ON THE SECURITY OF MULTI-PARTY PROTOCOLS IN DISTRIBUTED SYSTEMS

Danny Dolev[†]

Institute of Mathematics and Computer Science
Hebrew University, Jerusalem

Avi Wigderson

Electrical Engineering and Computer Science Department
Princeton University
Princeton, NJ 08544

ABSTRACT

Security of protocols for network communication has received considerable attention in recent years. We concentrate on ensuring the security of cryptographic protocols in distributed systems.

In a distributed system, beyond eavesdropping, a saboteur may impersonate another user or alter messages being sent. A saboteur who is also a user may send conflicting messages or use other illegal messages in order to uncover secret information.

The problem we address, in its most general form, is: "given a multi-party protocol which is provably secure when all the participants monitor every message being sent, can the protocol be modified to be secure in a distributed system?"

We use the Byzantine Agreement, Crusader Agreement, and other specific checks to improve protocols by making them secure in a general distributed network. We examine the trade-off between detection of faulty behaviour and the number of messages exchanged.

1. Introduction

The main purpose of a cryptosystem is to protect private information. Cryptographic protocols enable users to employ the cryptosystem while com-

[†] Part of this work has been done while the first author visited IBM Research Center, San Jose, California.

municating with each other. A cryptosystem is secure if it cannot be "broken" using various mathematical tools. The security of a cryptographic protocol is measured by its ability to prevent eavesdroppers and other non-participants from understanding the information being exchanged. Distributed cryptographic protocols are intended to guarantee the security of private information in case one cannot trust even the users participating in the protocol itself. We will make sure that private information will not be disclosed even in the case that several participants collaborate in attempting to cause such disclosure.

The security of cryptographic protocols has been discussed in previous papers (DEK, DLM, DY, LW, NS). In this paper we assume that we are given a protocol which is secure in a *round-table environment* (or round-table secure). That is, the protocol is secure provided every user is able to see every message being exchanged between every pair of users in the system, provided users are consistently following the protocol, and provided deviations from the protocol can be detected. We will show how to modify such a protocol to make it secure in a distributed system in which a user can see only information he himself receives. For example, all the protocols presented in (LW) are round-table secure. But the same protocols, when used in a distributed system are no longer secure.

The difference between a round-table and a distributed system is that in the latter a user can slightly deviate from the protocol in a way that the recipient of the message cannot be aware of, and by doing so he is able to uncover secret information. As a matter of fact, a saboteur in a distributed system can do can do whatever he wants as long as the user receiving the message does not suspect a misbehaviour, or even if he suspects, he is unable to obtain a "proof of misbehaviour".

The methods we will give for improving round-table secure protocols by making them distributed-secure produces protocols that are fault-tolerant in a very broad sense. The protocols will be able to sustain **any** malicious behaviour without ever revealing an individual's secret. We make no assumptions about the type of faulty behaviour nor about the communication network. We guarantee that in the case that the participants are all faithful, the protocol proceeds as usual. A malfunction may cause the protocol to stop. However, even in this case a saboteur cannot uncover private information. The algorithms we use will try to overcome the faulty behaviour, but we cannot guarantee to identify all the faulty users, as one might have hoped. The reason is that a faulty user can behave in a way that does not disclose his faultiness. However in cases where faultiness may endanger the security of the protocol, a faulty user will be detected.

2. The Model

Let U be a set of users participating in a given network. For simplicity we assume that all the users in U participate in the protocol. Moreover, we assume that the network is such that every two users can communicate directly. To relax this assumption one can follow the results of (Da, LSP) and obtain similar restrictions on the network connectivity. Let CU be the subset of correct (or faithful) users, and TU be the rest of the users, named the traitors (faulty users or saboteurs). We assume that faithful users do not know which users are members of which sets. Faithful users behave correctly; they follow the protocols and correctly execute the algorithms. There are no assumptions on the behaviour of saboteurs; they may even collaborate in trying to break the protocol. Notice that saboteurs may know who are the members of each set. Moreover, the can behave faithfully, or at least pretend to do so. We do assume that saboteurs cannot break the cryptosystem.

Let t be an upper bound on the number of saboteurs that may participate in the protocol, that is, t is the cardinality of the set TU. Our results are parametrized by t.

We assume that users communicate synchronously and thus an absence of a message can be detected. The synchronous behaviour can be relaxed, but then one needs an upper bound for the time it takes for a faithful user to respond to a message he receives (FLP).

In the analysis that follows, we assume that no message has been stopped by eavesdroppers. To overcome the stopping we need to include this misbehaviour in the parameter t in some way (if this possibility was not limited, one faulty user could have isolated every other user from all others). We will not cover this extension here.

Let MSG be the set of possible messages. We assume MSG is closed under the operation of forming sequences from MSG.

A *synchronized multi-party protocol P for a set of users U* is a finite sequence of phases $G_1, G_2, \cdots, G_l$. Each phase is a directed graph $G_k(U, E_k, T_k)$ where nodes correspond to the users in U, and the edges E_k are labeled from MSG according to T_k. For every directed edge (u,v) in E_k, $T_k(u,v)=f(k, I_u, M(k,u))$, where f is a function of I_u, the private information of user u, and $M(k,u)$, that contains the information u obtains through phase k from messages he receives. ($M(1,u)$ is empty for every faithful user u).

The phases are numbered with consecutive positive integers. At each phase k, a user should send the messages on his outedges and receive the messages on his inedges.

A protocol works in a *round−table* environment if for every faithful user u, $M(k,u) = M(k-1,u) \cup \{T_k(a,b) \mid a,b \in U\}$. In other words, every faithful user has the same information about every message that was sent in each phase.

A protocol is *round−table secure* if it is secure under the condition that a faithful user sends a message at the kth phase only if for every pair of faithful users a,b we have $M(k-1,a)=M(k-1,b)$.

A protocol works in a distributed system if at every phase k and for every faithful user u, $M(k,u) = M(k-1,u) \cup \{T_k(a,u) \mid a \in U\}$. In other words, a faithful user obtains only the messages he received at each phase.

We assume the exsistence of a cryptographic signature scheme with the following properties:

(1) Every user x has a distinct signature function S_x with which he signs messages.

(2) Every user can identify the signature of every other user and can extract the message from the signed message.

(3) No user (not even a saboteur) can forge the signature of another user.

(4) Every change in a signed message which is not done by the user who signed it can be detected by any other user.

(5) The signature functions do not commute, i.e. for every message M $S_x S_y(M) \neq S_y S_x(M)$ whenever $x \neq y$.

A signature scheme with the above properties can be constructed using a (secure) public-key cryptosystem (DH, RSA). We further assume that the cryptographic system used for signatures is independent of the one used for encryption.

using the cryptographic signature scheme one can obtain messages with several signatures, can extract the contents of a message, and can identify the various signatures (and their order) a message carries. This will be the way we will use the signature scheme in the coming sections.

We use the following notation: A user notarizes a message if he signs it and sends it to all users. (Our algorithm works also if faulty users are able to remove signatures notarizing a message in an undetectable way). For simplicity we will assume that when a faithful user sends a message he also sends it to himself. A message is said to be *k-dense* if its contents is followed (notarized) by exactly k distinct signatures. Notice that the contents of a message can be another message which by itself carries signatures. We assume that the content of a message is uniquely defined. (In fact, it follows from our definition of a protocol).

3. Distributed Agreements

The security of a round-table secure protocols is based on the fact that all the faithful users obtain the same information about every message being exchanged in each phase. The idea is that by obtaining that information they can check and find deviations from the protocol and stop the protocol before the saboteur is able to break it. We will try to bring the users in a distributed system to an agreement on what messages each user has sent. By doing that we will induce the security of a protocol in a distributed system from its security in a round-table environment. Our exact goal is : "Given a round-table secure protocol P, produce a modified distributed protocol P' such that at every phase of P' which correspond to a phase of P there is an agreement about the messages sent in the previous phase of P, or at the first time that this does not hold, at least one faithful user holds a "proof" about the faultiness of some user".

Two types of distributed agreements have been discussed previously in the literature: Byzantine agreement and Crusader agreement. These two agreements will enable us to improve a round-table secure protocol. For the Crusader agreement let us assume that there exists a transmitter who is supposed to

send his value to the netw
do so.

Crusader Agreement

(1) If the transmitter is
value he has sent.

(2) All the faithful use
transmitter should a

The elements of the
faithful they reach agre
the transmitter is intro
(the users do not necess
includes all the faithfu
transmitter, and the se
hold such a proof. All n
value as the value of the

In the following alg
transmitter should conta

The Crusader Algori

C1. The transmitter not

C2. If at the end of pha
then notarize it at
transmitter is fault

C3. If at the end of ph
values or at least t
are received define
only notarized valu

Theorem 1:
If the cardinality of th
algorithm reaches the C

The proof is simila
a faithful user can hold
signed also by the tr
transmitter is faulty. N
itself is faulty.

Lemma 1:
A faithful user holds a
the transmitter is faith
faultiness.

The natural idea i
about every message
agreement among the
that at the end of the

send his value to the network and that all the users know when he is supposed to do so.

Crusader Agreement (Da)

(1) If the transmitter is faithful, then all the faithful users should agree on the value he has sent.

(2) All the faithful users who do not hold a "proof" of faultiness of the transmitter should agree on the same value.

The elements of the Crusader agreement are that as long as all users are faithful they reach agreement about every value being sent. When faultiness of the transmitter is introduced then the faithful users are divided into two sets (the users do not necessarily know who are the members of each set). One set includes all the faithful users holding a "proof" of the faultiness of the transmitter, and the second set is the set of all the faithful users who do not hold such a proof. All members of the second set should decide on the same value as the value of the transmitter.

In the following algorithm a notarized message carrying a value of the transmitter should contain also the transmitter's signature.

The Crusader Algorithm:

C1. The transmitter notarizes his value at phase 1.

C2. If at the end of phase 1 a single value signed by the transmitter is received, then notarize it at the next phase. Otherwise notarize the message "the transmitter is faulty".

C3. If at the end of phase 2 at least 2 2-dense messages containing different values or at least $t+1$ signed messages saying that the transmitter is faulty are received define it to be a *proof of faultiness*. Otherwise agree on the only notarized value you have received.

Theorem 1:
If the cardinality of the set of faulty users is bounded by t, then the Crusader algorithm reaches the Crusader agreement at the end of phase 2.

The proof is similar to the proof in (Da). Observe that the proof of faultiness a faithful user can hold at the end of the algorithm is either two different values signed also by the transmitter or t+1 signed messages claiming that the transmitter is faulty. Neither of these can be produced unless the transmitter itself is faulty.

Lemma 1:
A faithful user holds a proof of faultiness only if the transmitter is faulty. If the transmitter is faithful, then no user (even a faulty one) can hold a proof of faultiness.

The natural idea is to run the algorithm to obtain the Crusader agreement about every message $T_k(u,v)$ being sent and by that to reach some sort of agreement among the faithful users about the various messages. The problem is that at the end of the Crusader agreement some of the faithful users do agree on the message but some do not. So if those who did not find a "proof" of faultiness continue to follow the protocol they will send the next messages when the conditions about $M(k,u)$ do not hold.

We can obtain a round-table environment by running a Crusader agreement on every message and then adding another phase in which only users who hold a proof of faultiness will send it and the rest will wait a phase without doing anything. By doing this we ensure that every user who does not receive a proof of faultiness by the end of that phase can be sure that all the faithful users have obtained agreement on the value of the transmitter at the end of the Crusader agreement. Therefore if he continues that protocol he cannot cause any security problem because of the fact that the round-table conditions held at the previous phase.

If a faithful user obtains a proof of faultiness at the end of the Crusader algorithm, then by the end of the additional phase all the faithful users will learn about that. This will prevent them from continuing the protocol and will stop the protocol at that point without enabling the faulty users to break it.

Observe that it still may happen that some of the faithful users will see a proof of faultiness at the end of the additional phase and will stop taking part in the protocol because of that. This will bring about a situation in which not all the users have the same information. But in this case those that will continue will be covered because of the round-table condition and those that have stopped will try to stop the protocol before further phases will be completed.

To do this we actually need to run another algorithm that will distribute that proof of faultiness among the faithful users in order to bring them to an agreement about the faultiness of the transmitter. For that we use the Byzantine agreement. For the Byzantine agreement let's assume that users may hold a legal proof (of faultiness) and that every user can check its legality.

Byzantine Agreement (Da, DS, LSP, PSL)

(1) If a faithful user holds a legal proof, then all the faithful users should agree on the existence of a legal proof.

(2) All the faithful users should reach the same agreement.

The difference between the two agreements seems to be small but it is important. The Byzantine agreement requires reaching agreement no matter what. The Byzantine agreement requires more phases than the Crusader agreement. We will run it after completing the Crusader agreement in parallel with the rest of the protocol in such a way that if a fault has been found, the Byzantine agreement will broadcast it and will stop the rest of the protocol.

The Byzantine agreement will be used here to agree on the faultiness of some user. For this reason the version of the algorithm presented here will be optimized for this use.

The Byzantine Algorithm:

B1. Every faithful user who holds a legal proof notarizes his proof at phase 1.

B2. If at the end of phase
by a user who did no
phase and agrees on t

B3. If by the end of phase
agree on the proof of

Theorem 2:

If the cardinality of the s
algorithm reaches the By
phase $k<t+1$ some faithf
phase $k+1$ all the faithful

The proof is simila
definition of Byzantine ag

4. Algorithm for Improvi

Let P be a given pro
from it a protocol P' with
secure for a distributed s
phases as P, s.t. each pha

A proof of faultiness

(1) it is about sending c
phase $3k$-2.

(2) it contains either tv
user a at that phase
$T_k(a,b)$ claiming no

(3) it is k-dense.

(4) it is received by pha

The legality of a
assumptions, by every
then the receiving user

To simplify the arg
message a user has to
user who suppose to re

Crusader-Byzantir

Define P' to be the prot

CB1. At phase 3k-2: for
faultiness by now
according to the
send the message
algorithm takes tv

CB2. At phase 3k: If a
b at phase $3k$-2,
proof of fault
$M(k,u)=M(k-1,u$

ɔt find a "proof" of faulti-
next messages when the

ıg a Crusader agreement
ich only users who hold a
ɔhase without doing any-
es not receive a proof of
ll the faithful users have
the end of the Crusader
annot cause any security
conditions held at the

:he end of the Crusader
e faithful users will learn
›rotocol and will stop the
.o break it.

faithful users will see a
d will stop taking part in
ation in which not all the
that will continue will be
e that have stopped will
npleted.

thm that will distribute
ier to bring them to an
that we use the Byzan-
ıme that users may hold
k its legality.

thful users should agree

ɘnt.

.s to be small but it is
; agreement no matter
ıan the Crusader agree-
eement in parallel with
been found, the Byzan-
the protocol.

ree on the faultiness of
ı presented here will be

his proof at phase 1.

B2. If at the end of phase $k<t+1$ a k-dense legal proof of faultiness is received by a user who did not notarize it before, then he notarizes it at the next phase and agrees on the proof of faultiness.

B3. If by the end of phase $t+1$ a $t+1$-dense proof of faultiness is received, then agree on the proof of faultiness.

Theorem 2:

If the cardinality of the set of faulty users is bounded by t, then the Byzantine algorithm reaches the Byzantine agreement at the end of phase t+1. If at some phase k<t+1 some faithful user agrees on a legal proof of faultiness, then by phase k+1 all the faithful users will agree on a legal proof of faultiness.

The proof is similar to the proof in (DS) although our algorithm and definition of Byzantine agreement is somewhat different.

4. Algorithm for Improving Security

Let P be a given protocol. The following algorithm will show how to obtain from it a protocol P' with the property that if P is round-table secure, then P' is secure for a distributed system. Furtheremore, P' will have three times as many phases as P, s.t. each phase k in P corresponds to phases $3k$-2, $3k$-1, $3k$ in P'.

A proof of faultiness is called legal if

(1) it is about sending or not sending some $T_k(a,b)$ that had to be sent by a at phase $3k$-2.

(2) it contains either two different values being sent (and signed) by the faulty user a at that phase or $t+1$ signatures of users who are supposed to receive $T_k(a,b)$ claiming not to receive it at that phase.

(3) it is k-dense.

(4) it is received by phase $k' < 3k$ of P'.

The legality of a given proof of faultiness can be checked, due to our assumptions, by every user who receives it. If a proof of faultiness is not legal, then the receiving user can ignore it.

To simplify the arguments in the following algorithm we assume that every message a user has to send according to protocol P contains the name of the user who suppose to receive that message in P.

Crusader-Byzantine algorithm

Define P' to be the protocol obtained from P as follows: for every $k > 0$

CB1. At phase 3k-2: for every faithful user u, if u did not agree on a legal proof of faultiness by now, then for every $T_k(u,v)$ that u is supposed to send according to the protocol P at phase k he uses the Crusader algorithm to send the message "$T_k(u,v)$, the phase is $3k$-2". (Recall that the Crsader algorithm takes two phases, $3k$-2 and $3k$-1).

CB2. At phase 3k: If a faithful user u holds a proof of faultiness about some user b at phase $3k$-2, then he starts a Byzantine agreement to send that legal proof of faultiness to all the users. Otherwise, he defines $M(k,u)=M(k-1,u) \cup \{T_k(a,b) \mid \text{u has agreed on at phase } 3k\text{-}1\}$.

CB3. If $3k$-2 is the earliest phase about which a faithful user u holds a legal proof of faultiness, then u "STOP"s the protocol at phase $3k+t+1$.

Lemma 2:

In the Crusader-Byzantine algorithm, if the cardinality of the set TU is bounded by t and a faithful user holds a legal proof of faultiness at some phase k, then by phase $k+1$ all the faithful users will hold a proof of faultiness. Moreover, if a legal proof of faultiness about some phase $3r$-2 has not been received by any faithful user by phase $3r+t$, then no faithful user will ever later accept any proof of faultiness about phase $3r$-2 as being legal.

The proof follows from the properties of the Byzantine agreements and the way we use them in the Crusader-Byzantine algorithm.

Notice that we can save one phase in lemma 2 and in stopping the protocol if we would exclude the faulty user while running the Byzantine agreement about his fault.

Lemma 3:

In the Crusader-Byzantine algorithm, if a faithful user u decides to send $T_k(u,v)$ at phase $3k$-2 of P, then $M(k-1,a)=M(k-1,b)$ for every pair a,b of users.

These lemmas enable us to prove the main theorem about the security of the new protocol P'.

Theorem 3:

If the protocol P is round-table secure and the cardinality of TU is bounded by t, then the protocol P' obtained from P using the Crusader-Byzantine algorithm is secure. Moreover, in case the protocol P' stops with a proof of faultiness, then a faulty user is found and this fact is known to all the users.

Observe that if the security of P holds only in the case where t is bounded by some function of $n=|U|$, then in the new protocol P' it will also be bounded by that function of n. For example the round-table secure protocols in (LW) require that $t<n-1$ or $t<n/2$, and therefore in the protocol one obtains from it using the Crusader-Byzantine algorithm, t should also be be similarly bounded for ensuring the security of the protocol.

5. Conclusions

The number of phases in the new protocol, P', is only three times as many as the number of phases in P. However, The number of messages that P' uses is more than that of P by a factor of n^2. Using better algorithms for reaching Byzantine agreement and similar ideas in the Crusader agreement one can reduce this factor to nt. One can save even more, using the algorithm in (DR), but that algorithm will require many more phases for reaching the agreement about the faultiness of a user. Observe that if one is not interested in reaching agreement about the faultiness of a user, then the number of messages can be further reduced, and so can the number of phases.

The algorithm we pr
protocols from that of a r
centrate on producing r
using our algorithm to b
needed for obtaining the
protocols to be secure in

The ideas presented
every two users can com
for improving protocol
assumptions than the rou

References

(Da) D. Dolev, "The Byzar
3, no. 1, pp.14-30, 19

(DEK)D. Dolev, S. Even, a
CRYPT82, Santa Bar

(DH)W. Diffie, and M. Hel
Information Theory,

(DLM)R. A. DeMillo, N.
Proceedings of the
May 1982.

(DR) D. Dolev, and R. Rei
Agreement," Procee
ples of Distributed S

(DS) D. Dolev, and H. R.
Agreement," Procee
Computing, May 198

(DY) D. Dolev, A. C. Yao
IEEE Trans. on Info

(FLP)M. J. Fischer, N. A.
Consensus with One

(LSP)L. Lamport, R. Sh
ACM Trans. on Prog

(LW) R. J. Lipton, and
unpublished manus

(NS) R. M. Needham, an
Large Networks of

(PSL)Presence of Fault

(RSA)R. Rivest, A. Sham
tures and public-k

The algorithm we presented gives a way to induce security of distributed protocols from that of a round-table protocol. Thus, a protocol designer can concentrate on producing round-table secure protocols, and then convert them using our algorithm to be secure in a distributed system. Further research is needed for obtaining the most efficient way to transform nondistributed secure protocols to be secure in a distributed system.

The ideas presented in the paper can also be used for networks in which not every two users can communicate directly. In addition, the ideas can be used for improving protocols that are secure in environments fulfilling weaker assumptions than the round-table environment.

References

(Da) D. Dolev, "The Byzantine Generals Strike Again," Journal of Algorithms, vol. 3, no. 1, pp.14-30, 1982.

(DEK) D. Dolev, S. Even, and R. M. Karp, "On the Security of Ping-Pong Protocols," CRYPT82, Santa Barbara, Aug. 1982.

(DH) W. Diffie, and M. Hellman, "New Direction in Cryptography," IEEE Trans. on Information Theory, IT-22, 6, pp. 644-654, 1976.

(DLM) R. A. DeMillo, N. A. Lynch, and M. Merritt, "Cryptographic Protocols," Proceedings of the 14th ACM SIGACT Symposium on Theory of Computing, May 1982.

(DR) D. Dolev, and R. Reischuk, "Bounds on Information Exchange for Byzantine Agreement," Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Systems, Aug. 1982.

(DS) D. Dolev, and H. R. Strong, "Polynomial Algorithms for Multiple Processor Agreement," Proceedings of the 14th ACM SIGACT Symposium on Theory of Computing, May 1982.

(DY) D. Dolev, A. C. Yao, "On the Security of Public Key Protocols," to appear, IEEE Trans. on Information Theory.

(FLP) M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," unpublished manuscript, Aug. 1982.

(LSP) L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Trans. on Programming Languages and Systems, to appear.

(LW) R. J. Lipton, and A. Wigderson, "Multi-Party Cryptographic Protocols," unpublished manuscript, May 1982.

(NS) R. M. Needham, and M. D. Schroeder, Using Encryption for Authentication in Large Networks of Computers," CACM, vol.21, no. 12, pp. 993-999, 1978.

(PSL) Presence of Faults," JACM, vol. 27, no. 2, pp. 228-234, 1980.

(RSA) R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," CACM, vol. 21, pp. 120-126, 1978.