# Generalized Meeting Businessmen Problem

Khalil Challita *Member, IAENG*

*Abstract*—**In this paper we address the problem where a distributed group of people wish to meet securely over the Internet. Obviously, basic encryption techniques fail to achieve this aim since they involve the encoding or decoding of messages between two parties only. For this purpose, we propose a cryptographic protocol that allows any number of people to meet remotely while keeping their discussions secure from any potential eavesdropper. Our solution uses a trusted third party and is based on a refined version of the original Otway-Rees protocol. Prior to the establishment of a secure communication channel, we start by authenticating all the users involved. We also provide a formal proof of the correctness of the suggested protocol.**

*Index Terms*—**Public-key cryptography; Cryptographic protocols; Trusted party; Authentication protocols; Secure multi-party computation.**

## I. Introduction

EXCHANGING confidential messages has long been of interest to people working in different fields. Historically speaking, they resorted to classical encryption techniques where two persons required to share a secret symmetric key, or possibly used steganographic techniques that helped them hide the existence of the message itself. Today's needs have obviously changed since the design of the first basic ciphers. Most military, governmental and commercial organizations rely heavily on the Internet to communicate and to exchange sensitive information. The invention of pubic-key cryptography in 1976 by Diffie and Hellmann [17], considered by many to be the most important invention of the whole history of cryptography, allowed researchers to elaborate very sophisticated and original protocols that pervade our everyday life. Simple activities such as email checking or online monetary transactions involve the implicit use of cryptographic protocols. Schneier [33] suggested to classify protocols based on their properties. We distinguish from his work four major categories: basic, intermediate, advanced, and esoteric protocols.

For example, and to cite just a couple of non traditional protocols, one may have a look at the Dining Cryptographers' Problem and the Millionaires' Problem. Both problems are impossible to solve using symmetric-key cryptography.

In the original version of the dining cryptographers' problem [12], three cryptographers have a dinner. Afterwards, one of them or the National Security Agency (NSA) pays for the dinner. In the former case, Chaum's protocol uses secure multi-party communication to keep secret the identity of the cryptographer who paid for it (in other words, the other two cryptographers are unable to tell who paid for the dinner); whereas in the latter case they can all agree to the fact that the NSA made the payment. Another interesting example was given by Yao [38]. He designed a protocol that allows two rich persons to tell which one is wealthier than the other without revealing the amount of money they possess. For example, if person A has 15 millions dollars and person B has 21 million dollars, then at the end of the protocol they both know that B is richer than A, but no clue whatsoever is given about their respective fortune.

It is a well-known fact how hard it is to design a secure cryptographic protocol [26]. Mao gives several examples where even experts designed flawed protocols that were used for several years. Some famous examples include the Needham Schroeder protocol [29] and the Woo-Lam protocol [36], [37]. Hence the need of formal methods to design better protocol in the sense that such protocols must be proven secure against specific types of attacks [1], [28], [32], [3], [4], [5], [39], [7], [8], [43]. Other researchers analyzed the security of protocols. One may cite Canetti [10], [11], [41], [35] and Goldreich [19]. Recently, more protocols were designed to address issues related to classical key authentication problems [42] as well as to new esoteric protocols such as the bitcoin backbone protocol [40].

In this paper we address the problem where several persons wish to meet remotely and securely over the Internet. We extend the work done in [13] where we suggested a protocol based on a modified version of the Needham-Schroeder, and that involved only three participants. Therefore we propose a completely new protocol based on Otway-Rees' one this time. Our protocol starts by authenticating the users before encrypting the exchanged messages. Note that our solution is scalable and can include any number of participants.

This paper is divided as follows. We start in Section II by presenting some relevant work related to secure multi-party computation. In Section III, we list the properties a cryptographic protocol must possess in order to enable any number of persons to meet securely over the Internet. Section IV focuses on approaches that involve an arbiter to solve our problem, whereas we propose in Section V solutions without any arbiter. We suggest in Section VI the generalized meeting businessmen protocol that solves our problem, before proving its correctness in Section VII.

## II. Related work

As it was first described by Yao [38], secure multiparty computation (SMPC) allows a set $P = \{p_1, p_2, \cdots, p_n\}$ of $n$ players to compute an $n$-party function $f = f(f_1, f_2, \dots, f_n)$ of their inputs without revealing any additional information to corrupted players other than the output of the function itself. At the end of the protocol, we have the following guarantees: the output is correct and each player's input is kept secret from the others. We will use in Section VI some of the techniques described here as part of a cryptographic protocol to allow any number of persons (who wish to meet) to participate in the generation of a symmetric key $K$.

Manuscript received September 28, 2018; revised January 18, 2019.

K. Challita is with Notre Dame University - Louaize, Computer Science Department, e-mail: kchallita@ndu.edu.lb.

Some research related to SMPC can be found in [22], [14], [18]. Maurer [27] presented a simple approach to SMPC that is not based on advanced sub-protocols (such as zero knowledge proofs). He designed protocols that are secure for mixed (i.e. active and passive) corruption. In a passive corruption scheme, a player is aware of the whole internal information of the corrupted player but does not deviate from the protocol, whereas in an active scenario an adversary $A$ can take full control of a player and make him perform illegal actions. We assume here that $A$ is able to control $t < n$ players. We shall denote by $t$ the maximum number of players an adversary is able to corrupt. In the worst-case scenario we have $t = n - 1$, where all but one player are corrupted. Goldreich *et al.* [20] proved that security can be achieved if $t < n/2$ for passively corrupted players. They also showed that security holds for passive corruption of players in case we have $t < n$. Later on, Ben-Or *et al.* [6] established the fact that SMPC is possible if and only if $t < n/2$ are passively corrupted, and if and only if $t < n/3$ are actively corrupted. Another interesting result was proposed by Goyal *et al.* [21], where they designed a SMPC protocol that remained secure even in the case when more than half of the players were dishonest. Instead of relying on costly zero-knowledge proofs techniques, their approach uses cut-and-choose techniques. It achieves security in the case where any number of players collude together. They extended a two-party protocol to a multi-party case where the protocol they suggested is secure against covert adversaries. Their solution uses several sub-protocols (such as *simultaneous commitment* and *open commitment*) and requires $O(n^3ts|C|)$ bits of communication. Note that $s$ denotes the statistical security parameter, $|C|$ the size of the circuit, and $\epsilon = 1 - 1/t$ the deterrence probability.
We next turn our attention to selecting an appropriate authentication protocol to be part of our solution in Section VI. The Otway-Rees [31] protocol helps two users achieve mutual authentication and allows them to share a secret symmetric key. Below is a variation of the protocol as given in [25]. Later on, Backes [2] established the cryptographic security of this protocol.

**Modified version of the Otway-Rees protocol**

1) Alice $\rightarrow$ Bob : $M$, $ID_A$, $ID_B$, $N_A$
2) Bob $\rightarrow$ T : $M$, $ID_A$, $ID_B$, $N_A$, $N_B$
3) T $\rightarrow$ Bob : $M$, $\{K_{AB}\}_{K_{AT}}$, $\{K_{AB}\}_{K_{BT}}$, $[N_A, ID_B, \{K_{AB}\}_{K_{AT}}]_{K_{AT}}$, $[N_B, ID_A, \{K_{AB}\}_{K_{BT}}]_{K_{BT}}$
4) Bob $\rightarrow$ Alice : $M$, $\{K_{AB}\}_{K_{AT}}$, $[N_A, ID_B, \{K_{AB}\}_{K_{AT}}]_{K_{AT}}$

The original version of the Woo-Lam protocol [36], [37] and their fixes were shown to be vulnerable against a wide range of attacks. Thus we consider a refined specification of the protocol [26] that withstands all known attacks against the original version, such as reflection and parallel-session attacks.
The below protocol allows Alice to authenticate herself to Bob, assuming that both Alice and Bob share secret symmetric keys with a trusted entity $T$, denoted by $K_{AT}$ and $K_{BT}$, respectively.

**Refined version of the Woo-Lam protocol**

1) Alice $\rightarrow$ Bob: $ID_A$
2) Bob $\rightarrow$ Alice: $N_B$
3) Alice $\rightarrow$ Bob: $[N_B]_{K_{AT}}$
4) Bob $\rightarrow$ T: $[ID_A, N_B, [N_B]_{K_{AT}}]_{K_{BT}}$
5) T $\rightarrow$ Bob: $[N_B]_{K_{BT}}$
6) Bob accepts if the integrity $[N_B]_{K_{BT}}$ verification returns true.

One of the first public-key authentication protocols was given by Needham and Schroeder [29]. We present below a refined version of their protocol that is not vulnerable against Lowe's attack [23]. In this secure version, Alice and Bob exchange messages by encrypting them using the other party's public key, denoted by $PU_A$ and $PU_B$, respectively.

**Refined version of the Needham-Schroeder public-key authentication protocol**

1) Alice $\rightarrow$ Bob: $[\{N_A\}_{PU_B}, ID_A]_{PU_A}$
2) Bob $\rightarrow$ Alice: $[\{N_A, N_B\}_{PU_A}]_{PU_B}$
3) Alice $\rightarrow$ Bob: $[\{N_B\}_{PU_B}]_{PU_A}$

Another relevant protocol to our problem was introduced by Bellare and Rogaway [4], where they provided a formal proof of its security. The MAP1 protocol allows an arbitrary number of players to mutually authenticate themselves. We also assume here that the players share a secret symmetric key.

**MAP1 protocol**

1) Alice $\rightarrow$ Bob: $ID_A$, $N_A$
2) Bob $\rightarrow$ Alice: $[ID_A, ID_B, N_A, N_B]_{K_{AB}}$
3) Alice $\rightarrow$ Bob: $[ID_A, N_B]_{K_{AB}}$

We next specify the requirements a protocol must possess in order to allow any number of persons to meet securely.

## III. Requirements and limitations

Our aim in this section is to specify the requirements a cryptographic protocol must have in order to allow any number of persons to meet remotely and securely. A straightforward approach is to draw the parallel with what happens during an actual meeting where several persons gather to discuss private issues.
It is worth noting that any protocol that solves this problem will have the same limitations to the ones that may happen in an actual (i.e. physical) meeting. For example, we cannot guarantee the fact that two of the persons who are in a meeting do not cheat by using another communication channel (e.g. using a phone), or simply by whispering if they are sitting next to each other. Therefore, we will list later on the assumptions under which this problem has a solution.
We next define some words that will be of help for us throughout the remaining of this paper.

*Definition 1:* (Basic terminology)

1) A **ring of trusted people** are trusted persons whose role is to sign and validate any exchanged message between the participants.
2) The persons who take part in a meeting are called the **participants**.
3) Any person that is not supposed to be part of the meeting is called a **stranger**.

Let us assume that several persons are physically meeting together. 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 supposed to hear what is being discussed.

Hence the following four requirements a protocol must satisfy:

1: Every person must be aware of the identities of all the other participants in the meeting.
2: A message sent by a participant is revealed to all the other ones.
3: No stranger can eavesdrop on their conversation.
4: The identity of the person who sent a message is clear to all the participants.

The first requirement is about authentication. The second deals with the delivery of a message. The third and fourth messages enforce confidentiality and proof of origins, respectively.

Other events/violations may happen during a meeting, such as:

1) A participant leaves the meeting earlier than the others.
2) A participant arrives late to the meeting.
3) A participant leaves the meeting temporarily.
4) (Collusion) A number of participants do not follow the rules and try to cheat by hiding some messages from the others.

As we already said in the introduction, basic encryption schemes cannot solve this problem since they are concerned with securing the communications between just two parties. Therefore the need of a specific protocol to address this issue. But first we enumerate the main limitations inherent to any cryptographic protocol that intends to solve our problem.
Let us assume that $n$ participants need to have a secret meeting. For this purpose they must share one common symmetric key, denoted by $K_n$. Before designing a protocol that solves our problem, one must take into account the following scenarios:

1) During a meeting, a participant could reveal sensitive information to an outsider by using another medium than the protocol itself. For example by using a phone, or even by sending him/her messages using another secret key than $K_n$.
2) Two or more participants may collude together and hide information from the others.

Nothing can be done to protect against the first point. Since the participants are not physically present in the same room, no one can monitor what the others are doing. Note that the same may happen during a physical meeting where a person is capable of revealing part of the conversation to a stranger. Therefore, any cryptographic algorithm that solves our problem will not take into account this case.
On the other hand, we will see later on that we are able to deal with the second point by using a well defined signature scheme.
Based on this brief analysis, the secure multiparty meeting problem has a solution under the following assumption:

*Assuming that the ring of trusted people is composed of $m$ persons, then there are at least $0 < m' \leq m$ honest participants.*

In other words, the group of $m'$ honest persons are trusted to abide by the protocol. This assumption is true in many real-life scenarios where some people meet with the intention to keep their conversation secret.
On the other hand, there could be $t \geq 0$ persons that try to cheat by forging a message for example, or by hiding a wire or a camera to disclose part of the conversation with an outsider.
We next give two obvious remarks.

*Remark 1:* The above assumption does not forbid two or more participants from colluding together.

We must be able to answer the following question: What is the maximum number of participants that may collude together while preserving the correctness of the protocol?
In other words, we must determine some $t < n$ where we assume that $t$ participants try to cheat without violating the correctness of the protocol (i.e. they will be detected).

*Remark 2:* Any solution to our 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 this problem, taking into account the previously mentioned assumption.
Obviously, we must start by authenticating the participants of a meeting.
We will consider protocols that involve a ring of trusted people that contain at least three persons. If only two persons are involved, then they can use a traditional encryption scheme to communicate together.

## IV. Protocols with an arbiter

Recall that any 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 the meeting businessmen 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 and distribute it to the participants. The ring of trusted people must take part in this process. Note that the arbiter may or may not be involved here (see Figure 1 for example).

Step 3. Before being sent, a message must be approved (e.g. signed) by all or part of the ring of trusted people.

Step 4. For each message that is sent, verify that all the participants received and read it.

*Remark 3:* (It follows from Step 3) Every message must be signed by the ring of trusted people (or part of it), before being considered as valid.

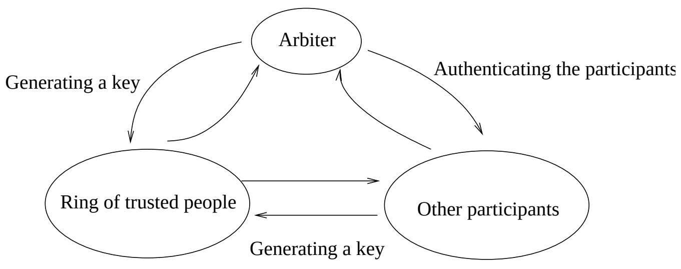

Fig. 1. The Arbiter takes part in generating and exchanging a secret key.

Steps 1 and 2 are may be repeated more than once, in case a person arrives late to a meeting or leaves it temporarily. In such a case authentication may be required again, and we could generate and distribute new secret key. Note that in Step 2 we have the option to allow the arbiter to read the content of the exchanged messages or not. The former case applies when the arbiter participates in the generation of the session key and distribute it to all the participants. Steps 3 and 4 are the most costly since they need to be repeated each time a new message is exchanged among the participants. To reduce the amount of work to be done here, we may use a probabilistic algorithm that randomly chooses a number of participants that belong to the ring of trusted people to approve/verify a message. For example, if the size of the ring is 10, our algorithm may choose randomly 4 persons to verify a given message.

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 public-key authentication cryptosystem such as the refined version of the Needham-Schroeder protocol as described in Section II (or a secure version of the Denning-Sacco protocol [16]) to authenticate the users and to help them generate a session key. This step is performed at the beginning of the meeting and should be repeated each time the number of participants changes (e.g. someone joins the meeting later on, or temporarily leaves, or simply leaves the meeting earlier).
2) When a participant wishes to send a message, he/she has to send it to the ring of trusted people (or part of it) who signs it and then sends it to the other participants.
3) The meeting ends upon the request of the mediator (person who called for the meeting) by sending a special request to the ring of trusted people to signal the end of the meeting.

Note that in this case we must try to forbid a group of $t$ participants to collude together by exchanging a message that is not valid. We should address the case where part of these persons are in the ring of trusted people.

This issue must be resolved empirically. We have a trade-off between the amount of work to be done and the correctness of the protocol in case we have collusion between a number of participants. We next deal with the general case.

Assume that we have a number $t$ of people who try to cheat. Three cases are to be consider:

1) All of them belong to the ring of trusted people (worst case).
2) All of them are outside the ring of trusted people (best case).
3) Some of them belong to the ring of trusted people while others don't.

The best case scenario does not affect the correctness of the protocol since the messages must be validated by the ring of trusted people, and all of the participants who try to cheat are outside of it. In the worst case scenario a violation of the protocol may happen. Assume that the ring of trusted people contain $m$ persons and that we use some random algorithm to select $k \leq m$ participants from the ring to validate a message. Then it is impossible for us to stop these $t$ persons from cheating if, for example, they decide to: generate a new message, or falsify an existing message, or even sign an old message and then send it as being *valid* to the remaining participants.

To prevent this from happening, we need to have at least one honest person who is part of the ring of trusted people. We can enforce this property as described below. Assume that the maximum number of persons who try to collude is $t$, then we must add to this group at least one honest user. This can be reflected in the following remark.

*Remark 4:* A probabilistic algorithm must randomly select at least $t + 1$ persons from the ring of trusted people to sing/verify a message.

If this condition is not satisfied, then a violation of the protocol cannot be avoided.

Our protocol shall include a modified version of any currently available authentication protocol that is concerned with authenticating only two users. Also note that the initial versions of several protocols were flawed (see for example [15]). A more detailed explanation of the security of these protocols is provided in Section V.

One might replace Case 1 (1.) by any basic authentication protocol using an arbiter, and then let the participants (or just the ring of trusted people) 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), a representative of the ring of trusted people 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 this representative (in this case he/she must be a trusted entity). Refer to Figure 2 for an illustration of Case 1 above.

Trying to reduce the amount of work to be done 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 representative of the ring of trusted people challenges some randomly selected participants to check whether or not they received the message.

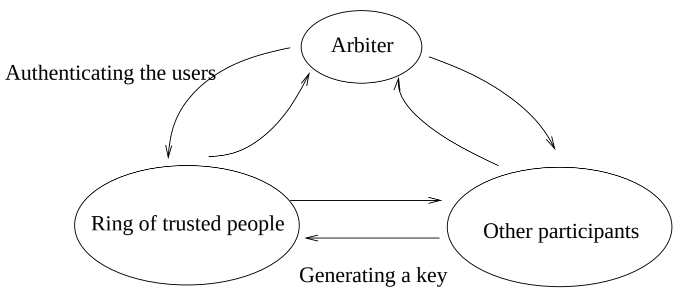

Fig. 2. The Arbiter only takes part in authenticating the participants.

Then two persons could cheat by sending each other a message without sharing it with the remaining participants. In this case the representative 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 [31], the woo-lam protocol as described in Section II, (or the Neuman-Stubblebine protocol [30], [34]) 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.
We next deal with the case of designing a protocol without the arbiter's involvement in the generation of a secret key.

## V. Protocols without an arbiter

We start here by discussing Case 3 where we assume that the representative of the ring shares a secret master key with all the other participants. The moderator belongs to the ring of trusted people and plays a central role by validating messages. See Figure 3 for example, where we have a peer-to-peer scheme for exchanging messages that are validated by the moderator.

**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 such as the refined version of the Needham-Schroeder protocol (see Section II).
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 ring of trusted people (or to a random number $k'$ of persons of the ring) who sign it and then send 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.

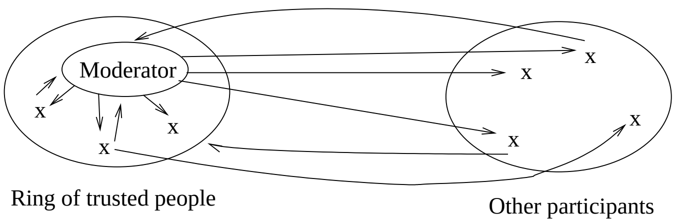

Fig. 3. The moderator is responsible for validating all the messages.

This protocol assumes that the participants can verify the signature of the trusted people who sign the message. They can use secure multi-party computation (SMPC) as introduced by Yao [38] to generate a session key because there is no arbiter.
Secure multi-party computation allows a group 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.

The fourth case described in Section III 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 participants authenticate themselves using a sound authentication protocol, such as the MAP1 protocol as given in Section II.
2) The ring of trusted people generate a session key. The key is then sent to all the participants.
3) Each participant sends his/her message to the others.
4) The meeting ends upon the request of the mediator.

In this case the participants cannot start the meeting without generating a secret key. All they can do is to rely on the ring (or part of it) to generate and distribute a symmetric key. The representative of the ring could play the role of an arbiter here. In Step 2 above, sending the message to the representative in order to forward it adds no security to the protocol. Indeed, a participant may cheat by sending a message to a number $k < n$ of participants using the session key generated and distributed by the representative. 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 (i.e. authenticate the person who initially wrote the message). Upon the receipt of a message, no one can be sure about the identity of the sender.
Based on our analysis in this section and in Section III, we can certify that no practical solution to our problem exists without the use of public-key cryptography. Thus practical implementations of the secure multiparty protocol must be based on Case 1 or Case 3 described in Section III.

## VI. The generalized meeting businessmen protocol

In this section we propose a protocol that uses symmetric-key cryptography to authenticate the participants and a public-key method to exchange messages. We assume here that the ring of trusted people generates the session key $K$ to be used by the participants. We also assume that all the participants can verify the signature of the arbiter (i.e. trusted entity). Recall that selecting a secure authentication protocol can be a daunting task. We discuss below the Woo-Lam and the Needham-Schroeder protocols.
The original Woo-Lam protocol [36] contained several security flaws. Furthermore, it turned out that a series of fixes proposed by Woo and Lam [37] were also flawed: they were vulnerable against reflection attacks. An early attack on this protocol was discovered by Abadi and Needham [1], where they illustrated a parallel session attack and suggested a fixed version of the protocol. Later on, Clark and Jacob [15] showed that the fixed version was also vulnerable against a reflection attack. On the other hand, Denning and Sacco [16], and also Lowe [23] discovered an attack on the Needham-Schroeder public-key authentication protocol [29]. As explained by Mao [26], the fixed version was also insecure. Mao and Boyd [25] provided a fixed version of the Woo-Lam and Needham-Schroeder protocols that is based on a new specification of authentication protocols.
After considering several approaches, we found out that the refined version of the Otway-Rees protocol (as described by Mao and Boyd [25]) was best suited for solving our problem. The below protocol can be applicable to any number of users. Recall the following notations:

- The participants are denoted by the capital letters $P_i$, ($i \in S$) where $S$ is a finite set; and the arbiter by $T$.
- $PU_{P_i}$ (resp. $PR_{P_i}$) denotes the public (resp. private) key of $P_i$.
- $Cert_{P_i}$ is the public-key certificate of $P_i$ (it contains the identity and the public key of $P_i$ among other information, all of which are signed by the arbiter $T$).
- $K$ is the session key.
- $sig_{P_i}$ is the signature of $P_i$.
- $T_{P_i}$ is a timestamp issued by $P_i$.
- $K_{P_iT}$ is the symmetric key $P_i$ shares with the arbiter $T$.
- $N_{P_i}$ denotes a nonce generated by $P_i$.

We next assume that every trusted entity (i.e. person who belongs to the ring of trusted people) is responsible for authenticating a group of participants. Afterwards, all the authenticated users will be announced to the people who are meeting.
Note that each time a participant wishes to leave or join the meeting a new session key must be generated by the ring of trusted people.

**The generalized meeting businessmen protocol**

1) $P_i \to P_j : M, P_i, P_j, N_{P_i}, [M, P_i, P_j, N_{P_i}]_{K_{P_iT}}$
2) $P_j \to T : M, P_i, P_j, N_{P_i}, N_{P_j}, [M, P_i, P_j, N_{P_i}]_{K_{P_iT}}, [M, P_i, P_j, N_{P_j}]_{K_{P_jT}}$
3) $T \to P_j :$
$M, [K_{P_iP_j}]K_{P_iT},$
$[M, N_{P_i}, [K_{P_iP_j}]K_{P_iT}]K_{P_iT},$
$[K_{P_iP_j}]K_{P_jT},$
$[M, N_{P_j}, [K_{P_iP_j}]K_{P_jT}]K_{P_jT},$
4) $P_j \to P_i :$
$M, [K_{P_iP_j}]K_{P_iT},$
$[M, N_{P_i}, [K_{P_iP_j}]K_{P_iT}]K_{P_iT}$

The aim after this protocol is to authenticate all the people and to exchange symmetric keys amongst users.
We kept the notation as in the original version of the protocol, but in this case we could replace $K_{P_iP_j}$ by $K$, since the latter is the session key to be used by all the participants. The authentication and key distribution protocol must be executed each time the number of persons who meet changes (e.g. someone leaves or joins the meeting).
In the first step user $i$ sends to user $j$ a message $M$, their identities, a nonce, plus the same information encrypted using the symmetric key user $i$ shares with the trusted entity $T$. In step two, and after receiving a message from user $i$, user $j$ sends the same information to the arbiter $T$ plus identical information generated by $j$, encrypted using the secret symmetric key $j$ shares with the arbiter. In the following steps the arbiter sends the symmetric key $K_{P_iP_j}$ to both users $i$ and $j$ to be used in order to encrypt all the messages they wish to exchange.
After completing this protocol, each participant is aware of the others' identities and possesses the session key $K$.
A graphical representation of the generalized meeting businessmen protocol (GMBP) is given in Figure 4.

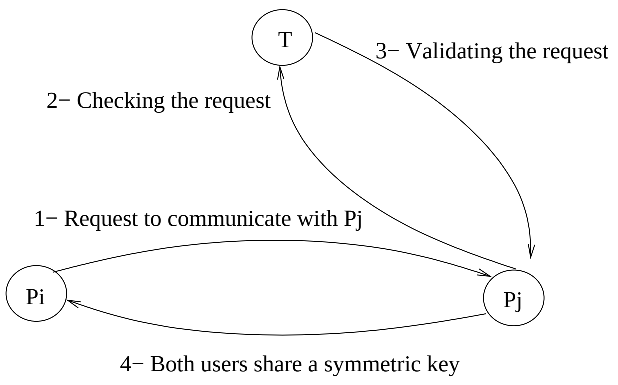
T
3− Validating the request
2− Checking the request
1− Request to communicate with Pj
Pi
Pj
4− Both users share a symmetric key

Fig. 4. The GMBP: authentication, key distribution, and message exchange.

Now each time a participant wishes to send a message, he/she has to send it first to the arbiter, who then signs it and sends it to the others.
For example, if $P_i$ wants to send a message to all the participants, we may have:

1) $P_i \to T : ID_T, E_K(M, ID_T)$
2) $T \to P_k : E_K(E_{PR_T}(M, ID_{P_i}))$

Here $k \in S$, which means that the message is sent to all the participants. But the problem with this approach is that a person $P_k$ cannot be sure that $P_i$ wrote $M$, since all the participants possess the key $K$ and someone could have impersonated $P_i$. Furthermore, step two is very costly since the whole message is encrypted using the private key of the arbiter $E_{PR_T}$.
We can overcome this problem by combining hash functions with public-key cryptography as shown below.

1) $P_i \to T : ID_{P_i}, T_{P_i}, E_K(M, \mathbf{sig_{P_i}}(\mathbf{hash}(\mathbf{M}), \mathbf{T_{P_i}}))$

2) $T \rightarrow P_k : T_T, E_K(M, \mathbf{sig_T}(\mathbf{hash(M)}, \mathbf{ID_{P_i}}, \mathbf{T_T}))$

Depending on the implementation of the protocol, the arbiter $T$ may be replaced with a person from the ring of trusted people (this would relieve the arbiter from signing and verifying all the messages). In step 1, the arbiter decrypts the second part of the message to ensure that the message has not been modified, and that it has been sent by user $P_i$; therefore providing a proof of origin and preventing illegal tampering with the message.
An attacker is unable to read its content because it has been encrypted with $K$. The timestamp $T_{P_i}$ protects against replay attack: The arbiter decrypts the message and verifies that the decrypted value of the timestamp matches the one sent in clear text.
In step 2 above, $T$ sends the message to all the participants. Upon receipt of this message, any user can verify that it is not an old message and that $P_i$ is the one who originally wrote it. In order to save encryption/decryption time, $T$ signs the hash of the message with his/her private key , the ID of $P_i$, and the timestamp $T_T$.
Note that the last two steps can be repeated as many times as needed.
Once the meeting is over, the representative of the ring of trusted people (or the moderator) sends a special message to all the participants to announce the end of the meeting. For example $T$ may send

$$T, End, E_K(sig_T(End, T))$$

to all the participants. The timestamp $T$ is necessary to thwart replay attacks. Otherwise, an attacker may replay an old message to falsely announce the end of the meeting.

## VII. Correctness of the GMBP

We use the BAN logic [9] to prove formally the soundness of the GMBP protocol. This predicate logic can be interpreted for users (i.e. participants), messages, keys, and statements. We start with a quick overview of this logic and see how to apply it to our case.
$P \mid\equiv X$ : Participant $P$ considers the statement $X$ as true.
$P \mid\sim X$ : Participant $P$ once said $X$.
$P \triangleleft X$ : Participant $P$ has received the statement $X$.
$\sharp X$ : The statement $X$ is new. No participants received it previously.
$P \rightarrow X$ : Participant $P$ should be trusted over the truth of the statement $X$.
$P \overset{K}{\leftrightarrow} Q$ : Participants $P$ and $Q$ use the secret key $K$ to communicate. No one is capable of eavesdropping on their conversation.
Below are some inference rules relevant to our case. These rules allow us to reason about the predicates defined above.

**The message meaning rule**

$$\frac{P\mid\equiv (Q \overset{K}{\leftrightarrow} P), \quad P \triangleleft \{X\}_K}{P\mid\equiv (Q\mid\sim X)} \tag{1}$$

This rule says that if participants $P$ and $Q$ share the same secret key $K$, and if $P$ receives a message $X$ encrypted using $K$, then $P$ is sure that $Q$ is the author of $X$.

**The freshness rule**

$$\frac{P\mid\equiv \sharp(X)}{P\mid\equiv \sharp(X, Y)} \tag{2}$$

If $P$ believes that the statement $X$ is fresh (i.e. has not been sent previously), then $P$ also believes that the combined statements $X$ and $Y$ are fresh too. This rule is useful when, for instance, $P$ receives a message $X$ and a key $K$. Then he/she is sure that the message and key were not sent before.

**The nonce verification rule**

$$\frac{P\mid\equiv \sharp(X), \quad P\mid\equiv (Q\mid\sim X)}{P\mid\equiv (Q\mid\equiv X)} \tag{3}$$

If $P$ believes that the statement $X$ is fresh and that $Q$ once said $X$, then $P$ is sure that $Q$ believes in $X$.

**The trusted party rule**

$$\frac{P\mid\equiv tp(Q), \quad P\mid\equiv (Q\mid\equiv X)}{P\mid\equiv X} \tag{4}$$

If $Q$ is a trusted party (in our case the arbiter) and $P$ knows that $Q$ believes in statement $X$, then $P$ also considers $X$ to be true. In other words, $Q$ must be trusted unconditionally by $P$.

**The session key rule**

$$\frac{P\mid\equiv (\{P, Q\} \triangleleft K), \quad P\mid\equiv \sharp(K)}{P\mid\equiv (P \overset{K}{\leftrightarrow} Q)} \tag{5}$$

If $P$ knows that both $P$ and $Q$ received the key $K$, and also that $K$ is fresh, then $P$ is sure that he/she shares $K$ with participant $Q$.

We next turn our attention to proving the soundness of the GMBP. We assume that the set of participants includes a trusted party denoted by $T$.

*Definition 2:* A protocol is sound if the following holds: after completion, all inferred formulas are regarded as true by the participants.

We next show that the symmetric key $K$ that is generated during the GMBP is fresh, and that it can be used securely by all the participants to encode their communications.

*Proposition 1:* The generalized meeting businessmen protocol is sound with respect to generating a session key.

*Proof:* Formally speaking, we have to show that the following holds for any participants $P_i$ and $P_j$:
$P_i\mid\equiv (P_j\mid\equiv P_i \overset{K}{\leftrightarrow} P_j)$ and $P_j\mid\equiv (P_i\mid\equiv P_i \overset{K}{\leftrightarrow} P_j)$.
In other words, both $P_i$ and $P_j$ believe that the key $K$ is safe once they start using it.
First we assume that the arbiter is trusted by every participant: $\forall i$, $P_i\mid\equiv tp(T)$.
Now we show that $\forall i, j$, $P_i\mid\equiv P_i \overset{K}{\leftrightarrow} P_j$.
Indeed, since any participant $P_i$ considers the following statements to be true: $T$ once approved $K$, and $T$ knows that he/she used $K$ with $P_i$, and also that $P_i$, $P_j$ and $T$ used $K$; then we can derive $P_i\mid\equiv P_i \overset{K}{\leftrightarrow} P_j$.
In notation we have:
$P_i\mid\equiv (T\mid\sim K)$, and
$P_i\mid\equiv (T\mid\equiv T \overset{K}{\leftrightarrow} P_i)$, and

$P_i|\equiv (T|\equiv \{T, P_j\} \triangleleft K)$, and
$P_i|\equiv (T|\equiv \{T, P_i\} \triangleleft K)$.
Added to this, we know that if $T$ once sent $K$ and $P_j$, then the trusted party $T$ transmitted $K$: $\dfrac{T|\sim (K, P_j)}{T|\sim K}$
Applying the freshness rule and the nonce verification rule, we conclude that $P_i|\equiv P_i \overset{K}{\leftrightarrow} P_j$.
Symmetrically speaking, we also have $P_j|\equiv P_i \overset{K}{\leftrightarrow} P_j$.
By extension, we conclude that $P_i|\equiv (P_j|\equiv P_i \overset{K}{\leftrightarrow} P_j)$ and $P_j|\equiv (P_i|\equiv P_i \overset{K}{\leftrightarrow} P_j)$; which concludes our proof. ■

Therefore the key generated by the GMBP can be used by all the participants to securely encode their communications.

## VIII. Conclusion

We tackled in this paper the problem of several persons who wish to meet securely over the Internet. Obviously, a standard protocol cannot solve this issue since basic encryption techniques involve only two parties. Before generating a session key, any approach must start by authenticating all the users. After a quick overview of some authentication protocols, we listed the requirements and the limitations of any potential solution to our problem. Moreover, after describing several possible designs based on the presence or absence of an arbiter, we suggested a cryptographic protocol that is based on a modified version of the Otway-Rees authentication protocol. A trusted entity is involved in generating a session key each time the protocol is run. We also used the BAN logic to prove the soundness of the GMBP.

## References

[1] Martin Abadi and Roger Needham, Prudent Engineering Practice for Cryptographic Protocols, *IEEE Transactions on Software Engineering*. **22** (1996) pp. 6–15.
[2] M. Backes, Real-or-random key secrecy of the Otway-Rees protocol via a symbolic security proof. *Electr. Notes Theoretical Computer Science*. **155** (2006) pp. 111-145.
[3] 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* (1995) pp. 57–66.
[4] M. Bellare and P. Rogaway, Entity authentication and key distribution, *Advances in Cryptology - Proceedings of CRYPTO 93, Lecture Notes in Computer Science*. (1994) pp. 232–249.
[5] M. Bellare and 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* (1998) pp. 419–428.
[6] M. Ben-Or, S. Goldwasser and A. Widgerson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, *Proc. 20th Symp. on Theory of Computing (STOC), ACM* (1988) pp. 1–10.
[7] J. Blackledge and O. Iakovenko, Resilient Digital Image Watermarking for Document Authentication, *IAENG, International Journal of Computer Science*, volume 41, issue 1, (2014) pp. 1–17.
[8] S. Boonkrong and C. Somboonpattanakit, Dynamic Salt Generation and Placement for Secure Password Storing, *IAENG, International Journal of Computer Science*, volume 43, issue 1, (2016) pp. 27–36.
[9] M. Burrows, M. Abadi, and R. Needham, A logic of authentication, *Technical report SRC* (1989).
[10] R. Canetti, Security and composition of multi-party cryptographic protocols, *Journal of Cryptology* (2000) pp. 143–200.
[11] R. Canetti, Security and Composition of Cryptographic Protocols: A tutorial, *SIGCAT News* , (2006), pp. 67–92.
[12] D. Chaum, The dining cryptographer problem: Unconditional sender and receiver untraceability, *Journal of Cryptology,* (1988), pp. 65–75.
[13] K. Challita, Protocols for the meeting businessmen problem *Annales UMCS Informatica* (2013) pp. 37–47.
[14] D. Chaum and C. Crépeau and I. Damgård, Multi-party unconditionally secure protocols, *In Proc. 20th Symposium on the Theory of Computing (STOC)*, (1988), pp. 11–19.
[15] J. Clark and J. Jacob, A survey of authentication protocol literature: version 1.0 (1997).
[16] D.E. Denning and G.M. Sacco, Timestamps in Key Distribution Protocols, *Communications of the ACM*, (1981), pp. 533–536.
[17] W. Diffie and M.E. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory*, (1976), pp. 644–654.
[18] Laura Giordano and Alberto Martelli, Verifying agents' conformence with multiparty protocols, *Springer-Verlag Berlin Heidelberg*, (2009), pp. 17–36.
[19] Oded Goldreich, Cryptography and cryptographic protocols, *Distributed Computing*, (2003), pp. 177–199.
[20] O. Goldreich, S. Micali and A. Widgerson, How to play a mental game - A completeness theorem for protocols with honest majority, *Proc. 19th Symp. on Theory of Computing (STOC), ACM* (1987) pp. 218–219.
[21] Vipul Goyal and Payman Moassel and Adam Smith, Efficient Two Party and Multi Party Computation Against Covert Adversaries, *Eurocrypt*, (2008), pp. 289–306.
[22] Gillat Kol and Moni Naor, Cryptography and game theory: Designing protocols for exchanging information, *International Association for Cryptographic Research*, (2008), pp. 320–339.
[23] G. Lowe, An attack on the Needham-Schroeder public-key authentication protocol, *Information Processing Letters* (2008), pp. 320–339.
[24] W. Mao and C. Boyd, Towards a Formal Analysis of Security Protocols, *Proceedings of the Computer Security Foundations Workshop VI* (1993), pp. 147–158.
[25] W. Mao and C. Boyd, Methodical use of cryptographic transformations in authentication protocols, *IEEE Proceedings, Comput. Digit. Tech.* (1995), pp. 272–278.
[26] Wenbo Mao, Modern cryptography: Theory and practice, *Prentic Hall, 1st edition*, (2003).
[27] Ueli Maurer, Secure Multi-Party Computation Made Simple, *Discrete Applied Mathematics* (2006), pp. 370–381.
[28] C.A. Meadows, Formal Verification of Cryptographic Protocols: A Survey, *Advances in Cryptology, ASIACRYPT, Proceedings Springer-Verlag*, (1995), pp. 133–150.
[29] R.M. Needham and M.D. Schroeder, Using Encryption for Authentication in Large Networks of Computers, *Communications of the ACM* (1978), pp. 993–999.
[30] B. C. Neuman and S. Stubblebine, A note on the use of timestamps as nonces, *Operating Systems Reviews* (1993), pp. 10–14.
[31] D. Otway and O. Rees, Efficient and Timely Mutual Authentication, *Operating Systems Review* (1987), pp. 8–10.
[32] R.D. Rubin and P. Honeyman, Formal Methods for the Analysis of Authentication Protocols, *Draft manuscript* (1994).
[33] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, *Wiley, 2nd Edition* (1994).
[34] B. C. Neuman and S. Stubblebine, A note on the use of timestamps as nonces, *Operating Systems Reviews* (1993), pp. 10–14.
[35] W. Simpson and K. Foltz, Ports and Protocols Extended Control for Security, *IAENG, International Journal of Computer Science*, volume 44, issue 2, (2017) pp. 227–240.
[36] T.Y.C. Woo and S.S. Lam, Authentication for Distributed Systems, *Computers*, (1992) pp. 39–52.
[37] T.Y.C. Woo and S.S. Lam, A lesson on authentication protocol design, *Operating systems Reviews*, (1992) pp. 24–37.
[38] A. C. Yao, Potocols for secure computations, *In Proc. 23rd IEEE Symposium on the Foundations of Computer Science (FOCS)*, (1982) pp. 160–164.
[39] Y. Lindell, Secure multiparty computation for privacy preserving data mining, *Journal of Privacy and Confidentiality*, (2009).
[40] J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, *Cryptology-EUROCRYPT*, (2015).
[41] R. Amin, G.P. Biswas, An Improved RSA Based User Authentication and Session Key Agreement Protocol Usable in TMIS, *Journal of Medical Systems*, (2015).
[42] C. Boyd, A. Mathuria, Protocols for Authentication and Key Establishement, *Springer*, (2013).
[43] M. Younes and A. Jantan, Image Encryption using Block-Based Transformation Algorithm, *IAENG, International Journal of Computer Science*, volume 35, issue 1, (2008), pp 15–23.