, 1)$, thus concluding the proof. □

Validity in the authenticated setting. We now prove that $\mathsf{GC}^*_{\mathsf{Auth}}$ satisfies validity in the authenticated setting. Throughout the rest of the proof of the validity property, we suppose that all honest parties propose the same value v.

Lemma 12. *If any $(n - t_i)$-certificate $\mathcal{C}(v')$ is formed, then $v' = v$.*

Proof. As $\mathcal{C}(v')$ is formed and $n - t_i > t_s$ (given that $n > t_s + 2t_i$), there exists an honest party that sends an $\mathsf{init}(v')$ message for its proposal v'. Given that all honest parties propose v, $v' = v$. □

Lemma 13. *If any honest party locks some value v' in round 2, then $v' = v$.*

Proof. Let p_i denote any honest party that locks some value v' in round 2. Therefore, p_i receives an $\mathsf{init}(v')$ message for v' from an honest party as $n - t_i > t_s$ (as $n > t_s + 2t_i$). As all honest parties propose v, $v' = v$ and the lemma holds. □

Lemma 14 (t_s-validity). $\mathsf{GC}^*_{\mathsf{Auth}}$ *satisfies authenticated t_s-validity.*

Proof. Consider any honest party p_i that decides; let p_i decide (v', g'). We distinguish two cases:
- Let party p_i lock in round 2. In this case, $v' = v$ (due to Lemma 13) and $g' = 1$. Therefore, the statement of the theorem holds in this case.
- Let party p_i decide in round (i.e., step) 4. Here, every honest party that proposes to Π_{MR} does so with value v. Indeed, if any honest party p_j updates its y_j variable in round 3, Lemma 12 proves that y_j holds value v. Hence, the validity property of Π_{MR} ensures that $v' = v$ and $g' = 1$.

Having considered both cases, the proof is concluded. □

Consistency in the authenticated setting.

Lemma 15. *No two $(n - t_i)$-certificates $\mathcal{C}(v)$ and $\mathcal{C}(v' \neq v)$ can be formed.*

Proof. By contradiction, suppose two certificates $\mathcal{C}(v)$ and $\mathcal{C}(v' \neq v)$ are formed. Therefore, there are $n - t_i + n - t_i - n = n - 2t_i$ parties that participated in forming both certificates. As $n > t_s + 2t_i$, $n - 2t_i > t_s$, which further implies that there is at least one honest party that participated in forming both certificates. As this is impossible, we reach a contradiction. □

Lemma 16. *If any honest party locks $(v, 1)$ in round 2, then all honest parties decide $(v, 1)$.*

Proof. First, all honest parties that lock in round 2 lock $(v, 1)$ (due to Lemma 15). Moreover, every other honest party p_i sets its y_i variable to v in round 3 (due to Lemma 15 and by protocol construction). Therefore, the validity property of Π_{MR} ensures that all honest parties that decide in round 4 decide $(v, 1)$. □

Lemma 17 (t_s-consistency). $\mathsf{GC}^*_{\mathsf{Auth}}$ *satisfies authenticated t_s-consistency.*

Proof. To prove the lemma, we consider two possible scenarios:

- There exist an honest party that locks $(v, 1)$ in round 2. In this case, all honest parties decide $(v, 1)$ (by Lemma 16).
- No honest party decides in round 2. In this case, the consistency property follows directly from the consistency property of Π_{MR}.

As the statement of the lemma holds in both cases, the proof is concluded. □

Complexity. The protocol Π_{MR} (using [MR21a]) has bit complexity of $O(\lambda n^2)$ for inputs of size $O(\lambda)$, and our addition to the protocol incurs an additive factor of $O(\lambda n^2)$ of bits communicated. Thus in total $\mathsf{GC}^*_{\mathsf{Auth}}$ has a bit complexity of $O(\lambda n^2)$. This concludes the proof of Theorem 3. □

5.2 Sabotaged Graded Consensus with Authenticated Validity

We now move on to our second graded consensus protocol, the dual version of $\mathsf{GC}^*_{\mathsf{Auth}}$ for the sabotaged case. Specifically, we aim to design a protocol, $\mathsf{GC}^*_{\mathsf{Sab}}$, that functions as a standard GC in the sabotaged case, but maintains validity in the authenticated case. Formally, we prove the following.

Theorem 4. *Let t_s, t_i such that $2t_i + t_s < n, t_i \leq t_s < \frac{n}{2}$. There exists an $T_2 = O(1)$ round deterministic protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ that satisfies the following in the context of the graded consensus primitive (See Definition 2) in a Δ-synchronous network if the round distance between protocol initiation times of any two honest parties is at most 1.*

- *Authenticated t_s-validity.*
- *Authenticated t_s-termination.*
- *Sabotaged t_i-security.*

Furthermore, given Π_{AW} has at most $O(\lambda n^2)$ bit complexity and constant round complexity, so too does $\mathsf{GC}^*_{\mathsf{Sab}}$.

Our protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ (Figure 5) assumes a sabotaged t_i-secure graded consensus protocol, like the constant-round round protocol Π_{AW} of Attiya and Welch [AW23] that is sabotaged t_i-secure, and authenticated t_s-terminating for any $t_i \leq t_s < \frac{n}{2}, t_s + 2t_i < n$, even in *asynchrony*. As in $\mathsf{GC}^*_{\mathsf{Auth}}$, parties first *echo* their signed value. As our goal is authenticated t_s-validity, parties form a $n - t_s$ certificate on $\mathsf{echo}(v)$ if possible. In the third round, we now require that if parties have both received certificates only for one value v, and additionally received $n - t_s$ *echos* of v, that they *vote* for v by multicasting $\mathsf{vote}(v)$. In the fourth round, parties overwrite their input to Π_{AW} if they receive at least $n - t_s - t_i$ $\mathsf{vote}(v)$ messages for unique v, lock their output to $(v, 1)$ if they receive $n - t_s$ $\mathsf{vote}(v)$ messages, and then execute Π_{AW}, outputting the result if there is no locked value.

$\mathsf{GC}^*_{\mathsf{Sab}}(v_i)$
(Pseudocode for a party p_i)
Initialization: $y_i = v_i, g_i = 0$

- Round 1 (**Echo**): Multicast $\mathsf{echo}(v_i)$.
- Round 2 (**Forward**): If $\exists v \in V$ s.t. received at least $n - t_s$ $\mathsf{echo}(v)$ messages, use TS to create certificate $\mathcal{C}(v)$ and multicast it.
- Round 3 (**Vote**): If p made a certificate $\mathcal{C}(v)$ at Round 2 for value $v \in V$, didn't receive a certificate $\mathcal{C}(v')$ for any other value $v' \in V$, and received $n - t_s$ $\mathsf{echo}(v)$, multicast $\mathsf{vote}(v)$.
- Round 4+: If $\exists v \in V$ s.t. received at least $n - t_s$ $\mathsf{vote}(v)$ messages, set $v_i = v$ and **lock** $y_i = v, g_i = 1$. Otherwise if received at least $n - t_s - t_i$ $\mathsf{vote}(v)$ for a unique value, set $v_i = v$. Run $(y_i, g_i) \leftarrow \Pi_{\mathsf{AW}}(v_i)$.
- **Decision:** Output (y_i, g_i).

Fig. 5: Sabotaged t_i-secure and authenticated t_s-valid $\mathsf{GC}^*_{\mathsf{Sab}}$ given a sabotaged t_s-secure graded consensus Π_{AW}.

Proof (of Theorem 4). Termination clearly holds in both settings by the behaviour of the protocol. We now move on to the other properties.

Let an honest party p_i start executing $\mathsf{GC}^*_{\mathsf{Sab}}$ in some global round ρ_i. Then, party p_i executes Round $x \in \{1, 2, 3, 4\}$ of $\mathsf{GC}^*_{\mathsf{Sab}}$ in global rounds $\rho_i + 2(x - 1)$ and $\rho_i + 2(x - 1) + 1$.

Authenticated t_s-validity. We first prove that $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies authenticated t_s-validity.

Lemma 18 (t_s-validity). *The protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies authenticated t_s-validity.*

Proof. Suppose all honest parties propose the same value v. As all honest parties overlap in each round of $\mathsf{GC}^*_{\mathsf{Sab}}$ for (at least) δ time, all honest parties receive $n - t_s$ $\mathsf{echo}(v)$ messages at the start of Round 2. Moreover, no certificate $\mathcal{C}(v' \neq v)$ can exist as that would imply that there exists a correct party whose proposal is $v' \neq v$. Hence, every honest party multicast $\mathsf{vote}(v)$ in Round 3, which then implies that every honest party outputs $(v, 1)$. Thus, the validity property is ensured in the signature world. □

Sabotaged t_i-validity. Then, we prove that $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies validity in the sabotaged setting. We start by proving that if all honest parties propose the same value v and an honest party creates a certificate $\mathcal{C}(v')$ in Round 2, then $v = v'$.

Lemma 19. *Suppose all honest parties propose the same value v. If an honest party creates a certificate $\mathcal{C}(v')$ in Round 2, then $v = v'$.*

Proof. If an honest party p_i creates a certificate $\mathcal{C}(v')$ in Round 2, party p_i has received an $\mathsf{echo}(v')$ message from an honest party as $n - t_s > t_i$ (given that $n > t_s + 2t_i$). Therefore, $v' = v$. □

Next, we prove that if all honest parties propose the same value v and an honest party multicast a $\mathsf{vote}(v')$ message in Round 3, then $v' = v$.

Lemma 20. *Suppose all honest parties propose the same value v. If an honest party multicasts a $\mathsf{vote}(v')$ message in Round 3, then $v' = v$.*

Proof. If an honest party p_i multicasts a $\mathsf{vote}(v')$ message in Round 3, party p_i has previously constructed a certificate $\mathcal{C}(v')$ in Round 2. By Lemma 19, $v' = v$, thus concluding the proof. □

Next, we prove that if all honest parties propose v to $\mathsf{GC}^*_{\mathsf{Sab}}$, then all honest parties propose v to Π_{AW} in Round 4.

Lemma 21. *Suppose all honest parties propose the same value v. Then, all honest parties propose v to Π_{AW} in Round 4.*

Proof. As v is proposed by every honest party, every honest party p_i has $v_i = v$. We show that if party p_i updates its v_i local variable, then it updates it to value v. Consider all possible places when party p_i could update its local variable v_i:

- Round 4 upon receiving $n - t_s$ $\mathsf{vote}(v')$ messages: As $n - t_s > t_i$ (given $n > t_s + 2t_i$), party p_i receives a $\mathsf{vote}(v')$ message from an honest party. Therefore, Lemma 20 proves that $v' = v$. Thus, the statement holds in this case.
- Round 4 upon receiving $n - t_s - t_i$ for a unique value v': As $n - t_s - t_i > t_i$ (given $n > t_s + 2t_i$), party p_i receives a $\mathsf{vote}(v')$ message from an honest party. Hence, $v' = v$ by Lemma 20, which proves the statement even in this case.

As v_i remains v at party p_i, the proof is concluded. □

Finally, we prove that $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies validity in the sabotaged setting.

Lemma 22 (t_i-validity). *The protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies validity in the sabotaged setting, i.e. $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies sabotaged t_i-validity.*

Proof. Suppose all honest parties propose the same value v. Consider any honest party p_i. We distinguish two possible cases:

- Let party p_i decide a pair (v', g') in Round 4. In this case, p_i receives $n - t_s$ $\mathsf{vote}(v')$ messages. As $n - t_s > t_i$ (given $n > t_s + 2t_i$), p_i receives a $\mathsf{vote}(v')$ message from an honest party. Therefore, Lemma 20 proves that $v' = v$ and $g' = 1$. Thus, the validity property is satisfied in this case.
- Let party p_i decide a value v' after running the Π_{AW} algorithm. By Lemma 21, all honest parties propose v to Π_{AW}. Due to the validity property of Π_{AW}, every honest party decides $(v, 1)$ from Π_{AW}, thus concluding the proof even in this case.

As the validity property is satisfied in both cases, the proof is concluded. □

Consistency in the sabotaged setting. Next, we prove that $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies consistency in the sabotaged setting. We first show that if one honest party sends a $\mathsf{vote}(v)$ message and another honest party sends a $\mathsf{vote}(v')$ message, then $v = v'$.

Lemma 23. *If an honest party p_i sends a $\mathsf{vote}(v)$ message and an honest party p_j sends a $\mathsf{vote}(v')$ message, then $v = v'$.*

Proof. By contradiction, let $v \neq v'$. As p_i (resp., p_j) sends a $\mathsf{vote}(v)$ (resp., $\mathsf{vote}(v')$) message, p_i (resp., p_j) creates a certificate $\mathcal{C}(v)$ (resp., $\mathcal{C}(v')$) in Round 2. Hence, party p_i receives a certificate $\mathcal{C}(v')$ in Round 3 and party p_j receives a certificate $\mathcal{C}(v)$ in Round 3. Therefore, we reach contradiction with the fact that parties p_i and p_j send $\mathsf{vote}(\cdot)$ messages. □

Finally, we are ready to prove the consistency property of $\mathsf{GC}^*_{\mathsf{Sab}}$.

Lemma 24 (t_i-consistency). *The protocol $\mathsf{GC}^*_{\mathsf{Sab}}$ satisfies consistency in the sabotaged setting.*

Proof. We distinguish two cases:

- Let there exist an honest party p_i that locks $(v, 1)$ in Round 4. Hence, honest party p_i receives $n - t_s$ $\mathsf{vote}(v)$ messages in Round 4. Now, consider any honest party p_j. We further consider two scenarios:
  - Let p_j lock (v', g') in Round 4. In this case, $g' = 1$. Moreover, as p_j decides in Round 4, p_j receives $n - t_s - t_i$ $\mathsf{vote}(v')$ messages. As $n - t_s - t_i > t_i$, Lemma 23 guarantees that $v' = v$.
  - Let p_j decide (v', g') after deciding (v', g') from Π_{AW}. As p_i receives $n - t_s$ $\mathsf{vote}(v)$ messages in Round 4, every honest party receives (at least) $n - t_s - t_i$ $\mathsf{vote}(v)$ messages in Round 4. Importantly, as $n - t_s - t_i > t_i$, Lemma 23 guarantees that no honest party receives $n - t_s - t_i$ $\mathsf{vote}(v')$, for any value $v' \neq v$. Hence, every honest party p_i sets its local variable v_i to v and proposes v_i to Π_{AW}. Finally, the strong validity property of Π_{AW} ensures $v' = v$.
- No honest party locks $(v, 1)$ in Round 4. In this case, the consistency property follows directly from the consistency property of Π_{AW}.

The lemma holds. □

Complexity. The protocol Π_{AW} (using [AW23]) has bit complexity of $O(\lambda n^2)$ for inputs of size $O(\lambda)$, and our addition to the protocol incurs an additive factor of $O(\lambda n^2)$ of communication bits. Thus in total $\mathsf{GC}^*_{\mathsf{Sab}}$ has bit complexity of $O(\lambda n^2)$. This concludes the proof of Theorem 4. □

6 Conclusion

In this work, we have constructed efficient crypto-agnostic Byzantine agreement, and in particular a protocol with $O(\lambda n^2)$ bit complexity and constant round complexity in the authenticated setting. Natural open problems are as follows:

- Our Juggernaut protocols use $O(\lambda n^2)$ bits only when the input message is of size $O(\lambda)$, and otherwise $O(Ln^2)$ for $L = \Omega(\lambda)$. It is thus natural to consider efficient crypto-agnostic BA for *long messages*. The main difficulty here is keeping complexity low while also providing security in the sabotaged or information-theoretic setting where it is difficult enough to build efficient protocols [CDG+24b] let alone in the crypto-agnostic setting.
- As Juggernaut optimises round complexity in the authenticated case, it may be of interest to instead optimise for the *sabotaged* case (i.e., not running the authenticated protocol in 'good' executions).
- Finally, extending our results to the model where the public key infrastructure may be inconsistent or arbitrarily broken as considered in [FHW04,FR09] may be of interest.

References

AC24. Gilad Asharov and Anirudh Chandramouli. Perfect (parallel) broadcast in constant expected rounds via statistical VSS. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 310–339. Springer, Cham, May 2024.

ADD+19. 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 *Financial Cryptography*, volume 11598 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2019.

ANS23. Ittai Abraham, Kartik Nayak, and Nibesh Shrestha. Communication and round efficient parallel broadcast protocols. Cryptology ePrint Archive, Paper 2023/1172, 2023.

AW23. Hagit Attiya and Jennifer L. Welch. Multi-valued connected consensus: A new perspective on crusader agreement and adopt-commit. In Alysson Bessani, Xavier Défago, Junya Nakamura, Koichi Wada, and Yukiko Yamauchi, editors, *27th International Conference on Principles of Distributed Systems, OPODIS 2023, December 6-8, 2023, Tokyo, Japan*, volume 286 of *LIPIcs*, pages 6:1–6:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

BCLZL23. Renas Bacho, Daniel Collins, Chen-Da Liu-Zhang, and Julian Loss. Network-agnostic security comes (almost) for free in DKG and MPC. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 71–106. Springer, Cham, August 2023.

BGP+89. Piotr Berman, Juan A Garay, Kenneth J Perry, et al. Towards optimal distributed consensus. In *FOCS*, volume 89, pages 410–415, 1989.

BGP92. Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science: research and applications*, pages 313–321. Springer, 1992.

BKL19. Erica Blum, Jonathan Katz, and Julian Loss. Synchronous consensus with optimal asynchronous fallback guarantees. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 131–150. Springer, Cham, December 2019.

BKL21. Erica Blum, Jonathan Katz, and Julian Loss. Tardigrade: An atomic broadcast protocol for arbitrary network conditions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 547–572. Springer, Cham, December 2021.

BKLZL20. Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 353–380. Springer, Cham, November 2020.

BOEY03. Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

BZL20. Erica Blum, Chen-Da Liu Zhang, and Julian Loss. Always have a backup plan: Fully secure synchronous MPC with asynchronous fallback. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 707–731. Springer, Cham, August 2020.

CCGZ19. Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. *Journal of Cryptology*, 32(3):690–741, July 2019.

CDG+24a. Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. Dare to agree: Byzantine agreement with optimal resilience and adaptive communication. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 145–156, 2024.

CDG+24b. Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, Manuel Vidigueira, and Igor Zablotchi. Error-free near-optimal validated agreement. *arXiv preprint arXiv:2403.08374*, 2024.

CGG+22. Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, Jovan Komatovic, Zarko Milosevic, and Adi Seredinschi. Crime and Punishment in Distributed Byzantine Decision Tasks. In *42nd IEEE International Conference on Distributed Computing Systems, ICDCS 2022, Bologna, Italy, July 10-13, 2022*, pages 34–44. IEEE, 2022.

CGG+23. Pierre Civit, Seth Gilbert, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. On the validity of consensus. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 332–343, 2023.

Cha90. David Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 591–602. Springer, New York, August 1990.

Che21. Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

CKPS01. Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 524–541. Springer, Berlin, Heidelberg, August 2001.

CW92. Brian A Coan and Jennifer L Welch. Modular construction of a byzantine agreement protocol with optimal message bit complexity. *Information and Computation*, 97(1):61–85, 1992.

DE24. Giovanni Deligios and Mose Mizrahi Erbes. Closing the efficiency gap between synchronous and network-agnostic consensus. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 432–461. Springer, Cham, May 2024.

DHLZ21. Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. Round-efficient byzantine agreement and multi-party computation with asynchronous fallback. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 623–653. Springer, Cham, November 2021.

DLZ23. Giovanni Deligios and Chen-Da Liu-Zhang. Synchronous perfectly secure message transmission with optimal asynchronous fallback guarantees. In Foteini Baldimtsi and Christian Cachin, editors, *FC 2023, Part I*, volume 13950 of *LNCS*, pages 77–93. Springer, Cham, May 2023.

DR85. Danny Dolev and Rüdiger Reischuk. Bounds on Information Exchange for Byzantine Agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985.

DRS90. Danny Dolev, Ruediger Reischuk, and H Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM (JACM)*, 37(4):720–741, 1990.

DS83. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

DW14. Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and MtGox. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 313–326. Springer, Cham, September 2014.

FHW04. Matthias Fitzi, Thomas Holenstein, and Jürg Wullschleger. Multi-party computation with hybrid security. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 419–438. Springer, Berlin, Heidelberg, May 2004.

FR09. Matthias Fitzi and Dominik Raub. Tight bounds for protocols with hybrid security. Cryptology ePrint Archive, Paper 2009/434, 2009. `https://eprint.iacr.org/2009/434`.

GK20. Juan A. Garay and Aggelos Kiayias. SoK: A consensus taxonomy in the blockchain era. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 284–318. Springer, Cham, February 2020.

GKKY10. S Dov Gordon, Jonathan Katz, Ranjit Kumaresan, and Arkady Yerukhimovich. Authenticated broadcast with a partially compromised public-key infrastructure. In *Stabilization, Safety, and Security of Distributed Systems: 12th International Symposium, SSS 2010, New York, NY, USA, September 20-22, 2010. Proceedings 12*, pages 144–158. Springer, 2010.

GKO+20. Juan A. Garay, Aggelos Kiayias, Rafail M. Ostrovsky, Giorgos Panagiotakos, and Vassilis Zikas. Resource-restricted cryptography: Revisiting MPC bounds in the proof-of-work era. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 129–158. Springer, Cham, May 2020.

IOZ14. Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, Berlin, Heidelberg, August 2014.

KK06. 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, Berlin, Heidelberg, August 2006.

LRM10. Christoph Lucas, Dominik Raub, and Ueli Maurer. Hybrid-secure mpc: Trading information-theoretic robustness for computational privacy. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 219–228, 2010.

LS22. Christoph Lenzen and Sahar Sheikholeslami. A recursive early-stopping phase king protocol. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, pages 60–69, 2022.

LSP82. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

LZLM+20. Chen-Da Liu-Zhang, Julian Loss, Ueli Maurer, Tal Moran, and Daniel Tschudi. MPC with synchronous security and asynchronous responsiveness. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 92–119. Springer, Cham, December 2020.

MMR15. Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous binary byzantine consensus with $t < n/3$, $o(n2)$ messages, and $O(1)$ expected time. *J. ACM*, 62(4):31:1–31:21, 2015.

MR21a. Atsuki Momose and Ling Ren. Optimal Communication Complexity of Authenticated Byzantine Agreement. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 32:1–32:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

MR21b. Atsuki Momose and Ling Ren. Optimal Communication Complexity of Authenticated Byzantine Agreement. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

PSL80. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, 1980.

PT84. Kenneth J Perry and Sam Toueg. An authenticated byzantine generals algorithm with early stopping. Technical report, Cornell University, 1984.

PW96. Birgit Pfitzmann and Michael Waidner. *Information-theoretic pseudosignatures and byzantine agreement for* $t \geq n/3$. Citeseer, 1996.