Authenticated Broadcast with a Partially Compromised Public-Key Infrastructure

S. Dov Gordon* Jonathan Katz† Ranjit Kumaresan†
Arkady Yerukhimovich‡

Abstract

Given a public-key infrastructure (PKI) and digital signatures, it is possible to construct broadcast protocols tolerating any number of corrupted parties. Existing protocols, however, do not distinguish between *corrupted* parties who do not follow the protocol, and *honest* parties whose secret (signing) keys have been compromised but continue to behave honestly. We explore conditions under which it is possible to construct broadcast protocols that still provide the usual guarantees (i.e., validity/agreement) to the latter.

Consider a network of n parties, where an adversary has compromised the secret keys of up to t_c honest parties and, in addition, fully controls the behavior of up to t_a other parties. We show that for any fixed $t_c > 0$ and any fixed t_a, there exists an efficient protocol for broadcast if and only if $2t_a + \min(t_a, t_c) < n$. (When $t_c = 0$, standard results imply feasibility for all $t_a < n$.) We also show that if t_c, t_a are not fixed, but are only guaranteed to satisfy the above bound, then broadcast is impossible to achieve except for a few specific values of n; for these "exceptional" values of n, we demonstrate broadcast protocols. Taken together, our results give a complete characterization of this problem.

1 Introduction

Broadcast protocols allow a designated party (the *dealer*) to distribute an input value to a set of parties such that (1) if the dealer is honest, all honest parties output the dealer's value (**validity**), and (2) even if the dealer is dishonest, the outputs of all honest parties agree (**agreement**). Broadcast protocols are fundamental for distributed computing and secure computation: they are crucial for simulating a broadcast channel over a point-to-point network, and thus form a critical sub-component of various higher-level protocols.

Classical results of Pease, Shostak, and Lamport [11, 9] show that broadcast is achievable in a synchronous network of n parties if and only if the number of corrupted parties t satisfies $t < n/3$. To go beyond this bound, some form of setup is required. The most commonly studied setup assumption is the existence of a public-key infrastructure (PKI) such that each party P_i has a

*Applied Communication Sciences, Piscataway, NJ. Work done while at the University of Maryland. Email: gordon.dov@gmail.com

†Department of Computer Science, University of Maryland, College Park, MD. This work was supported by NSF, the U.S. DoD/ARO MURI program, and the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001. Email: {jkatz,ranjit}@cs.umd.edu

‡MIT Lincoln Laboratory, Lexington, MA. Email: arkady@cs.umd.edu

public signing key pk_i known to all other parties. (Of course, this is only interesting if we assume the existence of secure digital signatures [6]). In this model, broadcast is possible for $t < n$ [11, 9, 2].

With few exceptions [4, 7] (see below), prior work in the PKI model treats each party as either totally honest, or as completely corrupted and under the control of a single adversary; the assumption is that the adversary cannot forge signatures of any honest parties. In many situations it makes sense to consider a middle ground: parties who honestly follow the protocol but whose signatures might be forged (e.g., because their signing keys have been compromised). Most existing work treats any such party P_i as corrupt, and provides no guarantees for P_i in this case: the output of P_i may disagree with the output of other honest parties, and validity is not guaranteed if P_i is the dealer. Clearly, it would be preferable to ensure agreement and validity for honest parties who have simply had the misfortune of having their signatures forged.

Here, we consider broadcast protocols providing exactly these guarantees. Specifically, say t_a parties in the network are actively corrupted; as usual, such parties may behave arbitrarily and we assume their actions are coordinated by a single adversary $\mathcal{A}$. We also allow for t_c parties who follow the protocol honestly, but whose signatures can be forged by $\mathcal{A}$; this is modeled by simply giving $\mathcal{A}$ their secret keys. We refer to such honest-behaving parties as *compromised*, and require agreement and validity to hold even for compromised parties.

Say t_a, t_c, n *satisfy the threshold condition* if $2t_a + \min(t_a, t_c) < n$. We show:

1. For any t_a, t_c, n satisfying the threshold condition, there is an efficient protocol achieving the notion of broadcast outlined above.

2. When the threshold condition is *not* satisfied, broadcast protocols meeting our notion of security are impossible (with the exception of the "classical" case where $t_c = 0$, in which case standard results imply feasibility).

3. Except for a few "exceptional" values of n, there is no *fixed* n-party protocol that tolerates all t_a, t_c satisfying the threshold condition with respect to n. For the exceptional values of n, we show protocols that *do* tolerate any t_a, t_c satisfying the threshold condition.

Taken together, our results provide a complete characterization of the problem for threshold adversaries. (Subsequent to our work, Hirt and Zikas [8] extended our results to the case of more-general access structures.)

Motivating the problem. Compromised parties are most naturally viewed as honest parties whose secret keys have been obtained by the adversary, e.g., because a user's secret key was weak, or because an adversary hacked into a user's system and obtained their secret key (but subsequently the honest party's computer was re-booted and now behaves correctly). Exactly this scenario is addressed by *proactive* cryptosystems [10] in a somewhat different context.

Our work also provides guarantees in case an honest user's signature might be *forged* (whether or not the adversary learns the user's secret key). Signature forgery can potentially occur due to cryptanalysis, poor implementation of cryptographic protocols, or side-channel attacks. In all these cases, an adversary might be able to forge signatures of a small number of honest parties without being able to forge signatures of everyone.

Prior work. Gupta et al. [7] also consider the problem of designing broadcast protocols providing agreement and validity for honest-behaving parties whose secret keys have been compromised, and claim results similar to ours. Our results improve upon theirs in several respects. For starters,

their positive result appears to be flawed[1]; in any case, the protocol they present has complexity exponential in n, whereas the protocol we show in this paper has complexity polynomial in n. Although Gupta et al. also claim impossibility when $2t_a + \min(t_a, t_c) \geq n$, our impossibility result is simpler and stronger in that it holds relative to a weaker adversary.[2] Finally, Gupta et al. treat t_a, t_c as known and do not consider the question of designing a fixed protocol achieving broadcast for any t_a, t_c satisfying the threshold condition (as we do in the third result mentioned above).

Fitzi et al. [4] consider broadcast in a model where the adversary can either corrupt a few players and forge signatures of *all* parties, or corrupt more players but forge *no* signatures; a statement of their result is given here as Theorem 11. Our work addresses the intermediate cases, where an adversary might be able to forge signature of some honest parties but not others.

Organization. Section 2 introduces our model and formal definition of broadcast in our setting. In Section 3 we show that for every t_a, t_c, n satisfying the threshold condition there exists an efficient protocol for broadcast. We show our impossibility results in Section 4: namely, that broadcast is impossible whenever t_a, t_c, n do not satisfy the threshold condition (except when $t_c = 0$), and — other than for certain exceptional values of n — there does not exist a fixed protocol achieving broadcast for all t_a, t_c satisfying the threshold condition. In Section 5 we give positive results for the exceptional values of n. Although dealing with these "outliers" may seem like a minor point, the exceptional values of n are all small and so may arise in practice.

2 Model and Definitions

We consider the standard setting in which n players communicate in synchronous rounds via authenticated channels in a fully connected, point-to-point network. (See below for further discussion regarding the assumption of authenticated channels.) We assume a public-key infrastructure (PKI), established as follows: each honest party P_i runs some key-generation algorithm Gen (specified by the protocol) to obtain a public key pk_i along with a corresponding secret key sk_i. Then all parties begin running the protocol holding the same vector of public keys $(pk_1, \ldots, pk_n)$, and with each P_i holding sk_i.

A party that is *actively corrupted* (or "Byzantine") may behave arbitrarily. All other parties are called *honest*, though we further divide the set of honest parties into those who have been *compromised* and those who have not been compromised, as discussed below. We view the set of actively corrupted players as being under the control of a single adversary $\mathcal{A}$ coordinating their actions. We always assume such parties are *rushing*, and may wait to see the messages sent by honest parties in a given round before deciding on their own messages to send in that round. Actively corrupted parties may choose their public keys arbitrarily and even based on the public keys of honest parties. We continue to assume, however, that all honest parties hold the same vector of public keys.

Some honest parties may be *compromised*; if P_i is compromised then the adversary $\mathcal{A}$ is given P_i's secret key sk_i. We stress that compromised parties follow the protocol as instructed: the only difference is that $\mathcal{A}$ is able to forge signatures on their behalf. On the other hand, we assume $\mathcal{A}$ is unable to forge signatures of any honest players who have *not* been compromised.

[1] The issue is that their analysis does not take into account the possibility that an honest (but compromised) party may receive a valid signature, signed under its own key, of a message it has not signed before.

[2] In [7], the adversary is assumed to have access to the random coins used by the compromised parties when running the protocol, whereas we do not make this assumption.

We assume authenticated point-to-point channels between *all* honest parties, even those who have been compromised. In other words, although the adversary can forge the signature of an honest party P_i who has been compromised, it cannot falsely inject a point-to-point message on P_i's behalf. This assumption is reasonable since different cryptographic keys (other than parties' signing keys) could be used for authenticating point-to-point communication; in small-scale networks, authenticated communication could even be established via physical means. Without the assumption of authenticated channels, no meaningful results are possible.

Definition 1 *A protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a dealer $D \in \mathcal{P}$ holds an initial input $m \in \mathcal{M}$, achieves* **broadcast** *if the following hold:*

Validity *If the dealer is honest, then all honest parties output m.*

Agreement *All honest parties output the same value.*

We stress that "honest" includes honest parties who have been compromised.

Although the above refers to an arbitrary input $m \in \mathcal{M}$ for the dealer, we assume for simplicity that the dealer's input is a bit b. Broadcast for arbitrary-length messages can be obtained from binary broadcast using standard techniques.

An adversary $\mathcal{A}$ is called a (t_a, t_c)-*adversary* if $\mathcal{A}$ actively corrupts up to t_a parties and additionally compromises up to t_c of the honest parties. In a network of n players, we call $\mathcal{A}$ a *threshold adversary* if $\mathcal{A}$ chooses t_a, t_c subject to the restriction $2t_a + \min(t_a, t_c) < n$, actively corrupts up to t_a parties, and compromises up to t_c honest parties.

3 Broadcast for (t_a, t_c)-Adversaries

In this section, we prove the following result:

Theorem 2 *Fix n, t_a, t_c. If $t_c = 0$ or $2t_a + \min(t_a, t_c) < n$, there exists an n-party protocol achieving broadcast in the presence of a (t_a, t_c)-adversary.*

When $t_c = 0$ no signatures can be forged, and so a standard protocol for authenticated broadcast suffices. If $t_a \leq t_c$ then $3t_a < n$ and the parties can run a standard (unauthenticated) broadcast protocol where the PKI is not used at all. The challenge is thus to design a protocol for $0 < t_c < t_a$, and we deal with this case in the remainder of the section.

In fact, we show how to achieve *weak broadcast* for $2t_a + t_c < n$. Weak broadcast is a relaxation of broadcast where validity holds as before, but agreement is weakened to require only that if any honest party outputs a value $b \in \{0, 1\}$ then every other honest party outputs either b or $\perp$. Since it is known [5, 4] that weak broadcast implies broadcast unconditionally (i.e., without using a PKI at all) when $2t_a < n$, this proves our desired result.

Theorem 3 *For any n, t_a, t_c with $2t_a + t_c < n$, the protocol in Figure 1 achieves weak broadcast in the presence of a (t_a, t_c)-adversary.*

Proof: We first prove validity. If D is honest and not compromised, then in round 1 each honest party $P_i \neq D$ receives (b, σ_D) with $\mathsf{Vrfy}_{pk_D}(b, \sigma_D) = 1$. So in round 2 this means that P_i receives

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote the set of parties. Let $D \in \mathcal{P}$ denote the dealer, $b \in \{0,1\}$ its input, and pk_D, sk_D its public and secret keys.

Round 1: D computes $\sigma_D \leftarrow \mathsf{Sign}_{sk_D}(b)$ and sends (b, σ_D) to every party.

Round 2: Each party P_i (other than D) does the following: let (b', σ_D) be the message that P_i received from D in the previous round. Compute $\sigma_i \leftarrow \mathsf{Sign}_{sk_i}(b' \| \sigma_D)$ and send $(b', \sigma_D, i, \sigma_i)$ to all other parties.

Round 3: $(c, \sigma_D, j, \sigma_j)$ is a *valid c-tuple for* P_j if (1) $\mathsf{Vrfy}_{pk_D}(c, \sigma_D) = 1$; (2) $P_j \neq D$; and (3) $\mathsf{Vrfy}_{pk_j}(c \| \sigma_D, \sigma_j) = 1$. A *valid tuple* is a valid 0-tuple or 1-tuple for some P_j. Each party P_i (other than D) sends to every other party all the valid tuples it received in the previous round.

Output determination: D outputs b. For every other P_i, let (b, σ_D) be the message P_i received from D in round 1. P_i outputs b if all the following hold: (1) $\mathsf{Vrfy}_{pk_D}(b, \sigma_D) = 1$; (2) P_i received valid b-tuples for at least $n - t_a - 1$ parties in round 2; and (3) P_i received valid $\bar{b}$-tuples for fewer than $n - t_a - 1$ parties in round 3. In any other case, P_i outputs $\perp$.

Figure 1: A protocol for weak broadcast, for fixed t_a, t_c with $2t_a + t_c < n$.

valid b-tuples for at least the $n - t_a - 1$ honest parties excluding D. Since D's signature cannot be forged, P_i receives no valid $\bar{b}$-tuples in round 3. We conclude that P_i outputs b.

The argument is similar if D is honest but compromised. Again, in round 1 each honest party $P_i \neq D$ receives (b, σ_D) from D with $\mathsf{Vrfy}_{pk_D}(b, \sigma_D) = 1$; in round 2, P_i receives valid b-tuples for at least the $n - t_a - 1$ honest parties excluding D. Now, however, since the adversary can forge signatures of D and $t_c - 1$ other honest parties, P_i may receive valid $\bar{b}$-tuples for up to $t_a + t_c - 1$ parties in round 3. But $t_a + t_c - 1 < n - t_a - 1$, so P_i still outputs b.

Finally, we prove weak agreement. Say some honest P_i outputs a bit b. This means that P_i must have received valid b-tuples for at least $n - t_a - 1$ parties in round 2. Since P_i forwards these to all other parties, any other honest party P_j receives valid b-tuples for at least $n - t_a - 1$ parties in round 3, and so cannot possibly output $\bar{b}$. ■

4 Impossibility Results

In this section we show two impossibility results. First, we show that there is no protocol achieving broadcast in the presence of a (t_a, t_c)-adversary when $2t_a + \min(t_a, t_c) \geq n$ and $t_c > 0$, thus proving that Theorem 2 is tight. We then consider the case when t_a, t_c are not fixed in advance, but instead all that is guaranteed is that $2t_a + \min(t_a, t_c) < n$. (I.e., we seek a single protocol that works for all values of t_a, t_c satisfying this condition.) We show that in this setting broadcast is impossible for all but a few exceptional values of n.

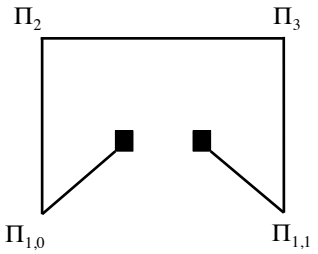

Figure 2: A modified network with two copies $\Pi_{1,0}, \Pi_{1,1}$ of P_1's program Π_1. There is no direct interaction between $\Pi_{1,0}$ and Π_3, or between $\Pi_{1,1}$ and Π_2.

4.1 The Three-Party Case

We first present a key lemma that will be useful for the proofs of both the results described above. Our proof of the lemma follows the presentation in [3] which considers a different security model.

Lemma 4 *There is no 3-party protocol among a set of parties $\{P_1, P_2, P_3\}$ that achieves broadcast when P_1 acts as the dealer and the adversary $\mathcal{A}$ can choose any one of the following corruption patterns:*

- *Actively corrupt P_1.*
- *Actively corrupt P_2, and compromise the secret key of P_1.*
- *Actively corrupt P_3, and compromise the secret key of P_1.*

Proof: Fix some protocol Π, and let Π_1, Π_2, Π_3 denote the program specified by Π for parties P_1, P_2, P_3 respectively.

We imagine running Π in the modified network shown in Figure 2. Here, two *independent* copies $\Pi_{1,0}, \Pi_{1,1}$ of Π_1 are run in the following way. All messages sent by $\Pi_{1,0}$ to P_3 are discarded, while all messages sent to P_2 are delivered to Π_2. Any messages expected from P_3 are replaced with null messages. Similarly, all messages sent by $\Pi_{1,1}$ to P_2 are discarded, while all messages sent to P_3 are delivered to Π_3. Any messages expected from P_2 are replaced with null messages. Messages sent by Π_2 to P_1 are routed to $\Pi_{1,0}$; thus, $\Pi_{1,1}$ does not receive any messages directly from Π_2. Similarly, all messages sent by Π_3 to P_1 are now routed to $\Pi_{1,1}$, and so $\Pi_{1,0}$ does not receive any messages directly from Π_3. Programs Π_2 and Π_3 interact with each other just as they would in a real network. We stress that $\Pi_{1,0}$ and $\Pi_{1,1}$ use the same public/secret keys, namely, the keys belonging to P_1.

Claim 5 *There exists an adversary $\mathcal{A}$ that actively corrupts P_1 and such that the joint view of P_2 and P_3 interacting with $\mathcal{A}$ in the real network is identically distributed to the joint view of Π_2 and Π_3 in the modified network.*

Proof: See Figure 3. $\mathcal{A}$ simply simulates the behavior of $\Pi_{1,0}$ and $\Pi_{1,1}$. That is, $\mathcal{A}$ internally runs (independent) programs $\Pi_{1,0}$ and $\Pi_{1,1}$. The messages $\mathcal{A}$ sends to P_2 (resp., P_3) are those sent by $\Pi_{1,0}$ to P_2 (resp., $\Pi_{1,1}$ to P_3); the messages $\mathcal{A}$ receives from P_2 (resp., P_3) are delivered to $\Pi_{1,0}$ (resp., $\Pi_{1,1}$). ■

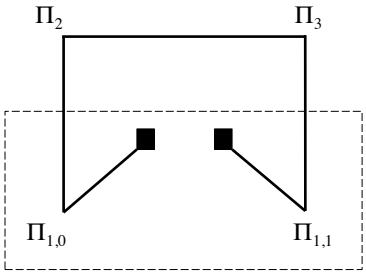

Figure 3: An adversary corrupting P_1 and interacting with P_2 and P_3 in the real network can simulate $\Pi_{1,0}$ and $\Pi_{1,1}$ interacting with Π_2 and Π_3.

Claim 6 *There exists an adversary $\mathcal{A}$ that actively corrupts P_2 and compromises P_1 and such that the joint view of P_1 and P_3 interacting with $\mathcal{A}$ in the real network is identically distributed to the joint view of $\Pi_{1,1}$ and Π_3 in the modified network.*

Proof: See Figure 4. $\mathcal{A}$ has sk_1 and sk_2 and so can internally run $\Pi_{1,0}$ and Π_2, forwarding messages between them internally. $\mathcal{A}$ always sends null messages to (the real) P_1, and ignores all messages from P_1. The messages $\mathcal{A}$ sends to P_3 are those sent by Π_2 to Π_3, and messages from P_3 are delivered to Π_2. ■

By an analogous argument, we have:

Claim 7 *There exists an adversary $\mathcal{A}$ that actively corrupts P_3 and compromises P_1 and such that the joint view of P_1 and P_2 interacting with $\mathcal{A}$ in the real network is identically distributed to the joint view of $\Pi_{1,0}$ and Π_2 in the modified network.*

We will now focus on an execution in the modified network. Say $\Pi_{1,0}$ has input 0 and $\Pi_{1,1}$ has input 1. Since Π guarantees validity, Claim 6 implies that Π_3 must output 1 and, similarly, Claim 7 implies that Π_2 must output 0. But then Claim 5 shows that Π does not achieve agreement. ■

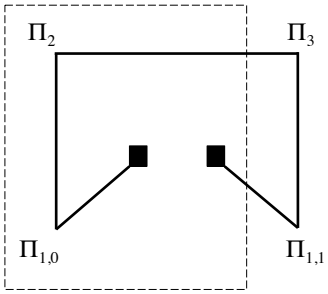

Figure 4: An adversary corrupting P_2, compromising P_1, and interacting with P_1 and P_3 in the real network can simulate Π_2 and $\Pi_{1,0}$ interacting with $\Pi_{1,1}$ and Π_2.

4.2 Impossibility of Broadcast when $2t_a + \min(t_a, t_c) \geq n$ and $t_c > 0$

In this section, we extend the three-party impossibility result to the general case using a standard player-partitioning argument.

Theorem 8 *Fix $n \geq 3$ and t_a, t_c with $t_c > 0$ and $2t_a + \min(t_a, t_c) \geq n$. There is no n-party protocol achieving broadcast in the presence of a (t_a, t_c)-adversary.*

Proof: Partition $\mathcal{P} = \{P_1, \ldots, P_n\}$ into three sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ such that $\mathcal{P}_1$ contains $\min(t_a, t_c, n-2)$ parties (note that $\mathcal{P}_1$ is non-empty), and $\mathcal{P}_2, \mathcal{P}_3$ each contain at least one and at most t_a parties. Such a partition is possible given the constraints.

Assume the existence of an n-party protocol Π (for $n \geq 3$) achieving broadcast in the presence of a (t_a, t_c)-adversary. We construct a three-party broadcast protocol Π' by having P_i (for $i \in \{1, 2, 3\}$) internally simulate an execution of Π by the parties in $\mathcal{P}_i$, in the obvious way. (The public key of each party in Π' now consists of multiple public keys of parties in Π. But nothing in the definition of the PKI model requires public keys of parties to have any particular form.)

Π' is secure against an adversary who actively corrupts P_1, as this corresponds to actively corrupting at most $\min(t_a, t_c, n-2) \leq t_a$ parties in Π. Protocol Π' is also secure against an adversary who actively corrupts P_2 and compromises the secret key of P_1, as this corresponds to actively corrupting at most t_a parties and compromising the secret key of at most $\min(t_a, t_c, n-2) \leq t_c$ parties in Π. Similarly, Π' is secure against an adversary who actively corrupts P_3 and compromises the secret key of P_1. But this contradicts Lemma 4. ∎

4.3 Impossibility of Broadcast with a Threshold Adversary

We now turn to the case of a threshold adversary. Recall that in this setting the exact values of t_a and t_c used by the adversary are not known; we only know that they satisfy $2t_a + \min(t_a, t_c) < n$ (and we allow $t_c = 0$). In what follows, we show that broadcast is impossible if $n \notin \{2, 3, 4, 5, 6, 8, 9, 12\}$. For the "exceptional" values of n, we demonstrate feasibility in Section 5.

Theorem 9 *If $n \leq 2\left\lfloor \frac{n-1}{3} \right\rfloor + \left\lfloor \frac{n-1}{2} \right\rfloor$, then there is no n-party protocol achieving broadcast in the presence of a threshold adversary. (The given bound on n holds for all $n > 1$ except $n \in \{2, 3, 4, 5, 6, 8, 9, 12\}$.)*

Proof: Once again, we use a player-partitioning argument. Partition $\mathcal{P} = \{P_1, \ldots, P_n\}$ into three non-empty sets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ such that $\mathcal{P}_1$ contains at most $\left\lfloor \frac{n-1}{2} \right\rfloor$ parties, and $\mathcal{P}_2, \mathcal{P}_3$ contain at most $\left\lfloor \frac{n-1}{3} \right\rfloor$ parties. Such a partition is possible given the bound on n.

Assume the existence of an n-party protocol Π achieving broadcast in the presence of a threshold adversary. We construct a three-party broadcast protocol Π' by having P_i (for $i \in \{1, 2, 3\}$) internally simulate an execution of Π by the parties in $\mathcal{P}_i$, in the obvious way.

Π' is secure against an adversary who actively corrupts P_1, as this corresponds to actively corrupting at most $t_a = \left\lfloor \frac{n-1}{2} \right\rfloor$ parties in Π (and so $2t_a + \min(t_a, t_c) = 2t_a < n$). Protocol Π' is also secure against an adversary who actively corrupts P_2 and compromises the secret key of P_1, as this corresponds to actively corrupting at most $t_a = \left\lfloor \frac{n-1}{3} \right\rfloor$ parties and compromising the secret key of at most $t_c = \left\lfloor \frac{n-1}{2} \right\rfloor$ parties in Π (and so $2t_a + \min(t_a, t_c) = 3t_a < n$). Similarly, Π' is secure against an adversary who actively corrupts P_3 and compromises the secret key of P_1. But this contradicts Lemma 4. ∎

5 Handling the Exceptional Values of n

We refer to $\{2, 3, 4, 5, 6, 8, 9, 12\}$ as the set of *exceptional values* for n. (These are the only positive, integer values of n for which Theorem 9 does not apply.) We show for all the exceptional values of n an n-party protocol achieving broadcast in the presence of a threshold adversary. Recall that a threshold adversary can choose to corrupt t_a parties, and compromise the secret keys of an additional t_c parties, for any t_a, t_c satisfying $2t_a + \min(t_a, t_c) < n$.

The cases $n \in \{2, 3, 4\}$. These cases are all handled relatively easily. For $n = 2$ we must have $t_a = 0$ and so broadcast is trivial to achieve. For $n = 3$ we can just run any protocol for authenticated broadcast; either $(t_a, t_c) = (1, 0)$, in which case no honest parties' signatures can be forged and so broadcast is achieved, or else $t_a = 0$, in which case the ability to forge signatures is of no use to the adversary (recall we assume that all point-to-point links are authenticated). For $n = 4$ we have $t_a \leq 1$ and $t_c \leq n$. Since $t_a < n/3$, here we may use any protocol for standard (unauthenticated) broadcast. Summarizing:

Theorem 10 *For $n \in \{2, 3, 4\}$ there is an n-party protocol achieving broadcast in the presence of a threshold adversary.*

The case $n = 5$. Here we may rely on the result of Fitzi et al. [4], which we re-state for convenience using our notation:

Theorem 11 ([4]) *Fix any n, t, T with $T < n/2$ and $T + 2t < n$. Then there exists an n-party protocol that achieves broadcast in the presence of either a $(T, 0)$-adversary or a $(t, n-t)$-adversary.*

Setting $t = 1$ and $T = 2$, this theorem implies a 5-party protocol that achieves broadcast in the presence of a threshold adversary. (Indeed, for a threshold adversary attacking a 5-party protocol either $t_a = 2$ and $t_c = 0$ or $t_a \leq 1$ and $t_c \leq n$.)

Corollary 12 *There is a 5-party protocol achieving broadcast in the presence of a threshold adversary.*

The case $n = 6$. For $n = 6$ we reduce to the case $n = 5$. Specifically, take the following protocol Π: The dealer signs its input b, sends b and its signature to everyone else, and terminates with output b. Each of the remaining parties then broadcasts what they received from the dealer using the 5-party protocol from Corollary 12. Let a *b-signature* be a valid signature of the dealer on the bit b. Each party P_i (other than the dealer) decides on its output as follows: If there is only one value b for which P_i has received a b-signature, then P_i outputs b. Otherwise, P_i outputs the value b for which the number of broadcast b-signatures is maximized. (Output 0 in case of a tie.)

For a threshold adversary either $t_a = 2$ and $t_c \leq 1$, or $t_a \leq 1$ and $t_c \leq 6$. If the dealer is honest and not compromised, clearly all honest parties will output the dealer's input b. In any other case, the corruption pattern among the five other parties (i.e., excluding the dealer) must satisfy the threshold condition (for those $n = 5$ parties); thus, broadcast is achieved in the second stage and agreement follows. Validity when the dealer is honest but compromised is implied by the fact that a majority of the five other parties are honest. We conclude:

Theorem 13 *There is a 6-party protocol achieving broadcast in the presence of a threshold adversary.*

The cases $n \in \{8, 9\}$. We first prove the following claim:

Claim 14 *Fix any n, t, T with $T < n/2$ and $T + 2t < n$. Then there exists an n-party protocol with the following properties:*

- *If the dealer is honest and not compromised, and at most T other parties are corrupted and all other parties' keys are compromised, then all honest parties output the dealer's input. (I.e., validity holds.)*
- *If the dealer is honest and compromised, and at most t parties are corrupted and all other parties' keys are compromised, then all honest parties output the dealer's input. (I.e., validity holds.)*
- *If T parties are corrupted and all parties' keys are compromised, weak agreement holds among the honest parties. (That is, if any honest party outputs $b \in \{0, 1\}$ then every other honest party outputs either b or $\perp$.)*

Proof: We use the protocol from [4]; only our analysis differs. Let b denote the dealer D's input. In the protocol only D signs anything; let a *b-signature* be a valid signature of D on the bit b. The protocol proceeds as follows:

Round 1: D computes $\sigma_D \leftarrow \mathsf{Sign}_{sk_D}(b)$ and sends (b, σ_D) to all parties.

Round 2: Each party P_i (other than D) does the following: let (b, σ_D) be the message that P_i received from D in the previous round. Echo this message to all other parties.

Output determination: Every party P_i (including D) decides as follows:

1. If P_i received b and a b-signature from D and at least $n - t - 1$ other parties, then output b.
2. Otherwise, if P_i received b and a b-signature from D and at least $n - T - 1$ other parties, and no $\bar{b}$-signature, then output b.
3. Otherwise, output $\perp$.

Note that key compromise is irrelevant except for possible compromise of D. The first two claims of the theorem are immediate. For the final claim, say P_i outputs a bit b and consider another honest party P_j. (Note that either P_i or P_j may be the dealer.) If P_i outputs b because of condition (1) this means P_i received b-signatures from at least $n - t$ parties overall. But then P_j received b-signatures from at least $n - t - T$ parties, and hence $\bar{b}$-signatures from at most $t + T < n - t$ parties overall. Moreover, P_j received at least one b-signature (from P_i). So P_j cannot output $\bar{b}$.

On the other hand, say P_i outputs b because of condition (2). We need to show that P_j does not output $\bar{b}$ by condition (2). (The case of condition (1) is already taken care of in the previous paragraph.) But since P_i received b-signatures from at least $n - T$ parties overall, P_j received a b-signature from at least $n - 2T \geq 1$ party. ■

Returning to the case of $n = 8$ and a threshold adversary, let Π_{bc} denote the protocol implied by Claim 14 for $n = 8$, $T = 3$, and $t = 2$ and consider the following 8-party protocol:

Round 1: D computes $\sigma_D \leftarrow \mathsf{Sign}_{sk_D}(b)$ and sends (b, σ_D) to all parties.

Round 2: Each P_i (other than D) does the following: let (b, σ_D) be the message P_i received from D. Compute $\sigma_i \leftarrow \mathsf{Sign}_{sk_i}(b \| \sigma_D)$ and broadcast $(b, \sigma_D, i, \sigma_i)$ using Π_{bc}.

Round 3: (c, σ_D) is a *valid c-tuple for D* if $\mathsf{Vrfy}_{pk_D}(c, \sigma_D) = 1$. The tuple $(c, \sigma_D, j, \sigma_j)$ is a *valid c-tuple for P_j* if (1) $\mathsf{Vrfy}_{pk_D}(c, \sigma_D) = 1$; (2) $P_j \neq D$; and (3) $\mathsf{Vrfy}_{pk_j}(c \| \sigma_D, \sigma_j) = 1$. A *valid tuple* is a valid 0-tuple or 1-tuple for some P_j (including possibly D). Each party sends to every other party all the valid tuples it received in previous rounds.

Output determination: Each P_i decides on its output as follows:

1. If P_i received valid b-tuples for at least six parties in rounds 1 and 2, then output b.
2. Otherwise, if P_i received valid b-tuples for at least five parties in rounds 1 and 2, and valid $\bar{b}$-tuples for at most four parties in round 3, output b.
3. Otherwise, if P_i received valid b-tuples for at least four parties in rounds 1 and 2, and no valid $\bar{b}$-tuples in round 3, output b.
4. Otherwise, output $\perp$.

We show that Π achieves *weak* broadcast; results of [5, 4] then imply a protocol for broadcast.

Theorem 15 *The 8-party protocol described above achieves weak broadcast in the presence of a threshold adversary.*

Proof: For a threshold adversary, either $t_a = 3, t_c \leq 1$ or $t_a \leq 2, t_c \leq n$. We consider these cases separately.

Say $(t_a, t_c) = (3, 1)$. By Claim 14, Π_{bc} achieves validity when an honest and non-compromised party acts as the dealer, and weak agreement otherwise. If D with input b is honest and not compromised, then each honest party receives a valid b-tuple for D in round 1, and for at least three other honest and non-compromised parties in round 2; it receives no valid $\bar{b}$-tuples in round 3. If D is honest and compromised, then each honest party receives a valid b-tuple from D in round 1, and for at least four other parties in round 2; moreover, it obtains valid $\bar{b}$-tuples for at most four parties in round 3. Thus, validity always holds. To prove weak agreement, say honest P_i outputs b and consider another honest party P_j. If P_i outputs b due to condition (1) then P_i has received valid b-tuples for at least five parties in round 2; weak agreement of Π_{bc} then implies that P_j has received $\bar{b}$-tuples for at most two parties in round 2, and thus at most three in rounds 1 and 2, and so cannot output $\bar{b}$. If P_i outputs b due to condition (2), then P_i has received valid b-tuples for at least five parties in rounds 1 and 2, which it sends to P_j in round 3. So P_j does not output $\bar{b}$. A similar argument holds if P_i outputs b due to condition (3).

Assume next that $(t_a, t_c) = (2, n-2)$. By Claim 14 we have that Π_{bc} achieves validity whenever an honest party acts as the dealer, and weak agreement otherwise. If D is honest with input b, then each honest party receives b-tuples from at least six parties in rounds 1 and 2 and so outputs b. Weak agreement holds by the same argument as above. ∎

Corollary 16 *There is an 8-party protocol achieving broadcast in the presence of a threshold adversary.*

For the case of $n = 9$, we reduce to the case $n = 8$. Specifically, take the following 9-party protocol: The dealer signs its input b, sends b and its signature to everyone else, and terminates with output b. Each of the remaining parties then broadcasts what they received from the dealer using the 8-party protocol implied by Corollary 16. (D is not involved in these executions.) Let a *b-signature* be a valid signature by D on the bit b. Each P_i (other than the dealer) decides on its output as follows: If there is only one value b for which P_i received a b-signature, then P_i outputs b. Otherwise, P_i outputs the value b for which the number of broadcast b-signatures is maximized (or 0 in case of a tie).

For a threshold adversary either $t_a = 4, t_c = 0$, or $t_a = 3, t_c \leq 2$, or otherwise $t_a \leq 2, t_c \leq n$. If the dealer is honest and not compromised, clearly all honest parties will output the dealer's input b. In any other case, the corruption pattern among the remaining parties (i.e., excluding the dealer) must satisfy the threshold condition for those 8 parties; thus, broadcast is achieved in the second stage and agreement follows. When the dealer is honest but compromised we have $t_c \geq 1$ and so $t_a \leq 3$; validity is then implied by the fact that a majority of the 8 parties other than the dealer are honest.

Theorem 17 *There is a 9-party protocol achieving broadcast in the presence of a threshold adversary.*

The case $n = 12$. The final case is handled using a variant of the protocol from Figure 1:

Round 1: D computes $\sigma_D \leftarrow \mathsf{Sign}_{sk_D}(b)$ and sends (b, σ_D) to every party.

Round 2: Each party P_i (other than D) does the following: let (b', σ_D) be the message that P_i received from D in the previous round. Compute $\sigma_i \leftarrow \mathsf{Sign}_{sk_i}(b' \| \sigma_D)$ and send $(b', \sigma_D, i, \sigma_i)$ to all other parties.

Round 3: $(c, \sigma_D, j, \sigma_j)$ is a *valid c-tuple for P_j* if (1) $\mathsf{Vrfy}_{pk_D}(c, \sigma_D) = 1$; (2) $P_j \neq D$; and (3) $\mathsf{Vrfy}_{pk_j}(c \| \sigma_D, \sigma_j) = 1$. A *valid tuple* is a valid 0-tuple or 1-tuple for some P_j. Each party P_i (other than D) sends to every other party all the valid tuples it received in the previous round.

Output determination: D outputs b. For every other P_i, let (b', σ_D) be the message P_i received from D in round 1. P_i outputs b' if $\mathsf{Vrfy}_{pk_D}(b', \sigma_D) = 1$ and one of the following conditions holds:

1. P_i received valid b'-tuples for at least 8 parties in round 2;
2. P_i received valid b'-tuples for at least 7 parties in round 2, and valid $\bar{b}'$-tuples for at most 6 parties in round 3;
3. P_i received valid b'-tuples for at least 6 parties in round 2, and no valid $\bar{b}'$-tuples in round 3.

Theorem 18 *The 12-party protocol described above achieves weak broadcast in the presence of a threshold adversary.*

Proof: For a threshold adversary, either $t_a = 5, t_c \leq 1$, or $t_a = 4, t_c \leq 3$, or $t_a = 3, t_c \leq n$.

Say D is honest with input b. Then all honest parties receive b and a valid signature on b from D in round 1. If $t_a = 3$ then all honest parties receive valid b-tuples for at least 8 parties in round 2.

If $(t_a, t_c) = (4, 3)$ then all honest parties receive valid b-tuples for at least 7 parties in round 2, and receive valid $\bar{b}$-tuples for at most 6 parties in round 3. If $(t_a, t_c) = (5, 1)$ then all honest parties receive valid b-tuples for at least 6 parties in round 2, and receive valid $\bar{b}$-tuples for no parties in round 3. Thus, validity holds in every case.

Assume D is corrupt. To prove weak agreement, say an honest party P_i outputs b and consider another honest party P_j. If P_i outputs b due to condition (1) then P_i received valid b-tuples for at least 8 parties in round 2; thus, P_j receives at least $8 - 4 = 4$ valid b-tuples in round 2 (recall that D is corrupt, so at most 4 of the remaining parties can be corrupt) and so at most 7 valid $\bar{b}$-tuples in that round. Since P_i sends its valid tuples to P_j, we see that P_j receives at least 8 valid b-tuples in round 3 and so cannot output $\bar{b}$.

If P_i outputs b due to condition (2) then we need to show that P_j cannot output $\bar{b}$ due to conditions (2) or (3). In this case P_i received valid b-tuples for 7 parties in round 2, but since it sends these to P_j we see that P_j cannot output $\bar{b}$. The case where P_i outputs b due to condition (3) is analogous. ∎

Once again, using [5, 4] this gives:

Corollary 19 *There is a 12-party protocol achieving broadcast in the presence of a threshold adversary.*

Acknowledgments. The authors would like to thank the anonymous reviewers for several helpful suggestions.

References

[1] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus. In *30th Proc. 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 410–415. IEEE, 1989.

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

[3] M. Fitzi. *Generalized Communication and Security Models in Byzantine Agreement.* PhD thesis, ETH Zurich, 2002.

[4] Matthias Fitzi, Thomas Holenstein, and Jürg Wullschleger. Multi-party computation with hybrid security. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 419–438. Springer, May 2004.

[5] Matthias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503. ACM, 2000.

[6] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[7] Anuj Gupta, Prasant Gopal, Piyush Bansal, and Kannan Srinathan. Authenticated Byzantine generals in dual failure model. In *ICDCN*, volume 5935 of *Lecture Notes in Computer Science*, pages 79–91. Springer, 2010.

[8] Martin Hirt and Vassilis Zikas. Player-centric Byzantine agreement. In *38th Intl. Colloquium on Automata, Languages, and Programming (ICALP), Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 281–292. Springer, 2011.

[9] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. Programming Languages and Systems*, 4:382–401, 1982.

[10] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *10th Proc. 10th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 51–59. ACM, 1991.

[11] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27:228–234, 1980.