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

(Extended Abstract)

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, Informatik 1
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* [19], 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* [8], 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) [16], Weak Byzantine Agreement [15], Atomic Commitment [21], Strong Consensus [9], Validated Byzantine Agreement [14].

In contrast to Secure Multi-Party Computation [11], the inputs of the parties do not need to be secret or independent.

Contract signing [6] 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 [21]. 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 [14]: 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, 10]. For an overview of recent work, see also [22].

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 4.1). 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 4.2.

Finally, in Section 5, 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 cannot forge signatures.

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

2.2 Preliminary Definitions

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

Definition 1. *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.

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 $H_1, H_2, H_3 \in \mathcal{H}$ [12].

[4] The corresponding notion from the area of secret sharing is access structure. An adversary structure [12], which consists of all sets of dishonest parties a protocol can withstand, is the complement of it.

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 we use in the following to describe validity conditions of agreement problems.

Definition 2. *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. 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 $y \in X^P$.

3 Definition of Verifiable Agreement

We first define agreement problems.

Definition 3. *An* agreement problem *for a validity function F is to devise a protocol* consensus[] *for parties $P_1, ..., P_n$. In order to start the protocol, a party P_i receives the input x_i. Upon termination, the protocol produces an output $y_i \in Y$ 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, 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: *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 4. *A* verifiable agreement *problem for a validity function F is to devise, in addition to the protocol* consensus[]*, the protocol* verify[] *which involves only one party P_i and a verifier V which does not have any knowledge about the execution of* consensus[] *or about possible previous runs of* verify[].

Party P_i starts verify[] *with thse input (tid, y) where tid is the transaction identifier of an execution of* consensus[] *and y is the result obtained from this execution. The verifier decides* accepted *or* verify_failed *for (tid, y). The following requirements must be satisfied in addition to those from Definition 3 for an honest V and all sets $H \in \mathcal{H}$ of actually honest parties:*

- Verifiability of Correct Result: *If $P_i \in H$ obtained the output y for some tid for $y \neq$* failed *from* consensus[]*, then V will obtain the result* accepted *for (tid, y).*
- Non-verifiability of failed: *The verifier V never accepts* failed *for any tid.*
- No Surprises: *If some $P_i \in H$ obtained y for some tid from* consensus[]*, then V never decides* accepted *for any party P_j, tid, and any $y' \neq y$.*
- Termination of verify[]: *Each V and each $P_i \in H$ eventually terminate.*

$\Diamond$

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

Definition 5. Contract signing *is a verifiable consensus problem described by the following validity function:*
$X := C \cup \{\text{reject}\}$*, where C is a finite set of contract texts that can be signed,* $Y := C \cup \{\text{failed}\}$*, and $\mathcal{H}$ is the power set of P. Then:*

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

$\Diamond$

In the full version of this paper, we show that the above definition and the "usual" definition from, e.g., [2] are equivalent.

4 Solvability of Verifiable Agreement

4.1 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 [19]. As the synchronous network is the most strong network model, this result implies non-solvability for all other network classes.

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

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.

Due to space limit, we omit the proofs from this extended abstract.

4.2 Verifiable Agreement for Honest Majorities

We show how to extend any agreement protocol for honesty structures satisfying the condition Q_2 to a verifiable agreement protocol.

Protocol 1. Let π be an agreement protocol (Definition 3) 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 $y \neq \mathsf{failed}$.

$\Diamond$

Theorem 2. *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 in this extended abstract.

Termination of consensus[] (Definition 3): 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 4): 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.

$\square$

Remark 1. If a protocol solves some agreement problem in asynchronous networks, the corresponding honesty structure must satisfy Q_3 [7]. 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 Discussion

5.1 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, 20], also make this assumption. This can be justified by relying on reliable group communication services, such as [17, 18].

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 [13]. For an overview of authentication mechanisms in ad hoc networks, including issues related to the public key infrastructure, see [1].

5.2 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 [23]. 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.

5.3 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. Blum. Three applications of the oblivious transfer. Technical report, Department of EECS, University of California, Berkeley, CA, 1981.
7. G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4), 1985.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. K. Kursawe. *Distributed Trust*. PhD thesis, Department of Computer Science, Saarland University, 2001.
15. L. Lamport. The weak byzantine generals problem. *J. ACM*, 30(3):668–676, 1983.
16. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
17. 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.
18. 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.
19. M. Pease, R. Shostak, and L. Lamport. Reaching agreement in presense of faults. *Journal of the ACM*, 27(2):228–234, April 1980.
20. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Trans. Parallel Distrib. Syst.*, 11(8):769–780, 2000.
21. 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.
22. 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.
23. L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.