Arbitrated multiparty protocol

Khalil Challita
Notre Dame University - Louaize
Department of Computer Science
Zouk Mosbeh, Lebanon

Abstract

Assume that several people wish to have a meeting. For this to happen, they usually have to meet somewhere. If they cannot meet physically, then they can take part in a video (or audio) conference to discuss whatever needs to be discussed. But what if their meeting is meant to be private? In this case they need a cryptographic protocol that allows them to exchange their ideas remotely, while keeping them secure from any potential eavesdropper. In this paper we list all the necessary requirements that a cryptographic protocol must have in order to allow several persons to exchange their ideas securely over the Internet. Moreover, and based on the standard taxonomy of cryptographic protocols, we suggest several approaches on how to design cryptographic protocols that enable us to achieve our aim. Finally, we propose the design of a protocol that solves our problem.

Key-words: Public-key cryptography, Cryptographic protocols, Trusted party, Secure authentication protocols.

1. Introduction

Cryptographic protocols can serve a wide variety of purposes. They play a key role in achieving confidentiality among users of a computing system when needed. In his book, Schneier [25] classifies protocols into four categories. He describes basic and intermediate protocols such as authentication, key exchange, bit commitment, and mental poker; as well as advanced and esoteric protocols such as zero knowledge proofs, blind signatures, secure elections, and digital cash.
The invention of public-key cryptography by Diffie and Hellmann [11] paved the way to the formulation of some sophisticated protocols that solved very interesting problems such as the Dining Cryptographers' Problem (introduced by Chaum [7]), and the Millionaires' Problem (introduced by Yao [28]), which could not be solved using private-key cryptography. In the dining cryptographers' problem, a group of cryptographers have a dinner. After it, someone pays for the meal, but no one is able to tell whether a cryptographer or the NSA paid for that meal. In the millionaires problem, two millionaires wish to know who is richer without revealing to the other the amount of his/her fortune. In his book, Mao [17] explains in a very clear and concise way how much hard it is to design secure protocols, even by experts in the field. He gives examples of protocols that were used for years before researchers discovered they were flawed (see for e.g. the Needham Schroeder protocol [21] and the Woo-Lam protocol [26] that we describe in more detail in Section 5). Several researchers [20, 24, 3, 4, 2] developed formal methods in order to analyze the security of authentication and key-exchange protocols. Canetti [5, 6] and Goldreich [13] analyzed the security of cryptographic protocols. Abadi and Needham [1] presented principles for designing secure cryptographic protocols.
One of the main contributions of this paper is to propose a new protocol that allows people to meet remotely. To our knowledge, no one has addressed and solved this problem so far. Classical encryption/decryption schemes cannot work because they just involve securing the communications between two users. In the case of three or more persons, we need a protocol that enables them to meet securely while satisfying the requirements of an actual meeting, where people meet physically in one room. Based on the standard taxonomy of cryptographic protocols, we also suggest several designs to solve our problems. In other words, we consider four different cases to design such a protocol: with or without an arbiter, and with or without using public-key cryptography. One of our main goals is to come up with a solution that is *inherently* correct and meets the requirements specified in Section 2. For example, we cannot guarantee the fact that two of the persons who are in a meeting do not cheat by exchanging secret messages by calling each other on the phone for example, or by using a private key they only share together. But the same scenario could happen during an actual meeting where two persons can have a private conversation during a break, or secretly talk together if they are sitting next to each others.

This paper is divided as follows. In Section 2, we list all the requirements a cryptographic protocol must meet in order to solve our problem. We also list the limitations of any such solution. In Section 3, we suggest the design of some protocols that use an arbiter for solving our problem. In Section 4, we propose several designs that do not use an arbiter. Before concluding, we propose in Section 5 a protocol that solves our problem. This protocol uses an arbiter and public-key cryptography, and is based on a refined version of the original Needham-Schroeder protocol [21].

2. Requirements and limitations

In this section we list the necessary requirements any cryptographic protocol should meet in order to solve our problem. To do so, we draw the parallel with what happens during an actual meeting where several persons gather to discuss private issues.
Moreover, we list all the limitations inherent to such a protocol. In other words, we state the assumptions under which this problem has a solution.
In our opinion, the approach we adopt can serve as a guideline to anyone who intends to design a solution to the problem.
We next give some basic terminology that will be helpful throughout the remaining of the text.

Definition 1 *1. The person who calls for a meeting is referred to as the* ***moderator****.*

2. *The persons who are in a meeting are called the* ***participants****.*
3. *Any person that is not supposed to be part of the meeting is called an* ***outsider****.*

Let us assume that several persons are in a meeting. Obviously, the following conditions hold:

- Every person is aware of the identities of the other participants;
- If a person says something, then it is heard by all the others, and they know who said it;
- No person outside of the meeting room is allowed to hear what is being discussed.

Hence the following 3 requirements:

1. Every businessman is fully aware of the identities of all the other participants in the meeting.
2. All the participants are aware of each other's communications. In other words, two persons should not be able to exchange any message without revealing its content to the others.
3. Their communications are kept confidential; meaning that no outsider can eavesdrop on their conversation.
4. Proof of origin: for each exchanged message, the participants must be aware of the identity of the person who wrote it.

Now basic encryption schemes (i.e. public or private key based cryptography) cannot solve this problem since they are concerned with securing the communications between just two parties. We need a specific protocol that solves our problem while meeting the abovementioned requirements.
But before suggesting some guidelines on how to build such protocol, we next state the main limitations inherent to any cryptographic protocol that intends to solve our problem. This part is of utmost importance since there may be some serious limitations that forbid us from finding a solution to our problem.

Let us assume that all the n participants share a secret key K_n. Two different scenarios may undermine any implementation of such a protocol:

- Scenario 1. During the meeting, a participant could reveal sensitive information to an outsider by using another medium than the protocol itself. For example by using a phone, by allowing him/her to sit next to him/her, or even by sending him/her messages using a secret key they both share.
- Scenario 2. Two participants may send each other a message using the key K_n, without sharing its content with the others.

The problem with the first scenario is that we cannot protect against it. This is because the participants are not physically present in the same room, and thus no one can control what the others are doing. It violates the first requirement since a participant may share all the conversations he/she is having with an outsider. In other words, it is as if $n+1$ persons were meeting instead of n, and the $(n+1)$st identity is kept secret to all the participants except for his/her accomplice.
The second scenario violates the second requirement. But we will see that we can deal with it.
Based on our previous analysis, the our problem has a solution under the following assumption:

All the participants are honest, in the sense that no one allows an outsider to share any part of his/her conversation.

This assumption is true in many real-life cases where a group of people who know each other wish to meet and do not wish to share any of their ideas with outsiders.
It is worth noting that the above assumption also applies

to an actual meeting where the participants gather in the same room, since no one can control the actions of all the participants. One could cheat (for example by hiding a wire, or a camera, or an audio receptor placed near his/her ear) and insidiously invite an outsider to the meeting.
We next state 2 obvious remarks.

***Remark* 1** *The above assumption does not forbid two or more participants from colluding together. This should be enforced by requirement number 2.*

***Remark* 2** *Any solution to this problem under the above assumption is at least as secure as any other means (e.g. phone or video conference) used to allow people to meet remotely.*

In the following sections, we suggest several approaches to help us design a solution to our problem, taking into account the previously mentioned assumption.
Needless to say, any solution should start by authenticating the participants of a meeting.

3. Protocols with an arbiter

Broadly speaking, a cryptographic protocol falls into one of the following 4 categories:

Case 1 Using an arbiter and public-key cryptography.

Case 2 Using an arbiter but without public-key cryptography.

Case 3 Without an arbiter but using public-key cryptography.

Case 4 Without an arbiter and without public-key cryptography.

In this section we indicate how to design protocols that use an arbiter (i.e. case 1 and case 2 above). These are just suggestions for future work. One may use them as guidelines to write a protocol that solves this problem.
The main steps for designing a protocol with an arbiter are the following:

Step 1. With the help of the arbiter, authenticate the participants.

Step 2. Generate a session key (with or without the help of the arbiter) and distribute it to the participants.

Step 3. For each sent message, verify that all the participants read it.

Steps 1 and 2 are done once. Note that in Step 2 we have the option to let the arbiter read the content of the exchanged messages or not. The former case applies when the arbiter generates the session key and distribute it to the participants. Step 3 is the most costly since it probably needs to be repeated each time a new message is exchanged among the participants. We next suggest several approaches to address this problem.

Case 1: Designing a protocol with an arbiter and with public-key cryptography.

1. Use a modified version of an authentication protocol (e.g. the Woo-Lam protocol [26], or the Needham-Schroeder protocol [21], or the Denning-Sacco protocol [10]) to authenticate the users and to generate a session key. This step is done once.

2. Each time a participant wishes to send a message, he/she has to send it to the arbiter (or the moderator) who signs it and then sends it to the other participants. No message is considered to be valid unless signed by the arbiter (or the moderator).

3. The meeting ends upon the request of the mediator (e.g. directly, or by sending a special request to the arbitrator to signal the end of the meeting).

We need to use a modified version of any currently authentication protocol, because they are all concerned with authenticating two users. Moreover, it was shown that all initial versions of these protocols were flawed (see for example [9]). A more detailed explanation of the security of these protocols is provided in Section 5.
One might replace Case 1 (1.) by any basic authentication protocol using an arbiter, and then let the participants (or just the moderator) generate a session key. In this case the arbiter will not be able to read the content of the exchanged messages.
In the above scenario (i.e. Case 1), the moderator initiates the protocol with the arbiter to authenticate all the users and to generate a session key. The second point in the above case is very costly and involves a lot of interactions from behalf the arbiter. A better solution would be to replace the arbiter with the moderator, since we assume that all the participants trust the moderator.
Trying to reduce the amount of work to be done by the moderator in Case 1 (2.) does not seem to work. For example, if we replace this step by the following ones:

2’. Each participant sends his/her message to the others.

2”. The moderator challenges some randomly selected participants to check whether or not they received the message.

Then 2 persons could cheat by sending each other a message without sharing it with the remaining participants. In this case the moderator will be unaware of the existence of that message! For this to work, no one must cheat; but then step (2”.) becomes useless.

Case 2: Designing a protocol that uses an arbiter but without public-key cryptography.

1. Use a modified version of the Otway-Rees protocol [23] (or the Neuman-Stubblebine protocol [22]) to authenticate the users and to generate a session key. This step is done once.
2. Each participant sends his/her message to the others.
3. The meeting ends upon the request of the mediator.

If we do not wish to use timestamps then we use the Otway-Rees protocol, otherwise we could use the Neuman-Stubblebine one.
In this protocol no message can be signed, so validating the origin of a message among a group of n persons who share the same secret key is impossible; unless in some extreme cases where all the participants trust each others and do not cheat. In other words, each time someone writes a message, he/she sends that message to all the other participants.
This concludes this section.
We now deal with the case were we have no arbiter.

4. Protocols without an arbiter

Case 4 of Section 3 cannot have a practical solution. Indeed, we need public key-cryptography to hope to find a solution to our problem. Consider for example the following protocol, where we assume that all the participants share a secret master key with the moderator:

Case 4: Designing a protocol without an arbiter and without public-key cryptography.

1. The moderator generates a session key. He/she then sends it to the participants encrypted using the master key he/she shares with each one of them, together with any additional information (e.g. identities of the other participants).
2. Each participant sends his/her message to the others.
3. The meeting ends upon the request of the mediator.

In this case the participants cannot collaborate in order to generate a secret session key. All they can do is to rely on the moderator to generate and distribute a new one. Implicitly, the moderator plays the role of an arbiter here. In Step 2 above, sending the message to the moderator in order to forward it adds no security to the protocol. Indeed, a participant may cheat by sending a message to a strict subset of the participants using the session key generated and distributed by the moderator. The fundamental problem that arises without the use of public-key cryptography is that there is no means of validating a message (e.g. by signing it). There is also the problem of message origin. Upon the receipt of a message, no one can be sure of the identity of the sender.
In the last case described below, we also assume that the moderator shares a secret master key with all the other participants.

Case 3: Designing a protocol without an arbiter but with public-key cryptography.

1. The participants authenticate themselves using a one-way function for example, or any basic authentication protocol.
2. The participants use secure multiparty computation to generate a session key.
3. Each time a participant wishes to send a message, he/she has to send it to the moderator who signs it and then sends it to the other participants. No message is considered to be valid unless signed by the moderator.
4. The meeting ends upon the request of the mediator.

This protocol assumes that the participants can verify the signature of the moderator. They can use secure multi-party computation (SMPC) as introduced by Yao [28] to generate a session key because there is no arbiter, and maybe not all of the participants trust the moderator to generate a secure session key (e.g. maybe the random number generator that he/she uses is weak).
Secure multi-party computation allows a set of n players to compute an arbitrary agreed function of their private inputs, even if an adversary may corrupt and control some of the players. Research related to SMPC can be found in [15, 12, 14, 19, 8].
Based on our analysis in this section and in Section 3, we can certify that no practical solution to our problem exists without the use of public-key cryptography. Thus practical implementations of this protocol must be based on Case 1 or Case 3 described in Section 3.

5. An arbitrated protocol: An example

In this section we propose a protocol based on Case 1 as explained in Section 3, where the moderator generates the session key to be used by the participants. Selecting

a secure authentication protocol can be a daunting task, as we next explain for the cases of the Woo-Lam and the Needham-Schroeder protocols. The original Woo-Lam protocol [26] contains several security flaws. A series of fixes for the Woo-Lam protocol proposed by Woo and Lam [27] were also flawed, since they suffer from reflection attack. An early attack on this protocol was discovered by Abadi and Needham [1], where they illustrate a parallel session attack and suggest a fixed version of the protocol. Later on, Clark and Jacob [9] showed that the fixed version is also vulnerable against a reflection attack. On the other hand, Denning and Sacco [10] and, later on, Lowe [16] discovered an attack on the Needham-Schroeder public-key authentication protocol [21]. As explained by Mao [17], the fixed version is also insecure. Mao and Boyd [18] provide a fixed version of the Woo-Lam and Needham-Schroeder protocols that is based on the specification of authentication protocols described in their paper.
We illustrate our protocol for 3 users in order to make it easier to read, bearing in mind that it can be extended naturally to include any number of participants. Our solution is based on the refined version of the Needham-Schroeder protocol given by Abadi and Needham [1].
Recall that the mediator is a trusted party, as explained in Section 3.
The notation used here is given below.

- The participants are denoted by the capital letters A, B, C; and the arbiter by T.
- PU_A (resp. PR_A) denotes the public (resp. private) key of A.
- $Cert_A$ is Alice's public-key certificate (it contains the identity and the public key of A among other information, all of which are signed by the arbiter T).
- K is the session key.
- sig_A is the signature of A.
- T_A is a timestamp issued by A.

The below authentication protocol is almost the same as
Authentication and key distribution

1. $A \rightarrow T : ID_A, ID_B, ID_C$
2. $T \rightarrow A : Cert_A, Cert_B, Cert_C$
3. $A \rightarrow B :$
$Cert_A, Cert_B, E_{PU_B}(sig_A(ID_A, ID_B, ID_C, K, T_A))$
$A \rightarrow C :$
$Cert_A, Cert_C, E_{PU_C}(sig_A(ID_A, ID_C, ID_B, K, T_A))$

We next comment the protocol's steps.
In the first step, the mediator (who is also a trusted party) informs the arbiter of the identities of all the participants.
In step 2, T sends to A the certificates of the participants. A retrieves from them the current public keys of the participants (which are signed by T).
In step 3, A sends to each participant their certificates, together with an encrypted message using the participant's public key. The message, which is signed by A, contains the identities of all the participants, the session key K to be used by them, and a timestamp issued by A.
After completing this protocol, each participant is aware of the others' identities and possesses the session key K.

Now each time a participant wishes to send a message, he/she has to send it first to the mediator, who then signs it and sends it to all the others.
For example, if B wants to *say* something, we could have:

1. $B \rightarrow A : ID_B, E_K(M, ID_B)$
2. $A \rightarrow C : E_K(E_{PR_A}(M, ID_B))$

But the problem here is that A cannot be sure that B wrote M, since all the participants possess the key K and someone could have impersonated A. Furthermore, step 2 is very costly since the whole message is encrypted using E_{PR_A}.
We can overcome this problem by combining hash functions with public-key cryptography, since the hash of a message is of small size.

1. $B \rightarrow A : ID_B, T_1, E_K(M, \mathbf{sig_B}(\mathbf{hash}(\mathbf{M}), \mathbf{T_1}))$
2. $A \rightarrow C : T_2, E_K(M, \mathbf{sig_A}(\mathbf{hash}(\mathbf{M}), \mathbf{ID_B}, \mathbf{T_2}))$

In step 1, A decrypts the second part of the message to ensure that the message has not been modified, and that it has been sent by B. An outsider cannot read its content because it has been encrypted with K. The timestamp $T1$ protects against replay attack: The moderator decrypts the message and verifies that the decrypted value of the timestamp matches the one sent in clear text.
In step 2, A sends the message to all the participants. Upon receipt of this message, C verifies that it is not a replay one and that B wrote it. In order to save encryption/decryption time, A signs with his/her private key the hash of the message, the ID of B, and the timestamp T_2.
Note that the last two steps can be repeated as many times as needed. Once the meeting is over, the mediator sends a special message to all the participants to announce it. For example A may send

$$T, End, E_K(sig_A(End, T))$$

to all the participants. The timestamp T is necessary to thwart replay attacks. In this case an attacker may replay the above message (if it does not contain T) to falsely announce the end of the meeting.

6 Conclusion and future work

In this paper we formulated and studied a new problem where several people wish to meet securely over an insecure channel. After a quick overview of some authentication protocols, we listed the requirements and the limitations of any possible solution to our problem. In order to be able to solve our problem, we saw for example that the moderator (i.e. the person who calls for the meeting in the real world) must be a trusted entity by the others, and that any cryptographic protocol that does not use public-key cryptography cannot solve it. Then we suggested several designs of cryptographic protocols based on the presence or absence of an arbiter. Finally, we proposed in Section 5 a solution to our problem that is partially based on a modified version of the Needham-Schroeder protocol.
There is still a lot of work to be done, at least in two directions.
(1) Implement the protocol given Section 5. But prior to this step we need to prove that our proposed solution is not vulnerable to known attacks such as replay, man-in-the-middle, reflection, or parallel-session.
(2) Design a generalized version of the protocol where participants can join/leave a meeting. This would capture the real case where people may arrive late to a meeting, or when some of the participants leave the meeting before it ends.

References

[1] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering, vol 22, n. 1*, pages 6–15, 1996.

[2] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key-exchange protocols. *Proc. 30th Symp. on Theory of Computing (STOC), ACM*, pages 419–428, 1998.

[3] M. Bellare and P. Rogaway. Entity authentication and key distribution. *Advances in Cryptology - Proceedings of CRYPTO 93, Lecture Notes in Computer Science*, pages 232–249, 1994.

[4] M. Bellare and P. Rogaway. Provably secure session key distribution - the three party case. *In Proc. of the 27th ACM Symposium on the Theory of Computing (STOC), ACM*, pages 57–66, 1995.

[5] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology, Vol. 13, No. 1,*, pages 143–202, 2000.

[6] R. Canetti. Security and composition of cryptographic protocols: A tutorial. *SIGCAT News*, pages 67–92, 2006.

[7] D. Chaum. The dining cryptographer problem: Unconditional sender and receiver untraceability. *Journal of Cryptology, v1, n.1*, pages 65–75, 1988.

[8] D. Chaum, C. Crépeau, and I. D. rd. Multi-party unconditionally secure protocols. *In Proc. 20th Symposium on the Theory of Computing (STOC)*, pages 11–19, 1988.

[9] J. Clark and J. Jacob. A survey of authentication protocol literature: version 1.0. *Online document*, 1997.

[10] D. Denning and G. Sacco. Timestamps in key distribution protocols. *Communications of the ACM, v. 24, n. 8*, pages 533–536, 1981.

[11] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644–654, 1976.

[12] L. Giordano and A. Martelli. Verifying agents' conformence with multiparty protocols. *Springer-Verlag Berlin Heidelberg*, pages 17–36, 2009.

[13] O. Goldreich. Cryptography and cryptographic protocols. *Distributed Computing*, pages 177–199, 2003.

[14] V. Goyal, P. Moassel, and A. Smith. Efficient two party and multi party computation against covert adversaries. *Eurocrypt*, pages 289–306, 2008.

[15] G. Kol and M. Naor. Cryptography and game theory: Designing protocols for exchanging information. *International Association for Cryptographic Research*, pages 320–339, 2008.

[16] G. Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, pages 131–133, 1995.

[17] W. Mao. Modern cryptography: Theory and practice. *Prentic Hall, 1st edition*, 2003.

[18] W. Mao and C. Boyd. Methodical use of cryptographic transformations in authentication protocols. *IEEE Proceedings, Comput. Digit. Tech.*, pages 272–278, 1995.

[19] U. Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, pages 370–381, 2006.

[20] C. Meadows. Formal verification of cryptographic protocols: A survey. *Advances in Cryptology, ASIACRYPT, Proceedings Springer-Verlag*, pages 133–150, 1995.

[21] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM, v. 21, n.12*, pages 993–999, 1978.

[22] B. C. Neuman and S. Stubblebine. A note on the use of timestamps as nonces. *Operating Systems Reviews, v. 27, n.2*, pages 10–14, 1993.

[23] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review, v. 21, n.1*, pages 8–10, 1987.

[24] R. Rubin and P. Honeyman. Formal methods for the analysis of authentication protocols. *Draft manuscript*, 1994.

[25] B. Schneier. Applied cryptography: Protocols, algorithms, and source code in c. *Wiley, 2nd Edition*, 1994.

[26] T. Woo and S. Lam. Authentication for distributed systems. *Computers*, pages 39–52, 1992.

[27] T. Woo and S. Lam. A lesson on authentication protocol design. *Operating systems Reviews*, pages 24–37, 1994.

[28] A. C. Yao. Potocols for secure computations. *In Proc. 23rd IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 160–164, 1982.