Inonymous: Anonymous Invitation-Based System

Sanaz Taheri Boshrooyeh(✉) and Alptekin Küpçü

Department of Computer Engineering, Koç University, İstanbul, Turkey
{staheri14,akupcu}@ku.edu.tr

Abstract. In invitation-based systems, a user is allowed to join upon receipt of a certain number of invitations from the existing members. The system administrator approves the new membership if he authenticates the inviters and the invitations, knowing who is invited by whom. However, the inviter-invitee relationship is privacy-sensitive information and can be exploited for inference attacks: The invitee's profile (e.g., political view or location) might leak through the inviters' profiles. To cope with this problem, we propose *Inonymous*, an anonymous invitation-based system where the administrator and the existing members do not know who is invited by whom. We formally define and prove the inviter anonymity against honest but curious adversaries and the information theoretic unforgeability of invitations. *Inonymous* is efficiently scalable in the sense that once a user joins the system, he can immediately act as an inviter, without re-keying and imposing overhead on the existing members. We also present *InonymouX*, an anonymous cross-network invitation-based system where users join one network (e.g., Twitter) using invitations of members of another network (e.g., Facebook).

Keywords: Invitation-based system · Anonymity · Unforgeability · Cross-network invitation

1 Introduction

An invitation-based system consists of a server (administrator) and a group of members. New users join the system only by obtaining invitations from a certain number of existing members. Each invitation confirms some level of trust to the invitee. This authentication method is also known as *trustee-based social authentication*. Invitation-based systems benefit from trustee-based authentication for the initial registration of a user to the system. Afterward, any authentication technique e.g., a password, can be utilized for the further logging into the system.

Invitation-based systems are employed due to various reasons such as a limited number of server resources to cover an arbitrary number of users, improving the quality of services by constraining the number of members, securing the system against fake users, and providing data or service privacy for the system. As a well-known historical example, Google applied invitation-based registration in the early stages of its new services like Gmail, Orkut, and Google Wave [1].

J. Garcia-Alfaro et al. (Eds.): DPM/CBT 2017, LNCS 10436, pp. 219–235, 2017.
DOI: 10.1007/978-3-319-67816-0_13

In invitation-based systems, the administrator knows the identity of the user's inviters to authenticate and manage new registrations. In some other cases, not only the administrator but also other members of the system are informed about the correspondence of a newcomer and his inviters. For example, in Telegram chat application, once a new user joins a group, its referee's identity is broadcasted to the group members. The user's referees are mostly among the user's acquaintances (e.g., colleagues, home mates, family members, close friends) who have many common preferences with the user. Due to this reason, information like location, religious beliefs, sexual orientation, and political views can be inferred about a user by analyzing the common features among his inviters [3,8]. Thus, the set of user's referees is privacy-sensitive information.

Related Works: Typically, trustee-based social authentication is considered as a backup authentication method, rather than the primary one. A backup authentication method is employed where the user fails to pass the primary authentication e.g., forgetting the password [4]. The account holder determines a set of trustees to the server in advance. When the user loses access to his account, the server sends recovery codes to the trustees. Upon collection of enough number (recovery threshold) of codes from the trustees, the user recovers his access. *Forest fire attack* [9] is the most significant security issue in this context where an attacker compromises a few seed accounts and exploits this to steal other accounts. The main security measures against forest fire attack are increasing the recovery threshold [10], assigning time to live to the recovery codes [10,13], keeping the identity of trustees hidden from the recovery requester [9], bit stuffing [9], and using encrypted recovery codes [13]. None of the mentioned solutions preserve the anonymity of inviters as it is not a concern in those applications. Instead, the identity of a trustee (i.e., inviter) is by default known to the server so that the server can communicate with the trustees for the account recovery procedure. On the contrary, in invitation based systems, the anonymity of trustees (inviters) is a security concern due to the inference attacks. This indicates that the existing trustee-based solutions are not applicable to invitation based systems as they disregard the anonymity of trustees.

To cope with this problem, we developed an anonymous invitation-based system named *Inonymous*. Our system overview is depicted in Fig. 1. *Inonymous* consists of three entities: A server (administrator), existing members (inviters) and a newcomer (invitee). The invitee receives invitations from a subset of existing members i.e., inviters. The invitee knows the inviters beforehand via some other means outside the network to be joined to. In the Gmail example, Google employees and their families/friends are the inviters and invitees, respectively. The invitee combines the invitations into a single invitation letter and submits to the server. If the invitation is verified by the server, the invitee joins the system. No interaction is required between inviters and the server. In contrast to the prior studies, in *Inonymous* inviters can anonymously add new users. This anonymity is not only against the server but also against the other members including inviters of the same invitee. Despite the anonymity of the inviters, the server can still verify the integrity of the invitations.

That is, a malicious invitee cannot join the system without threshold many legitimate invitations. We formally define inviter anonymity and invitation unforgeability and provide game based proofs of security. Furthermore, *Inonymous* efficiently enables the recently invited users to act as inviters. This is done instantly and without rekeying the system and contacting existing members.

Note that, while *Inonymous* guarantees the anonymity of the inviters in the phase of registration, it does not tackle with the anonymous interaction and relationship of users within the system. For example, while a user is invited to a Facebook group using *Inonymous*, his interaction with other group members might imply some information about his potential inviters. Thus, the anonymous interaction of users must be addressed independently and is out of the scope of this paper. But recall that the application of invitation-based system is not limited to the social networks e.g., a cloud does not constitute a social network while it may employ invitation-based notion to offer a limited service or storage to the recommended customers.

Additionally, we construct *InonymouX*, an anonymous cross-network invitation-based system on top of *Inonymous* which can be of independent interest. In the cross-network design, a user joins one system e.g., Twitter, by obtaining invitations from members of another network e.g., Facebook. The cross-network design is beneficial especially to bootstrap a system, for example in the case where a research group wants to hire qualified researchers from another group. A qualified researcher is the one with enough recommendations i.e., invitations from his own group.

Our contributions are as follows:

- *Inonymous* is the **first anonymous** invitation-based system that provides *inviter anonymity* and *invitation unforgeability*.
- We provide **formal security definitions and proofs** for both security objectives.
- *Inonymous* is **efficiently scalable** in terms of the number of inviters.
- We propose the **first cross-network anonymous** invitation-based system called *InonymouX* where the possibility of inter-network invitation is provided.

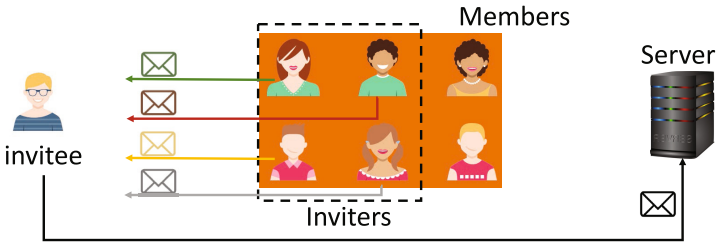

Fig. 1. *Inonymous* system overview.

2 Model

Inonymous is composed of three entities: a server, a set of existing members and a new user who is willing to join the system. The server is responsible for managing and validating users' registrations and generating certificates upon the occurrence of a new membership. The existing members are given necessary information which enables them to make anonymous individual invitations for their trusted ones. The newcomer becomes the member of the system if he obtains a certain number (denoted by t as threshold) of invitations from the existing members i.e., inviters. The invitee knows his inviters prior to joining the system. He collects and aggregates individual invitations to make a single final invitation letter. The aggregation of invitations has the main effect in inviters anonymity. Once the newcomer hands over his invitation to the server, the server is in charge to authenticate the invitation and provide necessary information for him. We assume that the system starts with at least t initially registered members (e.g., Google employees in the Gmail example), who are given credentials to join the system by the server directly. Henceforth, those existing members start inviting others.

Throughout the paper, we assume secure and authenticated channels per communication. In *Inonymous*, we seek two security objectives:

- **Inviter anonymity:** By inviter anonymity, we aim at protecting the identity of the inviters against the server and other members. The invitations should not leak any information about the inviters. We assume that the adversary is the server who may collude with a subset of a newcomer's inviters (obviously not all of them). The adversary is presumed to be honest but curious. We formally prove that the identity of non-colluding inviters remains anonymous to the adversary. Note that the newcomer is supposed to be concerned about his privacy hence does not reveal the identity of his inviters to the adversary, otherwise, the inviter anonymity is meaningless. This is defined in Sect. 7.1 as a game where the adversary controls the server and $t-1$ inviters.
- **Invitation unforgeability:** The invitation unforgeability indicates that a user i.e., an adversary who has an insufficient number ($t^{'}$) of inviters ($t^{'} < t$) should not be able to join the system even if he acts maliciously. The adversary can join if he forges some invitations on his own. Recall that the assumption of $t^{'} < t$ is not a restriction imposed by our system but it is a requirement in any invitation-based system. Indeed, threshold-many members collude then they can control the system in the sense that they can add an arbitrary number of users to the system by generating valid invitations. Thus, a collusion of t members threats any invitation-based system independent of how the system is cryptographically designed. We define invitation unforgeability in Sect. 7.2 as a game where the adversary controls up to $t-1$ existing users, but the server and other members are honest.

Overview: *Inonymous* is managed by a server who owns a master value and a decryption key. The server shares the master value among the existing members

using (t)-Shamir secret sharing scheme where t is the threshold value. Each newcomer requires t invitations to join the system. Each invitation is the masked version of an inviter's master share alongside with the encryption of masking value that is pseudorandomly generated. Once the invitee obtains his invitations, he can unify them into a single invitation by utilizing the homomorphic property of Shamir shares and El Gamal encryption scheme. Invitations are tied to a specific invitee using a server generated token given to each invitee.

3 Preliminaries

Negligible Function. A function f is called negligible if for every polynomial $p(.)$ there exists integer N such that for every $n > N$, $f(n) < \frac{1}{p(n)}$.

Pseudo Random Generator. A deterministic polynomial time function P: $\{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ (where $l(.)$ is a polynomial) is called Pseudo Random Generator (PRG) if $n < l(n)$ and for any probabilistic polynomial-time distinguisher D there exists a negligible function $negl(.)$ such that:

$$|Pr[x \leftarrow \{0,1\}^n : D(P(x)) = 1] - Pr[y \leftarrow \{0,1\}^{l(n)} : D(y) = 1]| = negl(n) \quad (1)$$

Shamir Secret Sharing Scheme. Secret sharing is a tool by which a secret is shared among several parties such that the secret is recoverable in the presence of a certain number of shareholders. The Shamir secret sharing scheme [2,6] works based on polynomial evaluations. The secret owner selects a polynomial f of degree $t-1$ randomly and sets the secret data S as the evaluation of that function at point 0 i.e., $f(0) = S$. Since each polynomial of degree $t-1$ can be uniquely reconstructed by having t distinct points of that function, t Shamir shareholders are able to reconstruct the secret. Shamir shares are homomorphic under addition operation i.e., let $[s_1]$ and $[s_2]$ be Shamir shares of S_1 and S_2, then $[s_1] + [s_2]$ constitutes a share of $S_1 + S_2$.

Multiplicative Homomorphic Encryption Scheme. A public key encryption scheme consists of three algorithms $\pi = (KeyGen, Enc, Dec)$. π is called multiplicative homomorphic encryption if for every a and b, $Enc(a) \otimes Enc(b) = Enc(a \cdot b)$ where a and b belong to the encryption message space and $\otimes$ is an operation over ciphertexts. As an example, in El Gamal encryption [11], $\otimes$ corresponds to a simple multiplication of two ciphertexts. Additionally, we have $Enc(a)^c = Enc(a^c)$ where a is a plain message and c is any integer. Throughout the paper, we consider El Gamal scheme as our underlying encryption scheme.

Signature Scheme. A signature scheme [12] consists of three algorithms $\gamma = (SGen, Sign, SVrfy)$. A pair of keys (sk, vk) is generated via $SGen$ where sk is the signature key and vk is the verification key. The signer signs a message m using sk by computing $\eta = Sign_{sk}(m)$. Given the verification key vk, a receiver of signature runs $SVrfy_{vk}(\eta, m)$ to verify.

Bilinear Map. Consider G_1 and G_2 as multiplicative groups of prime order q. Let g_1 be the generator of G_1. We employ an efficiently computable bilinear map $e : G_1 \times G_1 \rightarrow G_2$ with the following properties [14]

– Bilinearity: $\forall u, v \in G_1$ and $\forall a, b \in \mathbb{Z}_q : e(u^a, v^b) = e(u, v)^{a \cdot b}$.
– Non-degeneracy: $e(g_1, g_1) \neq 1$.

Computational Diffie-Hellman Assumption. Given a cyclic group G of prime order q with a generator g, and two randomly selected group elements $h_1 = g^{r_1}, h_2 = g^{r_2}$, the Computational Diffie-Hellman (CDH) assumption [5] is hard relative to G if for every PPT adversary A there exists a negligible function $negl(\lambda)$ where λ is the security parameter, such that:

$$\Pr[A(G, q, g, h_1, h_2) = g^{r_1 \cdot r_2}] = negl(\lambda)$$

4 Construction

Inonymous consists of six algorithms: *SetUp, Token generation (Tgen), Invitation generation (Igen), Invitation collection (Icoll), Invitation Verification (Ivrfy)* and *Registration (Reg)*. Figure 2 visualizes the interaction of entities and the order of execution of algorithms in *Inonymous*. The *server* runs the *SetUp* algorithm to set the system's parameters. Every new user (invitee) must obtain a token from the server. To generate a token, the *server* runs *Tgen* algorithm. The invitee receives the token and delivers to his inviters. Tokens are used in the invitation generation and tie each invitation to its invitee (i.e., the invitations issued using different tokens cannot be used interchangeably). This makes sure that invitees cannot cheat by combining invitations that are for different purposes. Provided a token, an *inviter* executes *Igen* algorithm to make an invitation. The *invitee* aggregates the individual invitations by running *Icoll* algorithm and delivers a unified invitation letter to the server. If the *server* authenticates the invitations by running *Ivrfy* algorithm, then the user is allowed to join the system. The *server* generates data necessary for the new member to be able to invite others by executing the *Reg* algorithm. Thenceforth, the invitee who is now a new member is able to add other users to the system. Detailed descriptions of the algorithms follow (the entity running the algorithm is indicated inside square braces).

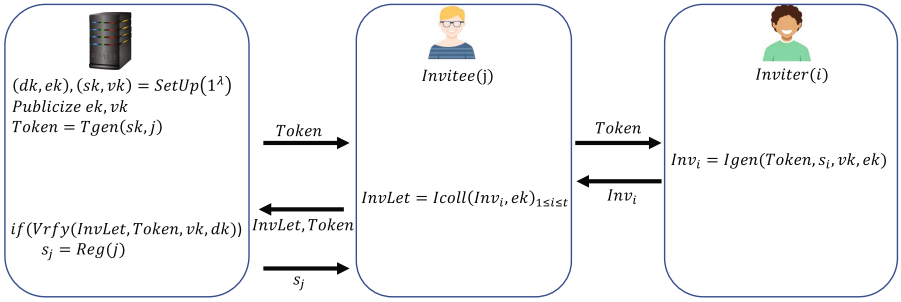

Fig. 2. The order of algorithms' execution in *Inonymous*.

SetUp(1^λ). This algorithm is run by the server who inputs the security parameter λ and generates system parameters as follows.

- Two big primes p and q such that $q|p-1$.
- g is a generator of a cyclic subgroup G of order q in Z_p^*. Let $h \in Z_p$ and $h \neq 0$, then g satisfies $g = h^{\frac{p-1}{q}} \bmod p$.
- El Gamal encryption scheme $\pi = (EGen, Enc, Dec)$ with the key pair (ek, dk) denoting encryption key and decryption key, respectively. dk remains at the server while ek is published publicly.
- A signature scheme $\gamma = (SGen, Sign, SVrfy)$. The signature and verification keys (sk, vk) are generated according to $SGen$. vk is publicized.
- A pseudo random generator $PRG{:}\{0,1\}^\lambda \rightarrow Z_q$
- A master value $S \leftarrow Z_q$
- A randomly chosen polynomial function $F(y) = f_{t-1}y^{t-1} + ... + f_1 y + f_0$ of degree $t-1$ whose coefficients $f_1, ..., f_{t-1}$ belong to Z_q and $f_0 = S$.

We assume that there are (at least) t users initially registered in the system. Each registered user is known by a unique numerical index i. Each member has the evaluation of function F on his own index i.e. the i^{th} member is given master share $s_i = F(i)$.

Token Generation: A new user who tends to join the system, initially must connect to the server and obtain an index and the corresponding token. For this sake, the server executes the token generation algorithm shown in Algorithm 4.1. In this procedure, the server assigns the user a unique index and a corresponding token. Indices can simply be given to the users sequentially based on their arrival order hence the j^{th} coming user receives the index value of j. Then, the server computes a token as $\omega = g^r$ (line 2) where r is a randomly selected value (line 1). The server certifies the association of user index and ω by generating a signature on their concatenation (line 3). The server's generated certificate constitutes the user's token (line 4). Observe that the server need not remember any information regarding a registration attempt. Thus, generated tokens can simply be discarded and only the last value of j (the number of token requests) need to be remembered, *not* incurring any storage load on the server per token.

Algorithm 4.1. Tgen [Server]

Input: sk, j
Output: $Token$

1 $r \leftarrow Z_q$
2 $\omega = g^r$
3 $\eta = Sign_{sk}(j||\omega)$
4 $Token = (\eta, j, \omega)$

Invitation Generation: The user, after obtaining his token $Token$, asks his inviters to issue an invitation letter. Each inviter uses his master share s_i to

compute an invitation letter as indicated in Algorithm 4.2. Firstly, the referee authenticates the token (line 1). Then, he computes a masked version of his master share as given in lines 2–4. δ_i is the masking value which is the output of the PRG (line 3). He also encrypts the masking value under the server's public encryption key (line 5). Note that the invitation letter is tied to the token as τ_i is the combination of the token and the inviter's master share.

Algorithm 4.2. Igen [Inviter]

Input: $Token$, s_i, vk, ek
Output: Inv_i

1 **if** $Svrfy_{vk}(\eta, j||\omega)$=$accept$ **then**
2 $\quad r \leftarrow \{0,1\}^{\lambda}$
3 $\quad \delta_i = PRG(r)$
4 $\quad \tau_i = \omega^{s_i+\delta_i}$
5 $\quad e\delta_i = Enc_{ek}(\omega^{\delta_i})$
6 $\quad Inv_i = (\tau_i, e\delta_i)$

Invitation Collection: Invitation Collection ($Icoll$) is run by the new user i.e., invitee, once he obtains a set of t individual invitations. He computes the final invitation letter i.e. $InvLet$ as indicated in Algorithm 4.3. The final invitation letter is indeed the aggregation of the individual invitations Inv_i (line 3–4). For aggregation, we benefit from the homomorphic property of both Shamir shares and encryption scheme under addition and multiplication operations, respectively. The invitee re-masks the aggregated invitation letter by contributing to the sum of masking values with another randomness i.e., δ^*. The re-masking is required to cancel out the effect of Lagrange coefficients and make the final masking value independent of B_i values. Recall that the Lagrange coefficients are dependent on the inviters' indices. This way, we achieve inviter anonymity.

We expand lines 3 and 4 of Algorithm 4.3 in Eqs. 2 and 3, respectively, where B_i values are the Lagrange coefficients computed with respect to the index of the i^{th} inviter. As indicated, T is composed of the token ω and a masked version of master value i.e., $S + \Delta$. $e\Delta$ constitutes the encryption of masking value Δ. The encryption of masking value i.e., $e\Delta$ would be required at the server for the verification purpose (see invitation verification).

$$T = \omega^{\delta^*} \cdot \prod_{i=1}^{t} \tau_i^{B_i} = \omega^{\delta^*} \cdot \prod_{i=1}^{t} \omega^{B_i \cdot s_i + B_i \cdot \delta_i} = \omega^{\delta^* + \sum_{i=1}^{t} B_i \cdot s_i + \sum_{i=1}^{t} B_i \cdot \delta_i} = \omega^{S + \delta^* + \sum_{i=1}^{t} B_i \cdot \delta_i} = \omega^{S+\Delta} \tag{2}$$

$$e\Delta = Enc_{ek}(\omega^{\delta^*}).\prod_{i=1}^{t} e\delta_i^{B_i} = Enc_{ek}(\omega^{\delta^*}).\prod_{i=1}^{t} Enc_{ek}(\omega^{B_i \cdot \delta_i}) = Enc_{ek}(\omega^{\delta^* + \sum_{i=1}^{t} B_i \cdot \delta_i}) = Enc_{ek}(\omega^{\Delta}) \tag{3}$$

Algorithm 4.3. Icoll [Invitee]

Input: $\{Inv_i = (\tau_i, e\delta_i) | 1 \leq i \leq t\}$, ek
Output: $InvLet$

1 $r \leftarrow \{0,1\}^{\lambda}$
2 $\delta^* = PRG(r)$
3 $T = \omega^{\delta^*} \cdot \prod_{i=1}^{t} \tau_i^{B_i}$
4 $e\Delta = Enc_{ek}(\omega^{\delta^*}) \cdot \prod_{i=1}^{t} e\delta_i^{B_i}$
5 $InvLet = (T, e\Delta)$

Invitation Verification: This protocol (shown in Algorithm 4.4) is invoked by the server to verify the validity of a new user's invitation letter $InvLet$ corresponding to a $Token$. First, the token is authenticated (line 1). When the authentication phase passed, the server checks the validity of the invitation letter. He first decrypts the masking value (line 2) and checks whether the invitations are issued by existing members and really intended for the new user (line 3). If all the invitations are generated correctly then the verification of line 3 will be accepted (the correctness results from Eqs. 2 and 3). If all the verification steps passed successfully, then the server accepts the user's membership request.

Algorithm 4.4. IVrfy [Server]

Input: $InvLet = (T, Delta), Token = (\eta, j, \omega), vk, dk$
Output: $reject/accept$

1 **if** $Svrfy_{vk}(\eta, j||\omega) = accept$ **then**
2 $\quad \omega^{\Delta} = Dec_{dk}(e\Delta)$
3 $\quad$ **if** $\omega^{S} \cdot \omega^{\Delta} = T$ **then**
4 $\quad\quad$ return accept

Registration: When a user passes the verification phase, the server runs the registration algorithm given in Algorithm 4.5 to issue the new member's master share s_j. s_j is the evaluation of function F on the point j that was in the user's token (line 1). Hereinafter, the new user is able to invite other users to the system.

Algorithm 4.5. Reg [Server]

Input: j
Output: s_j

1 $s_j = F(j)$

5 *InonymouX*: Anonymous Cross Network Invitation-Based System

Consider the situation where one system e.g. Twitter offers a special service for users of another system e.g. Facebook. You may assume other scenarios as well.

We name Twitter as the *host network* i.e. the network serving a special service whereas Facebook is called the *guest network* whose users will benefit from the services offered by the host network. A user of the guest network is served by the host network by convincing the host server on being invited by adequate inviters from the guest network. To do so, one simple but cumbersome solution is to follow the regular invitation-based system i.e., each time a guest user wants to join the host network, the guest server authenticates that user and communicates the authentication result to the host server. However, this solution requires two servers keep in contact with each other and imposes unnecessary overhead on the guest server. Whereas in our proposal i.e. $InonymouX$, we provide an efficient solution for cross network invitations which is independent of two servers interaction. in $InonymouX$, the host server is given enough information to authenticate guest users by his own. The solution is as follows.

The guest network with the master value S_{guest} publicizes $g^{S_{guest}}$ alongside the signature verification key vk_{guest}. On the other side, the host network announces an encryption key denoted by ek_{host}. Members of the guest network proceed as in the regular invitation procedure where the inviters use the encryption key of host network to encrypt their masking values. Indeed, in Algorithms 4.2 and 4.3, the inviter uses ek_{host} as input. Therefore, the invitation letters received by the host server are of the form $InvLet = (T, e\Delta)$ where $e\Delta$ is an encrypted masking value under ek_{host}. The host server runs a different verification routine, which is given in Algorithm 5.1. We assume the existence of a bilinear map e: $G \times G \rightarrow G_2$ where G and G_2 are multiplicative groups of prime order q. The only difference between Algorithm 5.1 and Algorithm 4.4 is at the second verification step i.e., line 3. The correctness holds by the bilinearity of the bilinear map e, as in Eq. 4.

$$\begin{aligned} e(\omega, g^{S_{guest}}) \cdot e(\omega^{\Delta}, g) &= e(\omega, g)^{S_{guest}} \cdot e(\omega, g)^{\Delta} = e(w, g)^{S_{guest}+\Delta} = e(w^{S_{guest}+\Delta}, g) \\ &= e(T, g) \end{aligned} \tag{4}$$

Algorithm 5.1. XIVerify [Host Server]

Input: $InvLet = (T, e\Delta), Token = (\eta, j, \omega), vk_{guest}, dk_{host}, g^{S_{guest}}$
Output: *reject*/*accept*

1 **if** $Svrfy_{vk_{guest}}(\eta, j||\omega)$=*accept* **then**
2 | $\omega^{\Delta} = Dec_{dk_{host}}(e\Delta)$
3 | **if** $e(\omega, g^{S_{guest}}) \cdot e(\omega^{\Delta}, g) = e(T, g)$ **then**
4 | | return accept

6 Performance

In this section we aim at analyzing the running time of each algorithm. Table 1 shows the results in millisecond. We used DSA signature scheme [7] with the key size of 1024 bits. The required number of inviters i.e., t is set to 5. The running time is measured on a standard laptop with 8 GB 1600 MHz DDR3 memory

Table 1. Running time of *Inonymous* Algorithms.

SetUp	Tgen	Igen	Icoll	IVerify	Reg
842 ms	3.46 ms	37.4 ms	35.3 ms	29.4 ms	0.129 ms

and 1.6 GHz Intel Core i5 CPU. Although the running time of SetUp algorithm has a huge difference with the other algorithms, it is run only once by the server just to bootstrap the system.

7 Security

In this section, we provide security definitions for inviter anonymity and invitation unforgeability, and then prove the security of *Inonymous*.

7.1 Inviter Anonymity

Security Definition: An invitation-based system protects inviter anonymity if a new user having enough inviters can convince the server without revealing the identity of his inviters. We model this security objective as a game denoted by $InvAnonym_A(\lambda)$ played between a challenger and an adversary. The members controlled by the adversary and challenger are called colluding and non-colluding members, respectively. The adversary controls the server as well. The challenger acts as a new user who wants to join the system and is required to obtain t invitations from t different members. We assume that $t-1$ inviters come from the colluding members (clearly it is the maximum power that can be considered for the adversary). The adversary selects two non-colluding members. The challenger uses one of them as the remaining inviter and registers into the system. If the adversary cannot guess the identity of the non-colluding inviter with more than a negligible advantage, then the system provides inviter anonymity.

Inviter Anonymity Experiment $InvAnonym_A(\lambda)$

1. The adversary outputs encryption key ek and signature verification key vk.
2. The challenger registers polynomially many users denoted by U to the system.
3. The adversary selects two users $u_0, u_1 \in U$ and generates a token $Token$. Also, the adversary outputs $t-1$ individual invitations for the given $Token$.
4. The challenger tosses a coin and selects a bit value b accordingly. Then, the challenger generates an invitation letter $InvLet$ using u_b as one of the inviters in addition to the $t-1$ invitations received from the adversary, and sends $InvLet$ to the adversary.
5. The adversary guesses a bit $b^{'}$ indicating that which of the two users u_0, u_1 is used as the inviter.
6. The output of game is 1 if $b == b^{'}$, 0 otherwise.

Definition 1. *An invitation-based system has inviter anonymity if for every probabilistic polynomial time adversary A there exists a negligible function negl(.) such that:*

$$Pr[InvAnonym_A(\lambda) = 1] = \frac{1}{2} + negl(\lambda)$$

Security Proof: Before the formal proof, let us summarize informally. In *Inonymous*, the anonymity of inviter relies on the security of the pseudo random generator. The invited user delivers to the server (the adversary) an invitation letter of the form $InvLet = (\omega^{S+\Delta}, e\Delta)$ where S is the server's master value, $e\Delta$ is the encryption of Δ and $\Delta = \omega^{\delta^* + \sum_{i=1}^{t} B_i \cdot \delta_i}$ (δ^* is the masking value added by the invitee, δ_i is inviter's masking value resulted from a *PRG* and B_i is the Lagrange coefficient computed based on the inviter's index). The adversary may get some idea about the inviters' identity by extracting the Lagrange coefficients from Δ value (Lagrange coefficients are the function of inviters' indices). Two cases may occur. If the random values δ_i and δ^* are selected truly at random, then we know that Δ is also a random value and conveys nothing about the Lagrange coefficient B_i. Though, if δ_i and δ^* are the output of a *PRG* then the adversary may have advantages to extract the Lagrange coefficients. We denote the adversary's advantage by ϵ. If ϵ is non-negligible, it implies that we can distinguish between a *PRG* and a random number generator hence we break the security of the *PRG*. In the following we provide the formal proof.

Theorem 1. *If PRG is a pseudo random generator then Inonymous provides inviter anonymity.*

We reduce the security of *Inonymous* to the security of the employed *PRG*. If there exists a PPT adversary A who breaks the inviter anonymity of *Inonymous* with non-negligible advantage then we can construct a PPT adversary B who distinguishes between a random generator and a pseudo random generator with the same advantage of A. Assume A's success probability is

$$Pr[InvAnonym_A(\lambda) = 1] = \frac{1}{2} + \epsilon(\lambda) \tag{5}$$

B runs A as its subroutine to distinguish the pseudo random number generator from the truly random generator. B is given a vector of values in Z_q denoted by $\vec{\delta} = (\delta', \delta'')$ and aims at specifying whether $\vec{\delta}$ is selected truly at random or is the output of a *PRG*. B invokes A as his subroutine and emulates the game of inviter anonymity for A as follows. If A succeeds then B realizes that $\vec{\delta}$ is pseudo random, otherwise random.

1. B is given the security parameter λ and a vector of two values denoted by $\vec{\delta} = (\delta', \delta'')$ where $\delta', \delta'' \in Z_q$. Adversary A outputs the encryption and signature public keys ek and vk, respectively.

2. B registers polynomially many users into the system. U indicates the set of indices registered by B.
3. A outputs two users $u_0, u_1 \in U$ and a token $Token = (\eta, u^*, \omega)$. u^* is the index of the new user. A also submits $t-1$ invitation letters i.e. $Inv_i = (\tau_i, e\delta_i)$ for $1 \leq i \leq t-1$.
4. B selects a random bit b and creates an invitation letter from u_b as $Inv_{u_b} = (\tau_{u_b}, e\delta_{u_b}) = (\omega^{s_{u_b}+\delta^{'}}, Enc_{ek}(\omega^{\delta^{'}}))$. He finally computes
$T = \omega^{\delta^{''}} \cdot \tau_{u_b}^{B_{u_b}} \cdot \prod_{i=1}^{t-1} \tau_i^{B_i}$
and
$e\Delta = Enc_{ek}(\omega^{\delta^{''}}) \cdot Enc_{ek}(\omega^{\delta^{'}})^{B_{u_b}} \cdot \prod_{i=1}^{t-1} e\delta_i^{B_i} = Enc_{ek}(\omega^{\delta^{''}+\delta^{'} \cdot B_{u_b} + \sum_{i=1}^{t-1} \delta_i \cdot B_i})$.
B_i and B_{u_b} denote the Lagrange coefficients. B submits $InvLet = (T, e\Delta)$ to the adversary A.
5. A outputs a bit $b^{'}$.
6. If $b = b^{'}$ then B outputs 0, otherwise 1.

Let $\vec{\delta}$ be a truly random vector. Once the adversary decrypts $e\Delta$ he obtains

$$\Delta = \omega^{\delta^{''}+\Gamma}$$

where

$$\Gamma = \delta^{'} \cdot B_{u_b} + \sum_{i=1}^{t-1} \delta_i \cdot B_i$$

Γ is a function of inviters indices due to the presence of Lagrange coefficients whereas $\delta^{''}$ is a random value completely independent of inviters. If $\vec{\delta}$ is a random vector then $\delta^{''}$ is also a random value from $\mathbb{Z}_q$. Therefore, in $\omega^{\delta^{''}+\Gamma}$, Γ is indeed masked with $\delta^{''}$ ($\delta^{''} + \Gamma \mod q$ is a completely random element of $\mathbb{Z}_q$). By this masking, Δ becomes completely independent of Lagrange coefficients and A has no advantage to infer the inviters identity. Thus, A's advantage is exactly $\frac{1}{2}$ i.e.,

$$Pr[B(\vec{\delta} \leftarrow Z_q) = 1] = Pr[b = b^{'}] = \frac{1}{2} \tag{6}$$

but if δ is the output of a PRG then

$$Pr[r \leftarrow \{0,1\}^{\lambda} : B(\vec{\delta} = PRG(r)) = 1] = Pr[b = b^{'}] = \frac{1}{2} + \epsilon(\lambda) \tag{7}$$

where $\frac{1}{2} + \epsilon(\lambda)$ is the success probability of A as assumed in our proof in Eq. 5. By combining Eqs. 6 and 7 we have

$$|Pr[r \leftarrow \{0,1\}^{\lambda} : B(\vec{\delta} = PRG(r)) = 1] - Pr[B(\vec{\delta} \leftarrow Z_q) = 1]| = \epsilon(\lambda) \tag{8}$$

Equation 8 corresponds to the security definition of PRG (see Eq. 1). Thus, if $\epsilon(\lambda)$ is non-negligible then the distinguisher B can distinguish a PRG from a random generator which contradicts with the security definition of PRG. Therefore, $\epsilon(\lambda)$ must be negligible according to the PRG definition. This concludes the security proof of inviter anonymity of *Inonymous*.

Discussion: We proved inviter anonymity against an honest but curious adversary who follows the algorithm descriptions, whereas a malicious adversary breaks the anonymity of the inviters in the following attack scenario. First note that for the inviter anonymity, the adversary is the server who is colluding with $t-1$ inviters. According to the inviter anonymity game definition, the server obtains $InvLet = (T = \omega^{S+\Delta}, e\Delta)$. As we discussed, if all the inviters act honestly and use their real master shares for the invitation generation, then the adversary obtains the w^S value. According to Shamir secret sharing scheme, even if the adversary knows $t-1$ inviters, the remaining inviter can be any of the existing shareholders, hence the inviter anonymity holds. Now, consider that the colluding $t-1$ inviters put zeros instead of their real master shares i.e., $s_1 = ... = s_{t-1} = 0$ (wlog. $1, ..., t-1$ are the indices of colluding inviters). Then, the server obtains $w^{S'}$ with the following value: $S' = s_1.B_1 + ... + s_{t-1}.B_{t-1} + s_t.B_t = s_t.B_t$. The adversary can simply try all the combinations of generated master shares s_t with different possible values for B_t and figure out the honest inviter's index (the possible number of values is linear in the number of remaining inviters, which in the inviter anonymity game is 2, and in practice corresponds to the number of registered users). This attack is defeated only if the inviters act honestly and follow the genuine routine of algorithms.

7.2 Invitation Unforgeability

Security Definition: In an invitation based system, the invitation unforgeability indicates that people who do not have enough inviters ($<t$) should not be able to join the system. Hence, no adversary can forge invitations by his own. We define the following game denoted by $InvUnforge_A(\lambda)$ running between a challenger and an adversary. The adversary controls a set of $t-1$ members denoted by Q. The adversary may query as many tokens as he wants and queries the challenger to check the validity of his pseudo-invitation letters. Finally, if the adversary registers to the system successfully, it shows that the invitations are forgeable, otherwise the system has invitation unforgeability. Note that we assume secure authenticated channels between entities hence we ignore the threat of eavesdropping.

Invitation Unforgeability experiment $InvUnforge_A(\lambda)$:

1. The challenger runs the setup algorithm. The adversary is given the encryption key ek, the signature verification key vk, as well as the security parameter λ.
2. The adversary registers a set of $t-1$ users denoted by Q.
3. The adversary asks the challenger to issue a token. The challenger generates a token for the next available index j. This step may be repeated polynomially many times upon the adversary's request.
4. The adversary queries invitation verification function on the invitations of his own choice. The challenger responds accordingly.
5. The challenger outputs a token denoted by $Token^*$. The adversary outputs an invitation letter $InvLet$ corresponding to the given token.
6. If the output of $IVrfy(InvLet, Token^*, vk, dk)$ is accepted then the game's output is 1 indicating the adversary's success, 0 otherwise.

Definition 2. *An invitation-based system has invitation unforgeability if for every probabilistic polynomial time adversary A there exists a negligible function negl(.) such that:*

$$Pr[InvUnforge_A(\lambda) = 1] = negl(\lambda)$$

Security Proof. Forging invitations is information-theoretically infeasible. In fact, the adversary, in the best case, has $t-1$ inviters hence $t-1$ evaluations of function $F(.)$. Recall that a valid invitation letter contains the master value S i.e. $\omega^{S+\Delta}$ (S is the evaluation of $F(0)$). Since the adversary is confined to $t-1$ points on this polynomial, he cannot reconstruct the S value in any way. Without the final share of S, $\omega^{S+\Delta}$ is a random element. Therefore, *Inonymous* has information-theoretic invitation unforgeability.

7.3 Security of *InonymouX*

InonymouX provides inviter anonymity as *Inonymous* does, hence the same proof of Sect. 7.1 applies here. However, invitation unforgeability needs a different proof, due to the publicity of $g^{S_{guest}}$, which provides computational information to the adversary. Note that a final aggregated invitation letter has the form of $T = \omega^{S_{guest}+\Delta} = g^{r \cdot S_{guest} + r \cdot \Delta}$. For an adversary who has ω and $g^{S_{guest}}$, making a valid invitation letter corresponds to solving the Computational Diffie-Hellman (CDH) problem i.e., given $g^{S_{guest}}$ and g^r compute $g^{r \cdot S_{guest}}$. Thus, assuming that CDH is hard to solve, then the invitation unforgeability holds for *InonymouX* as well. Full reduction follows.

Security Proof

Theorem 2. *If the computational Diffie-Hellman problem is hard to solve relative to G, then InonymouX has invitation unforgeability.*

If there exists a PPT adversary A who breaks the invitation unforgeability with probability $\epsilon(\lambda)$, then we construct an adversary B who solves CDH problem with $\epsilon(\lambda)$ probability. The adversary A acts as a new user with t-1 inviters. The adversary B plays as host and guest servers and the rest of members. B simulates the invitation unforgeability game for adversary A to break CDH problem.

1. B is given (G, q, g, g^r, g^S) (G is a cyclic group of order q with generator g) and the security parameter λ. B generates signature key pair vk, sk and public encryption key pair ek, dk. B sends g^S, vk and ek to A. Note that S is the master value which is unknown to B.
2. Adversary A registers $t-1$ users to the system. As B does not know the master value S, he generates random master shares for A's requests. Adversary A cannot distinguish between the real master shares and the randomly generated ones. Note that for every given $t-1$ points, we can construct a polynomial F of degree t such that $F(0) = S$. Thus, as long as the adversary A has only $t-1$ points of F, he does not distinguish whether B knows the master value (i.e., has generated real master shares) or is a simulator.
3. The adversary A asks for polynomially many tokens from B.
4. The adversary A outputs an invitation letter and asks B to verify it. A may repeat invitation verification query polynomially many times. B runs Algorithm 6 to answer the queries.
5. The challenger outputs $Token^*$ as (η, adv, w) where adv is an index, $\eta = Sign_{sk}(adv||w)$ and $w = g^r$ (B sets w as one of the inputs given in the CDH game). A outputs an invitation letter:

$$InvLet = (T = w^{S+\Delta}, e\Delta = Enc_{ek}(\omega^{\Delta})).$$

If $XIvrfy(InvLet, Token^*) = accept$ then B computes $w^{\Delta} = Dec_{dk}(e\Delta)$. As the solution for CDH problem, B outputs

$$T.(w^{\Delta})^{-1} = w^{S+\Delta}.w^{-\Delta} = w^S = g^{r.S} \tag{9}$$

If A manages to output a valid invitation letter $InvLet$, then B can extract the solution of CDH problem from that invitation letter as indicated in Eq. 9. Therefore,

$$\Pr[\text{B breaks CDH}] = \Pr[\text{A breaks Invitation Unforgeability}] = \epsilon(\lambda)$$

If $\epsilon(\lambda)$ is non-negligible then CDH problem is also solved with non-negligible probability. This implies a contradiction for the hardness assumption of CDH, hence we conclude that $\epsilon(\lambda)$ must be negligible. Therefore, invitation unforgeability of $InonymouX$ is proven.

8 Conclusion

We proposed *Inonymous*, an anonymous invitation-based system by seeking two security objectives i.e., *inviter anonymity* against the system administrator and

existing members and *invitation unforgeability* against newcomers with insufficient inviters. We present security definition and formal proof for each security objective. The anonymity of inviter relies on the security of the employed pseudo random generator (for masking value generation) and the invitation unforgeability is information-theoretically proven due to the Shamir secret sharing scheme's security. We also proposed $InonymouX$, an anonymous cross-network invitation-based system by a slight modification on *Inonymous* so that users of one network can act as inviters for another network. $InonymouX$ invitation unforgeability assumes Computational Diffie-Hellman problem. In the future, we aim to provide efficient user revocation capability as well as inviter anonymity against malicious adversaries.

Acknowledgements. We acknowledge the support of the Royal Society of UK Newton Advanced Fellowship NA140464 and European Union COST Action IC1306.

References

1. http://www.macworld.com/article/1055383/gmail.html
2. Bogdanov, D.: Foundations and properties of Shamir's secret sharing scheme research seminar in cryptography. University of Tartu, Institute of Computer Science, 1 May 2007
3. Chaabane, A., Acs, G., Kaafar, M.A., et al.: You are what you like! information leakage through users interests. In: Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS) (2012)
4. Gong, N.Z., Wang, D.: On the security of trustee-based social authentications. IEEE Trans. Inf. Forensics Secur. **9**(8), 1251–1263 (2014)
5. Gu, K., Jia, W., Chen, R., Liu, X.: Secure and efficient proxy signature scheme in the standard model. Chin. J. Electron. **22**(4), 666–670 (2013)
6. Harn, L., Lin, C.: Authenticated group key transfer protocol based on secret sharing. IEEE Trans. Comput. **59**(6), 842–846 (2010)
7. Kravitz, D.W.: Digital signature algorithm. US Patent 5,231,668, 27 July 1993
8. Mahmood, S.: Online social networks: privacy threats and defenses. In: Chbeir, R., Al Bouna, B. (eds.) Security and Privacy Preserving in Social Networks, pp. 47–71. Springer, Vienna (2013)
9. Malar, G.P., Shyni, C.E.: Facebook's trustee based social authentication
10. Parameswari, S.M., Sukumaran, S.: Trustee based authentication mechanism for social network. Int. J. Latest Res. Sci. Technol. **4**, 84–88 (2015)
11. Rao, F.-Y.: On the security of a variant of ELGamal encryption scheme. IEEE Trans. Dependable Secure Comput. (2017)
12. Roy, A., Karforma, S.: A survey on digital signatures and its applications. J. Comput. Inf. Technol. **3**(1), 45–69 (2012)
13. Sharimila, K., Janaki, V., Nagaraju, A.: Enhanced user authentication techniques using the fourth factor "some body the user knows". In: Proceedings of International Conference on Advances in Computer Science, AETACS. Elsevier (2013)
14. Yu, J., Kong, F., Cheng, X., Hao, R., Li, G.: One forward-secure signature scheme using bilinear maps and its applications. Inf. Sci. **279**, 60–76 (2014)