2011 International Conference on Broadband and Wireless Computing, Communication and Applications

# Building Trust for $\lambda$-Congenial Secret Groups

Di Ma
University Of Michigan-Dearborn
dmadma@umd.umich.edu

Claudio Soriente
Universidad Politécnica de Madrid
csoriente@fi.upm.es

***Abstract*—Establishing trust while preserving privacy is a challenging research problem. In this paper we introduce $\lambda$-congenial secret groups which allow users to recognize trusted partners based on common attributes while preserving their anonymity and privacy. Such protocols are different from authentication protocols, since the latter are based on identities, while the former are based on attributes. Introducing attributes in trust establishment allows a greater flexibility but also brings up several issues. In this paper, we investigate the problem of building trust with attributes by presenting motivating examples, analyzing the security requirements and giving an informal definition. We also survey one of the most related techniques, namely private matching, and finally present solutions based on it.**

## I. Introduction

Social networks and Web 2.0 applications allow effective information sharing but raise a number of privacy challenges. As a result, privacy preserving authentication techniques have gained dramatic importance in building secure distributed systems. In this paper, we introduce protocols to setup $\lambda$-congenial secret groups in two-party and multi-party scenarios. The goal is to effectively establish a secure, anonymous and unobservable communication channel among two or more parties that are mutually "congenial", i.e., they all fulfill a number of common requirements encoded in attributes. Unlike authentication protocols where trust is built on identities, in a $\lambda$-congenial secret group, trust is built on attributes. Hence, $\lambda$-congenial groups are a perfect match for environments where fine-grained authentication, anonymity and privacy are essential requirements.

By congenial group, we mean a group where all members possess one or more common attributes which represent level of rank, authority, skills and so on. A congenial group is called $\lambda$-congenial if all members share $\lambda$ common attributes. Such groups may be found in a several domains. For instance, parties from a given state might form a 1-congenial group where the one common attribute is their state; parties from the same university within that state might form a 2-congenial group where the two attributes are the state and the university.

A $\lambda$-congenial group is called secret if the trust setup protocol is both anonymous and privacy-preserving. By anonymous, we mean that the identity of any participant is kept hidden to other users engaged in the protocol and to both active and passive adversaries. By privacy-preserving, we mean that from the transcripts of the protocol, an adversary cannot learn whether the congenial group has been established or not. We also mean that if the protocol is unsuccessful, none of the participants learns any of the attributes of other parties involved in the protocol, i.e. all participants will disclose their attributes to each other only if the congenial group can be established.

The definition of $\lambda$-congenial group allows users to require a degree of trust based on a subset of their own attributes. If a participant acknowledges that the others share a certain number of attributes with her, she can trust them and thus is willing to interact. If the attributes are issued by a group authority, the protocol also guarantees that a member of a group is a legitimate owner of the attributes used to set up the group.

In real world scenarios, users might resort to trusted third parties, who get proof of users' attributes and output a list of the members of the congenial group. This unconditional trust is fraught with security risks; the trusted party may be dishonest or compromised, as it is an attractive target. Moreover, in many real life scenarios, an online trusted third party may just not be available.

Our motivating example comes from the intelligence community. For security purposes, agents of this community do not know each other and they are required not to disclose their identities to others at any time. Each member may have some skills or roles in the organization and members might be required to cooperate. It is easy to understand the need of a protocol that allows agents to group together on a role/skills basis, without leaking any confidential information. The goal is to form a task force without revealing personal information to parties that will not be part of it. As an example, suppose some agents are needed to cooperate to accomplish a task. They want to cooperate with members who hold the same rank, speak a given language, have certain background and know the secret task code. Before cooperation can start, they need to acknowledge each other as agents and as owners of the required skills; still, they are not willing to disclose their affiliation and their skills to any other party that lacks the requirements for cooperation. A 4-congenial secret group and subsequently an anonymous and unobservable communication channel can be established if all the participants engage in the protocol using as attributes their rank, language, background and task code. If any of the participants does not have one of the required attributes, the communication channel cannot be established and all the participants learn nothing beside the fact that they cannot form the congenial group.

In this paper, we define congenial group systems where members setup $\lambda$-congenial secret groups through trust setup

978-0-7695-4532-5/11 $26.00 © 2011 IEEE
DOI 10.1109/BWCCA.2011.35

protocols. Our protocols achieves completeness, anonymity, privacy-preserving, impersonator resistance, unspoofability and unlinkability through a combination of techniques of private-matching, secret sharing and group key distribution. At the same time, low computational overhead make our protocols suitable for a range of scenarios, allowing trust establishment even with resource constrained devices.

The rest of the paper is organized as follows. Section II gives a brief introduction to our main building blocks while Section III presents the security model for $\lambda$-congenial group trust setup protocols. Section IV gives definition and overview of a congenial group system and presents the system operations. In Section V we presents our two-party and multi-party trust setup protocols. Section VI analyzes the security properties while Section VII evaluates the performances. Finally Section VIII provides a brief overview of related work and Section IX concludes the paper.

## II. Preliminaries

This sections provides an overview of the two main building blocks used in our secret trust setup protocols.

### A. Private Matching

Private matching or privacy-preserving set intersection refers to the problem of computing common elements, i.e. the intersection over two or more private datasets. A private matching scheme is designed to guarantee that participating parties do not learn anything more than they would have learnt, had they given the data to a trusted third party and got back the answer. Private matching schemes proposed so far fall into two categories: those employing commutative encryption schemes [1], [2] and those employing polynomials and homomorphic encryption schemes [3], [4].

Two-party private matching protocols based on commutative encryption are proposed in [1], [2]. In these protocols, each party's input is encrypted twice by both the parties. Since encryption is commutative, resulting values from the two encrypted sets will be equal if and only if the original values were the same (the items were present in both the sets).

Freedman et al. proposed protocols for two-party private matching based on the representation of sets as roots of polynomials [3]. Their work does not use properties of the polynomial beyond evaluation at given points. Kissner et al. explored the power of polynomial representation of multi-sets, using operations on polynomials to obtain composable privacy-preserving multisets operations [4]. Both schemes assume set elements are drawn from a large domain so that an element drawn uniformly from this domain has negligible probability of representing an element in the set.

Two-party private matching schemes [1], [2], [3] can be extended to support multi-party private matching. We compare the communication cost and computation cost of the extended multi-party private matching schemes with [4] in Table I. Note that the computation cost of [3] can be optimized using Horner's rule from $O(nk^2)$ to $O(nk \log\log k)$ and Table I only lists un-optimized results.

TABLE I
COMPARISON OF COMMUNICATION AND COMPUTATION COSTS AMONG SCHEMES [2], [3] AND [4] UNDER HONEST-BUT-CURIOUS ADVERSARY MODEL. THERE ARE $N \geq 2$ PLAYERS, $c < N$ DISHONESTLY COLLUDING, EACH WITH AN INPUT MULTISET OF SIZE $k$. THE DOMAIN OF THE MULTISET ELEMENTS IS $D$.

| Schemes | Communication Cost | Computation Cost |
|---|---|---|
| Agrawal et al. [2] | $O(N^2 k \log \lvert D \rvert)$ | $O(Nk^2)$ |
| Freedman et al. [3] | $O(N^2 k \log \lvert D \rvert)$ | $O(Nk^2)$ |
| Kissner et al. [4] | $O(cNk \log \lvert D \rvert)$ | $O(ck^2)$ |

### B. Secret Sharing

Secret sharing was introduced by Shamir [5] in 1979. The basic idea in secret sharing is to divide a secret into pieces and distribute them to different parties so that the secret can only be reconstructed when the shares are combined; individual shares are of no use on their own.

More formally, in a $(q, n)$-threshold secret sharing scheme, there is one *dealer* and $n$ *players*. The dealer divides the secret into $n$ shares and gives each player a share in such a way that any group of $q$ (the threshold) or more players can together reconstruct the secret but no group of less than $q$ players can.

An efficient $(n, n)$ secret sharing scheme, which means all shares are necessary to recover the secret, is as follows. The secret is encoded as an integer $s$. Each player $i \in \{1, \ldots, n-1\}$ is given a random integer $r_i$. The $n$-th player receives $s - \sum_{i=1}^{n-1} r_i$. The secret is the sum of the shares of all players.

## III. Security Properties

A $\lambda$-congenial secret group protocol must be complete, anonymous, privacy-preserving, impersonator-resistant, unspoofable and unlinkable.

*Complete*. Informally, completeness states that if two or more honest parties run the protocol to establish the congenial group with the same set of $\lambda$ attributes, then each party will output "accept" at the end of the protocol.

*Anonymous*. Identities of the parties engaged in establishing a congenial group must remain hidden to any other party involved in the protocol and to both passive and active adversaries.

*Privacy-Preserving*. If two or more parties run the protocol to establish the congenial group with a different set of attributes, they will only learn that the congenial group cannot be established, i.e. they will not learn any information about the attributes owned by other parties involved in the protocol.

*Impersonator Resistance*. An adversary who is not a member of the system and wants to join a congenial group pretending to own some attributes, will be "rejected" by others with high probability.

*Unspoofability*. Anyone who wants to join a congenial group without having all the necessary attributes, cannot do so with high probability.

*Unlinkability*. Given the transcripts of two or more different protocol executions, deciding whether these two transcripts involve the same user is computationally hard.

## IV. System Model

### A. System Overview

A congenial group system consists of a *group authority* ($GA$), $N$ users, a set of attributes $X$, and several groups, each of them associated with a subset of $X$.[1] The $GA$ authorizes user $i$ to be a legitimate member of a group $G$ by assigning to $i$ the attributes of the group, namely $X_G$. We will refer to the set of attributes given to user $i$, by $X_i$. Users with the same set of attributes are logically affiliated with the same group, but are not aware of the group they belong to, its size, and of other members in it. Members of the system can use a subset of their attributes to form a congenial group $CG$; such group will be a $\lambda$-congenial group if $|CG| = \lambda$.

The set of attributes that defines each congenial group may overlap. In Figure 1, each circle denotes a group. Member $i$ belongs to a group $G$ if $X_i \supset X_G$. An arrow from $G'$ to $G''$ means an overlap between $X'_G$ and $X''_G$; more specifically $G' \supset G''$. Thus members of $G'$ are automatically members of $G''$. That is, a member in a group $G$ will be able to establish congenial groups with any members in any group within the subtree rooted in $G$. Members of isolated groups (e.g., $I$) and leaf groups (e.g., $B$, $E$, $F$ and $G$), can only establish congenial groups with members of the same group.

Admission and revocation of users to the system are managed by the $GA$ like in a key distribution scheme [6]. As in such schemes, time is divided in rounds and a transition from round $t$ to round $t+1$ is necessary each time one or more users are revoked at once. In a specific round, members can use their attributes to engage in as many congenial group establishments as desired.

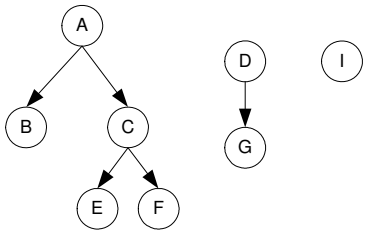


Fig. 1. **Sample group assignment.**

A legitimate member $i$ after getting $X_i$ can use part (or all) of the attributes in $X_i$ as she needs to initiate/join a congenial group through a trust setup protocol. However, a legitimate member $i$ of the system might not be entitled to join $CG$ if $X_i \not\supseteq CG$. To avoid impersonator attacks, attributes are encrypted with a group key, while to avoid spoofing attacks, they are also masked with nonces. Hence, attributes $X_i$ are not directly used in the trust establishment protocol, but they are replaced with an encrypted version. (Details of encryption are given below). In particular, we will refer to the set of encrypted attributes of user $i$ as $C_i$; elements in $C_i$ will be referred as secrets. Both $X_i$ and $C_i$ are kept secret and the latter are revoked by the $GA$ when $i$ is no longer a legitimate member of the system.

When a member $i$ wants to form a $\lambda$-congenial secret group, she selects $\lambda$ secrets from $C_i$ as input to the trust establishment protocol. Suppose there are $m$ players, the protocol will be successful iff $|\bigcap_{i=1}^{m} C_i| = \lambda$. For example, in Figure 1 if member $i$ from group $A$ wants to form a congenial group with members of group $E$, she may only use group $E$ set of secrets as her input in the protocol. Members from groups A, C and E are all able to join the congenial group. However, no one can get any information about which groups the others belong to. That is, members know nothing about the group affiliation, size, hierarchy, etc.

### B. System Operations

The congenial group system consists of four operations: *system-init*, *member-admit*, *member-revoc* and *trust-setup*. The $GA$ is required to be online only for the first three operations. The protocols for the congenial group formation do not need interaction with the $GA$.

We describe the first three operations here while we present our two-party and multi-party trust setup protocols in the next section. Our notation is summarized in Tab. II:

TABLE II
Notations

| Notation | Meaning |
|---|---|
| $MK^t$ | group key at round $t$ |
| $X$ | set of all secrets (each user will be given a subset of X) |
| $X_i$ | set of secrets owned by user $i$ |
| $x_{i_k}$ | $k$-th element of $X_i$ |
| $n_k^t$ | nonce associated to the $k$-th element of $X$ at round $t$ |
| $c_k^t$ | $c_{i_k}^t = E_{MK^t}(x_{i_k} \bigoplus n_k^t)$ |
| $C^t$ | set of all $c_k$ at round $t$ |
| $C_i^t$ | set of $c_{i_k}^t$ owned by user $i$ at round $t$ |

*System-Init*. At round 0 the $GA$ initializes the system with a set of attributes $X = \{x_1, \ldots, x_n\}$. Also it picks a random group key $MK^0$ for a symmetric encryption scheme (e.g., AES). Finally, for each $x_k$ in $X$, the $GA$ picks a nonce $n_k^0$ and computes $c_k^0 = E_{MK^0}(x_k \bigoplus n_k^0)$, where $\bigoplus$ is the bitwise XOR operation. Since the domain of the attributes might be small and a member of the system might have non-negligible probability to guess them, nonces are drawn from a large domain.

*Member-Admit*. At round $t$, user $i$ wants to be included in the system. The $GA$ determines $i$'s legitimate status and sends her $MK^t$, $X_i$ and $C_i^t = \{c_k^t = E_{MK^t}(x_k \bigoplus n_k^t) : x_k \in X_i\}$, where $E_\cdot(\cdot)$ is the symmetric encryption algorithm.

*Member-Revoc*. Members are revoked when the system advances from one round to the following one. Assume user $i$ is revoked when moving from round $t$ to round $t+1$. At this time, the $GA$ randomly picks a new group key, namely $MK^{t+1}$ and securely distributes it among all legitimate users. The latter can be achieved via broadcast encryption techniques [7]. Then, the $GA$ and each legitimate user $i$ computes the following: for each $x_k \in X_i$, they calculate $n_k^{t+1} = H(n_k^t, MK^{t+1})$ and $c_k^t = E_{MK^{t+1}}(x_k \bigoplus n_k^{t+1})$.

[1] Actual groups are the elements of the power set $P(X)$.

## V. Trust Setup Protocols

Our two-party and multi-party trust setup protocols are based on the two-party private matching scheme due to Freedman et al. [3] and the multi-party private matching scheme due to Kissner et al. [4], respectively.

The main idea is to enhance private matching protocol in order not to leak private information when the congenial group cannot be established.

To simplify the discussion, we assume that the protocols are carried out within one round, so we drop all superscripts referred to the actual round. Also, from now on we will use $X_i$ and $C_i$ to denote both the set of attributes and secrets owned by user $i$ or the set of attributes and secrets that she inputs to the trust setup protocol.

### A. All-or-Nothing

We modify the private-matching schemes so that our trust setup protocols have the "all-or-nothing" property [8]. That is, if the protocol succeeds, every player recognizes that all the other players have input the same set of secrets; if the protocol fails, no player gets to know anything beyond the fact that not all the parties used the same set of secrets to run the protocol. We adapt an $(n, n)$ secret sharing scheme described in Section II-B to afford "all-or-nothing" as follows.

In both private matching schemes [3], [4], user elements are represented as roots of polynomials. User $i$ defines a polynomial $P_i(\cdot)$ whose roots are the set of her secrets and encrypts its coefficients with an homomorphic encryption system. Encrypted coefficients are sent to user $j$. For each $x_k \in X_j$, user $j$ computes $y_k = r_k \cdot P_i(x_k) + 1$, where $r_k$ is a random nonce, and sends the results back to $i$. If $x_k \in X_j$ is a root of $P_i$, after decryption of $y_k$, user $i$ gets "1", otherwise, she gets a random value. Hence, $i$ learns the intersection between $X_i$ and $X_j$.

In our protocols, each user $i$ computes a hash of the attributes she uses as inputs. In particular, user $j$ computes $h_j = H(x_{j_1}, \ldots, x_{j_\lambda})$ (we assume attributes are always sorted lexicographically) and divides $h_j$ using a $(\lambda, \lambda)$ secret sharing scheme into $\lambda$ shares: $h_{j_1}, \ldots, h_{j_\lambda}$. Then, for each $c_k \in C_j$, user $j$ evaluates the received polynomial $P_i$ as $r_k \cdot P_i(c_{k_j}) + h_{k_j}$, where $r_k$ is a random nonce.

In the two-party trust setup protocol, when $i$ gets all the encrypted evaluations from $j$ and decrypts them, if the sum of the decrypted results $\hat{h}_j$ is equal to $h_i$, then $i$ outputs "accept" to indicate that $j$ is acknowledged as a member of the congenial group. The protocol succeeds when both $i$ and $j$ output "accept". Otherwise, the protocol fails.

In the multi-party trust setup protocol, all players work together to get a polynomial representation $P$ of the intersection of their inputs and then all of them evaluate $P$. If there are $m$ players $(1, \ldots, m)$, after the group decryption, the sum of the results is $h = h_1 + \cdots + h_m$. If all the players used the same set of secrets as input, then $h_1 = \ldots = h_m$. Player $i$ $(i = 1, \ldots, m)$ checks if $h_i = \frac{h}{m}$ and outputs "accept" if the equation holds. The protocol succeeds when all the players output "accept". Otherwise, the protocol fails.

### B. Two-Party Trust Setup Protocol

The trust setup protocol between two parties $i$ and $j$ is depicted in Figure 2. We assume before engaging in the protocol, each party computes a fresh key pair of a common homomorphic encryption system.[2] Algorithms $SS_{q,n}(\cdot)$ and $Rec_{q,n}(\cdot)$ are the algorithm to share and recover a secret according to a $(q, n)$-threshold secret sharing scheme, respectively. User $i$ defines a polynomial whose roots are the set of her secrets and encrypts it under her homomorphic public key. The encrypted coefficient are sent to the other party. Next, user $i$ shares the hash of her attributes (i.e., $h_i$) into $\lambda$ shares, according to a $(\lambda, \lambda)$-threshold secret sharing scheme. The received polynomial is evaluated for each secret and the result is first masked with randomness and later added to one of the share just computed. Results are sent to the other party. Finally, results of the polynomial evaluation by the other party are decrypted using the homomorphic secret key and input to recovery algorithm of the $(\lambda, \lambda)$-threshold secret sharing scheme. If the output of the recovery algorithm equals $h_i$, then user $i$ output "accept" otherwise she rejects.

The protocol succeeds when both $i$ and $j$ output "accept" and fails otherwise.

### C. Multi-Party Trust Setup Protocol

In a multi-party scenario, members collusion is an important issue. While in general private matching schemes, elements of any member are arbitrarily chosen, in our congenial group system, attributes and thereby groups are managed by the $GA$ in a way such that colluding attacks are discouraged: no one gets benefit by colluding with other users. This can be achieved using the following rules in group management:

- The set of secretes of an ancestor group is strict larger than the union of the set of secrets owned by all its offspring groups.
- Groups without ancestor-offspring relations should have disjoint set of secrets.

While in the two-party protocol each party generates a public/private key pair, in the multi-party scenario, the $GA$ generates a system wide public/private key pair $(sk, pk)$. The public key is known by everybody while the secret key is shared by all the legitimate members using a $(l, n)$ secret sharing scheme where $l$ is threshold and $n$ is the number of legitimate members in the congenial group system. This requires at least $l$ legitimate members to participate in the protocol in order to make group decryption available. As a consequence any group of $l$ legitimate members, even though they do not take part in the protocol, can decrypt the evaluations of the polynomial and learn information about the intersection of the actual players: if one of the colluding members has the same set of secrets that are involved in the protocol, they learn the intersection set. To solve this problem we introduce randomness in the shares, using the Burmester and Desmedt conference key distribution scheme [9]. Thus, during *System-Init*, the $GA$ sets up the conference key distribution scheme

[2] Reusing the same key pair would trivially violate user anonymity.

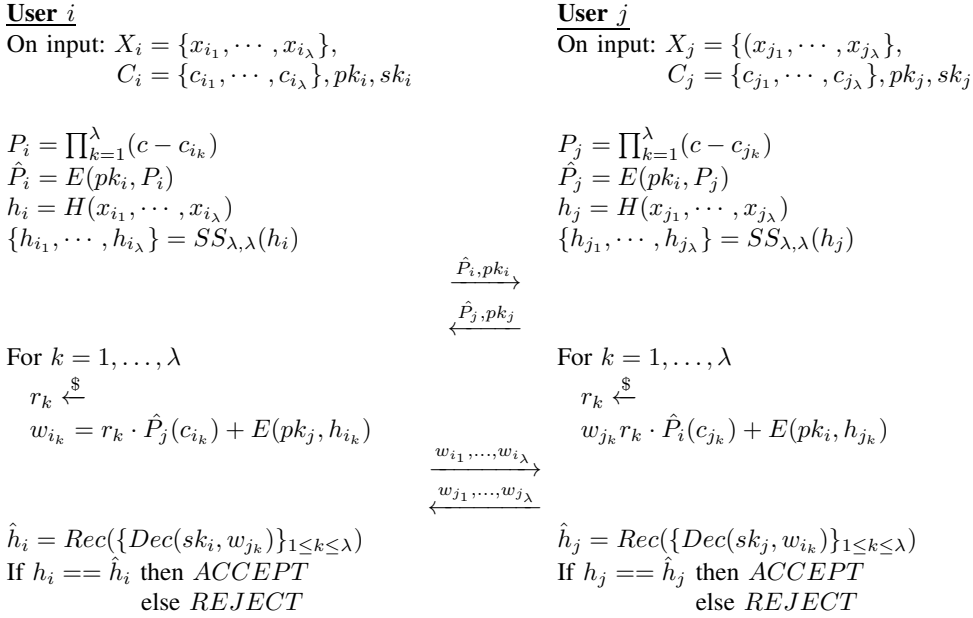


Fig. 2. Two-party trust establishment protocol.

parameters as follows: picks a prime $p = \Theta(2^{cN})$ where $N$ is a security parameter and $c \geq 1$ is a constant, chooses an element $\alpha \in Z_p$ of order $q = \Theta(2^N)$, and publishes $p$, $\alpha$ and $q$.

Suppose there are $m$ ($m \geq l$) parties playing the protocol, our multi-party protocol is shown in Figure 3. The multi-party trust setup protocol starts with all involved parties running the conference key scheme of [9] to agree on a random value $O_i$ which is to be used to randomize the attribute values. It involves two rounds of broadcast communications. Following the conference key agreement scheme is the multi-party secret matching scheme which also involves two rounds of broadcast communications. The protocol successes when all the players output "accept" and fails otherwise.

## VI. Security Analysis

In this section we analyze our protocols with the security properties listed in Section III.

### A. Completeness

In our two-party protocol, if $i$ and $j$ execute the protocol with the same set of attributes (i.e. $C_i = C_j$), each $c_{j_k}$ in $C_j$ is a root of $P_i$. Thus, for each $c_{j_k}$ in $C_j$, we get $E_{pk_i}(r_k \cdot P_i(c_{j_k}) + h_{j_k}) = E_{pk_i}(h_{j_k})$. User $i$ will receive $\lambda$ such values, decrypt them with her secret key and re-assemble them to obtain $h_j$. Since $h_j = H(c_{j_1}, \cdots, c_{j_\lambda}) = H(c_{i_1}, \cdots, c_{i_\lambda}) = h_i$, then $i$ outputs "accept". A similar argument holds for $j$.

In our multi-party protocol, the resulted polynomial $P$ in step 2 is the polynomial whose roots are the intersection of the inputs from all parties. Every party evaluates $P$ with her set of inputs. If all parties input the same set of secrets, then $h_1 = \cdots = h_m$ and $P$ is evaluated $m$ times in the same points. Thus $s = h_1 + \cdots + h_m = m \cdot h_i$. So if $h_i = \frac{s}{m}$, player $i$ outputs "accept".

### B. Anonymity

Since our protocols are based on user attributes rather than identities, no unique value is bound to the identity of any given party that runs the protocols. Thus their identity is kept secret.

### C. Privacy

In the two-party protocol, suppose that $i$ and $j$ run the protocol with different set of attributes (i.e. $C_i \neq C_j$). For each $c_{j_k} \in C_i \cap C_j$, $E_{pk_i}(r_k \cdot P_i(c_{j_k}) + h_{j_k}) = E_{pk_i}(h_{j_k})$. For each $c_{j_k} \notin C_i \cap C_j$, $E_{pk_i}(r_k \cdot P_i(c_{j_k}) + h_{j_k}) =$ random. Since $j$ compute the shares randomly, $i$ will not be able to distinguish between one of the shares $h_{j_k}$ and a random value. Thus $i$ will not be able to tell which of her attributes are owned by $j$. A similar argument holds for $j$.

In the multi-party protocol, since the evaluated polynomial $P$ has as roots the intersection of all the users' inputs and shares are computed randomly by all the parties, no one can distinguish between one of the shares $h_{j_k}$ and a random value; moreover, no one can link a pair of shares to any user.

### D. Impersonator Resistance

Both protocols use elements drawn from $C$. Since its domain is very large, an adversary has very little probability to guess any of those elements. Furthermore, if she wants to be acknowledged in a $\lambda$-congenial group, she has to guess $\lambda$ secrets. Even if the adversary recovers all the secrets, she will not be able to tell which attributes are owned by the members of the congenial group, since she misses both $MK^t$ and the nonces.

| **Each user $i$** | **computes** | **broadcasts** |
|---|---|---|
| *Move 1:* | $b_i \xleftarrow{\$} Z_q$<br>$z_i = \alpha^{b_i} \pmod p$ | |
| | | $\xrightarrow{z_i}$<br>$\xleftarrow{z_j (j \neq i)}$ |
| *Move 2:* | $w_i = (z_{i+1}/z_{i-1})^{b_i} \pmod p$ | |
| | | $\xrightarrow{w_i}$<br>$\xleftarrow{w_j (j \neq i)}$ |
| *Move 3:* | $o_i = (z_{i-1})^{mb_i} \cdot (w_i)^{m-1} \cdot (w_{i+1})^{m-2} \cdots w_{i-2} \pmod p$<br>$P_i = \prod_{k=1}^{\lambda} (c - c_{i_k})$<br>$\hat{P}_i = E(pk, P_i)$<br>$R_i \leftarrow R^{\lambda}[x]$<br>$\phi_i = \hat{P}_i \times R_i$ | |
| | | $\xrightarrow{\phi_i}$<br>$\xleftarrow{\phi_j (j \neq i)}$ |
| *Move 4:* | $F = \sum_{j=1}^{l} \phi_j$<br>$h_i = H(x_{i_1}, \cdots, x_{i_\lambda}, o_i)$<br>$(h_{i,1}, \cdots, h_{i\lambda}) = SS_{\lambda,\lambda}(h_i)$<br>For $k = 1, \cdots, \lambda$<br>$r_k \xleftarrow{\$}$<br>$v_{i_k} = r_k \times E(pk, F(c_{i_k} \oplus o_i)) + h_{i_j}$ | |
| | | $\xrightarrow{V_i = \{v_{i_1}, \cdots, v_{i\lambda}\}}$<br>$\xleftarrow{V_j (j \neq i)}$ |
| **All participants** | | |
| *Move 5:* | (together decrypt each element of $V = \{V_1, \cdots, V_m\}$)<br>For $k = 1, \cdots, m$<br>$\hat{V}_i = \{Dec(sk, v_{i_1}), \cdots, Dec(sk, v_{i_\lambda})\}$<br>$s_i = Rec(\{\hat{V}_i\}) = Rec(\{Dec(sk, v_{i_k})\}_{1 \leq k \leq \lambda})$<br>$s = \sum_{k=1}^{\lambda} Rec(d)$ | |
| **Each user i** | | |
| *Move 6:* | If $h_i == s/m$ then $ACCEPT$<br>else $REJECT$ | |

Fig. 3. Multi-party trust establishment protocol.

### *E. Unspoofability*

Let $i$ and $j$ have set of attributes $X_i$ and $X_j = X_i \bigcup x_z$ respectively, i.e. $i$ has all but one of the $j$'s attributes. Suppose they play the trust setup protocol using all of their attributes and suppose $i$ pretends to have $x_z$. Even though $i$ knows $MK^t$ and guesses $x_z$ ($X$ domain might be small) she has very little probability to guess the nonce (which are drawn from a large domain) to compute $E_{MK^t}(x_z \bigoplus n_z^t)$. Thus she is unable to produce all the $c_i$ ($i = 1, \cdots, |X_j|$) to join the congenial group.

### *F. Unlinkability*

In the two-party setting, each time a different key pair is used to achieve unlinkability. Since a semantically secure homomorphic encryption scheme is used, two executions of the protocol performed by a legitimate member will not give to any adversary enough information to link the two executions to any user. In the multi-party setting, a random polynomial is used each time the protocol is executed. Again, no enough information are given to any adversary to link two executions of the protocol to any user.

## VII. Performance Analysis

In this section we analyze the computation and communication overhead of our protocols for the $GA$ and a legitimate member.

### *A. System-Init*

Only the GA is involved in the *system-init* operation during which it chooses a random key for a symmetric encryption system, generates $|X|$ nonces and performs $|X|$ symmetric key encryptions. The $GA$ must also setup parameters for the conference key distribution scheme as in [9].

### B. Member-Admit

The $GA$ sends to the new user the group key $MK$, $X_i$ and the corresponding ciphertexts $C_i$. There is no computation overhead while the communication overhead is linear in the number of attributes provided to the user.

### C. Member-Revoc

When the system moves to a new round, malicious users detected during the previous round are revoked.[3] A new group key is randomly chosen by the $GA$ and distributed among all legitimate users through broadcast encryption, e.g., [7]. Let $\bar{X}$ be the union of all attributes owned by users being revoked at the current round. For each attribute in $\bar{X}$, the $GA$ computes a hash (to get the nonce for the current round) and a symmetric encryption operation. User $i$ is required to compute hash and encryption for each attribute in $X_i$.

### D. Two-party Trust-Setup

Part of the trust establishment protocol can be carried off-line. At this time, user $i$ (1) defines a polynomial of degree $\lambda$ through interpolation, (2) encrypts its coefficient using her fresh homomorphic public key, (3) computes the hash of $\lambda$ attributes and (4) breaks the computed hash in $\lambda$ shares.

During the online part of the protocol, user $i$ (1) picks $\lambda$ random values, (2) computes $\lambda$ public key encryptions and decryption, (3) performs $\lambda$ polynomial evaluations and (5) reassembles $\lambda$ shares. Each user also need to send $2\lambda$ ciphertexts and one public key. Table III shows communication and computation overhead of the trust setup protocol. In the two-party scenario, communication and online computation overhead is linear in the number of attributes of the congenial group. The offline part of the protocol is more involving.

TABLE III
OVERHEAD OF THE TWO-PARTY AND MULTI-PARTY TRUST SETUP PROTOCOLS.

| | **Communication** | **Computation (offline)** | **Computation (online)** |
|---|---|---|---|
| Two-Party | $O(\lambda)$ | $O(\lambda^2)$ | $O(\lambda)$ |
| Multi-Party | $O(m\lambda)$ | $O(\lambda^2)$ | $O(m\lambda)$ |

### E. Multi-party Trust-Setup

The multi-party trust setup protocol starts with all involved parties running the conference key scheme of [9] to agree on a random key. It requires two communication rounds and three modular exponentiations for each user.

Next each user employs interpolation to define a polynomial of degree $\lambda$, encrypts it and multiplies it with a random polynomial of the same degree. The encryption and multiplication involve $\lambda$ and $\lambda^2$ modular exponentiations respectively. This constitutes the offline part the protocol. When a user receives all the encrypted polynomials, she computes their sum with $\lambda(m-1)$ multiplications. The resulting polynomial is evaluated $\lambda$ times, once with each of the user input secrets. For verification each user performs $m\lambda$ decryptions.

[3]Techniques to identify malicious users are beyond our scope.

Each user also needs two broadcast messages to send its encrypted polynomial and later $\lambda$ evaluated results to others respectively.

Thus the multi-party trust setup protocol requires four communication rounds and the last two rounds involve $m$ participants sending coefficients encrypted polynomials or evaluations of polynomials with $\lambda$ values. The communication complexity of the multi-party protocol is $O(m\lambda)$. The total cost of offline computation per user is $O(\lambda^2)$ and the total cost of online computation per user is $O(m\lambda)$.

## VIII. Related Work

Two-party secret handshakes [10], [11] are mostly related to our work. They allow secret key establishment between two parties only if they have the same affiliation; otherwise no information is leaked. Multi-party secret handshakes were introduced by [12] and later enhanced in [13]. Our protocol does not lead to a secret key establishment but is only focused on trust establishment . (It could be easily extended to provide a secret key shared between the members of the congenial group).

However, secret handshakes only provide "static" affiliation, i.e., groups are defined by the authority and members can belong to one or more groups. Our protocols allow for "dynamic" affiliation. Given its set of attributes, a member can dynamically define an affiliation using any subset of its attributes. Ateniese et al. [14] introduced "Fuzzy Matching" for secret handshakes where Alice and Bob establish a secret only if the cardinality of attribute matching is above an arbitrary threshold. However, their protocol only allows two-party handshakes and requires bilinear map operations, what makes it more involving.

## IX. Conclusion

Motivated by the secret congenial group example in the intelligence community, with the objective to enhance the trust and privacy of electronic communities, we proposed a congenial group system where legitimate members form congenial groups through an anonymous and privacy-preserving trust setup protocol. Our trust setup protocols achieve completeness, anonymity, privacy-preserving, impersonator resistance, unspoofability and unlinkability through a combination of techniques related to private-matching, secret sharing and group key distribution. We believe that our work can be improved in both efficiency and flexibility: communication and computation costs can be optimized and other desirable properties can be achieved.

**Acknowledgments.** This research has been partially funded by the Madrid Regional Council – CAM under project CLOUDS (S2009TIC-1692), the Spanish Research Agency – MICINN under project CloudStorm (TIN2010-19077), and the European Commission under projects MASSIF (FP7-257475) and STREAM (FP7-216181).

## REFERENCES


[1] B. A. Huberman, M. K. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *ACM Conference on Electronic Commerce*, 1999, pp. 78–86.

[2] R. Agrawal, A. V. Evfimievski, and R. Srikant, "Information sharing across private databases," in *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2003, pp. 86–97.

[3] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2004, pp. 1–19.

[4] L. Kissner and D. X. Song, "Privacy-preserving set operations," in *25th Annual International Cryptology Conference(CRYPTO)*, 2005, pp. 241–257.

[5] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[6] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309–329, 2003.

[7] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *25th Annual International Cryptology Conference(CRYPTO)*, 2005, pp. 258–275.

[8] B. Schneier, *Applied cryptography*, J. W. . Sons, Ed., 1996.

[9] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system (extended abstract)," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 1994, pp. 275–286.

[10] D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H.-C. Wong, "Secret handshakes from pairing-based key agreements," in *IEEE Symposium on Security and Privacy (S&P)*, 2003, pp. 180–196.

[11] C. Castelluccia, S. Jarecki, and G. Tsudik, "Secret handshakes from ca-oblivious encryption," in *10th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'04)*, 2004, pp. 293–307.

[12] G. Tsudik and S. Xu, "Brief announcement: a flexible framework for secret handshakes," in *24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2005, p. 39.

[13] S. Jarecki, J. Kim, and G. Tsudik, "Authentication for paranoids: Multi-party secret handshakes," in *4th International Conferenceon Applied Cryptography and Network Security (ACNS)*, 2006, pp. 325–339.

[14] G. Ateniese, J. Kirsch, and M. Blanton, "Secret handshakes with dynamic and fuzzy matching," in *Network and Distributed System Security Symposium (NDSS)*, 2007.