Detectable Byzantine Agreement Secure Against Faulty Majorities

Matthias Fitzi* Daniel Gottesman† Martin Hirt* Thomas Holenstein*

Adam Smith‡

* ETH Zurich
Department of Computer Science
Switzerland
{fitzi,hirt,holenst}@inf.ethz.ch

† UC Berkeley
EECS: Computer Science Division
Soda Hall 585, Berkeley, CA 94720, USA
gottesma@eecs.berkeley.edu

‡ M.I.T. Laboratory for Computer Science
200 Technology Square, Cambridge, MA 02139, USA
adsmith@mit.edu

ABSTRACT

It is well-known that n players, connected only by pairwise secure channels, can achieve Byzantine agreement only if the number t of cheaters satisfies $t < n/3$, even with respect to computational security. However, for many applications it is sufficient to achieve *detectable broadcast*. With this primitive, broadcast is only guaranteed when all players are non-faulty ("honest"), but all non-faulty players always reach agreement on whether broadcast was achieved or not. We show that detectable broadcast can be achieved regardless of the number of faulty players (i.e., for all $t < n$). We give a protocol which is unconditionally secure, as well as two more efficient protocols which are secure with respect to computational assumptions, and the existence of quantum channels, respectively.

These protocols allow for secure multi-party computation tolerating any $t < n$, assuming only pairwise authenticated channels. Moreover, they allow for the setup of public-key infrastructures that are consistent among all participants — using neither a trusted party nor broadcast channels.

Finally, we show that it is not even necessary for players to begin the protocol at the same time step. We give a "detectable Firing Squad" protocol which can be initiated by a single user at any time and such that either all honest players end up with synchronized clocks, or all honest players abort.

Keywords: Broadcast, Byzantine agreement, multi-party computation, public-key infrastructure, quantum signatures.

1. INTRODUCTION

Broadcast (a.k.a. Byzantine agreement) is an important primitive in the design of distributed protocols. A protocol with a designated sender s achieves broadcast if it acts as a megaphone for s: all other players will receive the message s sends, and moreover if an honest player receives a message, then he knows that all other honest players received the same message — it is impossible even for a cheating s to force inconsistency in the outputs.

Lamport, Shostak, and Pease [26, 25] showed that if players share no initial setup information beyond pairwise authenticated channels, then in fact broadcast is possible if and only if $t < n/3$, where n is the number of players and t is the number of actively corrupted players to be tolerated by the protocol. By the impossibility proofs in [22, 13, 14], even additional resources (e.g., secret channels, private random coins, quantum channels and computers) cannot help to improve this bound unless some setup shared among more than just pairs of players is involved.

On the other hand, the picture changes dramatically if some previous setup is allowed. If secure signature schemes exist and the adversary is limited to polynomial time, then having pre-agreement on a public verification key for every player allows for efficient broadcast for any $t < n$ [25, 10]. Pfitzmann and

Waidner [27] showed that broadcast among n players during a precomputation phase allows for later broadcast that is efficient and unconditionally secure, also for any $t < n$. Those two works will be key pieces for our constructions.

Surprisingly, very strong agreement protocols are still achievable without previous setup. Fitzi, Gisin, Maurer, and von Rotz [16, 15] showed that a weaker variant of broadcast, *detectable broadcast*, can be achieved for any $t < n/2$. In a detectable broadcast, cheaters can force the protocol to abort, but in that case all honest players agree that it has aborted. This is ideal for settings in which robust tolerance of errors is not necessary, and *detection* suffices.

1.1 Contributions

We show that detectable broadcast is possible for any $t < n$, in three different models. The first protocol requires only pairwise authenticated channels, but assumes a polynomial-time adversary and the existence of secure signature schemes. The second protocol requires pairwise secure channels, but is secure against unbounded adversaries. The third protocol requires authenticated classical channels and (insecure) quantum channels, but also tolerates unbounded adversaries.

THEOREM 1. *Detectable broadcast is achievable based on (1): a network of pairwise authenticated channels and assuming a secure signature scheme; or based on (2): a network of pairwise secure channels with no computational assumptions; or based on (3): a network of pairwise authenticated channels and insecure quantum channels with no computational assumptions.*

The protocol for (1) requires $t + 3$ rounds and $O(n^3k)$ overall message bits to be sent by correct players, where k is the length of a signature. The protocol for (2) requires $t + 5$ rounds, and roughly $O(n^8(\log n + k)^3)$ overall message bits to be sent, where k is the security parameter. The protocol for (3) requires $t + 5$ rounds and roughly $O(kn^4 \log n)$ bits (qubits) of communication. The message complexities above are stated with respect to message domains of constant size. The exact complexities with dependencies on the domain size are given later.

In particular, our results show that the impossibility of weak broadcast for deterministic protocols, due to Lamport [24], does not extend to randomized ones (Lamport's proof does apply to protocols with only public coins, but fails when players are allowed to have private random inputs).

Combined with results from the previous literature, our results yield a protocols for a "detectable" version of *multi-party computation* (MPC) with resilience $t < n$, in which the adversary may force the protocol to abort partway through, assuming only pairwise authenticated channels and the existence of trapdoor permutations. An MPC protocol allows players with inputs $x_1, ..., x_n$ to evaluate some function $f(x_1, ..., x_n)$ such that the adversary can neither corrupt the output nor learn any information beyond the value of f.

We give two techniques for transforming MPC protocols based on a broadcast channel into ones which assume no such primitive. The first technique preserves the standard definition on MPC for a faulty majority, but increases the round complexity considerably. The second technique yields more efficient protocols, at the price of *partial fairness* (see Section 6.1 for a definition).

In independent work, Goldwasser and Lindell [20] give a different transformation, which also eliminates the use of a broadcast channel. Their transformation achieves a weaker notion of agreement than ours—honest players may not always agree on whether or not the protocol terminated successfully. On the other hand, their transformation is more efficient in round complexity, and additionally achieves partial fairness, which is not guaranteed by our second transformation. We discuss their work further in Section 6.3.

Finally, it can be observed that, in order to achieve detectable broadcast or multi-party computation, no prior agreement among the players is necessary. This implies that such a protocol can be spontaneously initiated by any player at any time. It also implies that players in a synchronous network can achieve "detectable clock synchronization," namely either all honest players end up with synchronized clocks, or all honest players abort (not necessarily at exactly the same time).

1.2 Models and Definitions

Models: We consider a synchronous network in which every pair of players is connected by an unjammable, authenticated channel. That is, for every pair i, j, player p_i can always send messages directly to p_j. The adversary can neither prevent those messages from being delivered nor introduce new messages on the channel. By *synchronous*, we mean that all players run on a common clock, and messages are always delivered within some bounded time. Our protocols are secure even if the adversary can *rush* messages, i.e., even if messages from honest players are delivered before corrupted players send their messages.

Our protocols are secure against Byzantine (or "active") adversaries, that select up to t players and coordinatedly corrupt them in an arbitrarily malicious way. The corruptions may be *adaptive*, that is the adversary grows the set of corrupted players on the fly, based on the execution so far.

In this framework, we consider three models, denoted $\mathcal{M}_{auth}$, $\mathcal{M}_{sec}$, $\mathcal{M}_q$:

1. $\mathcal{M}_{auth}$: *Authenticated channels, computational security.* The adversary may read all communication in the network, even among honest players, but is limited to polynomial time computations (and may not tamper with the channels). Here, the only information not available to the adversary is the internal state and random coins of the honest players.
2. $\mathcal{M}_{sec}$: *Secure channels, unconditional security* (a.k.a. "information-theoretic security"). The channels between honest players are unreadable, but the adversary is computationally unbounded.
3. $\mathcal{M}_q$: *Quantum channels, unconditional security.* All pairs of players share an authenticated classical channel and an insecure quantum channel (which the adversary can tamper with). The adversary is computationally unbounded.

If one is only interested in feasability results, then one only needs to consider the second model, $\mathcal{M}_{sec}$. By (carefully) encrypting communication over authenticated channels, one can implement secure channels in $\mathcal{M}_{auth}$ [6], so in fact any protocol for $\mathcal{M}_{sec}$ is also a protocol for $\mathcal{M}_{auth}$. However, protocols designed specifically for $\mathcal{M}_{sec}$ can use computational cryptographic tools for greater simplicity and efficiency[1]. Similarly, any protocol for $\mathcal{M}_{sec}$ leads to a protocol for $\mathcal{M}_q$ (by implementing secure channels using quantum key distribution), but protocols designed specifically for $\mathcal{M}_q$ may be more efficient.

Definitions: In the definitions below, $\mathcal{D}$ may be any finite domain (say $\{0,1\}^m$). We require that the conditions for each task hold except with probability exponentially small in the security parameter k (super-polynomially small in the case of computational security). Protocols should have complexity polynomial in both n and k.

DEFINITION 1 ((Strong) Broadcast). A protocol among n players, where player s (called the *sender*) holds an input value $x_s \in \mathcal{D}$ and every player p_i ($i \in [n]$) finally decides on an output value $y_i \in \mathcal{D}$, achieves *broadcast* if it satisfies:

- **Validity:** If the sender is honest then all honest players p_i decide on the sender's input value, $y_i = x_s$.
- **Consistency:** All honest players compute the same output value $v \in \mathcal{D}$. △

DEFINITION 2 (Detectable Broadcast). A protocol among n players achieves *detectable broadcast* if it satisfies:

- **Correctness:** All honest players commonly accept or commonly reject the protocol. If all honest players accept then the protocol achieves broadcast.
- **Completeness:** If no player is corrupted during the protocol then all players accept.
- **Fairness:** If any honest player rejects the protocol then the adversary gets no information about the sender's input. △

It turns out that our protocols achieve a stronger notion, namely they "detectably" establish the setup needed to perform strong broadcast using the protocols of [10, 27].

DEFINITION 3 (Detectable Precomputation). A protocol among n players achieves *detectable precomputation for broadcast* (or *detectable precomputation*, for short) if it satisfies:

- **Correctness:** All honest players commonly accept or commonly reject the protocol. If all honest players accept then strong broadcast will be achievable.
- **Completeness:** If no player is corrupted during the protocol then all honest players accept.
- **Independence:** A honest player's intended input value for any precomputed broadcast need not be known at the time of the precomputation. △

Independence implies two important properties: first, the precomputation may be done long before the actual broadcasts it will be used for; and second, that the adversary gets no information about any future inputs by honest senders (i.e., that fairness as defined for detectable broadcast is guaranteed). In particular, this means that detectable precomputation implies detectable broadcast.

As opposed to detectable broadcast, the advantage of detectable precomputation for broadcast is that the preparation is separated from the actual execution of the broadcasts, i.e., only the precomputation must be detectable. As soon as the precomputation has been successfully completed, strong broadcast is possible secure against any number of corrupted players.

2. GENERIC PROTOCOL FOR DETECTABLE BROADCAST

Along the lines of [16], we give constructions of protocols for detectable precomputation, which implies detectable broadcast.

We present protocols for three models: a network of authenticated channels with computational security ($\mathcal{M}_{auth}$), secure channels with unconditional security ($\mathcal{M}_{sec}$), and quantum channels ($\mathcal{M}_q$). The protocols for models $\mathcal{M}_{auth}$ and $\mathcal{M}_q$ are more efficient, while the one for model $\mathcal{M}_{auth}$ is ultimately more general (since it can be used, with small modifications, in all three models).

Note that, although stated differently, the known results in [25, 10] and those in [27] can both be viewed in context of the second one: In models $\mathcal{M}_{auth}$ and $\mathcal{M}_{sec}$, a temporary phase wherein broadcast is achievable (for some reason) allows running a precomputation such that future broadcast will be achievable without any additional assumptions. In $\mathcal{M}_{auth}$, this precomputation can simply consist of having every player p_i compute a secret-key/public-key pair (SK_i, PK_i) and broadcast his public key PK_i. In $\mathcal{M}_{sec}$, more involved methods must be applied, but still, the principle of the precomputation is similar: the players broadcast some information that allows all players to consistently compute keys for a pseudo-signature scheme among the players.

Our construction for detectable precomputation is generic in the sense that any "reasonable" precomputation protocol exploiting temporary broadcast in order to allow for future broadcast can be transformed into a protocol for detectable precomputation. This transformation is based on an implementation of conditional gradecast, defined below, which is a variant of graded broadcast [12]. The independent work of Goldwasser and Lindell [20] calls this task "broadcast with designated abort."

DEFINITION 4 (Conditional Gradecast). A protocol among n players, where player s (called the *sender*) holds an input value $x_s \in \mathcal{D}$ and every player p_i ($i \in [n]$) finally decides on an output value $y_i \in \mathcal{D}$ and a grade $g_i \in \{0,1\}$, achieves *conditional gradecast* if it satisfies:

- **Value validity:** If the sender is honest then all honest players p_i decide on the sender's input value, $y_i = x_s$.
- **Conditional grade validity:** If all players are honest then all players p_i decide on grade $g_i = 1$.
- **Consistency:** If any honest player p_i gets grade $g_i = 1$ then all honest players p_j decide on the same output value $y_j = y_i$. △

[1]Protocols designed specifically for $\mathcal{M}_{auth}$ may also use potentially weaker computational assumptions. For example, our protocols require only the existence of one-way functions, while the general reduction from $\mathcal{M}_{sec}$ to $\mathcal{M}_{auth}$ requires semantically secure encryption.

Assume set $P = \{p_1, \ldots, p_n\}$ of players in model $\mathcal{M} \in \{\mathcal{M}_{auth}, \mathcal{M}_{sec}\}$. Let π be a precomputation protocol for model $\mathcal{M}$ where, additionally, players are assumed to be able to broadcast messages; and let $B = \{\beta_1, \ldots, \beta_n\}$ be a set of protocols for model $\mathcal{M}$ where each protocol β_i achieves broadcast with sender p_i when based on the information exchanged during an execution of π. Furthermore, assume that π satisfies the independence property of Definition 1.2 with respect to the protocols in B.

If protocol π always efficiently terminates even if all involved broadcast invocations fail and all protocols β_i always efficiently terminate even when based on arbitrary precomputed information then a protocol δ for detectable precomputation can be achieved from π and B as follows:

1. Run protocol π wherein each invocation of broadcast is replaced by an invocation of conditional gradecast with the same sender.
2. Each player p_i computes the logical AND over the grades he got during all invocations of conditional gradecast in (the modified) protocol π, $G_i = \bigwedge g_i$.
3. For each player $p_i \in P$, an invocation of protocol β_i is run wherein p_i inputs G_i.
4. Each player p_i accepts if and only if $G_i = 1$ and p_i received $G_j = 1$ by every player $p_j \in P$ during Step 3.

Note that the protocols β_i in Step 3 do not necessarily achieve broadcast since the invocation of π during Step 1 might have failed. However, they will always efficiently terminate by assumption. We now informally argue that protocol δ achieves detectable precomputation.

Correctness. Suppose p_j and p_k to be honest players and suppose that p_j accepts. Then $G_j = 1$ and hence all invocations of conditional gradecast during protocol π achieved broadcast (when neglecting the grade outputs) and hence, all protocols in B indeed achieve broadcast. Since p_j accepts, all players p_i broadcasted $G_i = 1$ during Step 3, especially all honest ones, and hence all honest players accept at the end.

Completeness. If no player is corrupted during the invocation of protocol δ then no player p_i ever computes a grade $g_i = 0$ and all players accept.

Independence. Independence directly follows from the assumed independence property of π.

Before giving a more detailed view on our concrete protocols for the models $\mathcal{M}_{auth}$ and $\mathcal{M}_{sec}$, we first describe a protocol that achieves conditional gradecast in both models $\mathcal{M}_{auth}$ and $\mathcal{M}_{sec}$.

Protocol `CondGradecast`: [with respect to sender s]

1. Sender s sends his input x_s to every player; player p_i receives y_i;
2. Every player p_i redistributes y_i to every other player and computes grade $g_i = 1$ if the value y_i was reconfirmed by everybody during Step 2, and else $g_i = 0$;

LEMMA 1. *Protocol* `CondGradecast` *achieves conditional gradecast for $t < n$.*

PROOF. Both validity conditions are trivially satisfied. On the other hand, suppose that p_i is honest and that $g_i = 1$. Since $g_i = 1$, every honest player p_j sent $y_j = y_i$ during Step 2, and hence consistency is satisfied. □

3. COMPUTATIONAL SECURITY

Let (G, S, V) be a signature scheme, i.e., secure under adaptive chosen message attack. Here G is a key generation algorithm, S is the signing algorithm and V is the verification algorithm. All algorithms take a unary security parameter 1^k as input. For simplicity, we assume that signatures are k bits long.

The Dolev-Strong protocol [10] achieves (strong) broadcast when a consistent public-key infrastructure has been setup:

DEFINITION 5 (Consistent Public-Key Infrastructure (PKI)). We say that a group of n players have a *consistent public-key infrastructure for* (G, S, V) if every player p_i ($i \in [n]$) has a verification key PK_i which is known to all other players and was chosen by p_i (in particular, the keys belonging to honest players will have been chosen correctly according to G, using private randomness, and the honest players will know their signing keys). Note that the cheaters' keys may (and will, in general) depend on the keys of honest players. △

PROPOSITION 2 (DOLEV-STRONG [10]). *Given a consistent PKI and pairwise authenticated channels, (strong) broadcast tolerating any $t < n$ cheaters is achievable using $t + 1$ rounds and an overall message complexity of $O(n^2 \log |\mathcal{D}| + n^3 k)$ bits where $\mathcal{D}$ is the message domain (including possible padding for session IDs, etc). An adversary which makes the protocol fail with probability ε can be used to forge a signature in an adaptive chosen message attack with probability ε/n.*

Let `DSBroadcast` denote the Dolev-Strong broadcast protocol. A precomputation "protocol" π among n players $P = \{p_1 \ldots, p_n\}$ allowing for future broadcast is to set up a consistent infrastructure. Given broadcast channels this is very simple: have every player generate a signing/verification key pair (SK_i, PK_i) and broadcast the verification key. Hence, by applying the generic transformation described in the previous section we get the following protocol for detectable precomputation:

Protocol `DetPrecomp`: [Protocol for Model $\mathcal{M}_{auth}$]

1. Every player p_i generates a signing/verification key pair $(\mathrm{SK}_i, \mathrm{PK}_i)$ according to G. For every player p_j protocol `CondGradecast` is invoked where p_j inputs his public key PK_j as a sender. Every player p_i stores all received public keys $\mathrm{PK}_i^{(1)}, \ldots, \mathrm{PK}_i^{(n)}$ and grades $g_i^{(1)}, \ldots, g_i^{(n)}$.
2. Every player p_i computes $G_i = \bigwedge_{k=1}^n g_i^{(k)}$.
3. For every player p_j an instance of `DSBroadcast` is invoked where p_j inputs G_j as a sender. Every player p_i stores the received values $G_i^{(j)}$ ($j \in \{1, \ldots, n\} \setminus \{i\}$) and $G_i^{(i)} := G_i$.
4. All players p_i accept if $\bigwedge_{j=1}^n G_i^{(j)} = 1$ and reject otherwise.

THEOREM 2. *Protocol* `DetPrecomp` *achieves detectable precomputation among n players for model $\mathcal{M}_{auth}$ tolerating any $t < n$ corrupted players. An adversary who can make the protocol fail with probability ε can be used to forge a signature with probability ε/n.*

PROOF. Completeness and independence are trivially satisfied. It remains to prove that the correctness condition is satisfied. Assume p_j and p_k to be two honest players and assume that p_j accepts. We show that hence p_k also accepts: Since p_j accepts it holds that $G_j = 1$ and hence $g_j^{(1)} = \cdots = g_j^{(n)} = 1$. By the definition of conditional gradecast, this implies that the players have set up a consistent PKI and that hence any invocation of Protocol `DSBroadcast` achieves broadcast. Since p_j accepts, all players p_i broadcasted $G_i = 1$ during Step 3 and hence all honest players decide to accept at the end of the protocol.

Up to this point, the proof up to this point is deterministic, assuming that no invalid signatures are ever forged. Thus, any adversary which breaks the protocol with probability ϵ must forge a signature with that probability. We can use that adversary in a chosen message attack by simulating the protocol, and choosing one honest player whose public key will be the key in the chosen message attack. The chances of choosing a player whose signature will be forged is at least ϵ/n. □

Note that Protocols `DetPrecomp` and `DSBroadcast` have another important property, i.e., that they keep the players synchronized: If the players accept then they terminate these protocols during the same communication round. This can be easily verified.

As Protocol `DSBroadcast` requires $t+1$ rounds of communication and an overall message complexity of $O(n^2 \log|\mathcal{D}| + n^3 k)$ bits to be sent by honest players, Protocol `DetPrecomp` requires $t+3$ rounds and an overall message complexity of $O(n^3 \log|\mathcal{D}| + n^4 k)$ bits. Any later broadcasts are conventional calls to `DSBroadcast`.

The message complexity of Protocol `DetPrecomp` can be reduced to $O(n^3(\log|\mathcal{D}| + k))$ bits overall by replacing the n parallel invocations of `DSBroadcast` during Step 3 by a consensus-like protocol with default value 1. For this, the n `DSBroadcast` protocols are run in parallel in a slightly modified way. In the first round, a sender p_s who accepts simply sends bit $G_s = 1$ without a signature (or no message at all), and only if he rejects sends bit $G_s = 0$ together with a signature on it. As soon as, during some round $r = 1, \ldots, t+1$, a player accepts the value $G_s = 0$ from one or more senders p_s because he received valid signatures from r different players (including p_s) on value $G_s = 0$ with respect to the protocol instance with sender p_s then, for exactly one arbitrary such sender p_s, he adds his own signature for 0 with respect to this protocol instance and, during the next round, relays all $r+1$ signatures to every other player, decides on 0, and terminates. If a player never accepts value $G_s = 0$ from any sender p_s then he decides on 1 after round $t+1$. If all players p_s are honest and send value $G_s = 1$ then, clearly, all players decide on 1 at the end. On the other hand, if any correct player decides 0 then all correct players do so. Finally, during this protocol, no player distributes more signatures than during a single invocation of Protocol `DSBroadcast`.

4. UNCONDITIONAL SECURITY WITH PSEUDO-SIGNATURES

We now consider the applying of the same framework to the setting of *pairwise secure* channels, but requiring information-theoretic security. The basic procedures come from [27], which is itself a modified version of the Dolev-Strong protocol, with signatures replaced by information-theoretically secure "pseudosignatures".

PROPOSITION 3 (PFITZMANN-WAIDNER, [27]). *There exist protocols* `PWPrecomp` *and* `PWBroadcast` *such that if* `PWPrecomp` *is run with access to a broadcast channel, then subsequent executions of* `PWBroadcast` *(based on the output of* `PWPrecomp`*) achieve strong broadcast secure against an unbounded adversary and tolerating any $t < n$.*

The total communication is polynomial in $n, \log|\mathcal{D}|, k, \log b$, where k is the security parameter (failure probability $< 2^{-k}$) and b is the number of future broadcasts to be performed. The number of rounds is at most $2n^2$.

The generic transformation described in Section 2 can be directly applied to the PW-precomputation protocol resulting in a protocol for detectable broadcast with at most $r = \lfloor \frac{7}{2}n^2 \rfloor + t + 1$ communication rounds and an overall message complexity of bits polynomial in n and the security parameter k. However, most of the PW-precomputation protocol consists of fault localization, i.e., subprotocols that allow to identify players that have been misbehaving. These steps are not required in our context since we are only interested in finding out whether faults occurred or not. We now give a protocol for the precomputation of one single broadcast with sender s where these steps are stripped off. Thereby, as in [27], $m = O(n^2(k + \log n))$ and the key size is $|\kappa| = |K| = O(\log\log|\mathcal{D}|(k + \log n + \log\log\log|\mathcal{D}|))$.

Protocol `SimpPWPrecomp`: [player p_i's view, $i \in [n]$]

1. For $(j = s, A)$, and (j, A), (j, B) $(j \in \{1, \ldots, n\} \setminus \{s\})$ in parallel:
2. For $k = 1 \ldots m$ in parallel:
3. If $i \neq j$: select random authentication key κ_i;
4. For $\ell = 1 \ldots 2n$ in parallel:
5. $\forall p_h$ $(h \neq i)$ agree on a pairwise key $K_{hi}^{(\ell)}$;
6. Broadcast $\kappa_i^{2\ell-1} + \sum_{h \neq i} K_{hi}^{(\ell)}$;
7. If $i = j$: broadcast "accept" or "reject";
8. Decide to accept $(h_i := 1)$ if and only if all signers p_j sent message "accept" with respect to (j, A) and (j, B); otherwise reject $(h_i := 0)$;

The proof of the following lemma follows from [27]:

LEMMA 4. *Given a broadcast channel, Protocol* `SimpPWPrecomp` *detectable precomputation for broadcast (with respect to Protocol* `PWBroadcast`*). It requires three rounds, two of which use the broadcast channel.*

Furthermore, for our purpose, Step 7 does not require broadcast but can be done by simple multi-sending, since invocations of precomputed broadcast will follow anyway (see Protocol `DetPrecomp` in Section 4).

Protocol `DetPrecomp`: [Model $\mathcal{M}_{sec}$, for b later broadcasts]

1. Execute protocol `SimpPWPrecomp` for $b + n$ future broadcasts wherein each invocation of broadcast is replaced by an invocation of Protocol `CondGradecast`.
2. Every player p_i computes $G_i = \bigwedge_{k=1}^{\ell} g_i^{(k)} \wedge h_i$ where the $g_i^{(k)}$ are all grades received during an invocation of conditional gradecast during Step 1 and h_i is the bit indicating whether p_i accepted at the end of Step 1.
3. For every player p_j an instance of `PWBroadcast` is invoked where p_j inputs G_j as a sender. Every player p_i stores the received values $G_i^{(j)}$ $(j \in \{1, \ldots, n\} \setminus \{i\})$ and $G_i^{(i)} := G_i$.
4. All players p_i accept if $\bigwedge_{j=1}^{n} G_i^{(j)} = 1$ and reject otherwise.

Again, as in the protocol for model $\mathcal{M}_{auth}$ of the previous section, the parallel invocations of Protocol `PWBroadcast` during Step 3 can be replaced by a consensus-like protocol saving a factor of n for the bit-complexity of the whole protocol.

THEOREM 3. *Protocol* `DetPrecomp` *achieves detectable precomputation for b future broadcasts among n players for model $\mathcal{M}_{sec}$ tolerating any number $t < n$ of corrupted players with the following property: If, for the underlying* `PWPrecomp`*, a security parameter of at least $k_0 = k + \lceil \log(n+b) \rceil$ is chosen then the overall error probability, i.e., the probability that either Protocol* `DetPrecomp` *or any one of the b broadcasts prepared for fails, is at most 2^{-k}.*

PROOF. The proof proceeds along the lines of the proof of Theorem 2. The error probability follows from the analysis in [27] for one single precomputation/broadcast pair. Invoking `PWPrecomp` with security parameter k_0 hence implies an overall error probability of at most $(n+b)2^{-k_0} \leq 2^{-k}$. $\square$

Protocol `SimpPWPrecomp` requires 4 rounds of message exchange and an overall message complexity of $O(n^7 \log\log|\mathcal{D}|(k_0+\log n+\log\log\log|\mathcal{D}|)^2)$ bits where k_0 is the security parameter of the underlying `PWPrecomp` and $\mathcal{D}$ is the domain of future messages to be broadcast (including possible padding for session IDs, etc.). Protocol `PWBroadcast` requires $t+1$ rounds of message exchange and an overall message complexity of $O(n^2 \log|\mathcal{D}| + n^6(k_0 + \log n)^2)$ bits. Since Protocol `DetPrecomp` precomputing for b later broadcasts invokes Protocol `SimpPWPrecomp` $(n+b)$ times in parallel and Protocol `PWBroadcast` n times in parallel, it requires overall $t+5$ communication rounds and an overall message complexity of

$$O(n^3 \log|\mathcal{D}| + n^7(n+b)\log\log|\mathcal{D}|(k+\log n+\log b+\log\log\log|\mathcal{D}|)^2)$$

bits to be sent by correct players. Again, as the protocols presented in the previous section, if the players accept then they terminate Protocols `DetPrecomp` and `PWBroadcast` during the same communication round.

Furthermore, as follows from Proposition 3, using the recycling techniques in [27], the bit complexity of Protocol `DetPrecomp` can be reduced to polylogarithmic in the number b of later broadcasts to be precomputed for, i.e., to polynomial in n, $\log|\mathcal{D}|$, k, and $\log b$.

5. UNCONDITIONAL SECURITY WITH QUANTUM SIGNATURES

In this section we consider a third network model $\mathcal{M}_q$, in which participants are connected by pairwise authenticated channels as well as pairwise (unauthenticated) *quantum* channels. As for $\mathcal{M}_{sec}$, we require unconditional security with a small probability of failure.

Now one can always construct secure channels on top of this model by using a quantum key distribution protocol (e.g. Bennett-Brassard [4]). This requires adding two rounds at the beginning of the protocol. For noiseless quantum channels, agreeing on a key of ℓ bits requires sending $\ell + O(k+\log\ell)$ qubits and $O(k+\log\ell)$ classical bits [1]. Note that the key distribution protocol may fail if the adversary intervenes, but in such a case the concerned players can set their grades to 0 in the later agreement protocol and all honest players will abort. All in all, this yields protocols with similar complexity to those of the previous section.

One can improve the complexity of the protocols significantly by tailoring the pre-computation protocol to the quantum model, and using the quantum signatures of Gottesman and Chuang [21] instead of the pseudosignatures of [27]. The idea is to apply the distributed swap test from [21] to ensure consistency of the distributed quantum keys. As in [27], one gets a broadcast protocol by replacing classical signatures in the Dolev-Strong protocol [10] with quantum signatures. Note that quantum communication is required only in the very first round of the computation, during which (possibly corrupted) EPR pairs are exchanged. Any further quantum transmissions can be done using quantum teleportation. Authentication of the initial EPR transmissions can be done with the protocols of Barnum et al. [1].

THEOREM 4. *There is a protocol which achieves detectable precomputation for b future broadcasts among n players for model $\mathcal{M}_q$ tolerating any number $t < n$ of corrupted players. The protocol requires $t+5$ rounds and $O(k_0 n^5 b_0)$ bits (qubits) of communication, where $k_0 = k + \log(b_0 n)$ and $b_0 = b\log|\mathcal{D}|$.*

6. SECURE MULTI-PARTY COMPUTATION

In this section we consider the effect of using detectable broadcast as a building block in larger protocols, especially ones for *multi-party computing*, also known as *secure function evaluation* or simply *secure computation*. In independent work, Goldwasser and Lindell [20] also consider such general transformations, albeit with different goals. We discuss their results more specifically in Section 6.3.

The results of the previous sections suggest two general techniques for transforming a protocol Π which assumes a broadcast channel into a "detectable" protocol Π' which only assumes pairwise communication channels, but which may abort. Suppose that there is an upper bound r on the number of rounds of interaction required by Π (such a protocol is called "fixed-round").

The first transformation is straightforward, and is also used in [16]: First, run a protocol for detectable precomputation for broadcast. If it is succesful, then run Π, replacing calls to the broadcast channel with executions of an authenticated broadcast protocol[2]. The resulting protocol takes $t + rt + O(1)$ rounds.

The second transformation is suggested by the constructions of the previous sections. First run Π, replacing all calls to the broadcast channel with executions of `CondGradecast`. Next, run a detectable broadcast protocol to attempt to agree on whether or not all the executions of `CondGradecast` were succesful (i.e. each player uses the protocol to broadcast the logical AND of his grades from the executions of `CondGradecast`). If all of the detectable broadcasts complete successfully with the message 1, then accept the result of Π; otherwise, abort. The resulting protocol takes $2r + t + O(1)$ rounds.

[2]Note that when using an authenticated broadcast protocol several times in a row, sequence numbers need to be added to the broadcast messages to avoid replay attacks.

Remarks: If protocol Π achieves unconditional security then pairwise secure channels are required for both these transformations. For computational security, authenticated channels are sufficient. Moreover, when applying the first transformation in the computational setting, no bound is needed ahead of time on the number of rounds of interaction (though it should nonetheless be polynomial in the security parameter).

At an intuitive level, the first transformation preserves any security properties of Π, except for robustness and zero error: robustness is lost since the adversaries can force the protocol to fail by interfering with the initial precomputation protocol, and zero error is lost since detectable broadcast must have some small probability of error when $t > n/3$ (by the result of Lamport [24]). In the following section, we formalize this intuition for the case of secure multi-party computation.

The second transformation is more problematic: if the protocol fails partway through, the adversary may learn information he is not supposed to. Moreover, if the protocol has side effects such as the use of an external resource, then some of those side effects may occur even though the protocol aborts. Nonetheless, in the case of multi-party computing, the transformation may be modified to avoid these problems.

6.1 Definitions of Security

Informally, an MPC protocol allows players with inputs $x_1, ..., x_n$ to collectively evaluate a function $f(x_1, ..., x_n)$ such that cheaters can neither affect the output (except by their choice of inputs), nor learn anything about honest players' inputs except what can be gleaned from the output. For simplicity, we consider deterministic, single-output functions (this incurs no loss of generality). Also, for this section of the paper, we restrict our attention to static adversaries (i.e. the set of corrupted players is decided before the protocol begins).

The security of multi-party computation is usually defined via an *ideal model* for the computation and a *simulator* for the protocol. In the ideal model, a trusted third party ($\mathcal{TTP}$) assists the players, and the simulator transforms any adversary for the real protocol into one for the ideal model which produces almost the same output[3]. The standard ideal model for MPC when $t \geq n/2$ essentially operates as follows [17]: The players hand their inputs to the $\mathcal{TTP}$, who computes the output (which is a special symbol $\perp$ if any of the corrupted parties refuse to cooperate). If Player 1 is honest, then all parties get the output. If Player 1 is corrupted, then the adversary $\mathcal{A}$ first sees the output and then decides whether or not to abort the protocol. If $\mathcal{A}$ decides to abort, then the $\mathcal{TTP}$ sets the output to $\perp$. Finally, the $\mathcal{TTP}$ hands this possibly aborted output to the honest parties.

The notion of security corresponding to this ideal model is called "the second malicious model" in [17], and "secure computation with abort" in [20]. For consistency with previous sections, we refer to it as *detectable* MPC with *partial fairness*. This highlights two important properties:

Partial fairness: The definition above does not imply *fairness*, in that the adversary might force the protocol to abort when he has more information on the outcome than the honest players. This is in fact unavoidable for the setting of $t \geq n/2$, since it is unavoidable in the 2-party setting (Cleve [8], Goldwasser and Levin [19]). Nonetheless, the definition does have a weaker property, called *partial fairness*: if a specific player (Player 1) is honest, then in fact the honest players get their outputs if and only if the the cheaters do.

Detectability: The definition also has another desirable property, which is the main focus of this paper: if any honest player aborts, then so do they all. We call this property *detectability* (of success) of the computation.

In Section 6.3, we will construct protocols which do not achieve partial fairness, but do still achieve detectability.

DEFINITION 6 (Detectable MPC without partial fairness). A protocol achieves *Detectable* MPC *without partial fairness* for the function f if it is secure with respect to the following ideal model:

1. All players send their inputs $x_1, ..., x_n$ to the $\mathcal{TTP}$. If Player i is corrupted and refuses to cooperate or sends an invalid input, then set $x_i = \perp$.
2. The $\mathcal{TTP}$ computes $y = f(x_1, ..., x_n)$ (where $f(x_1, .., x_n) = \perp$ if $x_i = \perp$ for any i).
3. The $\mathcal{TTP}$ sends y to the adversary, who responds $y' \in \{y, \perp\}$.
4. The $\mathcal{TTP}$ sends y' to all honest players, who output y'.

6.2 Multi-party Computation for $t \geq n/2$

We first discuss the *feasability* of MPC for $t \geq n/2$. Given a broadcast channel, there is a (fixed-round) protocol for standard MPC (i.e. detectable MPC *with* partial fairness). This comes essentially from Goldreich, Micali and Wigderson [18], though a more careful statement of the definition, protocol and proof of security appears in Goldreich [17]. That work also points out that replacing calls to the broadcast channel with an authenticated broadcast protocol — given a signature infrastructure — does not affect the security of the protocol[4]. Applying the first transformation to this protocol, we obtain:

THEOREM 5. *Suppose that trapdoor one-way permutations exist. Then there is a* MPC *protocol for detectable multi-party computation* with *partial fairness (secure computation with abort) of any efficiently computable function in the model of pairwise authenticated channels, which tolerates any $t < n$.*

PROOF. (sketch) This proof is straightforward, but we sketch it to illustrate the structure of the proof of Theorem 6. Let Π be the original MPC protocol, which assumes broadcast, and let Π' be the transformed protocol.

First, create an intermediate protocol $\bar{\Pi}$ which also assumes a broadcast channel. $\bar{\Pi}$ differs only in the addition of a first round, in which all players broadcast a single bit $b_i \in \{\perp, 1\}$. If any player broadcasts $\perp$, then players do not run Π; instead, they wait until r rounds have passed and abort. The protocol $\bar{\Pi}$ achieves MPC if and only if Π does, since the power to abort the protocol at the beginning can be simulated by handing an invalid input to the $\mathcal{TTP}$ in the ideal model.

[3] The output here is the joint distribution of the adverary's view and the outputs of the honest players.

[4] One point which [17] does not disuss explicitly is that sequence numbers are needed to ensure the independence of the various executions of the authenticated broadcast protocol. This is not a problem since the network is synchronous and so the sequence number is simply the round number.

Second, note that any adversary $\mathcal{A}'$ for Π' can be transformed into an adversary $\bar{\mathcal{A}}$ for $\bar{\Pi}$ by simulating the detectable broadcast and authenticated broadcast protocols, using inputs provided by the real honest parties over the broadcast channel. By the properties of the agreement protocols, the joint output of the honest parites and the adversary will be identical in the Π' and $\bar{\Pi}$. □

6.3 Round-efficient Multi-party Computation

The protocol of Theorem 5 achieves the same definition of security as the original protocol of [18, 17], without the assumption of a broadcast channel. However, it is inefficient—the first transformation multiplies the round complexity of the original protocol by t.

Instead, we consider applying the second transformation to MPC protocols: replace every call to the broadcast channel with an invocation of Conditional Gradecast, and agree at the end of the protocol on whether or not all the broadcasts were successful by running the detectable broadcast of Section 3.

As mentioned above, this transformation must be made carefully. In multi-party computing, it is important that the transformed protocol not leak any more information than did the original protocol. If honest players continue to run the protocol after an inconsistent Conditional Gradecast has occurred, the adversary could exploit inconsistencies to learn secret values (say by seeing both possible answers in some kind of cut-and-choose proof). To ensure that this is not a problem, once an honest player has computed a grade $g_i = 0$ in one of the executions of `CondGradecast`, he no longer continues running the original protocol Π. Instead, he only resumes participation during the final detectable broadcast phase, in which players agree on whether to accept or reject the final result.

The resulting protocol Π' does not achieve *partial fairness* (this was pointed out by Yehuda Lindell); the adversary can wait until he has seen the outcome of the protocol to decide whether or not to force it to abort. Nonetheless, the protocol does satisfy Definition 6.

THEOREM 6. *The resulting protocol* Π' *achieves detectable* MPC *without partial fairness in the model of pairwise authenticated channels.* Π' *takes* $2r + t + O(1)$ *rounds, where* r *is the round complexity of the original protocol* Π.

Before proving Theorem 6, we compare this result to the independent work of Goldwasser and Lindell [20]. They consider a general transformation which takes any protocol in a broadcast model, and replaces broadcasts with calls to a protocol identical to `CondGradecast`. They show that, suprisingly, the resulting protocol still satisfies a meaningful definition of security—it retains partial fairness, but loses detectability (the honest parties may not always agree on whether or not the protocol aborted). Their transformation has two additional benefits, however. First, it only triples the round complexity of the original protocol. Second, combined with known results, it yields protocols which are concurrently composable.

In contrast, we obtain slightly less efficient protocols which lose partial fairness, but retain the property that players agree on the success or failure of the protocol—the main focus of this paper.

PROOF. (sketch) Note that the structure of this proof is almost identical to that of the analogous theorem in [20], which considers a similar transformation. The proof in [20] has the advantage of being considerably more formal and detailed.

Let Π be the original MPC protocol, which assumes broadcast channels. We first transform Π into an intermediate protocol $\bar{\Pi}$ which also assumes a broadcast channel, but which only achieves detectable MPC without partial fairness—nothing stronger. Next, we show that any adversary $\mathcal{A}'$ for Π' (in the authenticated channels model) can be transformed into an adversary $\bar{\mathcal{A}}$ for $\bar{\Pi}$ (in the broadcast model) such that the output of $\bar{\mathcal{A}}$ is indistinguishable from that of $\mathcal{A}'$.

To obtain $\bar{\Pi}$ from Π:

1. Make all communication go over the broadcast channel. (There is no loss of security since we are not assuming secure channels for Π).
2. Add an extra round at the end of the protocol, in which all players broadcast a single bit $b_i \in \{\perp, 1\}$. Player i broadcasts $b_i = \perp$ if his output is $\perp$, and 1 otherwise. If any player broadcasts $\perp$, then all honest players set their output to $\perp$. (Note that all honest players already agree on their output. This additional round simply allows the adversary additional power to force an abort).
3. Modify the protocol so that any player can ask any other player to abort. If requested to abort, Player i will stop running the protocol and wait until the final round, at which time he will broadcast $b_i = \perp$.

$\bar{\Pi}$ achieves detectable MPC *without* partial fairness if Π does. Intuitively, this is because Π simply allows the adversary additional opportunities to force an abort; however, a formal proof requires care (see, e.g., [20] for a rigorous proof of the analgous theorem for their transformation). Note that $\bar{\Pi}$ has been forced to *not* achieve partial fairness.

Now consider any adversary $\mathcal{A}'$ for the protocol Π. We construct an adverary $\bar{\mathcal{A}}$ for $\bar{\Pi}$ who runs a simulated copy of $\mathcal{A}'$. For each round of $\bar{\Pi}$, $\bar{\mathcal{A}}$ waits until the simulated honest parties have sent their messages over the broadcast channel. Now $\bar{\mathcal{A}}$ simulates n simultaneous executions of `CondGradecast`, using the broadcast values as inputs for the honest players. For each simulated honest Player i which aborts in one of the executions of `CondGradecast`, $\bar{\mathcal{A}}$ sends a request to abort to the real Player i. Finally, for any `CondGradecast` which succeeded (for at least one honest player), $\bar{\mathcal{A}}$ sends the corresponding value over the real broadcast channel.

In the final Detectable Broadcast protocol, $\bar{\mathcal{A}}$ again simulates the honest players, using the values they broadcast in the final round as inputs. If the all the detectable broadcasts complete successfully with output 1, then $\bar{\mathcal{A}}$ sends 1's over the broadcast channel for the corrupted players; otherwise, $\bar{\mathcal{A}}$ has all the corrupted players send $\perp$.

By the properties of Conditional Gradecast and Detectable Broadcast, the output of the simulated $\mathcal{A}'$ will indistinguishable from that of $\mathcal{A}'$ when running in the real model, against Π'. □

Recently, it has been observed that combining various known techniques, one can construct protocols for detectable MPC which require only $O(\log n)$ rounds ([23], from [2, 3, 7, 11]). Combining this with our transformation, we obtain protocols which require $O(t + \log n)$ rounds of communication, without the assumption of a broadcast channel.

7. NON-UNISON STARTING AND PKI BOOTSTRAPPING

For all previous "detectable protocols" it was implicitly assumed that all players start the protocol in the same communication round. This requires agreement among the players on which protocol is to be run and on a point of time when the protocol is to be started. We now argue that this assumption is unnecessary — not even agreement on the player set among which the protocol will be run.

Coan, Dolev, Dwork, and Stockmeyer [9] gave a solution for the "Byzantine agreement with non-unison start" problem, a problem related to the firing squad problem introduced by Burns and Lynch [5]. Given the setup of a consistent PKI, their protocol achieves broadcast for $t < n$ even when not all honest players start the protocol during the same round, i.e., the broadcast can be initiated by any player on the fly. It turns out that this idea is also applicable to detectable broadcast. This allows a player p_I who shares authenticated (or secure) channels with all members of a player set P' to (unexpectedly) initiate a protocol among the players in $P = P' \cup \{p_I\}$ for a detectable precomputation. Let such a player p_I be called the *initiator* and the players in P' be called the *initiees* of the protocol. The following protocol description is split into the initiator's part and the initiees' part.

Protocol `InitAdHocComp`: [Initiator p_I]

1. Send to P' an initiation message containing a unique session identifier id, the specification of the player set P' and of a multi-party computation protocol Π among player set $P = P' \cup \{p_I\}$.
2. Perform protocol `DetPrecomp` among P' to precompute for all broadcast invocations required by protocol Π. In the final "broadcast round", instead of broadcasting the value G_I to indicate whether all conditional gradecast protocols achieved broadcast, the value $G_I \wedge S_I$ is broadcast where S_I indicates whether all players in P' synchronously entered the precomputation protocol and always used session identifier id.
3. Accept and execute protocol Π if and only if $G_I \wedge S_I$ and all players in P' broadcasted their acceptance at the end.

Protocol `AdHocComp`: [Initiee p_i]

1. Upon receipt of an initiation message by an initiator p_I:
   - decide whether you are interested to execute an instance of Π among player set P.
   - check that the specified id is not being used in any concurrent invocation.
   - check whether $p_i \in P'$.
   - check whether there are authenticated (or secure) channels between p_i and all other players in $P' \cup \{p_I\}$ as required by protocol Π.
2. If all checks in Step 1 were positive then perform protocol `DetPrecomp` among P' to precompute for all broadcast invocations required by protocol Π. In the final "broadcast round", instead of broadcasting the value G_i to indicate whether all conditional gradecast protocols achieved broadcast, the value $G_i \wedge S_i$ is broadcast, where S_i indicates whether all players in P' synchronously entered the precomputation protocol and always used session identifier id.
3. Accept and execute protocol Π if and only if $G_i \wedge S_i$ and all players in P' broadcasted their acceptance at the end.

Note that the first check in Step 1 of Protocol `AdHocComp` implicitly prevents the adversary from "spamming" players with initiations in order to overwhelm a player with work load. A player can simply ignore initiations without affecting consistency among the correct players.

THEOREM 7. *Suppose there is a player set P' and a player p_I such that p_I shares authenticated (or secure) channels with every player in P' (whereas no additional channels are assumed between the players in P'). Then p_I can initiate a protocol among player set $P = P' \cup \{p_I\}$ that achieves the following properties for $t < n$:*

- *All honest players in P either commonly accept or reject the protocol (instead of rejecting it is also possible that a player ignores the protocol, which is an implicit rejection). If they accept, then broadcast or multi-party computation will be achievable among P (with everybody knowing this fact).*
- *If all players in P are honest and all players are connected by pairwise authenticated (or secure) channels then all players accept.*

In particular, such a protocol can be used in order to detectably set up a consistent PKI without the need for a trusted party.

8. ACKNOWLEDGEMENTS

We thank Jon Katz, Idit Keidar, and Rafi Ostrovsky for helpful discussions. We thank Shafi Goldwasser and Yehuda Lindell for very helpful comments on drafts of this work and also for showing us an early version of [20]. The work of Adam Smith was supported by U.S. Army Research Office Grant DAAD19-00-1-0177.

9. REFERENCES

[1] H. Barnum, C. Crépeau, D. Gottesman, A. Smith, and A. Tapp. Authentication of quantum messages. Manuscript, 2001.

[2] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *Advances in Cryptology: CRYPTO '89*, pages 589–590, Berlin, Aug. 1990. Springer.

[3] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. of 22nd ACM Symposium on the Theory of Computing (STOC)*, pages 503–513, 1990.

[4] C. H. Bennett and G. Brassard. An update on quantum cryptography. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 475–480. Springer-Verlag, 1985, 19–22 Aug. 1984.

[5] J. E. Burns and N. A. Lynch. The byzantine firing squad problem. In F. P. Preparata, editor, *Advances in Computing Research, Parallel and Distributed Computing*, volume 4, pages 147–161. JAI Press, Inc., Greenwich, Conn., 1987.

[6] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Proc. 31st ACM Symposium on the Theory of Computing (STOC)*, pages 639–648, 1996.

[7] B. Chor and M. Rabin. Achieving independence in logarithmic number of rounds. In *Proc. 6th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 260–268, 1987.

[8] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 364–369, Berkeley, California, 28–30 1986.

[9] B. A. Coan, D. Dolev, C. Dwork, and L. Stockmeyer. The distributed firing squad problem. *SIAM Journal on Computing*, 18(5):990–1012, Oct. 1989.

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

[11] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *Proc. of CRYPTO 1989*, pages 526–544, 1989.

[12] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, Aug. 1997.

[13] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1:26–39, 1986.

[14] M. Fitzi, J. A. Garay, U. Maurer, and R. Ostrovsky. Minimal complete primitives for unconditional multi-party computation. In *Advances in Cryptology — CRYPTO 2001*, Lecture Notes in Computer Science, 2001.

[15] M. Fitzi, N. Gisin, and U. Maurer. Quantum solution to the Byzantine agreement problem. *Physical Review Letters*, 87(21), Nov. 2001.

[16] M. Fitzi, N. Gisin, U. Maurer, and O. von Rotz. Unconditional Byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Advances in Cryptology — EUROCRYPT 2002*, Lecture Notes in Computer Science, 2002.

[17] O. Goldreich. Secure multi-party computation, working draft, version 1.2, Mar. 2000.

[18] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on the Theory of Computing (STOC)*, pages 218–229, 1987.

[19] S. Goldwasser and L. A. Levin. Fair computation of general functions in presence of immoral majority. In *CRYPTO*, pages 77–93, 1990.

[20] S. Goldwasser and Y. Lindell. Secure computation without a broadcast channel. http://eprint.iacr.org/2002/040.ps, Apr. 2002.

[21] D. Gottesman and I. Chuang. Quantum digital signatures. Manuscript, 2001.

[22] A. Karlin and A. C. Yao. Manuscript.

[23] J. Katz and R. Ostrovsky. Personal communication.

[24] L. Lamport. The weak Byzantine generals problem. *Journal of the ACM*, 30(3):668–676, July 1983.

[25] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[26] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, Apr. 1980.

[27] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t >= n/3$. Research report, IBM Research, Nov. 1996. Submitted for Publication.