Verifiable Agreement: Limits of Non-Repudiation in Mobile Peer-to-Peer Ad Hoc Networks

Zinaida Benenson[1], Felix C. Freiling[2], Birgit Pfitzmann[3], Christian Rohner[1], and Michael Waidner[3]

[1] Uppsala University, Department of Information Technology
{zina,chrohner}@it.uu.se
[2] University of Mannheim, Computer Science Department
freiling@informatik.uni-mannheim.de
[3] IBM Research, Zurich Research Lab
{bpf,wmi}@zurich.ibm.com

Abstract. We introduce verifiable agreement as a fundamental service for securing mobile peer-to-peer ad hoc networks, and investigate its solvability. Verifiability of a protocol result means that the participants can prove that the protocol reached a particular result to any third party (the verifier) which was not present in the network at the time of the protocol execution.

1 Introduction

1.1 Motivation

The envisioned applications of ad hoc networks often follow the scenario where a group of nodes meets for a short time, conducts some transaction, such as a collaborative document editing session, a decision to take some coordinated action, or dissemination of information to all group members, and then breaks apart, perhaps forever. We call this type of the network *mobile peer-to-peer ad hoc network*.

In this scenario, there is no centralized logging of the transaction, no transaction witnesses, apart from the participants themselves. Thus, to make the result of the transaction binding, it should be made *verifiable*. That is, after the transaction is finished, each participant should be able to prove to some third party which was not present in the network at the time of the transaction that this particular transaction (1) happened, (2) was conducted by the certain group of participants, and (3) reached a particular outcome. We call this problem *Verifiable Agreement* on the transaction result. Requiring that each participant be able to carry out the proof without the help of any other participant seems to be the most safe decision, as there is no guarantee that any other participant would be reachable at the time when the proof is conducted.

Verifiable Agreement is a crucial problem for securing mobile peer-to-peer ad hoc networks. Indeed, especially if such networks are set up in emergency situations, with participants from different organizations or different countries, the participants may distrust each other. Unfortunately, as we show below, the non-repudiation of the decisions made in this situation can only be reached if the majority of participants can be trusted. This puts a strict restriction on the usage of this network type for trust-critical applications.

1.2 Agreement and Contract Signing

We denote by *Agreement* a class of problems where a set of n parties $P := \{P_1, \ldots, P_n\}$ start with initial inputs $x_1, \ldots, x_n$. Some parties might be dishonest and arbitrary deviate from their programs. All honest parties must eventually, or with some high probability, terminate and agree on a common result, y, which is "valid". *Validity* defines a particular agreement problem:

- In *Interactive Consistency* [20], the parties must agree on a vector y, where the ith element must be x_i for all honest parties P_i, otherwise it can be any value.
- In *Consensus* [9], if there is a value x such that $x_i = x$ for all honest parties P_i, then $y = x$.

Other agreement problems include Byzantine Generals Problem (also called Byzantine Agreement) [17], Weak Byzantine Agreement [16], Atomic Commitment [22], Strong Consensus [10], Validated Byzantine Agreement [15].

In contrast to Secure Multi-Party Computation [12], the inputs of the parties do not need to be secret or independent.

Contract signing [7] can be considered as an agreement problem where the parties must agree either on a contract text or on a special value failed, which means that no contract was signed. The signed contract can be an outcome of the contract signing protocol only if all honest parties want to sign the same contract text. The signed contract must be *verifiable.* Informally, verifiability can be described as follows:

- Each honest party can convince a verifier V, which knows nothing about a particular protocol run, that this protocol run yielded the result y.
- If some protocol run yielded the result y, no party can convince V that the protocol yielded some result $y' \neq y$.

The result failed is usually left non-verifiable. This reflects the real-world situation where no proof of the fact that a contract was *not* signed is required.

1.3 Related Work

Apart from contract signing, which has been an active research area for several decades, the only approach to make an agreement problem verifiable, as far as we know, is undertaken in [22]. There, a specification for verifiable atomic commitment for electronic payment protocols is presented, but no explicit definition of verifiability is given. A different notion of verifiable agreement was introduced in [15]: each honest protocol participant P_i can convince any other honest participant P_j of its result, but not necessarily any outsider. Multi-party contract signing protocols are presented, e.g., in [2, 5, 11]. For an overview of recent work, see also [23].

1.4 Outline and Contribution

After presenting the system model in Section 2, we give a unifying definition of agreement problems which facilitates rigorous proofs, and define verifiable agreement (Section 3).

We show that in case of dishonest majorities, verifiable agreement cannot be solved (Section 4). This puts a fundamental limit on non-repudiation of transactions in mobile peer-to-peer ad hoc networks. In contrast, some agreement problems, such as Interactive Consistency, can be solved for any number of dishonest parties. We present a verifiable agreement protocol for honest majorities in Section 5.

Finally, in Section 6, we discuss our system assumptions and the applications of verifiable agreement.

2 System Model and Preliminaries

2.1 System Model

Let $P = \{P_1, ..., P_n\}$ denote the set of participats of an agreement protocol, and V denote a verifier.

Let $|X| \geq 2$ and $|Y| \geq 2$ be two finite sets representing the inputs and the outputs of individual participants P_i. We assume (w.l.o.g.) that Y contains a distinguished element failed. For a subset of parties $H \subseteq P$ we denote by X^H the set of all $|H|$-dimensional vectors with elements from X.

The parties P_i can digitally sign messages, and all parties can verify their signatures. The signature on message m associated with party P_i is denoted by $\mathsf{sign}_i(m)$.

The *adversary* can a priori choose to corrupt a certain subset of parties. It has full control over the behavior and knowledge of dishonest parties (Byzantine failures).

We assume that the adversary, as well as all participating parties, are computationally bounded. In particular, the adversary cannot forge signatures.

We consider both synchronous and asynchronous networks with reliable communication channels.

In synchronous networks, communication proceeds in rounds. In each round, a party first receives inputs from the user and all messages sent to it in the previous round (if any), processes them, and may finally send some messages to other parties or give outputs to the user.

In asynchronous networks, the sent messages can be delivered in any order and there is no upper bound on the time of message delivery. However, as the communication channels are reliable, each sent message is guaranteed to arrive eventually. The control over the message delivery is given to the adversary. A party P_i may decide to stop waiting for a certain event E. That means the following: Before P_i starts waiting for E, it sends to itself a unique timeout message *timeout* and waits for this message, too. If *timeout* arrives first, the party stops waiting for E and proceeds.

We discuss the viability of these assumptions in the ad hoc networks in Section 6.

Every protocol instance has a unique identifier *tid*. We assume that all honest parties are willing to participate in a protocol run with a fresh protocol identifier.

2.2 Preliminary Definitions

Honesty structure formalizes for which sets of honest parties the problem should be solved[4].

Definition 1 (Honesty Structure). *An* honesty structure $\mathcal{H}$ *for a set of parties* P *is a set of subsets of* P *such that if* $H \in \mathcal{H}$ *and* $H \subseteq H' \subseteq P$ *then* $H' \in \mathcal{H}$.

The definition reflects the intuition that any protocol that works given a certain set H of honest parties should also work in case there are *more* honest parties.

Definition 2 (conditions Q_2 and Q_3, from [13]). *An honesty structure* $\mathcal{H}$ *satisfies condition* Q_2 *if* $H_1 \cap H_2 \neq \emptyset$ *for all* $H_1, H_2 \in \mathcal{H}$, *and it satisfies condition* Q_3 *if* $H_1 \cap H_2 \cap H_3 \neq \emptyset$ *for all* $H_1, H_2, H_3 \in \mathcal{H}$.

A *threshold honesty structure* $\mathcal{H}_t$ for a threshold $t < n$ is a set of subsets of P such that $\mathcal{H}_t = \{H \subseteq P : |H| > n - t\}$. A threshold honesty structure satisfies Q_2 or Q_3 if and only if $t \leq \frac{n}{2}$ or $t \leq \frac{n}{3}$, respectively. Thus, the condition Q_2 generalizes the notion of honest majority.

We now define validity functions which will be used in the following to describe validity conditions of agreement problems.

Definition 3 (Validity Function). *Let* $\mathcal{H}$ *be an honesty structure. A* validity function *for the sets* X, Y, *and* $\mathcal{H}$ *is a function* F *that maps pairs* $(H, x) \in \mathcal{H} \times X^H$ *to subsets of* Y, *the allowed outputs. It must satisfy the* Non-triviality *condition:*

– $\forall y \neq \mathsf{failed}\ \exists x \in X^P :\ y \notin F(P, x)$ *and*
– $\exists x \in X^P :\ F(P, x) \neq \{\mathsf{failed}\}$.

Non-triviality excludes all consensus problems which can be solved by the trivial protocol which always outputs a constant result y, or always fails. We do not exclude problems that allow the output failed for all initial inputs, because the result failed is sometimes unavoidable. However, in this case the non-triviality condition guarantees that there exists at least one protocol run which does not output failed. In the following, we give examples of validity functions for some well-known problems.

Consensus with $Y = X$ is described by:

$$F_C(H, x) := \begin{cases} \{x\} \text{ if } x_i = x\ \forall\ P_i \in H \\ X \quad \text{otherwise.} \end{cases}$$

Interactive Consistency with $Y = X^P$ is described by:

$$F_{IC}(H, x) := \{y \in X^P\ |y^i = x_i\ \forall\ P_i \in H,\ y^i \in X \text{ otherwise.}\},$$

where y^i denotes the ith element of the vector $y \in X^P$.

[4] The corresponding notion from the area of secret sharing is access structure. An adversary structure [13], which consists of all sets of dishonest parties a protocol can withstand, is the complement of it.

3 Definition of Verifiable Agreement

We first define agreement problems. Here and further in the sequel, the *timeout* messages refer only to the asynchronous model.

Definition 4. *An* agreement problem *for a validity function F (for an honesty structure $\mathcal{H}$ and input and output sets X and Y) is to devise a protocol* $\mathsf{consensus}[]$ *for parties $P_1, ..., P_n$. In order to start the protocol, a party P_i receives the input* $(\mathsf{consensus}, tid, x_i)$*. Here tid is a transaction identifier unique for all executions of* $\mathsf{consensus}[]$*, and $x_i \in X$ is P_i's local input. Upon termination, the protocol produces an output (tid, y_i) with $y_i \in Y$ for each P_i.*

The following requirements must be satisfied for all sets $H \in \mathcal{H}$ of actually honest parties and input vectors $x \in X^H$:

- Agreement: *There is a $y \in Y$ such that $y_i = y$ for all $P_i \in H$.*
- Validity: *$y_i \in F(H, x)$ for all $P_i \in H$.*
- Correct Execution: *If all parties are honest, and no party receives any timeout messages, then for all input vectors $x \in X^P$ with $F(P, x) \neq \{\mathsf{failed}\}$, the parties will never agree on $y = \mathsf{failed}$.*
- Termination of $\mathsf{consensus}[]$: *Eventually each $P_i \in H$ terminates and produces an output $y_i \in Y$.*

Correct Execution excludes protocols that always output failed.

We now formalize the *verifiability* of an agreement.

Definition 5. *A* verifiable agreement *problem for a validity function F is to devise, in addition to the protocol* $\mathsf{consensus}[]$*, the protocol* $\mathsf{verify}[]$ *which involves only one party P_i and a verifier V which does not have any knowledge about the execution of* $\mathsf{consensus}[]$ *or about possible previous runs of* $\mathsf{verify}[]$*.*

Party P_i starts $\mathsf{verify}[]$ *with input* $(\mathsf{show}, V, tid, y)$ *where tid is the transaction identifier of an execution of* $\mathsf{consensus}[]$ *and y is the result obtained from this execution. The verifier receives the input* $(\mathsf{verify}, P_i, tid)$ *and eventually obtains an output* (tid, d_V) *where* $d_V \in \{(y, \mathsf{accepted}), \mathsf{verify\_failed}\}$*. The following requirements must be satisfied in addition to those from Definition 4 for an honest verifier V and all sets $H \in \mathcal{H}$ of actually honest parties:*

- Verifiability of Correct Result: *If $P_i \in H$ obtained the output (tid, y) for $y \neq \mathsf{failed}$ from* $\mathsf{consensus}[]$ *and later receives the input* $(\mathsf{show}, V, tid, y)$*, and if V receives the input* $(\mathsf{verify}, P_i, tid)$*, then V will obtain the result* $d_V = (y, \mathsf{accepted})$*, provided no timeout messages are received during the protocol.*
- Non-verifiability of failed: *The verifier V never decides* $(\mathsf{failed}, \mathsf{accepted})$ *on any input.*
- No Surprises: *If some $P_i \in H$ obtained (tid, y) from* $\mathsf{consensus}[]$*, then V never obtains the result* $d_V = (y', \mathsf{accepted})$ *on input* $(\mathsf{verify}, P_j, tid)$ *for any party P_j and $y' \neq y$.*
- Termination of $\mathsf{verify}[]$: *Each V and each $P_i \in H$ eventually terminate.*

◊

No Surprises says, in particular, that if some honest party P_i never started or not yet finished $\mathsf{consensus}[]$ for some tid, then the verifier cannot accept any result y for tid from some party P_j, honest or dishonest, unless P_i is guaranteed to obtain y for tid.

We now show how to define the contract signing problem within our framework.

Definition 6. Contract signing *is a verifiable consensus problem described by the following validity function:*
$X := C \cup \{\mathsf{reject}\}$*, where* C *is a finite set of contract texts that can be signed,* $Y := C \cup \{\mathsf{failed}\}$*, and* $\mathcal{H}$ *is the power set of* P*. Then:*

$$F_{CS}(H, x) := \begin{cases} \{contr, \mathsf{failed}\} & \textit{if } \exists \ contr \in C \textit{ such that } x_i = contr \ \forall \ P_i \in H \\ \{\mathsf{failed}\} & \textit{otherwise.} \end{cases}$$

◊

It is possible to show that the above definition of contract signing and the "usual" definition from, e.g., [2] are equivalent. We omit this proof due to space limit.

4 Impossibility of Verifiable Agreement for Dishonest Majorities

We show that if Q_2 (which generalizes the notion of honest majority) is *not* satisfied, then the Verifiable Agreement problem cannot be solved even in synchronous networks. In contrast, some agreement problems, e.g., Interactive Consistency, can be solved deterministically in this setting for any honesty structure [20]. As the synchronous network is the most strong network model, this result implies non-solvability for all other network classes.

In section 4.1 we show that no deterministic protocol can solve verifiable agreement for dishonest majorities. In section 4.2 we show that any probabilistic verifiable agreement protocol in case Q_2 is not satisfied has the error probability at least inversely linear in the number of protocol rounds. This means that in order to make the error probability of the protocol exponentially small, an exponential number of rounds is needed, which is unacceptable due to the computationally bounded protocol participants.

4.1 No Deterministic Verifiable Agreement for Dishonest Majorities

Theorem 1. *No synchronous deterministic protocol can solve Verifiable Agreement if condition* Q_2 *is not satisfied.*

Proof. Let $\mathcal{H}$ be an honesty structure which does not satisfy Q_2, F be a validity function for $\mathcal{H}$ and input and output sets X and Y. Assume that there is some protocol π which solves the verifiable agreement problem specified by F deterministically in r rounds.

If $\mathcal{H}$ does not satisfy Q_2, then there are sets $H_1, H_2 \in \mathcal{H}$ such that $H_1 \cap H_2 = \emptyset$. We assume, w.l.o.g., that $H_1 \cup H_2 = P$, i.e., $H_1 = \bar{H}_2$.

In this case, it is possible to collapse all parties in H_1 into one (new) party $\tilde{P_1}$, and all parties in H_2 into a new party $\tilde{P_2}$. The new resulting protocol $\tilde{\pi}$ runs exactly as the protocol π, but all messages sent in π between the parties in the set H_1 (H_2, respectively) now belong to the internal state of party $\tilde{P_1}$ ($\tilde{P_2}$) in the corresponding protocol run of $\tilde{\pi}$. Therefore, it is sufficient to consider only the two-party case $P = \{P_1, P_2\}$ in our impossibility proof.

First note that the non-triviality of validity function F implies that there are at least two different results for π in the all-honest case. One of them must be verifiable (i.e. unequal to failed). Furthermore, if some result y is allowed in the all-honest case, then that y must be an allowed result in case only one of the parties is honest, as the dishonest party might *behave* like honest in a protocol run.

We now show that party P_1 cannot obtain any verifiable result without communication with party P_2. Assume that it can be done, i.e., P_1 can obtain some verifiable result y_1 from some protocol run. Then the non-triviality of F implies that there is some result $y \neq y_1$, verifiable or non-verifiable, which can be obtained in the all-honest case. Assume that P_1 is dishonest, P_2 is honest. If the parties run π for some *tid* and party P_1 behaves like honest, then they can obtain the result y. At the same time, party P_1 can execute π for the same *tid* without party P_2 and receive the verifiable result y_1, which contradicts No Surprises (Definition 5).

The remaining case is the one where P_1 and P_2 must be able to obtain some verifiable result from the protocol and need to communicate with each other in order to do so. We assume, w.l.o.g., that the parties send messages to each other in each round, as we can always force them to send dummy messages. Consider some protocol run run_1 where both parties P_1 and P_2 are honest and obtain a verifiable result y after r rounds, see Figure 1. Honest parties are drawn as circles, messages sent in each round from P_1 to P_2 and vice versa are shown as diagonal lines. The round where the party P_i gains the result y from the protocol run is indicated as a black point.

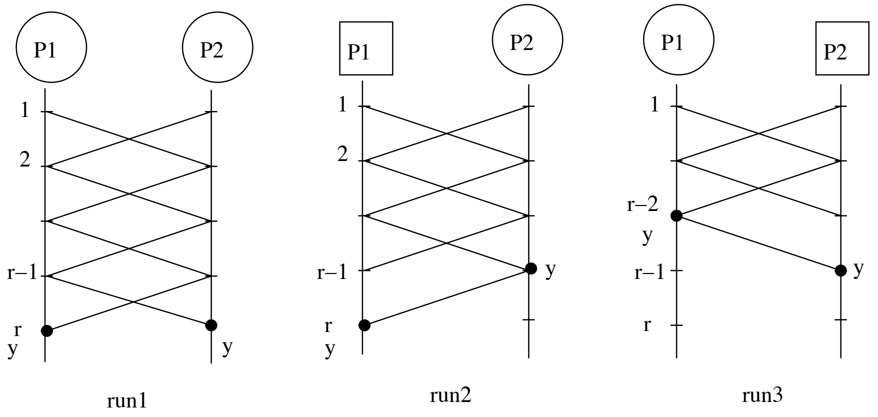

Fig. 1. No deterministic verifiable consensus for dishonest majorities. In run_1, both parties are honest. In run_2, party P_1 is dishonest. In run_3, party P_2 is dishonest.

Now we consider the protocol run run_2 where party P_1 is dishonest. It does not send any protocol messages in the last round, but all other messages are sent as in run_1. As the honest party P_2, however, send its messages in the last round, party P_1 gets all messages it needs to obtain the verifiable result y in round r. Then, as π must satisfy No Surprises (Definition 5), party P_2 must obtain the output y as well. As P_2 does not receive any messages after Round $r-1$, it obtains y in this round.

Thus, party P_2 can obtain y after it received all messages up to round $r-1$. Consider yet another protocol run, where P_2 is dishonest, but P_1 is honest. Until round $r-2$ both P_1 and P_2 send their messages exactly as in run_1. In round $r-1$, however, P_2 stops sending messages. It receives the messages from P_1 in round $r-1$ and therefore, obtains y. However, party P_1 must obtain y, too, as explained above. Thus P_1 can obtain y after round $r-2$.

We continue to construct the chain of protocol runs starting from run_1 where the parties need less and less messages to obtain y. In this manner, we will arrive at the the protocol run where parties P_1 and P_2 do not need to communicate at all to obtain y, which contradicts the initial assumption. Therefore, the protocol π does not exist.

□

4.2 Error Probability of Probabilistic Verifiable Agreement for Dishonest Majorities

Theorem 2. *The error probability of any probabilistic synchronous verifiable agreement protocol in case Q_2 is not satisfied is unacceptable large, i.e., at least inversely linear in the number of protocol rounds.*

Proof. We first note that any n-party verifiable agreement in case Q_2 is not satisfied can be transformed into a 2-party verifiable agreement problem, see Section 4.1. We then describe an adversary which makes the error probability of any 2-party verifiable agreement protocol which terminates in expected r number of rounds at least $\frac{1}{3r}$ only by causing the corrupted party to stop sending messages in some round i.

Let $P = \{P_1, P_2\}$ and $\mathcal{H} = \{\{P_1\}, \{P_2\}, \{P_1, P_2\}\}$, and let F be a validity function for $\mathcal{H}$. Let π be a probabilistic synchronous 2-party verifiable agreement protocol for F which terminates in an expected finite number of rounds.

We assume, w.l.o.g., that in a synchronous probabilistic protocol π between P_1 and P_2 the parties alternate in sending messages. Further, we assume that each party can get some result accepted without communication with the other party with probability at most $\frac{1}{3}$ and that there is a finite number of rounds r such that the result is accepted with probability at least $\frac{2}{3}$ after that round.

We complete the protocol such that it always runs at least r rounds. If it would terminate already in round $s < r$ we just let the parties wait till round r.

We consider only a very weak adversary: it can select the initial inputs for P_1 and P_2 and a number $i \in \{1, ..., r\}$ such that the protocol is interrupted in round i. The interruption is done by the party which should be sending in this round. The interrupting party accepts the message sent by the honest party in round $i-1$, but sends nothing in round i and ceases to participate in the protocol run. The corresponding honest party then gets its result in round r.

The adversary is successful if No Surprises or Verifiability of Correct Result (Definition 5) are violated, i.e. if the dishonest party is able to convince the verifier V of a certain value, but the honest party is not able to do this.

Let $F(\{P_1, P_2\}, (x_1, x_2)) \neq \{\mathsf{failed}\}$. Consider adversaries which select (x_1, x_2) as input vector. Let A_i be the adversary which interrupts the protocol run on input (x_1, x_2) in round $i \leq r$.

After the protocol is interrupted in round i, the uncorrupted party obtains some result in round r. The corrupted party obtains some result from the protocol run in round r, too, as it can pretend to be honest and to have not received any protocol messages after round i.

Let E_i denote the event that an honest verifier V accepts the output the dishonest party obtained after interrupting the protocol run in round i. The probability that it happens is denoted as $P(E_i)$. Then V accepts the output of the honest party in this protocol run with the probability $P(E_{i-1})$. E_0 denotes the probability that the output of the honest party is accepted if the interruption happens before round 1. Then $P(E_0) = P(E_1) \leq \frac{1}{3}$, $P(E_r) = \frac{2}{3}$, and the error probability of the protocol run with an adversary A_i is $P(E_i \wedge \overline{E_{i-1}})$, $i \leq r$.

Let A_{max} be the adversary A_i that maximizes the error probability. We first show the following lemma:

Lemma 1. *Let ϵ be a value such that for all rounds $i \leq r$ of a protocol $P(E_i \wedge \overline{E_{i-1}}) \leq \epsilon$. Then $\epsilon \geq (P(E_r) - P(E_0))/r$.*

Proof. (of the above lemma) For all $i \leq r$ we have $P(E_i) = P(E_i \wedge E_{i-1}) + P(E_i \wedge \overline{E_{i-1}}) \leq P(E_{i-1}) + \epsilon$, thus $P(E_i) - P(E_{i-1}) \leq \epsilon$. From that we can conclude that $P(E_r) - P(E_0) = (P(E_r) - P(E_{r-1})) + (P(E_{r-1}) - P(E_{r-2})) + ... + (P(E_2) - P(E_1)) + (P(E_1) - P(E_0)) \leq r\epsilon$ and thus, $\epsilon \geq (P(E_r) - P(E_0))/r$.

□

Proof of Theorem 2 (continued):

A_{max}'s probability of success is

$$\delta_{\mathsf{max}} := \max\{P(E_1 \wedge \overline{E_0}), ..., P(E_r \wedge \overline{E_{r-1}})\}$$

Thus we know that $P(E_i \wedge \overline{E_{i-1}}) \leq \delta_{\mathsf{max}}$. Applying Lemma 1 with $\epsilon := \delta_{\mathsf{max}}$ yields $\delta_{\mathsf{max}} \geq \frac{1}{3r}$, as $P(E_r) - P(E_0) \geq \frac{2}{3} - \frac{1}{3} = \frac{1}{3}$.

□

Remark 1.

A weaker version of Theorem 2 has been already proven in [6]: Their Theorem 1 shows that if a contract signing protocol terminates in r rounds, and if for some values δ, ϵ for all rounds i

$$P(E_i) > \delta \Rightarrow P(\overline{E_{i-1}} | E_i) < \epsilon \qquad (1)$$

then

$$r \geq \frac{\log(\delta)}{\log(1-\epsilon)} + 2$$

which is approximately equal to $\epsilon^{-1}\log(\delta^{-1})$.

Equation 1 implies $P(\overline{E_{i-1}} \wedge E_i) \leq \max\{\delta, \epsilon\}$, and applying Lemma 1 yields $\max\{\delta, \epsilon\} \geq 1/r$, which is basically Theorem 1 of [6].

The fact that Equation 1 implies our condition (but not vice versa) was already observed in [6]. Since we want to prove that there is *no* 2-party protocol which succeeds with high probability, the weakest possible definition of success is the most preferable one.

$\diamond$

5 Verifiable Agreement for Honest Majorities

We first show how to extend any agreement protocol for honesty structures satisfying the condition Q_2 to a verifiable agreement protocol (Section 5.1). We present a combined protocol for synchronous and asynchronous networks.

In Section 5.2, we present a contract signing protocol for honest majorities. Note that usually contract signing should be able to withstand any number of dishonest parties as long as at least one honest party participates in the protocol. In this case, there is no chance to sign a contract in an ad hoc peer-to-peer group. However, in case the majority of parties can be trusted, our protocol is the first one which enables contract signing in this setting.

5.1 A Verifiable Agreement Protocol for Honest Majorities

Protocol 1. Let π be an agreement protocol (Definition 4) for an honesty structure $\mathcal{H}$, input and output sets X and Y and a validity function F.

- consensus[]:
  1. The parties first run the protocol π on their inputs x_i for the identifier tid. As soon as a party P_i obtains output $y_i \neq \mathsf{failed}$, it sends $m_i := \mathsf{sign}_i(tid, y_i)$ to all participants.
  2. We call any set $M = \{\mathsf{sign}_{j_1}(tid, y), \ldots, \mathsf{sign}_{j_k}(tid, y)\}$ where $\{P_{j_1}, ..., P_{j_k}\} \in \mathcal{H}$ a *proof set* for (tid, y). P_i waits until it has received a proof set for (tid, y_i).
- verify[]: The verifier V accepts the result y for some tid if and only if it receives a proof set for $(tid,\, y)$ where $y \neq \mathsf{failed}$.

$\diamondsuit$

Theorem 3. *Protocol 1 solves Verifiable Agreement under the condition Q_2 in both synchronous and asynchronous networks.*

Proof. (sketch)

We only show the less obvious requirements.

Termination of consensus[] (Definition 4): Let $H \in \mathcal{H}$ be the actual set of honest parties. Then all honest parties $P_i \in H$ start π, terminate with the agreement on some result y and send the signed result (message m_i) to all parties (we assume that P_i sends m_i to itself as well). Thus, eventually each honest party P_i receives a proof set and terminates, as we assume reliable communication.

No Surprises (Definition 5): Let H be the actual set of honest parties, and assume that the verifier V receives a proof set for some $y \in Y$ with $H' \in \mathcal{H}$ as the set of all signatories of y. Since $H \cap H' \neq \emptyset$, there is at least one honest party $P_h \in H'$, and as P_h signed y, it must be the correct result.

□

Remark 2. If a protocol solves some agreement problem in asynchronous networks, the corresponding honesty structure must satisfy Q_3 [8]. If Q_3 is satisfied, then Q_2 is also satisfied. Therefore, in asynchronous networks, any agreement problem can be solved with verifiability, if it can be solved at all.

◇

5.2 Contract Signing for Honest Majorities

Applying the construction of Protocol 1 to the binary Consensus problem with $X = Y = \{0, 1\}$ (see Section 2.2), we construct contract signing for honest majorities.

Protocol 2. Let π be a protocol that solves binary Consensus, and let $(\mathsf{sign}, \mathit{tid}, x_i)$ be the input of the party P_i for the contract signing protocol. As previously, we present a combined protocol for synchronous and asynchronous networks.

(1) If $P_i \in H$ wants to sign the contract ($x_i = \mathit{contr}$), then it sets $c := (\mathit{tid}, \mathit{contr})$ and sends the promise to sign *contr* for *tid* to all parties: $m_{1,i} := \mathsf{sign}_i(c, \mathsf{sign})$. We call $M := \{\mathsf{sign}_1(c, \mathsf{sign}), ..., \mathsf{sign}_n(c, \mathsf{sign})\}$ a **minor proof set** for c.

P_i tries to collect such a minor proof set for c. On asynchronous networks, P_i can stop the collection process any time, on synchronous networks P_i waits until the next round. If P_i succeeded in collecting a minor proof set, then it sets $v_i := \mathsf{true}$, otherwise it sets $v_i := \mathsf{false}$.

If P_i does not want to sign the contract ($x_i = \mathsf{reject}$), then it sets $v_i := \mathsf{false}$.

(2) Protocol π is executed with input v_i. Let d_i be the result P_i obtains from this protocol run.

(3) If P_i decides $d_i = \mathsf{true}$ then it sends $\mathsf{sign}_i(c)$ to all parties.

We call any set $M_{tid} = \{\mathsf{sign}_{j_1}(c), ..., \mathsf{sign}_{j_k}(c)\}$ with $\{P_{j_1}, ..., P_{j_k}\} \in \mathcal{H}$ **major proof set** for *contr*. P_i waits until it receives such a major proof set and then stops.

The verifier V decides $(\mathit{tid}, \mathit{contr}, \mathsf{accepted})$ if and only if V receives a major proof set for *contr*.

Theorem 4. *Protocol 2 solves contract signing under condition Q_2.*

Proof. (sketch) We have to show that the protocol achieves verifiable consensus for the specific validity function F_{CS} (Def. 6). As previously, we present only the more important parts of the proof.

Validity (Def. 4): If not all honest parties wish to sign the same contract, or no contract at all, then no party will receive a minor proof set. Thus, all honest parties will start π with the input false, and π will yield the result false according to the definition of Byzantine Agreement. Therefore, all honest parties output failed, as required.

In case all honest parties wish to sign the same contract *contr* both outputs *contr* and failed are allowed.

No Surprises (Def. 5): Assume that the verifier V obtained the result $(tid, contr, \mathsf{accepted})$ from some party P_j. Then, P_j must have shown to V a major proof set for *contr* with the set of signatories H'. The condition Q_2 implies that there is some honest party P_h in H'. Then P_h received the minor proof set for *contr*, which means that all honest parties received the input *contr*. Besides, P_h must have received true from π, and therefore, all honest parties received or will receive the output true from π and therefore, the result *contr*.

□

6 Discussion

6.1 System Assumptions

Most debatable assumption in our system model is reliable communication. In fact, many cryptographic protocols for peer-to-peer ad hoc networks, most notably, group key agreement protocols [3, 4, 21], also make this assumption. This can be justified by relying on reliable group communication services, such as [18, 19].

Another assumption is the ability of the parties to digitally sign their messages. This requires a public-key infrastructure, such as described, e.g., in [14]. For an overview of authentication mechanisms in ad hoc networks, including issues related to the public key infrastructure, see [1].

6.2 Applications

Verifiable Agreement applies to situations where the result of a transaction should be used in the future. One class of such situations arises when a distributed database is implemented in the ad hoc network and is replicated across some specified nodes, the servers. In this case, transactions conducted in the absence of the servers, should be communicated to them as soon as possible. Consider, for example, the distributed public-key infrastructure in [24]. Using verifiable agreement of the client nodes on the exclusion of "bad" nodes form the network, each agreement participant can submit the exclusion decision to the service for the purpose of certificate revocation.

Another important scenario arises when the transaction conducted by the node in the peer-to-peer group should be used in another context. Consider a meeting which is set up in ad hoc manner, perhaps in an emergency situation, where several organizations from different organizations or countries do not trust each other. They collaboratively edit an important document which they should present in their organizations after the meeting. This document may be, e.g., the minutes of the meeting. It is important to fix the current document state, such that no single party is able to change the local copy of the document undetected. Usually, this can be done using a transaction logging by a trusted site. In the absence of a trusted site, the participants may sign the commitments to the document using a contract signing protocol. To do this, however, as we showed earlier, more than the half of the participants should be trusted not to cheat. In the full version of this paper, we present a contract signing protocol for honest majorities which can be used in the above situations.

6.3 Conclusion

We introduced the notion of Verifiable Agreement, and showed its applicability in mobile peer-to-peer ad hoc networks. Limits on the solvability of Verifiable Agreement show that the non-repudiation of any action without relying on an infrastructure requires placing trust into the majority of the participants.

References

1. N. Aboudagga, M. T. Refaei, M. Eltoweissy, L. A. DaSilva, and J.-J. Quisquater. Authentication protocols for ad hoc networks: taxonomy and research issues. In *Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, pages 96–104, New York, NY, USA, 2005. ACM Press.
2. N. Asokan, B. Baum-Waidner, M. Schunter, and M. Waidner. Optimistic synchronous multi-party contract signing. Technical Report Research Report RZ 3089, IBM Zurich Research Laboratory, 1998.
3. N. Asokan and P. Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.
4. D. Augot, R. Bhaskar, V. Issarny, and D. Sacchetti. An efficient group key agreement protocol for ad hoc networks. In *First International IEEE WoWMoM Workshop on Trust, Security and Privacy for Ubiquitous Computing*, 2005.
5. B. Baum-Waidner and M. Waidner. Round-optimal and abuse-free multi-party contract signing. In *International Colloquium on Automata, Languages and Programming*, LNCS 1853, 2000.
6. M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1), 1990.
7. M. Blum. Three applications of the oblivious transfer. Technical report, Department of EECS, University of California, Berkeley, CA, 1981.
8. G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4), 1985.
9. M. J. Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory*, pages 127–140, London, UK, 1983. Springer-Verlag.
10. M. Fitzi and J. A. Garay. Efficient player-optimal protocols for strong and differential consensus. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 211–220, New York, NY, USA, 2003. ACM Press.
11. J. A. Garay and P. D. MacKenzie. Abuse-free multi-party contract signing. In *Proceedings of the 13th International Symposium on Distributed Computing*, pages 151–165, London, UK, 1999. Springer-Verlag.
12. S. Goldwasser. Multi party computations: past and present. In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6, New York, NY, USA, 1997. ACM Press.

13. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 25–34, New York, NY, USA, 1997. ACM Press.
14. J.-P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–155, New York, NY, USA, 2001. ACM Press.
15. K. Kursawe. *Distributed Trust*. PhD thesis, Department of Computer Science, Saarland University, 2001.
16. L. Lamport. The weak byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.
17. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
18. J. Liu, D. Sacchetti, F. Sailhan, and V. Issarny. Group management for mobile ad hoc networks: design, implementation and experiment. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pages 192–199, New York, NY, USA, 2005. ACM Press.
19. J. Luo, P. T. Eugster, and J.-P. Hubaux. Pilot: Probabilistic lightweight group communication system for ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(2):164–179, 2004.
20. M. Pease, R. Shostak, and L. Lamport. Reaching agreement in presense of faults. *Journal of the ACM*, 27(2):228–234, April 1980.
21. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Trans. Parallel Distrib. Syst.*, 11(8):769–780, 2000.
22. L. Tang. Verifiable transaction atomicity for electronic payment protocols. In *ICDCS '96: Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS '96)*, Washington, DC, USA, 1996. IEEE Computer Society.
23. J. Zhou, J. Onieva, and J. Lopez. A synchronous multi-party contract signing protocol improving lower bound of steps. In *SEC 2006: 21st IFIP International Information Security Conference*, May 2006.
24. L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.