HILDESHEIMER
INFORMATIK-
BERICHTE
ISSN 0941-3014

Birgit Pfitzmann
Michael Waidner

Unconditionally Untraceable and Fault-tolerant Broadcast and Secret Ballot Election

3/92 (April 1992)

UNIVERSITÄT
HILDESHEIM

Dieser Bericht ist herausgegeben vom

Institut für Informatik

Postfach 10 13 63
31113 Hildesheim

Unconditionally Untraceable and Fault-tolerant Broadcast and Secret Ballot Election

— Extended Abstract, April 1992 —

Birgit Pfitzmann
Institut für Informatik
Universität Hildesheim
Samelsonplatz 1
W 3200 Hildesheim, Germany

Michael Waidner
Institut für Rechnerentwurf
und Fehlertoleranz
Universität Karlsruhe
Postfach 6980
W 7500 Karlsruhe, Germany

Abstract. One of the main goals of cryptographic protocols is to combine fault tolerance and privacy, under as few assumptions about possible opponents as possible. Recently, there have been many interesting results about multi-party computation protocols achieving these goals for general problem classes. However, in this generality, one must necessarily make assumptions about the number of attacking participants, and sometimes, there are computational assumptions, such as "one-way functions exist", too.

It is therefore useful to consider what interesting special cases can be realized under weaker assumptions. Two such cases, which both have been considered before, are untraceable broadcast and secret ballot election. Privacy, in these cases, means that it remains secret who sends which message, or who casts which vote; fault tolerance means that honest participants have a chance to send messages, and that honest voters' votes are counted and nobody has more than one vote.

For both problems, we present the first protocols which tolerate any number of attacking participants and are secure in an information-theoretic sense, i.e, they do not rely on a computational assumption.

Key words. Cryptographic protocols, communication, voting, privacy, untraceability, fault tolerance, unconditional security.

1 Introduction and Overview

1.1 Motivation

One of the main goals of cryptogaphic protocols is to combine privacy, i. e., secrecy of the inputs of individual participants, with fault tolerance, i. e., correctness of the result. Moreover, the weakest possible assumptions about possible opponents should be made. Many interesting techniques have been invented that translate any problem from a rather general class into a multi-party computation protocol that achieves these goals. However, in this generality, one must necessarily make assumptions about the number of attacking participants. Sometimes, computational assumptions, such as "one-way functions exist", are made, too.

Therefore, it remains useful to consider special problems that may be of practical interest separately, and to try to invent protocols solving these problems under weaker assumptions. We do this for two problems, untraceable broadcast and secret ballot election. In both cases, we present the first protocols which rely neither on an assumption about the number of attackers nor on a computational assumption.

Untraceable broadcast was previously considered in [Beav1_90, BoBo_90, Chau_81, Cha8_85, Chau_88, Waid_90, WaPf1_90]. It should enable each participant to broadcast an arbitrary number of messages from an arbitrary finite message set. Privacy, in this case, means that a protocol must not reveal any information about the senders' identities (untraceability), and fault-tolerance means that honest participants must not be prevented permanently from broadcasting.

Practical uses may be in transactions via public networks, where sometimes, one does not only want to keep the message contents secret, but also traffic data such as who communicates with whom. For discussions about potential uses, see, e. g., [Chau_81, Cha8_85, PfWa_86]; as two small examples, consider psychological advice via telephone, or a network where one wants to build an untraceable payment system on [Cha8_85, Chau_89, Damg1_90, OkOh_92, PfWa2_92].

Also see [Chau_81, Cha8_85, PfWa_86] for more discussions about practical issues like addressing on an untraceable broadcast channel, the combination with encryption of message contents, and authentication where that is desired.

Secret ballot election was previously considered in [Chau_81, Cha6_88, CoFi_85, DeLM_82, HuTe1_90, Iver_92, Yao_82]. Here, each voter selects as input a vote from an arbitrary finite message set. The public output of the protocol is a set of votes which should contain the vote of each honest voter, and altogether not more votes than there are voters. Again, privacy means untraceability of the published votes: The protocol must not reveal any information about which vote was cast by which honest voter.

Both our protocols are polynomial in the number of participants, n, and a security parameter, γ, and have an error probability of at most $2^{-\gamma}$. However, their complexity is quite high; thus we consider them as constructive proofs of existence rather than practically applicable protocols.

Remark: Our election protocol does not guarantee simultaneity, i. e., dishonest voters can choose their votes dependent on the honest voters' votes. This is unavoidable if any number of dishonest voters must be tolerated: A simultaneous election protocol can be used to realize a global random coin with exponentially small bias, and coin tossing is possible only for an honest majority of participants [Clev_86].

1.2 Assumptions and Relation to Other Work.

Some assumptions are necessary for all known protocols, including ours: We assume a synchronous network which enables each pair of honest participants to communicate securely. Obviously, secure channels are necessary to achieve fault tolerance. Additionally, we assume private channels between each pair of honest participants. Such channels are necessary for noncryptographically untraceable broadcast and secret ballot election. This can be derived from [AlSc_83] (which shows the same implication for an untraceable communication primitive slightly stronger than those considered here).

Some additional assumptions are often made:

- **Limited number of dishonest participants:** Let t be the maximum number of tolerated dishonest participants. Some protocols assume $t < n/2$, $t < n/3$, or that a set of $m \le n$ specific participants contains at least k honest participants. The last assumption is denoted by $k{:}m$.
- **Cryptographic assumptions:** Some protocols assume that dishonest participants are restricted to polynomial time computations, and that a cryptographic assumption is true (e. g., factoring

integers or computing discrete logarithms is infeasible in polynomial time). So far, no such cryptographic assumptions has been proved.

A protocol that tolerates a computationally unlimited attacker (and does therefore *not* depend on cryptographic assumptions) is called **noncryptographically secure**. Additionally, if it tolerates any $t < n$, it is called **unconditionally secure**. In this sense, the new protocols are unconditionally untraceable and fault-tolerant.

- **Reliable broadcast assumption:** This means that if a sender broadcasts a message, then all honest participants agree on a message v, and if the sender is honest, v is the message the sender meant to send [PeSL_80]. We do not care about this assumption much, since unconditionally reliable broadcast can be implemented using the Byzantine agreement protocol (BAP) from [PfWa_92, Waid_91] (with an exponentially small error probability). Further it assumes an **initialization phase** where physically reliable broadcast is available. This is substantially weaker than to assume reliable broadcast permanently: The initialization phase is logarithmic in the number of agreements and can take place an arbitrary time before the BAP is performed for the first time.

The following tables summarize the state of the art for broadcast and election protocols with respect to their assumptions on the number and computational capabilities of dishonest participants. Table 1 summarizes general fault-tolerant multi-party protocols because they can be used to implement untraceable broadcast and secret ballot election. Table 2 summarizes untraceable broadcast protocols, Table 3 secret ballot election protocols. The assumptions for untraceability and fault tolerance are independent. **CRYP** denotes a cryptographic assumption.

Of course, the tables are not complete, but they contain all protocols which describe substantial security improvements. They show that the new protocols are the first to guarantee untraceability *and* fault tolerance unconditionally.

Protocol	Assumptions for untraceability	Assumptions for fault tolerance	Comments
[GMW_87]	CRYP $\wedge$ $t < n/2$	CRYP $\wedge$ $t < n/2$	
[ChDG_88]	CRYP	$t = 0$	
[BeGW_88]	$t < n/3$	$t < n/3$	error-free
[ChCD1_88]	$t < n/3$	$t < n/3$	exp. small error
[RaBe_89, Beav5_91]	$t < n/2$	$t < n/2$	exp. small error

Table 1 General multi-party protocols

Protocol	Assumptions for untraceability	Assumptions for fault tolerance	Comments
[Chau_81]	CRYP $\wedge$ $1{:}m$	$m{:}m$	
[Chau_88]	unconditionally	$t = 0$	error-free (cf. Section 2.1)
[Chau_88]	unconditionally	CRYP	exponentially small error for untraceability
[Waid_90]	unconditionally	$t = 0$	error-free, no reliable broadcast needed
[WaPf1_90]	unconditionally	CRYP	exp. small error f. untraceability, no rel. broadc. needed
the new protocol	**unconditionally**	**unconditionally**	**exponentially small error**

Table 2 Untraceable broadcast

Protocol	Assumptions for untraceability	Assumptions for fault tolerance
[Chau_81]	CRYP ∧ 1:m	CRYP ∧ m:m
[DeLM_82]	CRYP	$t = 0$
[Yao_82]	unconditionally	$t = 0$
[CoFi_85]	CRYP ∧ 1:1	CRYP
[Cha6_88]	unconditionally	CRYP ∧ 1:m
[HuTe1_90]	CRYP	unconditionally
[HuTe1_90]	unconditionally	CRYP
the new protocol	**unconditionally**	**unconditionally**

Table 3 Secret ballot election

1.3 Structure of the paper

The new protocols are described in Section 3 (broadcast) and Section 4 (secret ballot election). They are based on some basic protocols, which are therefore sketched in Section 2.

1.4 Notation

$\{P_1, \ldots, P_n\}$ denotes the set of all participants. We say that a statement is true **almost certainly** if it is true with a probability at least $1–2^{-\gamma}$ for a parameter γ and all computations are polynomial in γ. If a participant **publishes** a message M this means that she *reliably* broadcasts M (either via a physically reliable broadcast network, or via a BAP).

2 Basic Protocols

This section contains three protocols, which are used in the sequel: the DC-protocol (Section 2.1), pseudosignatures (Section 2.2), and Byzantine agreement (Section 2.3). We describe informally what these protocols achieve, and we sketch those details which are necessary to understand the following sections, especially those details where we had to make modifications.

2.1 DC-Protocol with collision resolution

The basis of both new protocols is the DC-protocol [Cha8_85, Chau_88] with collision resolution [BoBo_90], which realizes an unconditionally untraceable, but easily disruptable, broadcast channel. It needs reliable broadcast [Waid_90], and private channels for secret key exchange.

Let G be an arbitrary finite Abelian group. For each execution of the DC-protocol, called a **DC-round**, each pair of participants $\{P_i, P_j\}$ needs a common secret key, $K_{ij} \in G$, and each participant P_i has a message $M_i \in G$. Each participant P_i computes and publishes her *local sum*

$$O_i := M_i + (K_{1,i} + \ldots + K_{i-1,i}) - (K_{i+1,i} + \ldots + K_{n,i}),$$

and computes the *global sum* $S := O_1 + \ldots + O_n$ from all published local sums. Obviously, $S = M_1 + \ldots + M_n$. If just one participant had chosen $M_i \neq 0$, this message has been sent successfully. Otherwise, a *collision* has occurred. These collisions are *resolved* by an untraceability preserving *multi-access protocol.* We will use the protocol from [BoBo_90], which is described in the last subsection.

The DC-protocol offers unconditional untraceability: The ensemble of the local sums contains no (Shannon) information about the M_i's except the global sum [Chau_88, Waid_90, Pfit_90]; thus sending is untraceable. Since all local sums are *published*, each participant receives the same messages; thus receiving is untraceable, too.

Unfortunately, the DC-protocol offers *no* fault tolerance: Each attacker can easily and untraceably disturb the whole DC-protocol by publishing wrong local sums.

One could punish attackers afterwards, if all participants had to publish the keys they used and if there are unambiguous rules which participants were allowed to choose which M_i's. This **investigation of DC-rounds** is described in [Chau_88]. After at most $n^2/2$ disturbed and publicly investigated rounds, all dishonest participants are **disqualified**. The untraceability of uninvestigated DC-rounds is not decreased.

The problem is that during a public investigation, the senders of sensitive messages would be traced, too. Thus [Chau_88] proposes to set **traps** for attackers, i. e., to send nonsensitive messages whose only purpose is to be disturbed by an attacker, so that an investigation can be carried out without loss in untraceability. The trap protocol in [Chau_88] should guarantee cryptographic fault tolerance, but needs some refinements and modifications before being secure [WaPf1_90]. Our broadcast protocol, too, is based on this trap idea.

To resolve collisions, we use a **multi-access protocol** described in [BoBo_90, Bos_92]: Let G be a finite field with char(G) > n. Each participant P_i chooses a message $R_i \in G$. If no participant disturbs the protocol, then in n DC-rounds the *multiset* R of all R_i's is computed. It is possible to execute all n DC-rounds simultaneously.

This protocol is used as a reservation technique: R_i is randomly chosen from G and called a **reservation message**. G must be sufficiently large so that, almost certainly, all honest participants choose different reservation messages. Let c_i be the number of occurrences of R_i in R, let *NotColl* the set $\{R_i \in R \mid c_i = 1\}$. Then the arithmetical order of all $R_i \in NotColl$ defines an order of all participants P_i with $c_i = 1$. For different ways of handling collisions and reservations, see Section 3, Phase 1, and Section 4, Phase 1.

2.2 Pseudosignatures

Pseudosignatures are an authentication technique which is unconditionally correct and unforgeable, almost certainly [ChRo_91, PfWa_92, Waid_91]. In addition to these properties, which are also satisfied by authentication codes [GiMS_74, Sim3_88], pseudosignatures are **λ-transferable** for a fixed parameter $\lambda \in \mathbb{N}$: A message pseudosigned by participant S can be transferred λ times, e. g., via $S = P_{i_0} \to P_{i_1} \to \ldots \to P_{i_\lambda}$, so that for each $j < \lambda$, if P_{i_j} accepts the message, then it knows that $P_{i_{j+1}}$ will accept the message, too, almost certainly. If P_{i_λ} forwards the message, it is not guaranteed that the (λ+1)-st recipient accepts the message, too. Thus λ must be chosen carefully, depending on the specific protocol.

For comparison, ordinary digital signatures (e. g., [RSA_78, GoMR_88]) should satisfy these requirements, too, and should even be transferable an arbitrary number of times. However, they are only cryptographically unforgeable [DiHe_76]. Pseudosignatures were introduced in [ChRo_91] for $\lambda = 2$, and generalized to arbitrary λ in [PfWa_92, Waid_91].

Each pseudosignature needs a sophisticated **initialization phase** (corresponding to the key distribution of ordinary digital signatures), where private channels and reliable broadcast are available: For a parameter $\Delta \in \mathbb{N}$ (which determines the error probability), let $m := (\lambda-1)\Delta+1$. There are m subphases where each participant sends a randomly chosen key of an authentication code untraceably, using the DC-protocol and a variant of the multi-access protocol sketched in Section 2.1 (instead of reservation messages, keys are sent). Thus the signer receives all the keys, but does not know which keys come from which participants. Nobody except the signer should receive these keys; therefore, while the keys are sent, the signer publishes randomly chosen values, instead of her correct local sums. Note that there is no vicious circle although fault tolerance is needed for the DC-protocol during this initialization: If the signer detects a disturbance, she can always cause an investigation and repeat the whole procedure until no disturbance has occurred, since the keys are not sensitive in themselves. A mechanism which enables the signer to detect disturbances is described in [PfWa_92, Waid_91].

To **pseudosign a message**, v, the signer uses all received keys to authenticate v. The set of all these authentications forms the pseudosignature ψ. If an honest participant P_i receives a pseudosignature ψ on a message v, she determines the number, a_i, of her authentications contained in ψ. According to a_i she assigns a **level of acceptance** to it: If $a_i \geq m-(h-1)\Delta$, then P_i ***h*-accepts** (ψ, v), $h = 1, \ldots, \lambda$. P_i **rejects** ψ if $a_i = 0$. If Δ is appropriately chosen, it is guaranteed that if an honest participant k-accepts ψ for any $k > 1$, then each other honest participant will $(k-1)$-accept ψ, almost certainly [PfWa_92, Waid_91].

Given a specific protocol and a security parameter γ, the main problem is to determine Δ so that this protocol has an error probability of at most $2^{-\gamma}$, in spite of the fact that the protocol might enable active attacks on the transferability. Surprisingly, this can be done for a large class of polynomial protocols in general, so that Δ is polynomial in γ, i. e., the error probability is exponentially small [PfWa_92, Waid_91].

We need pseudosignatures as a subprotocol of an unconditional BAP (Section 2.3), and as a subprotocol in Section 3 and Section 4.

2.3 Unconditional Byzantine Agreement

Based on pseudosignatures with $\lambda = n$, [PfWa_92, Waid_91] provide an unconditional Byzantine agreement protocol. It needs an initialization phase during which a certain number of pseudo-signatures are initialized. Therefore, our protocols need an initialization phase with a reliable broadcast network, too. The BAP has, necessarily, an exponentially small error probability [PeSL_80].

We need this BAP for two applications:

The first and obvious one is implementing reliable broadcast. This is not further mentioned, but implicitly assumed whenever a message is *published.*

Even if there is a physically reliable broadcast network, we need the BAP for the second application, finding agreement on the acceptability of a pseudosignature. For this, it is sufficient to know that, if a participant X sends a pseudosignature ψ to each other participant (it need not be published), the BAP can be used to guarantee that

- if X is honest, all honest participants will accept ψ, and
- even if X is dishonest, all honest participants will agree on whether to accept ψ or not.

The unconditional unforgeability of pseudosignatures guarantees that a *forged* ψ is never accepted by honest participants. (To be able to choose appropriate security parameters for ψ, the maximum number of trials to find agreement on the acceptability of ψ must be known and polynomial.)

3 Untraceable Broadcast

The following broadcast protocol is based on the DC-protocol and the trap idea from [Chau_88] (Section 2.1). The main problem with traps is that

- *before* a DC-round, it must be *unpredictable* for other participants (in particular, attackers) whether the round will be a trap, whereas
- *after* a DC-round it must be *unambiguously decidable* whether this round was a trap, so that traps, and only traps, can be investigated.

The first property is necessary for fault tolerance, the second for untraceability; thus our protocol must satisfy them both unconditionally.

The structure of our protocol is similar to that of the protocols described in [WaPf1_90] (Figure 1). We will mention the main differences between the new protocol and those of [WaPf1_90] at the end of this section.

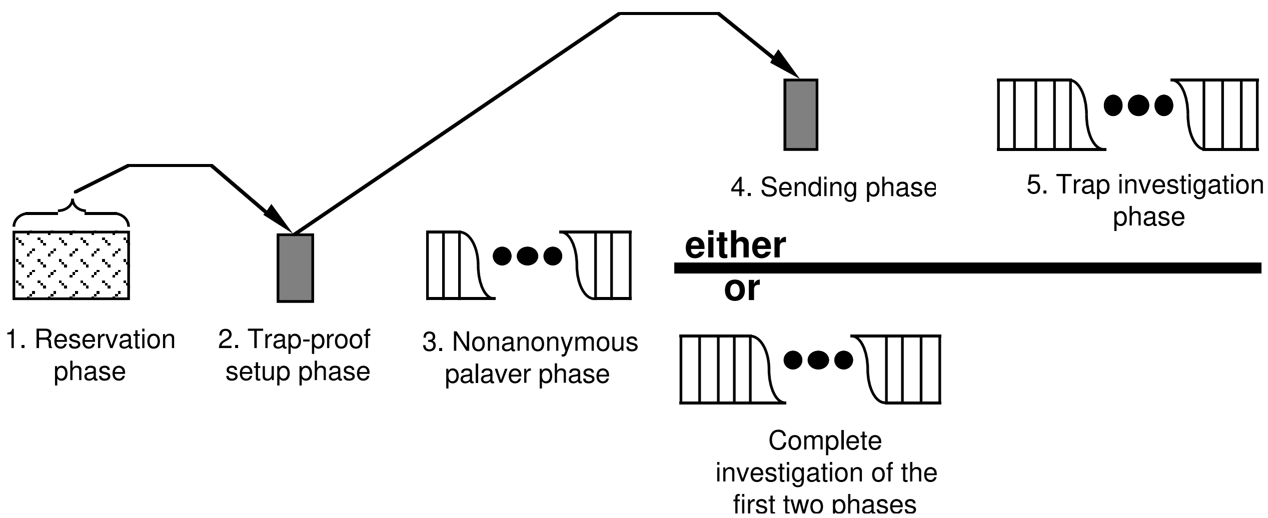

Figure 1 Broadcast protocol

During **Phase 1**, *each* participant tries to send a reservation message using the reservation technique sketched in Section 2.1. From *NotColl*, one reservation message, i.e., one participant, X, is randomly chosen by one participant, called the **current manager**. If $NotColl = \emptyset$, Phase 1 must be repeated (this will happen at most n times). The manager's role is discussed later in this section.

In **Phase 2**, a pseudosignature with X as the signer is initialized. This is possible because the undisturbed initialization protocol as sketched in Section 2.2 hides the signer's identity unconditionally: X publishes randomly chosen local sums, which are indistinguishable from the correct local sums of all other participants. If an initialization protocol is disturbed, an investigation is needed and Phase 1 and Phase 2 must be repeated. This can happen at most $n^2/2$-times (then all attackers are disqualified).

In **Phase 3**, each participant who detected a disturbance in Phase 1 or Phase 2 can nonanonymously initiate the investigation of all previous DC-rounds by publishing the message "investigation", and the attacker will be punished. If no disturbance is detected, the complainer is disqualified. This prevents attackers from always declaring the first two phases as disturbed. As said, after an investigation, the protocol restarts with Phase 1.

If no participant complains, **Phase 4** starts, and X sends either a real message or a trap. A trap must be the message 0. The probability μ of sending a trap is fixed for all participants.

If a trap is disturbed, X recognizes this, and, in **Phase 5**, she nonanonymously sends her pseudosignature ψ on the message "disturbance" to all other participants, who agree on whether to accept ψ using the unconditional BAP from Section 2.3. If ψ is accepted, the trap is investigated. Each participant who has sent anything $\neq 0$ is disqualified, and if no disturbance is found, the complaining X is disqualified. (The problem why we need the BAP and cannot simply publish the pseudosignature ψ in Phase 5 is that if X is an attacker and leaves some authentications out in ψ, some participants might accept ψ, and others reject it. Hence, when ψ has been published, there is not necessarily agreement whether the public investigation must be performed. This implies that, if X is honest, attackers could claim that they reject ψ, and thus prevent the public investigation, because these two cases are indistinguishable.)

Theorem 1: The protocol guarantees unconditional *untraceability* with an exponentially small error probability. ♦

Proof sketch: We cannot prove this theorem formally in this extended abstract. Mainly, we have to show that all the rules due to which an honest participant initiates an investigation ensure that the investigation will be successful, and that Phase 4 is investigated only if X decides to publish her pseudosignature. Then, the sending phase is as untraceable as the original DC-protocol. ❐

One might argue that it would be more efficient to use *all* reservations from *NotColl* in Phase 1. Unfortunately, this decreases untraceability: If *all* users send a trap or nontrap in Phase 4, it is obvious that the senders of investigated traps are not the senders of any nontrap! (In [Chau_88, WaPf1_90], this problem was overlooked.)

Output simulataneity: So far, this protocol is not very fault-tolerant (unless we would assume that in each round, all outputs would be made completely simultaneously): Consider one DC-round. If the attacker is the last participant to publish his local sum, he is in the fortunate position that he already knows what the correct global sum would be, and can manipulate it correspondingly by choosing an appropriate local sum. Thus, in Phase 1, he can cause collisions forever, and in Phase 4, he can distinguish traps and nontraps, and disturb nontraps only.

Unfortunately, this problem cannot be solved with perfect *fairness* if any $t < n$ must be tolerated noncryptographically. However, for *fault tolerance* it suffices to guarantee each honest participant a definite minimum bandwidth for sending messages: The participants take turns as **managers**. The current manager waits for all local sums before publishing her own one. (In Phase 1, this must be done for all n DC-rounds *simultaneously*). Since there are at least two honest participants (otherwise, untraceability becomes senseless), this prevents the attack for at least $2/n$ of all executions. The manager's role is transferred to the next participant

- if the current manager is disqualified, or
- if $NotColl = \emptyset$ in Phase 1, or
- when the execution is finished after Phase 4 or Phase 5.

Now we can evaluate the fault tolerance of the protocol: Phases 1-3 are iterated until no disturbance has occurred, i. e., at most $n^2/2$ times; thus this part cannot be permanently disturbed. If the current manager is dishonest, we cannot guarantee any fault tolerance in Phase 4. If the current manager is honest, then the attackers cannot distinguish traps and nontraps until Phase 5; hence, if they disturb Phase 4, they will be punished with probability μ in Phase 5. Thus, after expected $\mu\, n^2/2$ disturbed Phases 4 with an honest manager, all attackers have been disqualified. There are at least two honest participants. Therefore, if the attackers permanently disturb a fraction of more than $(n–2)/n$ of all protocol executions, they will be disqualified after expected less than $\mu\, n^3/2$ executions. Thus we obtain the following theorem:

Theorem 2: The protocol guarantees unconditionally, with an exponentially small error probability: A fraction of at least $2/n$ of the bandwidth provided by the undisturbed protocol cannot be disturbed permanently. ♦

Difference to previous protocols. The protocols described in [WaPf1_90] have a structure similar to Figure 1, but depend on completely different, only cryptographically secure techniques: Instead of pseudosignatures, an unconditionally committing commitment scheme is used [BrCC_88]. In Phase 2, X untraceably publishes her commitment to "trap" or "nontrap" (i. e., X must already decide whether she will send a trap in Phase 2). To initiate the investigation of a trap, she must open her commitment on "trap" in Phase 5. The problem solved here by the managers is solved in [WaPf1_90] by a commitment scheme, too. Since unconditionally committing *and* hiding commitment schemes are impossible [BrCC_88], this approach cannot be used for unconditional untraceability *and* fault tolerance.

4 Secret Ballot Election

The following secret ballot election protocol is based on pseudosignatures (Section 2.2) and the broadcast protocol described in Section 3 (Figure 2). (In [Cha6_88], a protocol is described which uses an untraceable broadcast protocol, too. However, this protocol depends on RSA signatures [RSA_78] and uses a different method to guarantee that each voter votes just once.)

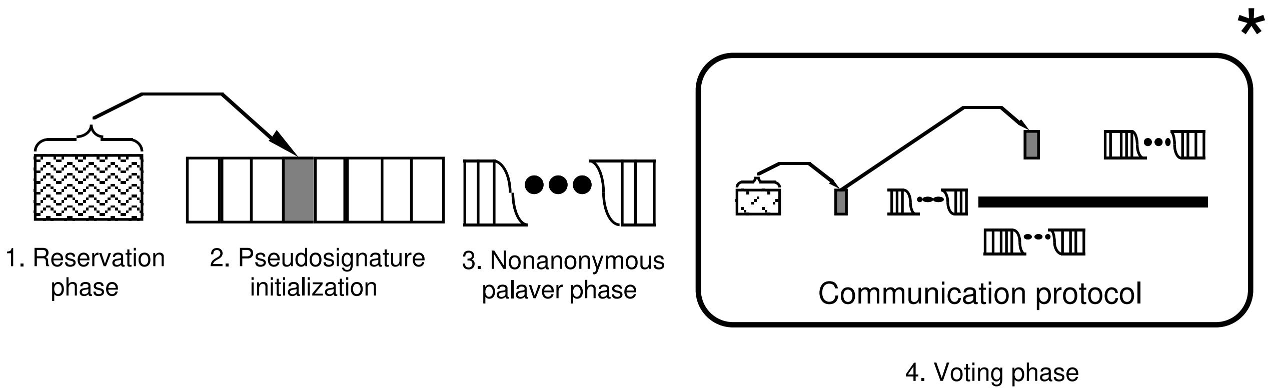

Figure 2 Secret ballot election protocol

Phases 1-3 of the protocol are very similar to the first three phases of the broadcast protocol (Section 3):

In **Phase 1**, each participant sends a reservation message. This time, almost certainly, each *honest* participant must be successful in Phase 1, even if the current manager is dishonest. Otherwise, it is not guaranteed that all honest participants can cast their votes. Thus, we cannot use the simple reservation technique sketched in Section 2.1. This is discussed further in the next subsection.

Instead of choosing one participant randomly, *all* reservations are used, and in **Phase 2**, *each* participant initializes a pseudosignature untraceably. This is done in the order defined by the reservation protocol in Phase 1. In **Phase 3**, each participant can initiate the investigation of the first two phases.

After Phase 3, each honest participant pseudosigns her vote using her pseudosignature initialized in Phase 2. In **Phase 4**, all these votes are published untraceably by the broadcast protocol described in Section 3. If a vote is published, the unconditional BAP (Section 2.3) is used to agree on whether to accept the vote or not (this is the same trick with ψ as in Section 3, Phase 5). If a vote has been accepted, the pseudosignature used to sign that vote will never be considered again. Since the broadcast protocol does not guarantee fault tolerance in a deterministic sense, we must wait until the probability that all honest participants have successfully sent their votes is sufficiently high. We can show that for $\mu = 1/2$, after $6n^4\gamma$ executions this probability is at least $1-n\,2^{-\gamma}$.

It remains to describe **Phase 1**:

Modified reservation protocol: The following reservation protocol uses the simple reservation protocol sketched in Section 2.1 iteratively. For simplicity, we call that protocol ***BoBo***. Let $Coll := R \setminus NotColl$ the multiset of all collided reservation messages. (For notation, see Section 2.1.)

From the first iteration of *BoBo*, all participants receive the multiset R and the set $NotColl$. If an honest participant does not find her reservation message in R, she complains in Phase 3.

If there are any collisions, we pass the role of manager to the next participant and *BoBo* is repeated, but only those participants P_j with $R_j \in Coll$ choose new reservation messages R_j randomly from $G \setminus NotColl$. All other participants choose the message 0. All participants receive the multiset R' of the new R_j's (not counting the zeros). Let $NotColl' := \{R_j \in R' \mid c_j' = 1\}$ and $Coll' := R' \setminus NotColl'$. If R' does not contain as many messages as $Coll$, or messages from $NotColl$, all honest participants stop this phase and complain in Phase 3. Furthermore, any honest participant P_i who has taken part in this iteration (i.e., chosen a new reservation message), but does not find her message in R', will complain in Phase 3. Otherwise, we set $NotColl := NotColl \cup NotColl'$. We repeat *BoBo* this way until $Coll' = \emptyset$, but at most n times.

If $Coll' \neq \emptyset$ after n repetitions, all honest participants will request an investigation in Phase 3, all participants who were involved in a collision in the n-th iteration are disqualified, and Phase 1 is repeated completely. (These disqualifications are correct: After n iterations, there must have been at least one iteration with an honest manager. After this iteration, all honest participants have successfully sent a reservation message. Hence, after the n-th iteration, all messages in $Coll'$ were sent by dishonest participants.)

After at most n iterations, Phase 1 terminates successfully with $Coll' = \emptyset$.

For this protocol, we can show the following Theorem 3:

Theorem 3. The protocol described above is an unconditionally untraceable and fault-tolerant secret ballot election protocol with an exponentially small probability of error. ♦

Proof sketch: The proof is in two parts:

1. Honest voters are untraceable. Mainly, this follows from Theorem 1 and from the fact that if an honest participant complains in Phase 3, the initiated investigation will be successful.
2. The set of all votes which are published and accepted in Phase 4 contains at most n votes, and it contains the votes of all honest participants. Mainly, this follows from Theorem 2 and the properties of pseudosignatures. ❐

5 Conclusion

We have presented the first untraceable and fault-tolerant protocols for broadcast and secret ballot election which tolerate any number of attacking participants without relying on any computational assumptions.

References

AlSc_83 Bowen Alpern, Fred B. Schneider: Key Exchange Using 'Keyless Cryptography'; Information Processing Letters 16 (1983) 79-81.

Beav1_90 Donald Beaver: Security, Fault Tolerance, and Communication Complexity in Distributed Systems; Ph. D. Thesis, Division of Applied Sciences, Harvard University, Cambridge, Massachusetts, May 1990.

Beav5_91 Donald Beaver: Secure Multiparty Protocols and Zero Knowledge Proof Systems Tolerating a Faulty Minority; Journal of Cryptology 4/2 (1991) 75-122.

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

BoBo_90 Jurjen Bos, Bert den Boer: Detection of disrupters in the DC protocol; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 320-327.

Bos_92 Jurjen Bos: Practical Privacy; Proefschrift, ISBN 90 6196 405 9, Technische Universiteit Eindhoven 1992.

BrCC_88 Gilles Brassard, David Chaum, Claude Crépeau: Minimum Disclosure Proofs of Knowledge; Journal of Computer and System Sciences 37 (1988) 156-189.

Chau_81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.

Cha8_85 David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044.

Chau_88 David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability; Journal of Cryptology 1/1 (1988) 65-75.

Cha6_88 David Chaum: Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA; Eurocrypt '88, LNCS 330, Springer-Verlag, Berlin 1988, 177-182.

Chau_89 David Chaum: Privacy Protected Payments – Unconditional Payer and/or Payee Untraceability; SMART CARD 2000, North-Holland, Amsterdam 1989, 69-93.

ChCD1_88 David Chaum, Claude Crépeau, Ivan Damgård: Multiparty unconditional secure protocols; 20th Symposium on Theory of Computing (STOC) 1988, ACM, New York 1988, 11-19.

ChDG_88 David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

ChRo_91 David Chaum, Sandra Roijakkers: Unconditionally Secure Digital Signatures; Crypto '90, LNCS 537, Springer-Verlag, Berlin 1991, 206-214.

Clev_86 Richard Cleve: Limits on the Security of Coin Flips When Half the Processors are Faulty; 18th Symposium on Theory of Computing (STOC) 1986, ACM, New York 1986, 364-369.

CoFi_85 Josh D. Cohen, Michael J. Fischer: A robust and verifiable cryptographically secure election scheme; 26th Symposium on Foundations of Computer Science (FOCS) 1985, IEEE Computer Society, 1985, 372-382.

Damg1_90 Ivan Bjerre Damgård: Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 328-335.

DeLM_82 Richard A. DeMillo, Nancy A. Lynch, Michael J. Merritt: Cryptographic Protocols; 14th Symposium on Theory of Computing (STOC) 1982, ACM, New York 1982, 383-400.

DiHe_76 Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654.

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

GMW_87 Oded Goldreich, Silvio Micali, Avi Wigderson: How to play any mental game - or - a completeness theorem for protocols with honest majority; 19th Symposium on Theory of Computing (STOC) 1987, ACM, New York 1987, 218-229.

GoMR_88 Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

HuTe1_90 Ming-Deh A. Huang, Shang-Hua Teng: Security, Verifiability, and Universality in Distributed Computing; Journal of Algorithms 11 (1990) 492-521.

Iver_92 Kenneth R. Iversen: A Cryptographic Scheme for Computerized General Elections; Crypto '91, LNCS 576, Springer Verlag, Berlin 1992, 405-419.

OkOh_92 Tatsuaki Okamoto, Kazuo Ohta: Universal Electronic Cash; Crypto '91, LNCS 576, Springer Verlag, Berlin 1992, 324-337.

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

Pfit_90 Andreas Pfitzmann: Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz; IFB 234, Springer-Verlag, Heidelberg 1990.

PfWa_86 Andreas Pfitzmann, Michael Waidner: Networks without user observability -- design options; Eurocrypt '85, LNCS 219, Springer-Verlag, Berlin 1986, 245-253; Computers & Security 6/2 (1987) 158-166.

PfWa_92 Birgit Pfitzmann, Michael Waidner: Unconditional Byzantine Agreement for any Number of Faulty Processors; STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, LNCS 577, Springer-Verlag, Heidelberg 1992, 339-350.

PfWa2_92 Birgit Pfitzmann, Michael Waidner: How to Break and Repair a "Provably Secure" Untraceable Payment System; Crypto '91, LNCS 576, Springer Verlag, Berlin 1992, 338-350.

RaBe_89 Tal Rabin, Michael Ben-Or: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority; 21st Symposium on Theory of Computing (STOC) 1989, ACM, New York 1989, 73-85.

RSA_78 R. L. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.

Sim3_88 Gustavus J. Simmons: A Survey of Information Authentication; Proceedings of the IEEE 76/5 (1988) 603-620.

Waid_90 Michael Waidner: Unconditional Sender and Recipient Untraceability in spite of Active Attacks; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 302-319.

Waid_91 Michael Waidner: Byzantinische Verteilung ohne kryptographische Annahmen trotz beliebig vieler Fehler; Universität Karlsruhe, Fakultät für Informatik, Dissertation, Karlsruhe 1991.

WaPf1_90 Michael Waidner, Birgit Pfitzmann: The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 690.

Yao_82 Andrew C. Yao: Protocols for Secure Computations; 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 160-164.