Univ. Karlsruhe, Febr. 7, 1991
STACS '91, Hamburg, Febr., 1991.

Unconditional Byzantine Agreement with Good Majority

Birgit Baum-Waidner, Birgit Pfitzmann, Michael Waidner

Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe
Postfach 6980, D-7500 Karlsruhe 1, FRG

Abstract. We present a protocol which achieves Byzantine Agreement (BA) if less than *half* of the processors are faulty and which does *not* rely on unproved computational assumptions such as the unforgeability of digital signatures. This is the first protocol which achieves this level of security.

Our protocol needs reliable broadcast and secret channels in a precomputation phase. For a security parameter k, it achieves BA with an error probability exponentially small in k, whereas all computations are polynomial in k and the number of processors, n. The number of rounds is linear in k and independent of n. The length of the precomputation phase is linear in n and proportional to the number of BAs based on it.

As a subprotocol, we present a coin flipping protocol on the same assumptions.

1 Introduction

1.1 Byzantine Agreement

Byzantine **A**greement **p**rotocols (BAPs) are an important primitive for distributed computations. They are intended to achieve reliable broadcast where this is not available physically and some processors may be faulty.

Correct agreement on the value "0" or "1" of a sender means [PeSL_80]:

a) All good processors agree on the same value $v \in \{0,1\}$.
b) If the sender is good, v is the value the sender meant to send.

Let n be the number of processors and t an upper bound on the number of faulty processors. Faulty processors are assumed to be malicious. Except for preventing good processors from communicating, they can do whatever they like.

In a strong sense, BA can be achieved if and only if $3t < n$ [PeSL_80]. More faults can be tolerated if one weakens the requirements: For the case where

- a small probability of error is acceptable,
- secure digital signatures exist, and
- reliable broadcast is available in a precomputation phase (for reliably distributing test keys of the signature scheme),

[PeSL_80] provides a BAP for arbitrary t with $t < n$. Such a BAP is usually called authenticated.

Remark: According to [FeMi_88], DOLEV and DWORK have proved that for tolerating $3t \geq n$, a precomputation phase using reliable broadcast is necessary.

Unfortunately, *all* known BAPs which tolerate a t with $3t \geq n$ are based on computational assumptions (e.g., "factoring is hard"), and *all* these assumptions are *unproved*. But even if one of them could be proved (which would imply a proof of P $\neq$ NP!), these BAPs require that faulty processors are polynomially bounded.

Hence, till now, there has been no really verified BAP for $3t \geq n$, and none which tolerates computationally unlimited faulty processors. In the following, we fill both gaps for $2t < n$.

A BAP which tolerates computationally unbounded faulty processors is called **unconditional**.

Remark: The first step towards more secure BA for $3t \geq n$ was taken in [WaPf_89]: There, an unconditional BAP for $3t < n$ is combined with an authenticated BAP for $t < n$ so that the combination achieves correct agreement if any of the two basic protocols would do so. The same construction can be applied to the following unconditional BAP for $2t < n$.

Remark: The BAPs of [PeSL_80] are not very efficient. Efficient protocols for $3t < n$ are, e.g., proposed in [DFFL_82, FeMi_88, GoPe_90]; an efficient authenticated protocol for $t < n$ is proposed in [DoSt_83].

Effect on multi-party computations: Provided reliable broadcast is available, [RaBe_89] describes how *any* multi-party protocol $\mathcal{P}$ can be transformed into a protocol $\mathcal{P}'$, so that $\mathcal{P}'$ ensures the privacy of each good processor's input and tolerates $t < n/2$ maliciously faulty processors. Together, this result and ours imply that the transformed protocol $\mathcal{P}'$ can be executed without reliable broadcast, if reliable broadcast was available in a precomputation phase. This precomputation phase can be executed before each processor knows its inputs to $\mathcal{P}$.

1.2 Overview

Our BAP is based on TAL RABIN's verifiable secret sharing scheme (VSS) for $2t < n$ [Rabi_88, RaBe_89]. However, RABIN assumes the availability of reliable broadcast for sharing and revealing secrets. It is fundamental for our BAP that for *revealing* a secret, reliable broadcast is not used. In §2, we present a modified protocol, VSS*, where this is the case.

We first use VSS* to construct an unconditional BAP for $2t < n$ with linearly small error probability (§3, Theorem 3). Part of this BAP is a coin flipping protocol which tolerates $2t < n$.

By iterating the protocol from §3, we achieve unconditional BA with exponentially small error probability for $2t < n$ (§4, Theorem 4).

In §5, we discuss an approach for achieving unconditional BA for arbitrary $t < n$.

1.3 Notations and Assumptions

Let $F = \mathrm{GF}(p)$, p prime, a large finite field, $P := \{P_1, \ldots, P_n\}$ the set of all processors, $\alpha_1, \ldots, \alpha_n$ fixed different elements of F, Π_t the set of all polynomials $f \in F[X]$ of degree $\leq t$, and for each $s \in F$ let $\Pi_t(s)$ be the set of all $f \in \Pi_t$ with $f(0) = s$. Let $k \in \mathbb{N}$ be a security parameter so that an error probability of 2^{-k} is acceptable.

All our protocols will be polynomial in $\log(p)$ and n. We will call a probability $Prob(p, n)$ "exponentially small" iff there is a polynomial $Q \in \mathbb{Z}[X,Y]$ so that $Prob(p, n) < 2^{-k}$ for all $k, n \in \mathbb{N}$, and all primes p with $\log(p) \geq Q(k, n)$. (The reason is that by choosing $\log(p) \geq Q(k, n)$, we obtain the desired error probability and a protocol polynomial in k and n.) Similarly, $Prob(p, n)$ is called "exponentially close to δ" iff $(\delta - Prob(p, n))$ is exponentially small.

If a statement is true with exponentially small probability, we say that it holds almost certainly.

As usual with BAPs, we assume that each pair of good processors can communicate securely. We also assume that the network is synchronous. Additionally, we assume that in a precomputation phase, i.e. *before* the value which is to be distributed by a BAP is chosen,

- each processor can reliably broadcast some information (like public keys of a signature scheme)
- each pair of processors has a secret and secure channel.

With these additional assumptions, the assumption that each pair of good processors can communicate securely during the protocol can be reduced to the assumption that faulty processors cannot prevent good processors from communicating (authentication codes, [GiMS_74, WeCa_81]).

2 A Modification to RABIN's VSS

The basis of our protocols is a modification to RABIN's verifiable secret sharing scheme (VSS) [Rabi_88, RaBe_89] which does not need reliable broadcast while the secret is revealed.

In §2.1, we informally define VSS, and describe what RABIN's VSS achieves and what is absolutely necessary to know about our modification to RABIN's VSS to understand §3 and §4. In §2.2, we repeat RABIN's VSS, and in §2.3 and §2.4, we describe our modification.

2.1 Some facts about verifiable secret sharing

A simple secret sharing scheme with threshold $t < n$ enables a processor, the dealer D, to share a secret $s \in F$ among all the other processors so that no set of t processors learns anything about the secret, but any set of t+1 processors can reconstruct the secret. There are schemes which preserve the privacy of s unconditionally [Blak_79, Sham_79] (cf. §2.2).

Simple secret sharing does not consider fault tolerance: A faulty dealer may distribute inconsistent shares so that s cannot be reconstructed unambiguously, or faulty processors may publish wrong shares so that a wrong "secret" is reconstructed. *Verifiable* secret sharing (VSS) avoids these problems [CGMA_85]: If any good processor thinks that a secret has been shared, it is guaranteed that it has indeed been shared correctly; and the good processors can reconstruct s, no matter how the t faulty processors behave.

Remark: VSS can only be achieved if $2t < n$, and like BA, VSS without any probability of error can only be achieved for $3t < n$ [BeGW_88], even if reliable broadcast is available [ChCD1_88].

RABIN's VSS is the first unconditional VSS for $2t < n$ [Rabi_88, RaBe_89]. It is polynomial in k and n, sharing a secret needs a number of rounds linear in n, revealing a secret needs a constant number of rounds, and the error probability is exponentially small. For reasons of completeness, RABIN's VSS, with one small change, is precisely described in §2.2.

RABIN's VSS uses reliable broadcast to distribute shares and to reveal secrets. In §2.3 and §2.4 we present a modified protocol VSS* where for revealing secrets, reliable broadcast is not used. Apart from that, it achieves exactly the same as RABIN's VSS.

One can understand our agreement protocols (§3 and §4) without reading the following three sections, if one believes the following: If a secret s has been shared successfully, each good processor P_i knows a share β_i of s, together with some check information which will convince each other good processor P_j that β_i was the share which P_i received from the dealer. Thus, if some processors send their shares and their check information to P_j, P_j can almost certainly decide which shares are correct. If P_j receives at least t+1 correct shares, it can reconstruct the correct s.

Apart from VSS*, our BAPs use a subprotocol WSS*, which is more efficient than VSS*, but less powerful. To understand our BAPs, WSS* can be replaced by VSS* in §3 and §4. (In §2.4, however, WSS* is essential as a subprotocol of VSS*.)

2.2 RABIN's VSS

RABIN's VSS is based on a simple, i.e. non-verifiable, secret sharing scheme [Blak_79, Sham_79]: To **share** the secret $s \in F$, dealer D chooses a polynomial $f \in \Pi_t(s)$ randomly and sends the **piece** $\beta_i := f(\alpha_i)$ to P_i, $i = 1, \ldots, n$. To **reveal** the secret, f is interpolated from any t+1 pieces. No set of less than t+1 pieces contains any information about s.

For *verifiably* sharing the secret s, RABIN uses the following construction: D shares s using a random polynomial $f \in \Pi_t(s)$, and each P_i commits to the piece β_i received from D. This commitment is made by a variant of secret sharing, called *weak secret sharing* (WSS). The correctness of both levels of secret sharing is verified in zero-knowledge. For revealing the secret, at least the commitments of all good processors are opened and f is interpolated from all correctly opened pieces β_i.

In the following, we describe this construction precisely and bottom-up. (For more explanations, cf. [RaBe_89].) We assume the availability of reliable broadcast for sharing *and* revealing secrets, and we assume $2t < n$.

The core of RABIN's WSS is **information checking** (IC), which is a weak, but unconditionally secure, substitute for digital signatures: Consider three processors. Intermediary *INT* receives β from dealer D. Until *INT* forwards β to recipient R, β must be kept secret. If D signed β, R could test whether β is authentic, and *INT* would be sure that R will accept β. IC simulates this situation without relying on cryptographic assumptions [RaBe_89, extracted from WSS]:

RABIN's IC Protocol

In all steps: if a processor does not send a required message, it is *disqualified*.

Phase 1: Preparing and verifying check vectors

[1] D chooses $\boldsymbol{b} \in (F\backslash\{0\})^{2k}$ and $\boldsymbol{y} \in F^{2k}$ randomly, determines $\boldsymbol{c} \in F^{2k}$ by computing $c_\iota := \beta + b_\iota y_\iota$, $\iota = 1, \ldots, 2k$, and sends $(\beta, \boldsymbol{y})$ to *INT* and the **check vector** $(\boldsymbol{b}, \boldsymbol{c})$ to R.

[2] *INT* verifies by a cut-and-choose technique [Rabi_78] that the check vectors have been formed correctly:

- *INT* chooses k indices $J \subset \{1, \ldots, 2k\}$ and broadcasts J.
- As an answer, R broadcasts $(\iota, b_\iota, c_\iota)$ for all $\iota \in J$.
- D checks R's answer and broadcasts either an approval or the correct $(\boldsymbol{b}, \boldsymbol{c})$. In the latter case, R takes this public $(\boldsymbol{b}, \boldsymbol{c})$ as its check vector.

INT publicly rejects this proof if either D has approved R's answer, but $c_\iota \neq \beta + b_\iota y_\iota$ for at least one $\iota \in J$, or D has broadcast a new check vector $(\boldsymbol{b}, \boldsymbol{c})$ and for at least one ι, $c_\iota \neq \beta + b_\iota y_\iota$.

[3] If *INT* rejects the proof, or if *INT* is disqualified, D broadcasts β. In this case, β is called a **public piece**.

Phase 2: Forwarding β to R

[1] If β is not a public piece, *INT* sends $(\beta, \boldsymbol{y})$ to R. R accepts β if $c_\iota = \beta + b_\iota y_\iota$ holds for at least one $\iota \notin J$, or β is a public piece.

Remark: The main difference to signatures is that reliable broadcast must still be available when D chooses the value β.

From [RaBe_89] we obtain the following IC-Lemma. It says that if at most one of the three processors D, *INT*, R is faulty, IC works correctly almost certainly:

Lemma 1 (IC-Lemma): a) If D and *INT* are good, *INT* accepts in Phase 1 and β is kept perfectly secret until Phase 2. **b)** If *INT* and R are good and D has not been disqualified in Phase 1, R accepts β in Phase 2 almost certainly. **c)** If D and R are good and R accepts β in Phase 2, then β is the authentic piece almost certainly. **d)** The protocol is polynomial in k and needs a constant number of rounds. ♦

Proof: a) - c) [RaBe_89, Lemmata 1-3]. **d)** Trivial. □

Next we consider **weak secret sharing (WSS)** [RaBe_89, WSS]. This results from simple secret sharing, if D ensures by IC that each P_i can convince each P_j of the authenticity of β_i. Thus D commits itself to s.

RABIN's WSS Protocol

Phase 1: Sharing the secret (≈ committing to s)

[1] D chooses $h \in \Pi_t(s)$ randomly and computes $\beta_i := h(\alpha_i)$, $i = 1, \ldots, n$. For each pair of two processors
(P_i, P_j), Phase 1 of IC with $(\beta, D, INT, R) = (\beta_i, D, P_i, P_j)$ is performed. (Thereby, β_i may become

public.) The corresponding check information ***y*** is called $\boldsymbol{y_{ij}}$.

Phase 2: Revealing the secret (≈ opening the commitment)

[1] D broadcasts polynomial h.

[2] Each P_i hands to each P_j its piece β_i, if β_i is not public yet, and the corresponding check information $\boldsymbol{y_{ij}}$.

[3] Each P_j checks, using its check vectors, for each received pair $(\beta_j, \boldsymbol{y_{ji}})$, if it accepts this piece β_j.

[4] If P_j has accepted a piece β_i or knows a public piece β_i with $h(\alpha_i) \neq \beta_i$, it broadcasts a vote to disqualify D.

[5] If at least t+1 processors have voted against D, D is disqualified. Otherwise, $h(0)$ is taken to be D's secret.

Lemma 2 (WSS-Lemma): After Phase 1, **a)** if D is good, the faulty processors have no information about s, **b)** no good processor has been disqualified, **c)** the good processors will almost certainly accept each other's pieces, and **d)** good processors will only accept authentic pieces, almost certainly.

e) After Phase 2, almost certainly no good processor has been disqualified and all good processors agree on whether to disqualify D or to accept $h(0)$. In the latter case, after Phase 1, the good processors' pieces β_i, and the public pieces, must have lain on this h.

f) The protocol is polynomial in k and n, and needs a constant number of rounds. ♦

Proof: a) [RaBe_89, Lemma 3]. **b)** Trivial. **c)** Lemma 1b. **d)** Lemma 1c. **e)** [RaBe_89, proof of the WSS Theorem]. **f)** The first part is obvious. All executions of IC Phase 1 in WSS Phase 1 and of IC Phase 2 in WSS Phase 2 Step [2] can be parallelized. Thus all steps of WSS need a constant number of rounds. □

VSS is now obtained by introducing the second level of secret sharing, and a zero-knowledge proof of the fact that D has really shared a secret and that all the commitments are correct [RaBe_89, VSS] (similar to [ChCD1_89]):

RABIN's VSS Protocol

Phase 1: Sharing the secret

[1] D chooses $f \in \Pi_t(s)$ randomly, and sends $\beta_i := f(\alpha_i)$ to P_i, $i = 1, \ldots, n$.

[2] Each P_i shares β_i among all the other processors by Phase 1 of WSS, using a random polynomial $h_i \in \Pi_t(\beta_i)$. If P_i is disqualified in WSS, D broadcasts β_i.

The correctness of Steps [1] and [2] is verified by a cut-and-choose procedure, which is performed in at most t+1 iterations. The iteration counter τ is a global variable, and initially, $\tau = 1$. Each iteration consists of at most n "small" cut-and-choose procedures; for the v-th small procedure, P_v acts as verifier.

[3.τ] D chooses kn polynomials $g_{v,1}, \ldots, g_{v,k} \in \Pi_t$, $v = 1, \ldots, n$, randomly and computes the pieces $\gamma_{v,u,i} := g_{v,u}(\alpha_i)$. For each P_i whose β_i is not public, D sends all $\gamma_{v,u,i}$'s to P_i. Otherwise, D broadcasts these $\gamma_{v,u,i}$'s.

[4.τ] Each P_i whose β_i is not public shares each $\gamma_{v,u,i}$ by WSS, using a random polynomial $h_{v,u,i} \in \Pi_t(\gamma_{v,u,i})$. P_i also shares $\delta_{v,u,i} := \beta_i + \gamma_{v,u,i}$ by WSS, using the polynomial $h_i + h_{v,u,i} \in \Pi_t(\beta_i + \gamma_{v,u,i})$. For this, each P_j computes its share locally as the sum of its shares in β_i and $\gamma_{v,u,i}$; P_i only distributes the check information and vectors. If P_i is disqualified during WSS, all good processors go to Step [6.τ].

[5.τ] Each P_v ($v = 1, \ldots, n$) who has not been disqualified decides randomly, for each $u = 1, \ldots, k$, to ask D to broadcast either $g_{v,u}$ or $g_{v,u} + f$.

- Verification of the public pieces: For each public piece β_j, each P_i checks $\gamma_{v,u,j} = g_{v,u}(\alpha_j)$ or $\delta_{v,u,j} = (g_{v,u} + f)(\alpha_j)$, resp. If one of these equations does not hold, all good processors disqualify D.

- Verification of "D has distributed consistent pieces": Each P_i checks $\gamma_{v,u,i} = g_{v,u}(\alpha_i)$ or $\delta_{v,u,i} = (g_{v,u} + f)(\alpha_i)$, resp. If P_i is not satisfied, it broadcasts this.
- Verification of "P_i has committed to β_i": Each P_i whose piece is not public reveals, using WSS, $\gamma_{v,u,i}$ or $\delta_{v,u,i}$, resp. If $\gamma_{v,u,i}$ or $\delta_{v,u,i}$ has been revealed successfully, but does not lie on $g_{v,u}$ or $g_{v,u} + f$, P_i is disqualified.

[6.τ] If, during Step [4.τ] and [5.τ], no processor was disqualified and all P_i's were satisfied with their pieces, all good processors decide that a secret has been shared successfully.

Otherwise, for each P_i which was disqualified or not satisfied in Step [4.τ] or [5.τ], D broadcasts β_i. If t+1 pieces β_i are now public, or the public pieces β_i do not lie on one polynomial in Π_t, all good processors disqualify D. Otherwise they set $\tau := \tau + 1$ and return to Step [3.τ].

If D has been disqualified, all good processors decide that no secret has been shared.

Phase 2: Revealing the secret.

[1] Each P_i reveals its piece β_i by Phase 2 of WSS, if β_i is not public yet.

[2] Each P_i forms a set S_i of all the pieces which it has accepted in Step [1] and the public pieces.

[3] Each P_i may take any t+1 pieces from S_i, interpolate the polynomial f, and take $f(0)$ to be the secret s.

Remark: In [Rabi_88, RaBe_89], a slightly different cut-and-choose procedure is used: Steps [3.τ] - [6.τ] are performed only once, and the nk questions in Step [5.τ] are asked and answered in nk subsequent rounds. We did not see that the public pieces would be sufficiently verified in that version (the proof of [RaBe_89, Lemma 7] never mentions them): If D and P_i are faulty, P_i may claim it is not satisfied in the very last round. D then broadcasts a β_i which does not lie on f. D can adapt the $\gamma_{u,v,i}$'s, which it must also broadcast, to the wrong β_i: if D has broadcast $g_{u,v}$ in Step [5.τ], it broadcasts $\gamma_{u,v,i} = g_{u,v}(\alpha_i)$, otherwise $\gamma_{u,v,i} := \delta_{u,v,i} - \beta_i$.

Lemma 3 (VSS-Lemma): After Phase 1, **a)** if D is good, the faulty processors have no information about s, **b)** no good processor has been disqualified, **c)** if the good processors decide that a secret has been shared, then all pieces held by non-disqualified processors, together with the public pieces, lie on one polynomial $f \in \Pi_t$, almost certainly.

d) If D has not been disqualified in Phase 1, then in Phase 2, all good processors reconstruct the polynomial f, and thus the correct secret, almost certainly.

e) The protocol is polynomial in k and n. The number of rounds of Phase 1 is linear in t, the number of rounds of Phase 2 is constant. ♦

Proof: a) [RaBe_89, Lemma 5]. **b)** [RaBe_89, Lemma 6, proof of the Main Theorem, Part 1]. **c)** In each iteration, at least $(n - t)k$ qestions in Step [5,τ] are chosen randomly, thus the cut-and-choose argument from [RaBe_89, Proof of Lemma 7] applies: Assume the good processors' pieces, together with the public pieces, do not lie on a polynomial $f \in \Pi_t$. Then in Step [5.τ], almost certainly either all good processors detect an inconsistency for a public piece and disqualify D, or a good processor P_j will not be satisfied; thus D broadcasts β_i, and the (τ+1)-st iteration is performed. **d)** [RaBe_89, proof of the Main Theorem, Part 2]. **e)** In each iteration, one new piece is public. Thus there are at most t+1 iterations. The rest is trivial counting. □

2.3 Very weak secret sharing

We first present a modification to WSS which needs no reliable broadcast while the secret is revealed. Its properties are similar to those of WSS, with one difference: If the dealer is faulty, some good processors may accept a secret (which is then the correct one), while others disqualify the dealer. We therefore call it *very weak secret sharing*. It is abbreviated by WSS*.

For §3, we have to provide for a situation where not all good processors take part in Step [1] of Phase 2, but in all the other steps.

WSS* Protocol

Phase 1: Sharing the secret. Like Phase 1 of WSS.

Phase 2: Revealing the secret. For each P_i:

[1] If P_i takes part here and its piece β_i is not public yet, it hands to each P_j its piece β_i and the corresponding check information $\boldsymbol{y}_{ij}$. (N.B.: we do not assume reliable broadcast for this step!)

[2] P_i checks, using its check vectors, for each received pair (β_j, $\boldsymbol{y}_{ji}$), if it accepts this piece β_j.

[3] If P_i has accepted at least t+1 pieces β_j, including public pieces and its own piece, it interpolates all accepted pieces by a polynomial h'. If $h' \notin \Pi_t$, P_i decides for itself to disqualify the dealer.

[4] If P_i does not disqualify the dealer, it takes $s = h'(0)$ to be D's secret.

Theorem 1 WSS*-Theorem

a) Before Phase 2, if D is good, the faulty processors have no information about s.

b) If D is good, and a good P_i has accepted at least t+1 pieces, P_i reconstructs the correct s almost certainly.

c) If D is good, and all good processors take part in Step [1] of Phase 2, they all reconstruct the correct s almost certainly.

d) For any D: Assume that after Phase 1, the good processors' pieces β_i, and the public pieces, lie on one polynomial $f \in \Pi_t(s)$. Then, if all good processors take part in Phase 2, each good processor either disqualifies D or reconstructs this s, almost certainly.

e) The protocol is polynomial in k and n, and the number of rounds is constant. ♦

Remark: Theorem 1d is the case where D behaves well in Phase 1, but not necessarily in Phase 2. This case becomes interesting in the context of VSS, where the dealer is forced to behave well in Phase 1 of WSS* by other means.

Proof of Theorem 1:

a) The secrecy only depends on Phase 1, which is unchanged. Thus Theorem 1a follows from Lemma 2a.

b) By Lemma 2d, the pieces are almost certainly authentic. Since D is good, t+1 authentic pieces define the original polynomial. Thus P_i reconstructs the correct s.

c) Lemma 2c and Theorem 1b.

d) Almost certainly, all the good processors accept all these pieces (Lemma 2c). Since there are at least n–t of them, and $n-t \geq t+1$, f is the only polynomial of degree t on which they all fit. Thus all good processors which accept any polynomial accept just this f.

e) Phase 1 is unchanged, and Phase 2 is even shortened. Thus Theorem 1e follows from Lemma 2f. □

2.4 VSS which needs no reliable broadcast while the secret is revealed

The following protocol VSS* is derived from the VSS protocol of [RaBe_89], and needs no reliable broadcast while the secret is revealed. We prove that it has the same properties as the original protocol.

VSS*: VSS Protocol which needs no reliable broadcast while the secret is revealed

Phase 1: Sharing the secret. Like Phase 1 of VSS. (In particular, where secrets are revealed for verification, WSS is used, not WSS*.)

Phase 2: Revealing the secret.

[1] Each P_i takes part in revealing the piece β_j of each other P_j by Phase 2 of WSS*, if β_j is not public yet.

[2] Each P_i forms a set S_i of all the pieces which it has accepted in Step [1] and the public pieces. (Now, $|S_i| \geq t$+1 holds because of Theorem 1c.)

[3]	Each P_i may take any t+1 pieces from S_i, interpolate the polynomial f, and take $f(0)$ to be the secret s.

Theorem 2 VSS*-Theorem

The VSS*-Protocol achieves VSS in the sense of Lemma 3. ♦

Proof of Theorem 2:

The conditions corresponding to Lemma 3a - e are called a' - e', respectively. a' - c' are trivial.

d') According to Theorem 1c, almost certainly each good processor P_j can reveal its piece β_j to all the other good processors in Step [1]. (Thus, in particular, the last sentence in Step [2] is correct.)

According to Theorem 1d, almost certainly, a faulty processor P_j cannot trick any good processor into accepting a piece different from that shared as β_j in Phase 1. Together with Lemma 3c, this implies that good processors only accept pieces lying on the same and correct polynomial (although here, in contrast to [RaBe_89], they do not necessarily have the same pieces). Hence they all recover the secret correctly.

e') The protocol is part of that in [RaBe_89]. Thus e' follows from Lemma 3e. □

3 BA with Linearly Small Error Probability and Coin Flipping

The following protocol needs a precomputation phase using both secret channels and reliable broadcast. Each precomputation phase can be used for just one agreement. We need *no* computational restrictions on the faulty processors and *no* unproved assumptions.

The parameter L determines the number of rounds needed (O(L)) and the error probability ($\approx 1/L$); in §4 we will choose $L = 3$.

Remark: The purpose of the values r_i, s_i, shared in the precomputation phase of the following protocol is: The sum of the r_i's serves as a random coin. This coin determines which of the L steps of Phase 1 determines the final result. Revealing the secret s_i will serve as proof to P_i that at least t+1 processors have taken a preliminary decision on the value "1".

Preliminary Byzantine Agreement Protocol with parameter *L*: For each processor P_i:

Precomputation

P_i selects values r_i, $s_i \in F$ randomly. It shares r_i, using Phase 1 of VSS*, and s_i, using Phase 1 of WSS*.

Phase 0: Distribution

If P_i is the sender, it sends a value $v \in \{0, 1\}$ to each processor.

P_i calls the value received from the sender v_i.

Phase 1: Agreement by revealing secrets

[0,b] If $v_i = 1$, P_i reveals its shares in all the secrets s_j, using Step [1] of Phase 2 of WSS*.

If P_i receives shares from another processor P_k, it checks if it can accept them all (according to Step [2] of Phase 2 of WSS*); if not, it accepts none of them. It forms a vector E_i of all the shares it has accepted. (One entry $E_{i,k}$: all shares of P_k in all s_j 's.) P_i also puts its own shares into E_i if and only if it has decided to publish them.

For l := 1 to L–1:

[l,a] If, after Step [l–1,b], E_i contains at least t+1 entries for the first time, then for each j, P_i uses Step [3] and [4] of Phase 2 of WSS* to recover s_j, and either disqualifies P_j or sends s_j to P_j.

[l,b] If P_i has received its secret s_i for the first time in Step [l,a], and not published its shares in Step [0,b], it publishes them now. This is carried out just as in Step [0,b].

Phase 2: Revealing the coin to fix the deciding step

P_i takes part in revealing all the secrets r_j (using Phase 2 of VSS*) and computes their sum R modulo L. As its final value, P_i takes "1" if E_i contained at least t+1 entries after Step [R], and "0" otherwise.

Lemma 4 (Coin Flipping Lemma): a) Almost certainly, after Phase 2 all good processors agree on R. **b)** Before Phase 2, the faulty processors have no information about R. **c)** The coin R is distributed randomly in $\{0, \dots, L-1\}$, with exponentially small bias. ♦

Theorem 3 Preliminary Byzantine Agreement Theorem

a) If all good processors receive the same value v_i in Phase 0, agreement on this value is achieved almost certainly.

b) Otherwise, agreement is achieved with probability δ exponentially close to $1 - 1/L$.

c) The protocol is polynomial in L, k, and n. The number of rounds of the Precomputation is linear in t. The number of rounds of the BAP is linear in L and independent of k and n. ♦

Proof of Lemma 4:

At least t+1 r_i's are chosen randomly by good processors. Because of Theorem 2, the faulty processors have no information about them when they choose their own r_i's. Thus the sum $R^* := r_1 + \dots + r_n$ is distributed randomly in F. Since $R = R^* \bmod L$, some values have a $1/p$-advantage. This means an exponentially small bias. The rest follows from Theorem 2. □

Remark: Since each set of t+1 processors contains at least one good processor, it would be sufficient if just t+1 processors chose a random value r_i [BrDo_84]. The application of VSS to the problem of coin flipping was first described in [CGMA_85].

Proof of Theorem 3:

a$_1$) Assume that all good processors have $v_i = 1$ after Phase 0. Then in Step [0,b] of Phase 1, they all decide to publish their shares in all the secrets s_j. By Lemma 2c, all good processors accept these shares almost certainly. Thus all good processors have at least t+1 entries in their vectors after this and all following steps. Therefore, no matter what the result of the coin flip in Phase 2 is, almost certainly they all decide for "1".

a$_2$) Assume that all good processors have $v_i = 0$. We will show that, almost certainly, no good processor P_i publishes its secrets. Therefore, no good processor P_j has any share of a good processor in its vector. Thus after any step, P_j has at most t entries. Hence, no matter what the result of the coin flip in Phase 2 is, P_j decides for "0".

Assume, to the contrary, that there is a first Step [l] where a good processor P_i publishes its shares. l cannot be 0, since $v_i = 0$. Hence P_i has received its secret s_i from a processor P_j in Step [l,a], $l > 0$. P_j must be faulty, because no good processor has any share of a good processor in its vector yet. However, since no good processor has published a share yet, the faulty processors can only guess s_i with exponentially small probability of success, and they have only $t(n-t)$ chances per round to send an s_i to a good processor P_i.

b) We show that, almost certainly, there is at most one step [l] in Phase 1 after which there is no agreement. Then the probability that no final agreement is achieved is just the probability that $R = l$. By Lemma 4, this probability is exponentially close to $1/L$.

Assume that there is a first step [l] after which there is no agreement. This means that at least one good participant P_i has at least t+1 entries in its vector, i.e. t+1 shares of each secret. Thus in Step [l+1,a], for each good P_j, P_i recovers s_j correctly and sends it to P_j, almost certainly (Theorem 1b). Consequently, almost certainly, each good P_j decides to publish its shares in [l+1,b]. All good processors accept these shares. Thus they have at least t+1 entries in their vectors after Step [l+1] and all following steps.

c) Since VSS* and WSS* are polynomial in k and n (Theorem 1e, 2), the first part is trivial. Since all secrets r_i, s_i can be shared (Precomputation) and revealed (Phase 1, 2) in parallel, the number of rounds of the Precomputation is linear in t and the number of rounds of the BAP is independent of k and n. Obviously, the number of rounds is linear in L. □

4 BA with Exponentially Small Error Probability

The following protocol mainly consists of k executions of the protocol from §3. Each time, the results of the previous execution are used as starting values in Phase 1. The idea is that if agreement is reached once, all following executions preserve this agreement.

Final Byzantine Agreement Protocol

Precomputation

Each processor P_i selects $2k$ values $r_{i,m}, s_{i,m} \in F$, $m = 1,\ldots,k$, randomly, and shares the $r_{i,m}$'s, using Phase 1 of VSS*, and the $s_{i,m}$'s, using Phase 1 of WSS*.

Max-Phase 0: Distribution

If P_i is the sender, it sends a value $v \in \{0, 1\}$ to each processor.

P_i calls the value received from the sender $v_i^{(0)}$.

For *m* := 1 to *k*:

Max-Phase *m*: Distributed consensus

Phase 1 and 2 of the Preliminary Byzantine Agreement Protocol are executed with $v_i = v_i^{(m-1)}$, $L = 3$, and using the values $r_{i,m}$, $s_{i,m}$. Call P_i's result $v_i^{(m)}$.

Final result

P_i's final result is $v_i^{(k)}$.

Theorem 4 Final Byzantine Agreement Theorem

a) If $2t < n$, the Final Byzantine Agreement Protocol achieves correct agreement almost certainly.

b) The protocol is polynomial in k and n. The number of rounds of the Precomputation is linear in t. The number of rounds of the BAP is linear in k and independent of n. ♦

Proof of Theorem 4:

a) First assume that correct agreement has been achieved after Max-Phase $m < k$. Then in Max-Phase $m+1$, all good processors start with the same values v_i. According to Theorem 3a, the same correct agreement is almost certainly preserved to the end of Max-Phase $m+1$.

Hence, the faulty processors must successfully prevent agreement in all Max-Phases in order to prevent final agreement. Each time, their success probability δ is exponentially close to $1/L = 1/3$ (Theorem 3b). Thus the overall probability is δ^k, which is exponentially close to $1/3^k$ and therefore exponentially small.

b) Follows immediately from Theorem 3c. □

5 Further Research: Tolerating any *t* < *n*

Until now, nothing has been known about unconditional BAPs for $2t \geq n$, neither a protocol nor an impossibility proof. We hope that we can provide an unconditional BAP for any $t < n$ soon. This optimism stems from the work of DAVID CHAUM and SANDRA ROIJAKKERS [ChRo_90], who proposed a scheme called "unconditionally secure signatures": After a complex precomputation phase (using reliable broadcast), a processor, the signer, can "sign" a bit so that the first recipient of the signature can, almost certainly, convince all the other participants of the validity of this signature.

However, in contrast to "true" digital signatures, the test whether a signature is valid depends both on the recipient and on via how many other recipients the signature has been passed to him. E.g., in the basic version, already the second recipient cannot be sure that a third good recipient will accept the signature. Thus these signatures cannot just be used in any protocol using "true" signatures. In particular, it is not yet known how large these signatures must originally be to be passed along *n* recipients, and this size might be exponential in *n*. (The tricks from [ChRo_90, §5] cannot be used in BA, since they assume that the signer is good.)

We conjecture that an improved version of CHAUM's and ROIJAKKERS's scheme can be used to convert the authenticated BAP of DOLEV and STRONG [DoSt_83] into an unconditional BAP which tolerates any $t < n$ and causes only an exponentially small error. By some regeneration tricks, it should then be possible to perform a polynomial number of BAs with a single precomputation phase of fixed length (similar to authenticated BAPs with "true" digital signatures).

Acknowledgements

We are pleased to thank *Manfred Böttger* and *Kristin Lucie Munthe* for checking the correctness of our modifications to RABIN's VSS; *David Chaum*, *Maarten van der Ham*, and *Sandra Roijakkers* for explaining us the "Unconditionally Secure Digital Signatures" [ChRo_90]; *Klaus Echtle*, *Winfried Görke*, *Andreas Pfitzmann*, and *Rüdiger Reischuk* for valuable comments; and the *German Science Foundation* (DFG) for financial support.

References

BeGW_88 Michael Ben-Or, Shafi Goldwasser, Avi Wigderson: Completeness theorems for non-cryptographic fault-tolerant distributed computation; 20th STOC, ACM, New York 1988, 1-10.

Blak_79 G. R. Blakley: Safeguarding cryptographic keys; AFIPS Conference Proceedings Vol. 48, National Computer Conference (NCC) 1979, 313-317.

BrDo_84 Andrei Z. Broder, Danny Dolev: Flipping coins in many pockets (Byzantine agreement on uniformly random values); 25th FOCS, IEEE Computer Society, 1984, 157-170.

CGMA_85 Benny Chor, Shafi Goldwasser, Silvio Micali, Baruch Awerbuch: Verifiable secret sharing and achieving simultaneity in the presence of faults; 26th FOCS, IEEE Computer Society, 1985, 383-395.

ChCD1_88 David Chaum, Claude Crépeau, Ivan Damgård: Multiparty unconditional secure protocols; 20th STOC, ACM, New York 1988, 11-19.

ChRo_90 David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; Crypto '90 - Abstracts, Santa Barbara, August 1990, 209-217.

DFFL_82 Danny Dolev, Michael. J. Fischer, Rob Fowler, Nancy A. Lynch, H. Raymond Strong: An Efficient Algorithm for Byzantine Agreement without Authentication; Information and Control 52 (1982) 257-274.

DoSt_83 Danny Dolev, H. Raymond Strong: Authenticated Algorithms for Byzantine Agreement; SIAM J. Comput. 12/4 (1983) 656-666.

FeMi_88 Paul Feldman, Silvio Micali: Optimal algorithms for byzantine agreement; 20th STOC, ACM, New York 1988, 148-161.

GiMS_74 E. N. Gilbert, F. J. Mac Williams, N. J. A. Sloane: Codes which detect deception; The Bell System Technical Journal BSTJ 53/3 (1974) 405-424.

GoPe_90 Oded Goldreich, Erez Petrank: The Best of Both Worlds: Guaranteeing Termination in Fast Randomized Byzantine Agreement Protocols; Information Processing Letters 36 (1990) 45-49.

PeSL_80 Marshall Pease, Robert Shostak, Leslie Lamport: Reaching Agreement in the Presence of Faults; Journal of the ACM 27/2 (1980) 228-234.

RaBe_89 Tal Rabin, Michael Ben–Or: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority; 21st STOC, ACM, New York 1989, 73-85.

Rabi_78 Michael O. Rabin: Digitalized Signatures; Foundations of Secure Computation, ed. by R.A. DeMillo, D.P. Dobkin, A.K. Jones, R.J. Lipton; Academic Press, N.Y. 1978, 155-166.

Rabi_88 Tal Rabin: Robust Sharing of Secrets when the Dealer is Honest or Cheating; Technical Report 1 1988, Computer Science Department, Hebrew University, Jerusalem, Israel.

Sham_79 Adi Shamir: How to Share a Secret; Communications of the ACM 22/11 (1979) 612-613.

WaPf_89 Michael Waidner, Birgit Pfitzmann: Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks; Fakultät für Informatik, Universität Karlsruhe, Interner Bericht 5/89, März 1989.

WeCa_81 M. N. Wegman, J. L. Carter: New Hash Functions and Their Use in Authentication and Set Equality; Journal of Computer and System Sciences 22 (1981) 265-279.