Authenticated Broadcast with a Partially Compromised Public-Key Infrastructure

S. Dov Gordon, Jonathan Katz*, Ranjit Kumaresan, and Arkady Yerukhimovich

Dept. of Computer Science
University of Maryland
{gordon, jkatz, ranjit, arkady}@cs.umd.edu

Abstract. Given a public-key infrastructure (PKI) and digital signatures, it is possible to construct broadcast protocols tolerating any number of corrupted parties. Almost all 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 who 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.) We also show that if t_c, t_a are not fixed, but are only guaranteed to satisfy the bound above, then broadcast is impossible to achieve except for a few specific values of n; for these "exceptional" values of n, we demonstrate a broadcast protocol. Taken together, our results give a complete characterization of this problem.

1 Introduction

Although Public Key Infrastructures (PKI) are heavily used in the design of cryptographic protocols, in practice they are often subject to key leakage, cryptanalysis and side channel attacks. Such attacks can make the resulting construction insecure as it's security depends on the security of the PKI. In this work, we consider the security that can be guaranteed even with a compromised PKI. In particular, we study the problem of *broadcast* in a setting where some of the honest players' signatures can be forged.

Broadcast protocols allow a designated player (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,

* Work done in part while visiting IBM. 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.

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 [12, 8] show that broadcast (and, equivalently, Byzantine agreement) 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 set-up is required. The most commonly studied set-up assumption is the existence of a public-key infrastructure (PKI) such that each party P_i has a public signing key pk_i that is known to all other parties (in addition to the cryptographic assumption that secure digital signatures exist). In this model, broadcast is possible for any $t < n$ [12, 8, 1].

With few exceptions [3, 5] (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. However, 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 when 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 *satisfy the threshold condition* with respect to some total number of parties n if $2t_a + \min(t_a, t_c) < n$. We show:

1. For any n and any t_a, t_c satisfying the threshold condition with respect to n, there is an efficient (i.e., polynomial in n) 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 $t_c = 0$; here standard results like [1] 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. (The positive result mentioned above relies on two different protocols, depending on whether $t_a \leq t_c$.) 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.

Motivating the problem. Compromised parties are most naturally viewed as honest parties whose secret (signing) keys have been obtained somehow by the adversary. E.g., perhaps an adversary was able to hack into an honest user's system and obtain their secret key, but subsequently the honest party's computer was re-booted and now behaves honestly. Exactly this scenario is addressed by *proactive* cryptosystems [11] and leakage-resilient cryptosystems [2], though in somewhat different contexts.

We remark, however, that our model is meaningful even if such full-scale compromise of honest users' secret keys is deemed unlikely. Specifically, our work provides important guarantees whenever there is a possibility that an honest user's signature might be *forged* (whether or not the adversary learns the user's actual secret key). Signature forgery can potentially occur due to cryptanalysis, poor implementation of cryptographic protocols [9, 10], or side-channel attacks [6, 7]. In all these cases, it is likely that 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. [5] also consider broadcast protocols providing agreement and validity for honest-behaving parties whose secret keys have been compromised. Our results improve upon theirs in several respects. First, we construct *efficient* protocols whenever $2t_a + \min(t_a, t_c) < n$, whereas the protocols presented in the work of Gupta et al. have message complexity exponential in n. Although Gupta et al. [5] 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.[1] 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. [3] 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. In our notation, their work handles the two extremes $t_a < n/3$, $t_c = n$ and $t_a < n/2$, $t_c = 0$. 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 provides a formal definition of broadcast in our setting. In Section 3 we show that for every n, t_a, t_c satisfying the threshold condition, there exists an efficient broadcast protocol. We show our impossibility results in Section 4: namely, broadcast is impossible whenever t_a, t_c do not satisfy the threshold condition (except when t_c is fixed to 0), and (other than for the exceptional values of n) there does not exist a single, 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, in fact all the exceptional values

[1] In [5], 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.

of n are small and so are more likely to arise in practice. Furthermore, dealing with these exceptional values is, in some sense, the most technically challenging part of our work.

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 party P_i runs a key-generation algorithm Gen (specified by the protocol) to obtain public key pk_i along with the 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 dependent 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 players may be *compromised*; if P_i is compromised then the adversary $\mathcal{A}$ is given that P_i's secret key sk_i. We stress that compromised players follow the protocol as instructed: the only difference is that $\mathcal{A}$ is now 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.

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. It is worth noting that this is a common assumption in many previous works relating to information-theoretic broadcast, byzantine agreement, and secure computation: for each of these problems, shared cryptographic keys cannot be used to ensure authenticated and/or secret channels against an all-powerful adversary. In practice, authenticated channels would be guaranteed using pairwise symmetric keys (that are less easily compromised or cryptanalyzed than signing keys), or could also be ensured via physical means in small-scale networks. Note that 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 distinguished dealer $D \in \mathcal{P}$ holds an initial input M, achieves* **broadcast** *if the following hold:*

Agreement *All honest parties output the same value.*

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

We stress that "honest" in the above includes those honest parties who have been compromised.

Although the above refers to an arbitrary input M for the dealer, we assume for simplicity that the dealer's input is a single bit. 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 1. *Fix n, t_a, t_c with $2t_a + \min(t_a, t_c) < n$. Then there exists a protocol achieving broadcast in the presence of a (t_a, t_c)-adversary.*

The case of $t_a \leq t_c$ is easy: $t_a \leq t_c$ implies $3t_a < n$ and the parties can thus run a standard (unauthenticated) broadcast protocol [12, 8] where the PKI is not used at all. (In this case, it makes no difference whether honest players are compromised or not.) The challenge is to design a protocol for $t_c < t_a$, and we deal with this case for the remainder of this section.

Let DS refer to the Dolev-Strong protocol [1] that achieves broadcast with a PKI, in the usual sense (i.e., when no honest parties' keys can be compromised), for any $t < n$ corrupted parties. (The Dolev-Strong protocol is reviewed in Appendix A.) We say that P_i calls an execution of the DS protocol *dirty* if P_i receives valid signatures by the dealer on two different messages, or never receives any valid signed messages from the dealer; i.e., if P_i detects that the dealer is either corrupted or compromised. P_i declares the execution *clean* otherwise. The following is easy to prove (the proof is omitted due to lack of space):

Lemma 1. *Consider an execution of protocol DS in the presence of t_a adversarial parties and t_c compromised honest parties, where $t_a + t_c < n$. Then:*

1. *All honest parties agree on whether an execution of DS is clean or dirty.*
2. *Agreement holds. (I.e., the outputs of all honest players are identical.)*
3. *If the dealer is honest and has not been compromised, then validity holds (i.e., all honest parties agree on the dealer's input) and the execution is clean. If the dealer is honest and the execution is clean, then validity also holds.*

Thus, DS fails to satisfy Definition 1 only when the dealer is *honest* but *compromised*. Our protocol (cf. Figure 1) guarantees validity even in this case (while leaving the other cases unaffected).

Protocol 1

Inputs: Let D be the dealer, with input bit b.

Computation:

1. D sends b to all other players. Let b_i be the value received by P_i from D in this step (if the dealer sends nothing to P_i, then b_i is taken to be some default value).
2. In parallel, each party P_i acts as the dealer in an execution of $\mathsf{DS}(b_i)$ (the original dealer D runs $\mathsf{DS}(b)$). We let $|\mathsf{CLEAN}_0|$ (resp., $|\mathsf{CLEAN}_1|$) denote the number of executions of DS that are both *clean* and result in output 0 (resp., 1).

Output: If $|\mathsf{CLEAN}_0| \geq |\mathsf{CLEAN}_1|$ then all parties output 0; otherwise, all parties output 1.

Fig. 1. A broadcast protocol for $t_c < t_a$ and $2t_a + t_c < n$.

Theorem 2. *Let $\mathcal{A}$ be a (t_a, t_c)-adversary with $t_c < t_a$ and $2t_a + t_c < n$. Then Protocol 1 achieves broadcast in the presence of $\mathcal{A}$.*

Proof. We prove agreement and validity. Note that $n > t_a + t_c$, so Lemma 1 applies.

Agreement: By Lemma 1, the output of each honest player is the same in every execution of DS in step 2, and all honest parties agree on whether any given execution of DS is clean or dirty. So all honest players agree on $|\mathsf{CLEAN}_0|$ and $|\mathsf{CLEAN}_1|$, and agreement follows.

Validity: Assume the dealer is honest (whether compromised or not). Letting t_h denote the number of honest, non-compromised players, we have $t_h+t_a+t_c = n > 2t_a + t_c$ and so $t_h > t_a$. Thus, there are t_h honest and non-compromised dealers in step 2 of Protocol 1, and (since D is honest) each of these runs $\mathsf{DS}(b)$ where b is the initial input of D. By Lemma 1, all honest players output b in (at least) these t_h executions, and each of these t_h executions is clean. Furthermore, there can be at most t_a clean executions resulting in output $1 - b$, as only adversarial players will possibly run $\mathsf{DS}(1 - b)$ in step 2. The majority value output by the honest players is therefore always equal to the original dealer's input b.

4 Impossibility Results

In this section we show two different impossibility results. First, we show that there is no protocol achieving broadcast in the presence of a (t_a, t_c)-adversary when $n \leq 2t_a + \min(t_a, t_c)$ and $t_c > 0$, thus proving that Theorem 1 is tight. We then consider the case when t_a, t_c are not fixed, but instead all that is guaranteed is that $2t_a + \min(t_a, t_c) < n$. (In the previous section, unauthenticated broadcast was used to handle the case $t_a \leq t_c$ and Protocol 1 assumed $t_c < t_a$. Here we seek a *single* protocol that handles both cases.) We show that in this setting, broadcast is impossible for almost all n.

Fig. 2. A mental experiment involving a four-node network.

4.1 The Three-Player Case

We first present a key lemma that will be useful for the proofs of both results described above. For this, define a general adversary $\mathcal{A}$ as follows:

Definition 2. *Let $\mathcal{S}$ be a set of pairs $\{(S_a^1, S_c^1), (S_a^2, S_c^2), \ldots\}$ where $S_a^i, S_c^i \subset \{P_1, \ldots, P_n\}$. An $\mathcal{S}$-*adversary *can choose any i, and actively corrupt any subset of the players in S_a^i and additionally compromise the secret keys of any subset of the players in S_c^i.*

We restrict our attention to the case of three parties (named A, B, and C) and $\mathcal{S}$ defined as follows:

$$\mathcal{S} = \left\{ \begin{array}{c} (\{A\}, \emptyset) \\ (\{B\}, \{A\}) \\ (\{C\}, \{A\}) \end{array} \right\}. \tag{1}$$

Lemma 2. *In the presence of an $\mathcal{S}$-adversary, for $\mathcal{S}$ defined as above, there does not exist a protocol achieving broadcast for dealer A.*

Proof. Suppose, towards a contradiction, that there exists a protocol Π for computing broadcast in the presence of an $\mathcal{S}$-adversary when A is the dealer. Let Π_A, Π_B, Π_C denote the code specified by Π for players A, B, and C, respectively.

Consider an experiment in which four machines are arranged in a rectangle (see Figure 2). The top left and top right nodes will run Π_B and Π_C, respectively. The bottom left node will run Π_A using input 1, and the bottom right node will run Π_A using input 0. Public and secret keys for A, B, and C are generated honestly, and both executions of Π_A use the same keys.

Claim. In the experiment of Figure 2, Π_B outputs 1.

Proof. Consider an execution in the real network of three players, in the case where A holds input 1 and the adversary corrupts C and compromises the secret key of A. The adversary then simulates the right edge of the rectangle from Figure 2 while interacting with the (real) honest players A and B (running the code for $\Pi_A(1)$ and Π_B, respectively). That is, every time the corrupted player C receives a message from B the adversary forwards this message to its internal copy of Π_C, and every time C receives a message from A the adversary forwards this message to its internal copy of $\Pi_A(0)$. Similarly, any message sent by Π_C

to Π_B is forwarded to the real player B, and any message sent by $\Pi_A(0)$ to $\Pi_A(1)$ is forwarded to the real player A. (Messages between Π_C and $\Pi_A(0)$ are forwarded internally.) This defines a legal $\mathcal{S}$-adversary. If Π is a secure protocol, validity must hold and so B in the real network (and hence Π_B in the mental experiment) must output 1.

Claim. In the experiment of Figure 2, Π_C outputs 0.

The proof is the same as above.

Claim. In the experiment of Figure 2, Π_B and Π_C output the same value.

Proof. Consider an execution in the real network of three players when the adversary corrupts A (and does not further compromise anyone). The adversary then simulates the bottom edge of the rectangle when interacting with the real players B and C, in the obvious way. Since this defines a legal $\mathcal{S}$-adversary, security of Π implies that agreement must hold between B and C in the real network and so the outputs of Π_B and Π_C must agree in the mental experiment.

The three claims are contradictory, and so we conclude that no secure protocol Π exists.

We remark that impossibility holds even if we relax our definition of broadcast and allow agreement/validity to fail with negligible probability.

4.2 Impossibility of Broadcast for $2t_a + \min(t_a, t_c) \geq n$

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

Proof. We prove the theorem by demonstrating that a broadcast protocol Π secure in the presence of a (t_a, t_c)-adversary with $2t_a + \min(t_a, t_c) \geq n$, yields a protocol Π' for 3-player broadcast in the presence of an $\mathcal{S}$-adversary for $\mathcal{S}$ as defined in the previous section. Using Lemma 2, this shows that such a protocol Π cannot exist. In fact, we show this even assuming the dealer is fixed in advance.

Assume that such a protocol Π exists. We construct a protocol Π' for 3-player broadcast by having each player simulate a subset of the players in the n-player protocol Π. The simulation proceeds in the obvious way, by having each of the 3 players run the code of the parties they simulate in Π. They forward any messages sent by the simulated parties to the player simulating the destination party, who uses these as incoming messages for his simulated players. To provide a PKI for the simulated protocol we view the keys of each of the 3 players as consisting of multiple keys. Player A's public key is $PK_A = (pk_1, \ldots, pk_a)$ and his secret key is $SK_A = (sk_1, \ldots, sk_a)$ for some number of simulated players a. The players in the 3-player protocol determine their outputs from the outputs of the players they simulate. If all players simulated by A output the same value b in the simulated protocol, then A outputs b. Otherwise, A outputs a special value $\perp$. Note that an adversarial player can only simulate adversarial players and an honest but compromised player can only simulate compromised players since the adversary learns all the secret keys of player A's simulated players when A's key is compromised.

We let A simulate a set of at most $\min(t_a, t_c)$ players, including the dealer, and let B and C each simulate at most t_a players. Since $2t_a + \min(t_a, t_c) \geq n$, it is possible to do this in such a way that each of the n original players is simulated by one of A, B, or C. We now consider each of the three allowed types of corruption for the adversary $\mathcal{A}$ as per Definition 2, and demonstrate that the corresponding corruption in the n-player protocol is also "legal": that is, we demonstrate that the allowed actions for $\mathcal{A}$ translate into adversarial actions for which the non-faulty players in Π terminate correctly, achieving broadcast in the simulated n-player protocol. This implies a secure 3-player broadcast protocol in the presence of $\mathcal{A}$.

Recall that, by assumption, Π is secure against a (t_a, t_c)-adversary; as long as no more than t_a players are corrupt, and no more than t_c are compromised, Π satisfies the requirements of authenticated broadcast. If $\mathcal{A}'$ chooses the pair $(\{A\}, \emptyset)$, all players simulated by A in Π' are corrupt and the players simulated by B and C are honest and non-compromised. Since, $\min(t_a, t_c) \leq t_a$, this is an allowed corruption for a (t_a, t_c)-adversary, and Π executes correctly implying that Π' terminates with the correct output. Next, if $\mathcal{A}'$ chooses $(\{B\}, \{A\})$ this will result in a $(t_a, \min(t_a, t_c))$ corruption. Since $\min(t_a, t_c) \leq t_c$, this corruption type is also permitted in Π, and Π' executes correctly. Finally, the corruption type $(\{C\}, \{A\})$ is handled identically to that of $(\{B\}, \{A\})$. Since we proved in Lemma 2 that no such protocol Π' exists, this proves the theorem.

4.3 Impossibility of Broadcast with a Threshold Adversary

We now turn to the case of the 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 do allow $t_c = 0$). In what follows, we show that secure 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 4. *If $n \leq 2\left\lfloor \frac{n-1}{3} \right\rfloor + \left\lfloor \frac{n-1}{2} \right\rfloor$, then there does not exist a secure broadcast protocol for n players in the presence of a threshold adversary. (Note that $n \leq 2\left\lfloor \frac{n-1}{3} \right\rfloor + \left\lfloor \frac{n-1}{2} \right\rfloor$ for all $n > 1$ except $n \in \{2, 3, 4, 5, 6, 8, 9, 12\}$.)*

Proof. Assume there exists a protocol Π for n satisfying the stated inequality. We show that this implies a protocol Π' for broadcast with 3 players in the presence of the adversary $\mathcal{A}$ from Definition 2. By Lemma 2, we conclude that Π cannot exist. In fact, we show this even assuming the dealer is fixed in advance.

We construct Π' using a player simulation argument as in the previous section. Let A simulate a set of at most $\left\lfloor \frac{n-1}{2} \right\rfloor$ players, and including the dealer. B and C each simulate at most $\left\lfloor \frac{n-1}{3} \right\rfloor$ players and at least one player. By the stated inequality, it is possible to do this in such a way that A, B, and C simulate all n players. We now show that the three allowed types of corruption for $\mathcal{A}$ (in the

3-party network) are also allowed corruption patterns for the n-player threshold adversary $\mathcal{A}'$

If $\mathcal{A}$ corrupts A, this corresponds to corruption of $\lfloor \frac{n-1}{2} \rfloor$ players in Π (and no compromised players). Since $2\lfloor \frac{n-1}{2} \rfloor < n$, this is a legal corruption pattern for a threshold adversary and Π should remain secure. If $\mathcal{A}$ corrupts B and compromises A, this corresponds to $t_a = \left\lfloor \frac{n-1}{3} \right\rfloor$ players and $t_c = \lfloor \frac{n-1}{2} \rfloor$ players in Π. Since $2 \left\lfloor \frac{n-1}{3} \right\rfloor + \min\{\left\lfloor \frac{n-1}{3} \right\rfloor, \lfloor \frac{n-1}{2} \rfloor\} = 3 \left\lfloor \frac{n-1}{3} \right\rfloor < n$, this is again a legal corruption pattern for a threshold adversary and Π should remain secure. The case when C is corrupted and A is compromised is exactly analogous.

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 4 does not apply.) We show for any exceptional value of n a broadcast protocol that is secure against any threshold adversary. Designing protocols in this setting is more difficult than in the setting of Section 3, since the honest parties are no longer assumed to "know" whether $t_a \leq t_c$.

Our protocol, which we refer to as $\mathsf{authLSP}$, is an authenticated version of the exponential protocol of Lamport et al. [8]; see Figure 3. Although the message complexity of this protocol is exponential in the number of players, the maximum number of players considered here is 12. In this full version of this work [4], we provide a more efficient protocol under the assumption that there is at least one honest and uncompromised player.

We say a message M is *valid* if it has the form $(v, s_{P_1}, \ldots, s_{P_i})$, where all P_j's are distinct, the string s_{P_j} is a valid signature on $(v, s_{P_1}, \ldots, s_{P_{j-1}})$ relative to the verification key of P_j, and one of the s_{P_j} is the signature of the dealer. (We note that $\mathsf{authLSP}$ is defined recursively, and the criteria for deciding if a message is valid is defined with respect to the dealer of the *local* execution.) We also assume implicitly that each message has a tag identifying which execution it belongs to. These tags (together with uncompromised signatures) will prevent malicious players from substituting the messages of one execution for those of another execution. We refer to v as the *content* of such a message. When we say that an execution of $\mathsf{authLSP}$ satisfies agreement or validity (cf. Definition 1), we mean that the output is a valid message whose *content* satisfies these properties. We note that in the protocol $\mathsf{authLSP}$, it is possible for honest players to have invalid input. In this case, we change the definition of validity slightly to require that all honest players (including the dealer) output messages with content 0. Finally, we let $t_h = n - t_c - t_a$ denote the number of honest and uncompromised parties. One useful observation about threshold adversaries that we will repeatedly use is that when $t_a > \left\lfloor \frac{n-1}{3} \right\rfloor$, it follows that $t_h > t_a$.

The next two lemmas follow readily from [8]; we do not prove them here.

Lemma 3. *If $n > 3m$ and $m \geq t_a$, then* $\mathsf{authLSP}(m)$ *achieves validity and agreement.*

Protocol $\mathsf{authLSP}(m)$

Inputs: The protocol is parameterized by an integer m. Let D be the dealer with input M of the form $M = (v, s_{P_1}, \ldots, s_{P_i})$ with $0 \le i \le n$ (M is not necessarily valid).

Case 1: $m = 0$

1. If the content of M is not in $\{0, 1\}$, D sets $M = 0$.[a] D sends $M_d = (M, \mathsf{Sign}_{sk_D}(M))$ to all other players and outputs M_d.
2. Letting M_i denote the message received by P_i, P_i ouputs M_i.

Case 2: $m > 0$

1. If the content of M is not in $\{0, 1\}$, D sets $M = 0$. D sends $M_d = (M, \mathsf{Sign}_{sk_D}(M))$ to all other players and outputs M_d.
2. Let $\mathcal{P}' = \mathcal{P} \setminus \{D\}$. For $P_i \in \mathcal{P}'$, let M_i denote the message received by P_i from D. P_i plays the dealer in $\mathsf{authLSP}(m-1)$ for the rest of the players in $\mathcal{P}'$, using message M_i as its input.
3. Each P_i locally does the following: for each $P_j \in \mathcal{P}'$, let M_j be the output of P_i when P_j played the dealer in $\mathsf{authLSP}(m-1)$ in step 2. For each M_j, P_i sets value b_j as follows:

$$b_j = \begin{cases} \text{the content of } M_j & \text{if } M_j \text{ is valid} \\ \perp & \text{otherwise} \end{cases}$$

(We stress that the above also includes the output of P_i when he was dealer in step 2.) P_i computes $b^* = \mathsf{majority}(b_j)$. If there is no value in strict majority, P_i outputs 0.
4. P_i outputs the first valid message M_j (lexicographically) with content b^*.

---

[a] As mentioned in the text, we assume the dealer also includes the appropriate tag identifying which execution M belongs to. We do not mention this again going forward.

Fig. 3. Protocol $\mathsf{authLSP}$.

Lemma 4. *If the dealer D is honest and $n > 2t_a + m$, then* $\mathsf{authLSP}(m)$ *achieves validity and agreement.*

We now prove several additional lemmas about $\mathsf{authLSP}$.

Lemma 5. *If the dealer is honest and uncompromised, then* $\mathsf{authLSP}(m)$ *achieves validity and agreement for any m.*

Proof. Let D be the dealer with input that has content b_d. (Recall that if $b_d \notin \{0, 1\}$, then D switches his input for valid input with content $b_d = 0$.) It follows from the protocol description that D outputs a valid message with content b_d. Furthermore, when an honest player is dealer in the recursive call in step 2, it has input and output with content b_d. Therefore, when honest P_i computes

$\mathsf{majority}(\mathsf{b_j})$ in step 3 of $\mathsf{authLSP}$, it sets the value $b_i = b_d$. On the other hand, since D is honest and uncompromised, the adversary cannot produce a valid message with content $1 - b_d$ (recall that for a message to be valid, it must contain the signature of the dealer). It follows then that $b_j \neq 1 - b_d$ for all values used to compute majority in step 3. Validity and agreement follow.

Lemma 6. *If the dealer is honest and compromised, and $t_h > t_a$, then protocol* $\mathsf{authLSP}(m)$ *achieves validity and agreement for any m.*

Proof. It is easy to see that the lemma holds for $m = 0$. Let us assume the lemma holds for $\mathsf{authLSP}(m-1)$, and consider $\mathsf{authLSP}(m)$. If an honest and uncompromised player is the dealer in step 2 of $\mathsf{authLSP}(m)$ (i.e. in the recursive call to $\mathsf{authLSP}(m-1)$), then by Lemma 5 this run achieves validity and agreement. If an honest but compromised player is the dealer in step 2, then it still holds in the recursive execution that $t_h > t_a$, since the dealer is not counted in t_h, and all other players participate in the execution of $\mathsf{authLSP}(m-1)$; by the induction hypothesis this execution achieves validity and agreement on output b_d as well. It follows that in step 3 of $\mathsf{authLSP}(m)$, for each honest player P_i, at least $n - t_a - 1$ of the b_j values equal b_d and at most t_a of the b_j values equal $(1 - b_d)$. Since $n - t_a - 1 \geq t_h > t_a$, b_d is the majority value for each honest player, and the lemma follows.

Theorem 5. *For any value $n \in \{2, 3, 4, 5, 6, 8, 9, 12\}$ there exists a protocol for n players that achieves broadcast in the presence of a threshold adversary.*

Proof. The case $n = 2$ is trivial. When $n = 3$, it follows from our constraints that $t_a \leq 1$ and $t_c = 0$, so we can run any authenticated byzantine agreement protocol. When $n = 4$, it follows from our constraints that $t_a \leq 1$, and therefore that $n > 3t_a$, so we can ignore the PKI and run a protocol that is secure without authentication. The remainder of the proof deals with $n \in \{5, 6, 8, 9, 12\}$.

Lemma 7. *For $n \in \{5, 6, 8\}$,* $\mathsf{authLSP}(\lfloor \frac{n-1}{3} \rfloor + 1)$ *achieves broadcast in the presence of a threshold adversary.*

Proof. We prove the lemma by considering all possible types of dealers. We let b_d denote the input bit of the dealer D.

D is honest and not compromised: This case follows from Lemma 5.

D is honest and compromised: Consider the following two scenarios:

$\mathbf{t_a} \leq \lfloor \frac{\mathbf{n-1}}{\mathbf{3}} \rfloor$: For $n \in \{5, 6, 8\}$, we have $n > 2\left\lfloor \frac{n-1}{3} \right\rfloor + \left\lfloor \frac{n-1}{3} \right\rfloor + 1 \geq 2t_a + m$, where the first inequality holds because of our assumption on n, and the second from our assumption on t_a. Applying Lemma 4 we get validity and agreement as claimed.

$\mathbf{t_a} > \lfloor \frac{\mathbf{n-1}}{\mathbf{3}} \rfloor$: Since we assume a threshold adversary, in this case we have $t_h > t_a$ (cf. section 2). Applying Lemma 6, agreement and validity follow.

D is malicious: Since we assume a threshold adversary, we have that $n > 2t_a$. We note that the malicious dealer is excluded from each of the executions of

$\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor)$ in step 2, and therefore, of the $n-1$ players that participate in those executions, only $t_a - 1$ are malicious. The reader can verify that for $n \in \{5,6,8\}$, $n-1 > 3\lfloor\frac{n-1}{3}\rfloor$, and that $\lfloor\frac{n-1}{3}\rfloor \geq t_a - 1$ (recalling that $t_a < \frac{n}{2}$). Applying Lemma 3, we have agreement in each execution of $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor)$ in step 2. Agreement in $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 1)$ follows when the players compute their output in steps 3 and 4.

Lemma 8. *For $n \in \{9, 12\}$, Protocol $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 2)$ achieves broadcast in the presence of a threshold adversary.*

Proof. We prove the lemma by considering all possible types of dealers.

D is honest and uncompromised: This case follows from Lemma 5.

D is honest and compromised: Consider the following two scenarios:

$\mathbf{t_a \leq \lfloor\frac{n-1}{3}\rfloor}$: For $n \in \{9,12\}$, we have $n > 2\lfloor\frac{n-1}{3}\rfloor + \lfloor\frac{n-1}{3}\rfloor + 2 \geq 2t_a + m$, where the first inequality holds by our assumption on n, and the second holds by our assumption on t_a. Applying Lemma 4 we get validity and agreement as claimed.

$\mathbf{t_a > \lfloor\frac{n-1}{3}\rfloor}$: Because we assume a threshold adversary, we have $t_h > t_a$ (cf. section 2). Applying Lemma 6, agreement and validity follow.

D is malicious: We consider the recursive execution of $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 1)$ in step 2, and prove agreement for each of the $n-1$ dealers. When the dealer in step 2 is honest and uncompromised, by Lemma 5 we have agreement in his execution. If the dealer is honest and compromised we consider two further possibilities. If $t_a \leq \lfloor\frac{n-1}{3}\rfloor$, then among the $n-1$ players participating in this recursive execution, of which at most $t_a - 1$ are malicious, we have

$$\begin{aligned} n - 1 > 3\left\lfloor\frac{n-1}{3}\right\rfloor - 1 &= 2\left(\left\lfloor\frac{n-1}{3}\right\rfloor - 1\right) + \left(\left\lfloor\frac{n-1}{3}\right\rfloor + 1\right) \\ &\geq 2\left(t_a - 1\right) + \left(\left\lfloor\frac{n-1}{3}\right\rfloor + 1\right). \end{aligned}$$

By Lemma 4, agreement follows. If the dealer is honest and compromised and $t_a > \lfloor\frac{n-1}{3}\rfloor$, then $t_h > t_a$ and by Lemma 6 agreement follows. If the dealer in step 2 is malicious, consider what happens in the *next* recursive step when the players execute $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor)$. Now two malicious dealers have been excluded: both the dealer in $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 2)$ and the dealer in $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 1)$. Noting that the maximum number of malicious players is 4 when $n = 9$ and 5 when $n = 12$ (because we have a threshold adversary), it follows that among the remaining $n-2$ players, $n-2 > 3(\lfloor\frac{n-1}{3}\rfloor)$ and $\lfloor\frac{n-1}{3}\rfloor \geq t_a - 2$. Applying Lemma 3, we have agreement for all dealer types in $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor)$, and agreement follows for all malicious dealers in the executions of $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 1)$. Since we have proven agreement for all dealer types in $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 1)$, we have agreement in the execution of $\mathsf{authLSP}(\lfloor\frac{n-1}{3}\rfloor + 2)$ as well.

This concludes the proof of Theorem 5.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

1. D. Dolev and H. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
2. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 293–302. IEEE, 2008. Full version available at `http://eprint.iacr.org/2008/240`.
3. M. Fitzi, T. Holenstein, and J. Wullschleger. Multi-party computation with hybrid security. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of *LNCS*, pages 419–438. Springer, 2004.
4. S. Gordon, J. Katz, R. Kumaresan, and A. Yerukhimovich. Authenticated broadcast with a partially compromised public-key infrastructure, 2009. Available at `http://eprint.iacr.org/2009/410`.
5. A. Gupta, P. Gopal, P. Bansal, and K. Srinathan. Authenticated Byzantine generals in dual failure model. In *Distributed Computing and Networking (ICDCN)*, volume 5935 of *LNCS*, pages 79–91. Springer, 2010.
6. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology — Crypto '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
7. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology — Crypto '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
8. L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine generals problem. *ACM Trans. Programming Language Systems*, 4(3):382–401, 1982.
9. MS00-008: Incorrect registry setting may allow cryptography key compromise. Microsoft Help and Support. See `http://support.microsoft.com/kb/259496`.
10. P. Q. Nguyen. Can we trust cryptographic software? Cryptographic flaws in GNU privacy guard v1.2.3. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of *LNCS*, pages 555–570. Springer, 2004.
11. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *10th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 51–59. ACM Press, 1991.
12. M. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

A The Dolev-Strong Protocol

For completeness, we present a modified version of the Dolev-Strong [1] protocol for authenticated broadcast. (See Figure 4.) A message M is called (v,i)-*valid* if it was received in round i and has the form $(v, s_{P_1}, \ldots, s_{P_i})$, where $P_1 = D$, all P_j's are distinct, and for every $j = 1, \ldots, i$ the string s_{P_j} is a valid signature

DS

Inputs: Let D be the dealer with input $b_d \in \{0,1\}^*$ and secret key sk_D.

1. (Round $r = 0$) D sends $(b_d, \mathsf{Sign}_{sk_D}(b_d))$ to every player.
2. In round $r = 1$ to n:
   1. Every player P_i checks every incoming message and discards any that are not $(\cdot, r)$-valid or that already contain P_i's signature. P_i orders the remaining messages lexicographically.
      – If the content, v, of all remaining messages is identical, P_i appends its signature to the first message (thus forming a $(v, r+1)$-valid message) and sends the result to all players.
      – If there exist 2 messages with different content, P_i appends its signature to the first 2 such messages and sends the result to all players.
2. Termination:
   1. If P_i ever received valid messages with different content, then it outputs a default value.
   2. If P_i only received valid messages for one value v, then it outputs v.
   3. If P_i never received a valid message for either $v \in \{0,1\}$ then it outputs a default value.

Fig. 4. The Dolev-Strong protocol for broadcast.

on $(v, s_{P_1}, \ldots, s_{P_{j-1}})$ relative to the verification key of P_j. We refer to v as the *content* of a valid message. If the dealer is honest and uncompromised, there will only be $(v, \star)$-valid messages for a single value v, in which case the players consider the execution *clean*. Otherwise, the execution is called *dirty*. We note that the protocol differs from the original Dolev-Strong [1] protocol only in round complexity: we always require $n+1$ rounds to ensure that all players agree whether the run was dirty. We also assume at least one honest and uncompromised player.