Player-Centric Byzantine Agreement

Martin Hirt[1] and Vassilis Zikas[2,⋆]

[1] Department of Computer Science, ETH Zurich
hirt@inf.ethz.ch
[2] University of Maryland, USA
vzikas@cs.umd.edu

Abstract. Most of the existing feasibility results on Byzantine Agreement (BA) are of an *all-or-nothing* fashion: in Broadcast they address the question whether or not there exists a protocol which allows *any* player to broadcast his input. Similarly, in Consensus the question is whether or not consensus can be reached which respects pre-agreement on the inputs of *all* correct players. In this work, we introduce the natural notion of *player-centric BA* which is a class of BA primitives, denoted as PCBA $= \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C}\subseteq\mathcal{P}}$, parametrized by subsets $\mathcal{C}$ of the player set. For each primitive PCBA$(\mathcal{C}) \in$ PCBA the validity is defined on the input(s) of the players in $\mathcal{C}$. Broadcast (with sender p) and Consensus are special (extreme) cases of PCBA primitives for $\mathcal{C} = \{p\}$ and $\mathcal{C} = \mathcal{P}$, respectively.
We study feasibility of PCBA in the presence of a general (aka non-threshold) mixed (active/passive) adversary, and give a complete characterization for perfect, statistical, and computational security. Our results expose an asymmetry of Broadcast which has, so far, been neglected in the literature: there exist non-trivial adversaries which can be tolerated for Broadcast with sender some $p_i \in \mathcal{P}$ but *not* for *some other* $p_j \in \mathcal{P}$ being the sender. Finally, we extend the definition of PCBA by adding fail corruption to the adversary's capabilities, and give exact feasibility bounds for computationally secure PCBA$(\mathcal{P})$ (aka Consensus) in this setting. This answers an open problem from ASIACRYPT 2008 concerning feasibility of computationally secure multi-party computation in this model.

1 Introduction

Byzantine agreement (BA) is one of the most studied problem in the areas of distributed protocols and multi-party computation. The problem was introduced by Lamport, Shostak, and Pease [26], where first solutions were also suggested. The high-level goal is to have n players agree on an output-value, where some (dishonest) players might try to prevent the others from reaching agreement. The potential dishonesty of players is modeled by considering a central adversary who corrupts players. The three most typical corruption types are active corruption (the adversary takes full control over the player), passive corruption (the adversary sees the player's internal state), and fail corruption (the adversary can make the player crash at some point during the protocol).

BA comes in two flavors, namely *Consensus* and *Broadcast*. In Consensus, every player has an input and it is required that they all agree on an output-value y (consistency), where if all correct players have the same input x then the output is $y = x$

⋆ Work done while the author was at ETH Zurich.

(validity). In Broadcast, only one player, called the *sender*, has input, and the requirements are that all players should agree on an output-value y (consistency), such that if the sender correctly follows the protocol then y equals his input (validity).[3]

A protocol is said to *perfectly $\mathcal{A}$-securely realize* Consensus or Broadcast, if it achieves the above properties with probability 1, in the presence of a computationally unbounded adversary $\mathcal{A}$. If a protocol satisfies the above properties, except with negligible probability, in the presence of a computationally bounded (resp. unbounded) adversary, then we say this protocol *computationally* (resp. *statistically*) *$\mathcal{A}$-securely realizes* the corresponding primitive.

Known results The first results on BA considered a threshold adversary who actively corrupts up to t players. In particular, in [27, 26] it was shown that when no setup is assumed, Consensus and Broadcast are possible if and only if less than a third of the players are malicious (i.e., $t < n/3$). This model has been extensively studied [10, 29, 12, 8, 9, 2, 17] and protocols with optimal resiliency and complexity (communication and computation) polynomial in the number of players were suggested. Later solutions [11, 4, 28, 6] considered a setting where a setup allowing digital signatures is available, and showed that Broadcast tolerating an arbitrary number of cheaters ($t < n$) is possible, whereas Consensus is possible if and only if $t < n/2$; for both primitives corresponding protocols with optimal resiliency and complexity polynomial in the number of players were suggested.[4] Lamport and Fischer [25] considered an adversary who can fail corrupt up to t players, and showed that any $n - 1$ players being fail corrupted can be tolerated for Broadcast. The above results were unified in [18] where it was shown that if at most t_a players are actively corrupted and, simultaneously, at most t_f are fail corrupted, and no setup is assumed, then $3t_a + t_f < n$ is a tight bound on feasibility of BA. In [23], it was observed that when a setup is assumed then the existing protocols for Broadcast and Consensus do not work for an adversary who can actively and, simultaneously, passively corrupt players. The reason is that in such a model the signatures of passively corrupted players are not reliable, as the adversary knows the signing keys and can trivially fake them. In [21] it is shown that given a Public-Key Infrastructure (PKI), an adversary who can actively corrupt up to t_a players and passively corrupt up to t_p players can be tolerated for Consensus if and only if $2t_a + \min\{t_a, t_p\} < n$.

Our Contributions We put forward a player-centric approach to BA by introducing the class $\text{PCBA} = \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C}\subseteq\mathcal{P}}$ parametrized by subsets $\mathcal{C}$ of the player set $\mathcal{P}$. Each primitive $\text{PCBA}(\mathcal{C}) \in \text{PCBA}$ has the same consistency property as traditional BA but the validity property is defined with respect to the specific set $\mathcal{C}$. In fact, Broadcast (with sender p) and Consensus are special cases of PCBA primitives for $\mathcal{C} = \{p\}$ and $\mathcal{C} = \mathcal{P}$, respectively.

We prove general negative and positive results translating feasibility statements for different PCBA primitives (i.e., for $\text{PCBA}(\mathcal{C})$ with different choices of $\mathcal{C}$), in the presence of a mixed active/passive adversary. In particular, we show under which conditions

[3] We point out that some works use the word "persistency" to refer to the validity property of Consensus; furthermore, in some works the term Byzantine agreement refers exclusively to Consensus.

[4] In fact feasibility of Broadcast for $t < n$ when a setup is available was also proved in [26], but the suggested protocol has exponential communication complexity.

we can construct PCBA($\mathcal{C}$) if we assume PCBA($\mathcal{C}'$) for $\mathcal{C}' \neq \mathcal{C}$. This characterization allows to translate feasibility results for PCBA($\mathcal{C}$) for specific choices of $\mathcal{C}$ to results about the traditional notions of BA (i.e., Broadcast and Consensus) and vice-versa. Furthermore, we provide exact feasibility bounds for PCBA, for an arbitrary choice of $\mathcal{C}$, tolerating a general adversary who might actively and passively corrupt players, simultaneously. Our results are for perfect security and, assuming a setup which allows for generation and verification of digital signatures, for statistical and computational security. Our characterization specifies the set of players who can securely broadcast their input. In fact, as we show, there are non-trivial adversary structures for which this set is neither empty nor the complete player set $\mathcal{P}$. To the best of our knowledge, this is the first work to explore this asymmetry of Broadcast. Note that in this model, with the exception of perfect security, exact bounds are not even known for traditional BA. All our protocols are efficient in the size of the player set and the representation of the inputs. Furthermore, unless some signature is forged, our protocols achieve perfect security.

As an extension of our results, we show how to define PCBA in a setting where the adversary can actively, passively, and fail corrupt players, simultaneously. For this setting, we give an exact feasibility bound for computationally secure PCBA($\mathcal{P}$) (aka Consensus), assuming a PKI. This result answers an open problem from ASIACRYPT 2008 that concerns feasibility of computationally secure multi-party computation (MPC) and secure function evaluation (SFE) in this model. In particular, in [23], a complete characterization of computationally secure MPC and SFE *assuming Broadcast* was proved. Because an exact bound for Consensus is trivially necessary for SFE (hence, also for MPC) and sufficient for Broadcast, our result fills the gap left open in that work.

Related Work BA in the general adversary model was considered in [16, 1], where exact feasibility bounds for an adversary who can actively corrupt and/or fail corrupt players were proved. However, these works consider the model without a setup and their impossibility results are *only* for Consensus. We point out that, in that model, adding passive corruption makes no difference for Consensus [5]. Feasibility of BA with active and passive corruption (and a trusted PKI) was previously studied in [21] for Consensus, and in [20] for Consensus and Broadcast. In both works, a threshold adversary is considered; the corresponding bound for Consensus is $2t_a + \min\{t_a, t_p\} < n$. In such a threshold world, constructing BA protocols for the corresponding bound turns out to be less involved than in the general-adversary setting, as one can consider the cases $t_a \leq t_p$ and $t_a > t_p$ separately, and construct one protocol for each. Finally, the intermediate ground between Broadcast and Consensus was partially explored in [13], where a variant of Consensus was considered with the property that if more than $n/2$ honest parties have the same input-value then the output is this value.

2 The Model

We consider a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of n players who can communicate with each other through a complete network of bilateral synchronous authenticated channels. Furthermore, we consider a *general active/passive adversary*, i.e., the adversary's corruption capability is characterized by an adversary structure which is a monotone set of pairs of player sets, i.e., $\mathcal{Z} = \{(A_1, E_1), \ldots, (A_m, E_m)\}$ (for some m). The adversary chooses

a class in $\mathcal{Z}$ non-adaptively i.e., before the beginning of the protocol; this class is denoted as $Z^\star = (A^\star, E^\star)$ and is called the *actual adversary class* or simply the actual adversary. The players in $A^\star$ and $E^\star$ are actively and passively corrupted, respectively. Note that $Z^\star$ is not known to the players and appears only in the security analysis. For notational simplicity we assume that $A \subseteq E$ for any $(A, E) \in \mathcal{Z}$ (intuitively, an actively corrupted player can behave as being passively corrupted). To simplify the description, we adopt the following convention: Whenever a player does not receive a message (when expecting one), or receives a message outside of the expected range, then the special symbol $\perp$ is taken for this message. Moreover, we say that a player is *correct* at a certain point of the protocol if he has followed the protocol instructions correctly up to that point.

Digital Signatures For computational and statistical security, we assume a trusted setup which allows the players to generate and verify digital signatures, e.g., a PKI, with the respective security. We make the standard assumption on the security of the used signature-scheme, namely existential unforgeability under chosen-message attacks. In slight abuse of notation, we refer to signatures that unconditionally satisfy this definition, except with negligible probability, as *information theoretically (i.t.)*, or *statistically* secure. Note the no construction of such i.t. secure signatures is known. Nevertheless, for our construction i.t. pseudo-signatures of the type used in [28] would also be sufficient. We denote by $\text{sig}_i(x)$ the signature of player p_i (i.e., generated using p_i's private key) to message x. We say that some value σ_i is a *valid* signature with signer p_i (or simply p_i's valid signature) on a message x, if the signature-verification algorithm (given p_i's public key) accepts this signature as valid for the message x. Without loss of generality, we assume that every signature includes a unique signer ID, round ID, and message ID so that it can be linked to the signer and the specific round of the protocol in which it was generated.

Passive Corruption and Forgery Passive corruption allows the adversary to see the internal state of the corrupted players. This includes the private (signing) keys of these players. Hence, for a passively corrupted player p_i, the adversary can trivially produce signatures with signer p_i on any message of her choice. Therefore, we will only use the term "forgery" for signatures of players who are not passively (or actively) corrupted.

3 Definition and Reductions

Consensus and Broadcast differ in their respective validity property. In particular, Broadcast defines validity with respect to the input of one specific player, the sender, whereas Consensus considers the inputs of *every* player in $\mathcal{P}$. A natural question which arises is: "why should one restrict the definition of BA primitives to these two extreme cases?" In fact, one can find real world scenarios where agreement on the inputs of a subset of parties is desirable, e.g., a network where a dedicated set of master-routers needs to agree on the status of certain links, in order to compute routing-paths. Furthermore, considering such intermediate cases might lead to more efficient protocols for BA and, more general, secure distributed computation in cases where only a subset of the parties need to provide input. This leads naturally to the definition of a new class of BA

primitives, called *player-centric BA* and denoted as PCBA $= \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C}\subseteq\mathcal{P}}$, which is parametrized by non-empty subsets $\mathcal{C}$ of the player set $\mathcal{P}$. All the members of the class PCBA have the same consistency property as in the original definitions of BA, but the validity property of each PCBA$(\mathcal{C}) \in$ PCBA is defined on the inputs of the players in $\mathcal{C}$. More precisely, in PCBA$(\mathcal{C})$, every $p_i \in \mathcal{C}$ has an input x_i and the goal is that all players in $\mathcal{P}$ agree on an output-value y, such that if every non-actively corrupted player in $\mathcal{C}$ has input x, then $y = x$. More formally, we say that a protocol *perfectly $\mathcal{Z}$-securely realizes PCBA$(\mathcal{C})$* among the players in $\mathcal{P}$, if it satisfies the following properties in the presence of a $\mathcal{Z}$-adversary:

- (Consistency) There exists some y such that every $p_j \in \mathcal{P} \setminus A^\star$ outputs y.[5]
- ($\mathcal{C}$-Validity) If every $p_i \in \mathcal{C} \setminus A^\star$ has the same input $x_i = x$, then every $p_j \in \mathcal{P} \setminus A^\star$ outputs $y = x$.

If a protocol satisfies the above properties except with negligible probability in the presence of a computationally bounded (resp. unbounded) $\mathcal{Z}$-adversary, then we say that the protocol *computationally* (resp. *statistically*) *$\mathcal{Z}$-securely realizes* PCBA$(\mathcal{C})$. As in most of the synchronous BA literature, all the protocols presented in this work trivially satisfy the following termination property: For every $p_i \in \mathcal{P} \setminus A^\star$ the protocol terminates after a finite number of rounds; to save space we omit it in our security analysis. We point out that the above definition requires that the inputs of all non-actively corrupted players (even those that are passively corrupted) are considered. This is the most natural way of defining PCBA in the mixed active/passive model and it is consistent with the past literature on secure distributed computation tolerating a mixed adversary, e.g. [14, 22, 24, 5, 23], as well as the literature tolerating passive corruption only (semi-honest model), e.g., [30, 19, 7].

Remark 1 (Authenticated Channels). Consistently with the existing BA literature, our definition of PCBA$(\mathcal{C})$ requires that the inputs of all $p \in \mathcal{C} \setminus A^\star$ (even those that are passively corrupted) are considered. To meet this requirement, authenticated channels are necessary. Indeed, when such channels are not given and are simulated, e.g., by digital signatures or MACs, then this requirement can trivially be violated. For a detailed discussion on secure computation without authentication we refer to [3].

Note that Broadcast (with sender p) and Consensus are special cases of PCBA for $\mathcal{C} = \{p\}$ and $\mathcal{C} = \mathcal{P}$, respectively. However, most past results on feasibility of Broadcast, including the ones considering a general adversary [22, 1], are concerned with whether or not there exists a protocol which achieve PCBA$(\{p\})$ *for every* $p \in \mathcal{P}$. In the remaining of this section we prove results which allow us to translate statements about feasibility of PCBA$(\mathcal{C})$ for different choices of $\mathcal{C} \subseteq \mathcal{P}$. All results in the current section hold for all three security levels, i.e., perfect, statistical, and computational; furthermore, all the negative results hold even when a trusted key-setup allowing digital signatures is assumed. The proofs have been moved to the full version of this paper.

An Inherent Impossibility As with Consensus, the definition of PCBA$(\mathcal{C})$ only makes sense if there are no two actively corruptible sets that cover the set $\mathcal{C}$. More

[5] Recall that $A^\star$ denotes the set of actively corrupted players.

precisely, for a player set $\mathcal{C} \subseteq \mathcal{P}$, let $\mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ denote the following condition:

$$\mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \Leftrightarrow \forall (A_1, E_1), (A_2, E_2) \in \mathcal{Z} : \; A_1 \cup A_2 \neq \mathcal{C}$$

One can verify that when $\mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ does not hold, then no protocol can achieve PCBA($\mathcal{C}$), as the parties would have to be able to distinguish the setting where the players in A_1 are corrupted from the setting where the players in A_2 are corrupted (even when the corrupted players behave correctly).

The following lemma states that if for a non-empty set $\mathcal{C}$ a $\mathcal{Z}$-secure protocol for PCBA($\mathcal{C}$) exists and the adversary cannot actively corrupt every $p_i \in \mathcal{C}$ simultaneously, then Broadcast with sender *any* player $p_i \in \mathcal{P}$ (i.e., any player in the complete player set) is possible.

Lemma 1 (Broadcast from PCBA($\mathcal{C}$)). *If for some (non-empty) set $\mathcal{C} \subseteq \mathcal{P}$ there exists a protocol for $\mathcal{Z}$-securely realizing PCBA($\mathcal{C}$) and the condition $\forall (A, E) \in \mathcal{Z} : \mathcal{C} \not\subseteq A$ holds, then for every $p \in \mathcal{P}$ there exists a protocol which $\mathcal{Z}$-securely realizes PCBA($\{p\}$) (i.e., Broadcast with sender p).*

The above lemma can be generalized to compare arbitrary subsets of $\mathcal{P}$ with respect to feasibility of PCBA as follows:

Lemma 2 (PCBA($\mathcal{C}'$) from PCBA($\mathcal{C}$)). *If for a (non-empty) set $\mathcal{C} \subseteq \mathcal{P}$, there exists a protocol for $\mathcal{Z}$-securely realizing PCBA($\mathcal{C}$) and the condition $\forall (A, E) \in \mathcal{Z} : \mathcal{C} \not\subseteq A$ holds, then for every (non-empty) set $\mathcal{C}' \subseteq \mathcal{P}$, for which the condition $\big((|\mathcal{C}'| = 1) \vee \mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C}')\big)$ holds, there exists a protocol which $\mathcal{Z}$-securely realizes PCBA($\mathcal{C}'$).*

Corollary 1. *Assuming that for some (non-empty) set $\mathcal{C} \subseteq \mathcal{P}$ the condition $\mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ holds, there exists a protocol for $\mathcal{Z}$-securely realizing PCBA($\mathcal{C}$) if and only if for every (non-empty) $\mathcal{C}' \subseteq \mathcal{C}$ for which $\mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C}')$ holds there exists a protocol which $\mathcal{Z}$-secure realizes PCBA($\mathcal{C}'$).*

4 Perfect Security

In this section we study the case of perfect security and prove an exact bound for player-centric BA tolerating a general active/passive adversary. The bound is stated in the following theorem:

Theorem 1. *Assuming $|\mathcal{P}| \geq 3$,[6] there exists a perfectly $\mathcal{Z}$-secure PCBA($\mathcal{C}$) protocol for some $\mathcal{C} \subseteq \mathcal{P}$ if and only if the condition $\mathrm{C}^{\text{perf}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ holds, where $\mathrm{C}^{\text{perf}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \iff \mathrm{C}^{(3)}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \wedge \big(|\mathcal{C}| = 1 \vee \mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})\big)$, and $\mathrm{C}^{(3)}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \iff \forall (A_1, E_1), (A_2, E_2), (A_3, E_3) \in \mathcal{Z} : \; A_1 \cup A_2 \cup A_3 \neq \mathcal{P}$.*

The sufficiency of the above condition is straight-forward: In [15] a Consensus (i.e., PCBA($\mathcal{P}$)) protocol was given which is $\mathcal{Z}$-secure when $\mathrm{C}^{(3)}(\mathcal{P}, \mathcal{Z}, \cdot)$ holds. Because

[6] The case $|\mathcal{P}| < 3$ is of no interest, as PCBA($\mathcal{C}$) is either impossible (which happens when $|\mathcal{C}| = 2$ and $\mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ is violated) or trivial.

the condition $\mathrm{C}^{(3)}(\mathcal{P},\mathcal{Z},\cdot)$ implies $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{P})$, if $|\mathcal{C}|=1$ then the sufficiency follows from Lemma 1 (feasibility of broadcast with sender the player $p\in\mathcal{C}$), otherwise Lemma 2 implies that there exist a PCBA($\mathcal{C}$) protocol for every $\mathcal{C}$ for which $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})$ holds. The necessity of the condition $\mathrm{C}^{\mathsf{perf}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})$ for PCBA($\mathcal{C}$) is proved in the full version of this paper.

5 Statistical and Computational Security (with Setup)

In this section we consider $\mathcal{Z}$-secure player-centric Byzantine Agreement in a setting where a setup allowing secure signatures. e.g., a Public-Key Infrastructure (PKI), is assumed. We point out that, unless some signature is forged, all the protocols in this section are perfectly secure. Therefore, our constructed protocols are as secure as the underlying signature scheme. The following theorem, states an exact bound for feasibility of PCBA($\mathcal{C}$) for an arbitrary set $\mathcal{C}\subseteq\mathcal{P}$, tolerating a $\mathcal{Z}$-adversary who can actively and passively corrupt players.

Theorem 2. *Assuming that $|\mathcal{P}|\geq 3$ and a setup which allows for generation/verification of computationally (resp. statistically) secure digital signatures is given, there exists a protocol which computationally (resp. statistically) $\mathcal{Z}$-secure realizes PCBA($\mathcal{C}$) for a non-empty set $\mathcal{C}\subseteq\mathcal{P}$ if and only if the condition $\mathrm{C}^{\mathsf{c/s}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})$ holds, where $\mathrm{C}^{\mathsf{c/s}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})\Leftrightarrow\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})\wedge\big((|\mathcal{C}|=1)\vee\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})\big)$, and*

$$\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})\Longleftrightarrow\begin{cases}\forall(A_1,E_1),(A_2,E_2),(A_3,E_3)\in\mathcal{Z}:\\ A_1\cup A_2\cup\big(E_1\cap E_2\cap A_3\big)=\mathcal{P}\Rightarrow(E_1\cap E_2\cap A_3)\cap\mathcal{C}=\emptyset\end{cases}$$

The necessity of the condition is proved in the full version. In the remaining of this section, we prove the sufficiency of $\mathrm{C}^{\mathsf{c/s}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})$ for the existence of $\mathcal{Z}$-secure PCBA($\mathcal{C}$). The proof proceeds in two steps: In a first step (Sub-section 5.1), we construct a PCBA($\{p\}$) protocol which is $\mathcal{Z}$-secure when the condition $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\{p\})$ is satisfied. In a second step (Sub-section 5.2), we use this protocol to construct a protocol for PCBA($\mathcal{C}$) which is $\mathcal{Z}$-secure when $\mathrm{C}^{\mathsf{c/s}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})$ is satisfied.

5.1 PCBA($\{p\}$) (Broadcast with sender p)

For simplicity we construct a *bit*-broadcast protocol; using standard techniques, one can extend it to broadcast any message. The high-level idea is the following: first, in a distribution phase, the sender p sends his input and his signature on it to every player; in a second phase, all the players run a protocol to establish a consistent view on the sender's value. Although this sounds similar to the standard approach for constructing a Broadcast protocol [8, 1] (i.e., first have p multi-send his input and then invoke Consensus), our second phase cannot be realized by a Consensus protocol as the condition $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\{p\})$ is weaker than $\mathrm{C}^{\mathsf{c/s}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{P})$ which is necessary for Consensus. Nevertheless, to realize the second phase we use an approach which is inspired by the methodology of [8] for achieving Consensus. More precisely, we build subprotocols which achieve gradually stronger consistency properties, and compose them

in a clever way to construct the Broadcast protocol. We denote these sub-protocols as MakeConsistent, GradeConsistency, and UseKing.

Because our protocols cannot achieve Consensus, the message which is distributed by the sender in the first phase plays a central role in the construction. In particular, in each sub-protocol, the players have input this message along with the sender's signature. To deal with a sender who never sends his signature to any player, we use the following technical trick: Each $p_i \in \mathcal{P}$ keeps a local bit (throughout the whole protocol) α_i which indicates whether or not, according to p_i's view, the sender p is actively corrupted. Initially $\alpha_i = 0$. If p_i detects that the sender is misbehaving then he sets $\alpha_i := 1$. When some p_i has set $\alpha_i = 1$ then p_i will accept any value as p's correct signature on any bit (without invoking the signature verification algorithm). This trick makes sure that, when no player receives a signature from p then every p_i sets $\alpha_i := 1$ and hence any value is acceptable as p's signature on any bit. As syntactic sugar we say that some value σ is a (p, p_i)-*acceptable* signature on x if $p_i \in \mathcal{P}$ accepts σ as p's signature on x (i.e, σ is valid or $\alpha_i = 1$); for a player set $\mathcal{C}$ we say that σ is a $(p, \mathcal{C})$-*acceptable* signature on x if for every $p_i \in \mathcal{C}$ the value σ is a (p, p_i)-acceptable signature on x.[7]

Remark 2 (The use of signatures). The player use signatures for detecting and exposing passive corruption. In particular, often in our sub-protocols a player p_i is instructed to send a message m to some p_j along with his signature $\text{sig}_i(m)$ on it, so that p_j has a proof that he indeed got this message from this sender in the corresponding round.[8] However, if p_i is passively corrupted, the adversary might introduce into the protocol arbitrary signatures with signer p_i. To cope with this behavior, we have p_j forward p_i's signature $\text{sig}_i(m)$ to every player as soon as he receives it. This way, if some party presents to p_j a (fake) signature with sender p_i and the same ID's (round and message ID) as $\text{sig}_i(m)$, then p_j can prove to every player that $p_i \in E^\star$.

In the following we sketch the sub-protocols MakeConsistent, GradeConsistency, and UseKing, and specify the achieved security properties. Due to space limitation, many of the protocols along with their security analysis have been removed, and will be included in the full version of this paper. We stress that the security properties are guaranteed *only when* at the beginning of the protocol the following two conditions hold: (1) Every $p_i \in \mathcal{P} \setminus A^\star$ holds as input a pair (x_i, σ_i) such that σ_i is a $(p, \mathcal{P})$-acceptable signature on x_i with round ID corresponding to the distribution phase, and (2) when $p \in \mathcal{P} \setminus A^\star$ then for some x and for every $p_i \in \mathcal{P} \setminus A^\star$: $x_i = x$ and $\alpha_i = 0$. To keep the description short, we introduce the following notation: We say that the input state is (p, x)-*well-formed* if it satisfies the above two conditions. The invariant in all sub-protocols is that if the input state is (p, x)-well-formed then the output state is also (p, x)-well-formed.

Distribution Phase Before describing the three sub-protocols, we describe the protocol Send (see next page) used in the distribution phase for p to send his input along with his signature. The protocol achieves a (p, x)-well-formed state, where x is p's input.

Lemma 3. *Assuming that no signature is forged, protocol* $\mathsf{Send}(\mathcal{P}, \mathcal{Z}, p, x)$ *achieves a* (p, x)*-well-formed state.*

[7] Note that, for any $p_j \neq p$, only p_j's valid signature, i.e., the one matching p_j's public key, can be (p_j, p_i)-acceptable.

[8] Recall that every signature has a unique signer ID, message ID, and round ID.

Protocol Send$(\mathcal{P}, \mathcal{Z}, p, x)$

1. p sends x along with his signature on it to every p_j who denotes the received value as x_j and the corresponding signature as σ_j; if p_j does not receive a message with a valid signature then he sets $\alpha_j := 1$.
2. Every $p_i \in \mathcal{P}$ forwards (x_i, σ_i) to every p_j. If p_j did not receive a consistent pair (x_j, σ_j) in Step 1 and receives one in Step 2 from some p_i then he adopts it. Otherwise p_j sets $(x_j, \sigma_j) := (0, \perp)$.

MakeConsistent As the name suggests, protocol MakeConsistent ensures that there are no inconsistencies among the outputs of non-actively corrupted players (however, some of them might output a special symbol "n/v", denoting that they have no output-value).[9] On a high level, the protocol works as follows: each p_i sends his input along with his signature on it to every party p_j; subsequently, every p_j forwards all the received values/signatures to every p_k, who uses the received signatures to detect passive corruption (see Remark 2). Each p_k checks if his view is consistent with pre-agreement on some value x; if this is the case, then he outputs x, otherwise he outputs "n/v".

GradeConsistency In GradeConsistency each $p_i \in \mathcal{P}$ outputs a pair (y_i, g_i) (along with a $(p, \mathcal{P})$-acceptable signature on y_i), where y_i is p_i's actual output-value and $g_i \in \{0, 1\}$ is a bit, called p_i's *grade*. The grade g_i has the meaning of the confidence level of p_i on the fact that agreement on y_i has been reached. In particular, if $g_i = 1$ for some $p_i \in \mathcal{P} \setminus A^\star$ then (p_i knows that) $y_j = y_i$ for every $p_j \in \mathcal{P} \setminus A^\star$. Moreover, when the non-actively corrupted players pre-agree on a value x, then they all output x with grade 1. The protocol GradeConsistency is included in the full version.

UseKing Here, there exists a distinguished player $p_k \in \mathcal{P}$, called the *king*. If $p_k \in \mathcal{P} \setminus A^\star$, then every $p_j \in \mathcal{P}$ outputs the same value $y_j = y$ along with a $(p, \mathcal{P})$-acceptable signature on it (consistency). Furthermore, independent of whether or not the king is correct, pre-agreement is preserved. The idea is simple: invoke GradeConsistency, and have the king forward his output to every party p_j, who adopts it when his grade (in GradeConsistency) was $g_j = 0$, and ignores it otherwise.

Broadcast We next describe our PCBA$(\{p\})$ (aka Broadcast) protocol: First, protocol Send is invoked. Subsequently, for $k = 1, \ldots, n$, UseKing is invoked with king $p_k \in \mathcal{P}$. The input to the first iteration of UseKing is the state which is output from Send, whereas for $k = 2, \ldots, n$ the input to the kth iteration of UseKing is the output of the $(k-1)$th iteration. If $p \in \mathcal{P} \setminus A^\star$ then the well-formness of the state (Lemma 3) ensures that from the first iteration of UseKing all $p_i \in \mathcal{P} \setminus A^\star$ have as input the input-bit of p, and the $\{p\}$-validity property of UseKing ensures that this agreement will be preserved in all iterations. In any case, $\mathbf{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \{p\})$ ensures that there is at least one honest player $p_\ell \in \mathcal{P}$; at the latest during the iteration of UseKing with king p_ℓ, agreement on the output will be achieved (king consistency), which is maintained in all future iterations of UseKing ($\{p\}$-validity).

Lemma 4. *Assuming that no signature is forged and the condition* $\mathbf{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \{p\})$ *holds, the protocol* Broadcast$(\mathcal{P}, \mathcal{Z}, p, x)$ *$\mathcal{Z}$-securely realizes PCBA$(\{p\})$ (i.e, Broadcast with sender p).*

[9] Observe that "n/v" is not the same as $\perp$.

5.2 PCBA($\mathcal{C}$) for an arbitrary $|\mathcal{C}| \geq 1$

Using our PCBA($\{p\}$) protocol, i.e., protocol Broadcast, we can achieve PCBA($\mathcal{C}$) for an arbitrary $\mathcal{C} \subseteq \mathcal{P}$. The corresponding protocol, denoted as $\mathsf{PCBA}_{\mathcal{C}}$, is described in the following; the input of each $p_i \in \mathcal{C}$ is denoted as x_i.

Protocol $\mathsf{PCBA}_{\mathcal{C}}(\mathcal{P}, \mathcal{Z}, x_1, \ldots, x_{|\mathcal{C}|})$

1. Every $p_i \in \mathcal{C}$ uses Broadcast to Broadcast x_i.
2. For each p_j: if $|\mathcal{C}| = 1$, then output the broadcasted value; otherwise, if there exists unique x s.t. $\exists (A, E) \in \mathcal{Z} : \{p_i \mid x_i \neq x\} \subseteq A$ then output x, otherwise output 0.

Lemma 5. *Assuming that no signature is forged, if the condition* $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \wedge \big((|\mathcal{C}| = 1) \vee \mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \{p\})\big)$ *holds for a set* $\mathcal{C} \subseteq \mathcal{P}$*, then the protocol* $\mathsf{PCBA}_{\mathcal{C}}$ *perfectly* $\mathcal{Z}$*-securely realized PCBA*($\mathcal{C}$)*.*

6 Extension: Adding Fail Corruption

We extend the definition of PCBA to consider an adversary who can actively, passively and fail corrupt players, simultaneously. In this setting, a general adversary is described by a structure which is a collection of triples (instead of pairs) of player sets, $\mathcal{Z} = \{(A_1, E_1, F_1), \ldots, (A_m, E_m, F_m)\}$, where the adversary of class (A, E, F) actively corrupts the players in A, passively corrupts the players in E, and fail corrupts the players in F. Consistently with our previous notation, we denote by $(A^\star, E^\star, F^\star)$ the class corresponding to the adversary's actual corruption choice. To simplify the notation, we assume that $A \subseteq F$ (anyway, an actively corrupted player can behave as being fail corrupted). We say that a player is *alive* at a certain point of the protocol if he has not crashed until that point. Note that a fail corrupted player is both correct and alive until the point when he crashes. In the following we give the definition of player-centric BA in this extended model and prove an exact feasibility bound for PCBA($\mathcal{P}$) (aka Consensus) for computational security assuming a trusted PKI.

A natural question which arises when fail corruption is considered in PCBA is how the inputs of fail corrupted players are accounted in the validity condition. Following the intuition that a fail corrupted player never gives a wrong input (but might give no input) we extend the definition of PCBA as follows: every p_i has input x_i and the goal is to agree on an output-value y, such that if every $p_i \in \mathcal{C} \setminus A^\star$ who is alive at the beginning has the same input $x_i = x$ then $y = x$. More formally, let $\mathcal{C} \subseteq \mathcal{P}$; we say that a protocol *perfectly* $\mathcal{Z}$*-securely realizes PCBA*($\mathcal{C}$), if it satisfies the following properties in the presence of a $\mathcal{Z}$-adversary:

- (Consistency) There exists some y such that every player p_i who is correct until the end of the protocol outputs y.
- ($\mathcal{C}$-Validity) If every $p_i \in \mathcal{C} \setminus A^\star$ who is alive at the beginning of the protocol has the same input $x_i = x$, then every (alive) $p_j \in \mathcal{P} \setminus A^\star$ outputs $y \in \{x, \text{“n/v”}\}$, where $y = x$, unless all players in $\mathcal{C} \setminus A^\star$ have crashed during the protocol execution.
- (Termination) For $p_i \in \mathcal{P} \setminus A^\star$ the protocol terminates after a finite number of rounds.

When a protocol satisfies the above properties except with negligible probability in the presence of a computationally bounded (resp. unbounded) adversary, then we say that the protocol *computationally* (resp. *statistically*) $\mathcal{Z}$*-securely realizes* PCBA($\mathcal{C}$).

PCBA($\mathcal{P}$) (Consensus) We next give an complete characterization of tolerable adversaries for computationally secure PCBA($\mathcal{P}$) (Consensus) in our model. Recall that the corresponding bound for (perfect) security without a setup can be derived in a straightforward manner from [1] (see [5] for details). The necessary and sufficient condition is stated in the following theorem which is proved in the full version of this paper:

Theorem 3. *Assuming $|\mathcal{P}| \geq 3$, if a setup allowing digital signatures is given, then a set of players $\mathcal{P}$ can computationally $\mathcal{Z}$-securely realize Consensus if and only if the following condition holds:* $\forall (A_1, E_1, F_1), (A_2, E_2, F_2), (A_3, E_3, F_3) \in \mathcal{Z}: \; A_1 \cup A_2 \cup \big(\big((E_1 \cap F_1) \cup (E_2 \cap F_2) \cup (E_1 \cap E_2)\big) \cap A_3\big) \cup (F_1 \cap F_2 \cap F_3) \neq \mathcal{P}$.

7 Conclusions and Open Problems

Most existing definitions of Byzantine Agreement are of an all-or-nothing type. Motivated by the above observation, we introduced a new class of player-centric BA primitives, denoted as PCBA $= \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C} \subseteq \mathcal{P}}$, which is parametrized by non-empty subsets $\mathcal{C}$ of the player set. For each PCBA($\mathcal{C}$) $\in$ PCBA, the validity condition depends on the inputs of the players in $\mathcal{C}$. We proved general negative and positive results, which associate feasibility of PCBA for different choices of the set $\mathcal{C}$. Furthermore, for a general active/passive adversary we proved exact feasibility bounds for PCBA for any choice of the set $\mathcal{C}$, for all three security levels, i.e., perfect, statistical, and computational. Moreover, we showed that there might be adversaries who can be tolerated for some specific sender p to broadcast his input, but not for any $p' \in \mathcal{P}$ being the sender.

As an extension of our results we provide a definition of PCBA tolerating an active/passive/fail adversary and prove an exact bound for PCBA($\mathcal{P}$) (aka Consensus) in this model. A complete characterization of PCBA for an arbitrary set $\mathcal{C}$ in this extended model is an interesting research direction. However, the unexpectedly high complexity of the tight bound for Consensus gives some evidence that such a characterization might be too complicated, and raises the question whether one should look for a different trichotomy of corruption types.

References

1. B. Altmann, M. Fitzi, and U. Maurer. Byzantine agreement secure against general adversaries in the dual failure model. In *DISC '99*, pp. 123–137, 1999.
2. A. Bar-Noy, D. Dolev, C. Dwork, and H. Strong. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. *Inf. Comput.*, 97(2):205–233, 1992.
3. B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. In *CRYPTO 2005*, pp. 361–377, 2005.
4. B. Baum-Waidner, B. Pfitzmann, and M. Waidner. Unconditional Byzantine agreement with good majority. In *STACS '91*, pp. 285–295, 1991.
5. Z. Beerliova-Trubiniova, M. Fitzi, M. Hirt, U. Maurer, and V. Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In *TCC 2008*, pp. 231–250, 2008.

6. Z. Beerliova-Trubiniova, M. Hirt, and M. Riser. Efficient Byzantine agreement with faulty minority. In *ASIACRYPT 2007*, pp. 393–409, 2007.
7. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88*, pp. 1–10, 1988.
8. P. Berman, J. Garray, and J. Perry. Towards optimal distributed consensus. In *FOCS '89*, pp. 410–415, 1989.
9. B. Coan and J. Welch. Modular Construction of Efficient Byzantine Agreement Protocols. In *PODC '89*, pp. 295–306, 1989.
10. D. Dolev, M. Fischer, R. Fowler, N. Lynch, and H. Strong. An efficient algorithm for Byzantine agreement without authentication. *Information and Control*, 52(3):257–274, 1982.
11. D. Dolev and H. Strong. Polynomial algorithms for multiple processor agreement. In *STOC '82*, pp. 401–407, 1982.
12. P. Feldman and S. Micali. Optimal algorithms for Byzantine agreement. In *STOC '88*, pp. 148–161, 1988.
13. M. Fitzi and J. Garray Efficient player-optimal protocols for strong and differential consensus. In *PODC '03*, pp. 211–220, 2003.
14. M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for privacy in unconditional multi-party computation. In *CRYPTO '98*, pp. 121–136, 1998. Corrected version available online.
15. M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT '99*, pp. 232–246, 1999.
16. M. Fitzi and U. Maurer. Efficient Byzantine agreement secure against general adversaries. In *DISC '98*, pp. 134–148, 1998.
17. J. Garay and Y. Moses. Fully polynomial Byzantine agreement in t+1 rounds. In *STOC '93*, pp. 31–41, 1993.
18. J. Garay and K. Perry. A continuum of failure models for distributed computing. In *WDAG '92*, pp. 153–165, 1992.
19. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *STOC '87*, pp. 218–229, 1987.
20. S. Gordon, J. Katz, R. Kumaresan, and A. Yerukhimovich. Authenticated broadcast with a partially compromised public-key infrastructure. In *SSS 2010*, pp. 144–158, 2010.
21. A. Gupta, P. Gopal, P. Bansal, and K. Srinathan. Authenticated Byzantine generals in dual failure model. In *ICDCN 2010*, pp. 79-91, 2010.
22. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *PODC '97*, pp. 25–34, 1997.
23. M. Hirt, U. Maurer, and V. Zikas. MPC vs. SFE: Unconditional and computational security. In *ASIACRYPT 2008*, pp. 1–18, 2008.
24. Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *CRYPTO 2006*, pp. 483–500, 2006.
25. L. Lamport and M. Fischer. Byzantine generals and transaction commit protocols. Technical Report Opus 62, SRI International (Menlo Park CA), TR, 1982.
26. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
27. M. Pease and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27:228–234, 1980.
28. B. Pfitzmann and M. Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *STACS '92*, pp. 337–350, 1992.
29. S. Toueg, K. Perry, and T. Srikanth. Fast distributed agreement. *SIAM J. Comput.*, 16(3):445–457, 1987.
30. A. Yao. Protocols for secure computations. In *FOCS '82*, pp. 160–164, 1982.