# Pseudo Trust: Zero-Knowledge Based Authentication in Anonymous Peer-to-Peer Protocols

Li Lu[1], Jinsong Han[2], Lei Hu[1], Jinpeng Huai[3, 4], Yunhao Liu[2], and Lionel M. Ni[2]

[1]State Key Lab of Information Security
Graduate School of Chinese Academy of Sciences
Beijing, China
{luli, hu}@is.ac.cn

[2]Dept. of Computer Science and Engineering
Hong Kong University of Science and Technology
Kowloon, Hong Kong
{jasonhan, liu, ni}@cse.ust.hk

[3]School of Computer Science
Beihang University
Beijing, China
huaijp@buaa.edu.cn

[4]State Key Lab of Software Developing Environment
Beihang University
Beijing, China
huaijp@buaa.edu.cn

## Abstract

*Most of the current trust models in peer-to-peer (P2P) systems are identity based, which means that in order for one peer to trust another, it needs to know the other peer's identity. Hence, there exists an inherent tradeoff between trust and anonymity. To the best of our knowledge, there is currently no P2P protocol that provides complete mutual anonymity as well as authentication and trust management. We propose a zero-knowledge authentication scheme called Pseudo Trust (PT), where each peer, instead of using its real identity, generates an unforgeable and verifiable pseudonym using a one-way hash function. A novel authentication scheme based on Zero-Knowledge Proof is designed so peers can be authenticated without leaking any sensitive information. With the help of PT, most existing identity-based trust management schemes become applicable in mutual anonymous P2P systems. We analyze the levels of security and anonymity in PT, and evaluate its performance using trace-driven simulations and a prototype implementation. The strengths of Pseudo Trust include the lack of need for a centralized trusted party or CA, high scalability and security, low traffic and cryptography processing overheads, and man-in-middle attack resistance. We aim for the Pseudo Trust design to be included in the P2P trust and anonymity context.*

1-4244-0910-1/07/$20.00 ©2007 IEEE.

## 1. Introduction

As an emerging model of communication and computation, peer-to-peer (P2P) networking has recently gained significant acceptance. Millions of users share huge amounts of resources by forming an abstract, logical network called an overlay network. Most widely-deployed P2P systems today, including Gnutella, KaZaA, and BitTorrent, employ a routed-search-and-direct-download mechanism. Peers are linked in the overlay network, each maintaining several logical neighbors. Query flooding is the most popular search method in such systems. If a peer receiving a query can provide the requested object, a response message is sent back to the requesting peer, and a direct download path is constructed between the downloader and the content provider.

One drawback of the above protocols is the fact that such P2P systems might compromise users' privacy. The IP addresses of object requesters and providers can easily be discovered and translated into users' names and postal addresses. Hence, many studies such as the Peer-to-Peer Personal Privacy Protocol ($P^5$) [22] and Anonymous Peer-to-Peer File Sharing (APFS) [20] focus on providing anonymous searching and downloading in P2P systems.

On the other hand, numerous concerns have been raised about the issue of providing authentic resources in P2P systems. To guarantee that real resources are received from authentic responders, some researchers have built trust models to help peers verify the validity of other entities [5, 6, 11, 13]. However, most trust models are identity-based, which means that for one peer to trust another, it needs to know the identity of the other peer. Thus, there exists an inherent tradeoff between trust and anonymity in P2P systems. To the best of our knowledge, there is no existing P2P protocol that provides mutual anonymity as well as trust management.

The purpose of designing an anonymous authentication protocol in P2P systems is motivated by a specific problem: how to support authentication without exposing the real identities of peers. In this paper, we propose the design of the Pseudo Trust (PT) protocol, in which each peer generates an unforgeable and verifiable pseudonym using a one-way hash function. Such one-way mapping can effectively defend against impersonation, forgery, and Man-In-Middle-Attacks, so that the pseudonyms can be used as the real IDs in P2P systems. This means that previous methods of identity-based trust management can be adopted. We also design a novel authentication scheme based on Zero-Knowledge Proof to help unfamiliar peers successfully complete authentication procedures during transactions. Thus, trust management can be pseudonym-based so that the real identities of peers are protected and users are able to verify each other without leaking any sensitive or private information. The salient features of Pseudo Trust include (1) achieving anonymity as well as authentication, (2) eliminating the support of a centralized CA system, and (3) resisting man-in-middle-attacks.

We discuss the implementation choices that were made for security and efficiency reasons, and conduct trace-driven simulations to evaluate the parameter selections and the performance of our protocol. We also implement a Pseudo Trust prototype within a 50-machine overlay in the Internet. Both our theoretical analyses and experimental results show that Pseudo Trust is effective, scalable, and completely decentralized with no need of a central CA. We believe the Pseudo Trust design should be included in peer-to-peer trust and anonymity contexts.

The rest of this paper is organized as follows. Section 2 introduces related works including trust management schemes, anonymous P2P protocols, and Zero-Knowledge Proof. Section 3 presents the PT design. Section 4 analyzes the anonymity and security degree of PT. Section 5 presents our trace driven simulations and the performance evaluation of the design. We introduce the PT prototype implementation and experimental results in Section 6, and conclude this work in Section 7.

## 2. Related Work

In this section, we briefly describe related works in authentication, anonymity, and Zero-Knowledge Proof.

### 2.1 P2P trust and authentication

A number of approaches have been proposed to provide trust and reliable authentication in the P2P systems. XREP [6] enables peers to evaluate and share other peer reputations by introducing a distributed polling algorithm. This study also employs confirmation voting procedures among randomly chosen peers in order to resist collusive cheating from cliques of malicious peers. P-Grid [2] utilizes on an efficient data management technique to construct a scalable trust model for decentralized applications. EigenTrust [11] builds a virtual global matrix to represent individual reputations. NICE [14] provides a platform to implement distributed cooperative applications. Based on trust chains, NICE computes a user reputation in a PGP-like model. By employing an asymmetric cryptographic algorithm, it requires peers to encrypt cookies to help others compute their reputations. Indeed, most P2P trust designs are identity-based, where one peer does not trust another before knowing its identity.

### 2.2 Anonymity

Privacy has becomes an increasingly salient issue, and considerable progress has been made with anonymous communications [20, 22, 25]. Several solutions achieve mutual anonymity for both initiators and responders in P2P systems, which generally aim to conceal the real identities of users during transactions. For example, in APFS [20], peers construct an anonymous path with tail nodes using an onion technique [25], providing complete and mutual anonymity for peers. Recent research has attempted to introduce reputation value into anonymous P2P systems [24], or construct a trust management based on proxy techniques [16]. However, failure to support authentication makes these approaches vulnerable to impersonation and man-in-middle attacks. Therefore, it is argued that increasing the privacy of peers increases the difficulties of ensuring authenticity and security. Obviously, there is a clear tradeoff between authentication and anonymity.

### 2.3 Zero-knowledge proof

The purpose of Zero-Knowledge Proof (ZKP) protocols is to help a prover convince a verifier that she holds some knowledge (usually secret), without leaking any information about this knowledge during the verification process (Zero-knowledge). The concept of ZKP was first introduced by Goldwasser et al. in [10], and has been employed in many authentication and identification protocols. Loosely speaking, a ZKP is an

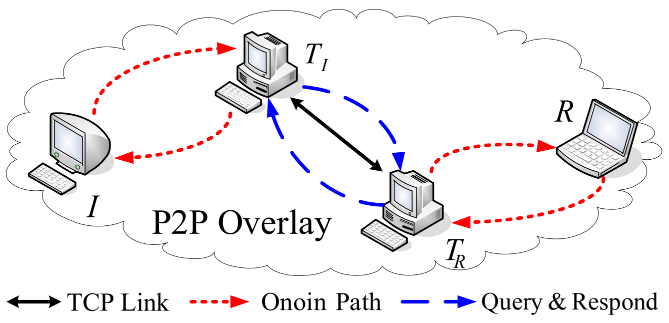

**Figure 1. PT query-downloading and authentication.**

interactive proof system which comprises a prover and a verifier. The principle rule is that the prover demonstrates knowledge of a secret to the verifier through several interactive rounds. During the process, the prover does not reveal any sensitive information of the secret to the verifier or any other parties. Each round involves a challenge (say, a question) from the verifier, and a response (say, an answer) from the prover. If the secrets are related to user identities, ZKP can be used for identification and, in this case, is called Zero-Knowledge Proof of Identity (ZKPI). The security of ZKPI protocols is often based on the intractability of factoring large integers [10, 19] or computing a discrete logarithm problem [3]. Some of them have been improved to employ mutual authentication and key exchanges [4]. However, since almost all ZKP-based identification schemes [12, 18, 23] depend on a trusted third party (such as a CA) as an authorized central server, they are not directly adopted by this design.

## 3. Pseudo Trust

The real and specific challenge that underlies the tradeoff between trust and anonymity is that on one hand, all existing P2P trust systems attempt to link each peer ID with a trust value; on the other hand, anonymous designs hide the real IDs of communicating parties during transactions. This is where our proposed Pseudo Trust design enters the picture. Instead of using their real IDs in a P2P society, can peers use pseudonyms to interact with others and accumulate their reputations?

Clearly, if we attempt to adopt such a mechanism, we need to guarantee that when a peer selects a pseudonym, it is not likely to be a name already being used by another peer; and that pseudonym impersonations must be made impossible. That is, each peer is able to verify whether the other party it is communicating with is the real holder of the claimed pseudonym.

In this section, we first give an overview of the design of PT, and then discuss its three key components, including Pseudo Identity Generation and Issuance, New Peer Initialization, and Authentication and Session Key Exchange.

### 3.1 Design overview

PT is applicable under most of today's widely-deployed P2P protocols, such as Gnutella and KaZaA. For simplicity of discussion, we take a Gnutella-like P2P environment as a platform that PT runs on. In Gnutella, a requesting peer issues its queries in a flooding manner. A query is broadcast and rebroadcast until a certain criterion is satisfied. If the peer receiving the query can provide the requested object, a response message containing the IP address of the responder is sent back to the source peer along the reversed path of the query.

To protect real identities, in the PT design, each peer is required to generate a *pseudo identity* (*PI*) before joining the system. As illustrated in Fig. 1, peers construct anonymous onion paths and find tail nodes based on the APFS protocol [20]. Other selections of anonymous protocol designs are possible, but such changes are out of the scope of this discussion.

### 3.2 Pseudo identity generation and issuance

In PT, each peer is required to generate two items before joining the system: a *pseudo identity* (*PI*) and a *pseudo identity certificate* (*PIC*).

A *PI* is used to identify and replace the real identity of a peer in a P2P system. In this way, a peer does not have to expose its real identity when communicating with others. Furthermore, a peer's reputation is also coupled with its *PI* instead of its real ID. To avoid impersonation, *PIC* is generated to authenticate the *PI* holder. Terms not defined here can be found in [9].

Let $\mathrm{ID} \in \{0,1\}^*$ denote the real identity of a peer $A$ ($\{0,1\}^*$ denotes a set of binary strings). $Z_n^*$ is a multiplicative group of integers modulo $n$.

*PI and PIC Generation*: PT adopts a hash function, such as SHA-1, to generate *PI and PIC* for each peer. We slightly modify the prototype SHA-1 and name the

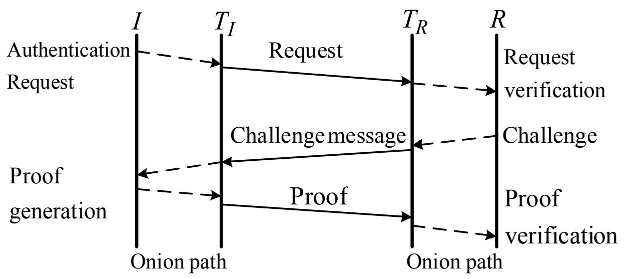

**Figure 2. Procedure of Zero-Knowledge Proof.**

revisions $h_i$ to fit the different inputs and outputs. $A$ randomly chooses two large primes $p_1$ and $p_2$, and calculates the integer $n = p_1 \times p_2$. Here, $A$ first uses $h_1 : \{0,1\}^* \times Z_n^* \times Z_n^* \to \{0,1\}^m$ to generate a *Seed*, where $m$ is the length of *Seed*, and $\{0,1\}^* \times Z_n^* \times Z_n^*$ is a Cartesian product. $A$ then computes its $Seed_A$ by:

$$Seed_A = h_1(\text{ID}, p_1, p_2) \in \{0,1\}^m$$

Having $Seed_A$, $A$ computes its $PI_A$ and $PIC_A$ as follows:

(1) Choose $k$ distinct integers, $j_1...j_k$. Compute $v_j = h_2(Seed_A, j_i, n)(\text{mod } n) \in Z_n^*$ for each small integer $j_i$, $i = 1...k$, such that $v_j$ is a quadratic residue (mod $n$), where $h_2$ is a hash function such that $h_2$:$\{0, 1\}^m \times N \times N \to Z_n^*$, and $N$ is the set of positive integers.

(2) Compute the smallest square root $s_j$ of $v_{j_i}$ mod $n$, where $i = 1...k$. Here we use $J = \{j_i\}$, $\{s_{j_i}\}^k$, $\{v_{j_i}\}^k$ to denote the sets of $j_i$, $s_{j_i}$, and $v_{j_i}$, respectively. According to the Number Theory, it is computationally infeasible to compute square roots modulo $n$ without knowing the factoring of $n$. The details can be found in §6.6 in [17].

(3) Compute $PI_A = h_3(Seed_A, n) \in \{0, 1\}^m$, where $h_3$ is a hash function such that $h_3$:$\{0, 1\}^m \times N \to \{0, 1\}^m$.

After the above operation, $A$ generates $PIC_A = \{PI_A, n, J, Seed_A\}$, and publishes its $PIC_A$ on public sites. Other peers can obtain the valid $PIC_A$ of peer $A$ from well-known sites for later verification. To protect the authenticity of $n$, we combine each $PI$ with $n$ through hash functions. Therefore, whether or not the public sites are secured, PT becomes a "counterfeit-sensitive" protocol.

## 3.3 New peer initialization

After joining the P2P system, peer $A$ constructs anonymous sessions with existing peers using the APFS [20] protocol. In this design, anonymous sessions are onion routes (*OR*) consisting of chosen peers. A tail node $T_A$ acts as an agent relaying a message for $A$. $T_A$ and other peers in this path do not know $A$ is at the end point position. In APFS, peers use a multicast technique to send the query via a tail node to some servers anonymously, which is similar to the flooding procedure used by PT. To allow $T_A$ to send messages back to $A$, $A$ constructs an *OR*:

$$OR_A = \{T_A, \{...A, \{mix\}K_A, ...\}K_{T_A}\}$$

Meanwhile, peer $A$ builds another onion path $OR_A$ for sending message to $T_A$ anonymously.

After the construction of anonymous sessions (note each node selects its tail node and builds two onion paths), each peer can anonymously issue queries for desired files. We use $I$ to denote an initiator. $I$ forwards a query $q$ for a certain requested file $f$, through $OR_I$ to its tail node $T_I$. $T_I$ then starts a flooding search in the P2P system.

When a peer receives the query and holds the requested file, it gets $I$'s credit based on the trust management mechanism to help it decide whether to act as a responder $R$ and provide the file. If it decides to provide the file, $R$ replies to this query through its tail node with a response including the IP address of its tail node and its reputation record.

$$I \xrightarrow{OR_I} T_I \xrightarrow{flooding} R : PIC_I, T_I, f, request$$

## 3.4 Authentication and session key exchange

PT employs a modified ZKP of Identification scheme. To adopt it into decentralized P2P networks, we remove the central authority servers in [8]. ZKP used in PT is based on the assumption that factoring a large integer is computationally infeasible.

When the query initiator, $I$, receives multiple responses, it selects those peers with high reputations as potential responders. Without loss of generality, suppose $R$ is selected as one of the responders. $I$ can initiate the authentication procedure to verify that the peer claiming to be the holder of $R$ is not lying. $I$ sends an authentication request to $R$ through the anonymous path, $I \to T_I \to T_R \to R$, where $I \to T_I$ and $T_R \to R$ are onion paths $OR_{T_I}$ and $OR_{T_R}$, and $T_I \to T_R$ is a TCP connection. If $R$ decides to prove that it is the holder of pseudo identity $R$, it sends the reply with its $PIC_R$ to its tail node $T_R$.

$$R \xrightarrow{OR_{T_R}} T_R : PIC_R, T_R, f, response$$

Following the APFS protocol, $T_R$ delivers the messages to $T_I$ directly through the TCP connection, and $T_I$ delivers the response to peer $I$ through $OR_{T_I}$.

$$T_I \xrightarrow{OR_{T_I}} I : PIC_R, T_R, f, response$$

Having $PIC_R$, $I$ initiates the *authentication procedure* as illustrated in Fig. 2. The authentication procedure includes two main phases: (1) $I$ acts as a prover to prove its validity to $R$, and (2) $R$ also proves its validity to $I$. These two phases are symmetrical and opposite in the proving direction. However, the order has been carefully chosen to avoid potential Dos attacks that may be launched onto $R$, which is further discussed in our full version paper [15].

To provide confidentiality and integrity to data exchanges after authentication, we embed a Diffie-Hellman Key Exchange protocol [7] into the authentication procedure to generate a session key held by $I$ and $R$ only. Three publicly known parameters $g$, $P$, and $Q$ are published by bootstrapping servers. $P$ (512 bits or longer) and $Q$ (160 bits) are prime numbers such that $Q$ divides $P$-1. The $g$, satisfying $g^Q = 1 \bmod P$, is chosen from (1, $P$-1) randomly. A detailed authentication procedure is described as follows.

(*a*) (*Authentication Request*) Before starting authentication, peer $I$ chooses two random numbers $x$ and $a$, where $x$ is a commitment for $R$ authenticating the $PI$ of $I$ in step (*f*), and $a \in [1, Q)$ is used to generate the session key. $I$ chooses $c \in (0, n_I)$ randomly and computes $x = c^2 \pmod{n_I}$, $g^a \bmod P$ (for simplicity, we denote $g^a \bmod P$ as $g^a$), and $u = h_4(x, PI_R, T_R, g^a)$. $I$ sends $\{x, u, g^a\}$ to Peer $R$, and keeps $a$ as a secret. Here, $h_4$ is a hash function $h_4 : Z_n^* \times \{0,1\}^m \times \{0,1\}^* \times Z_p^* \rightarrow \{0,1\}^k$ to generate $u$, and $u$ is $k$-bits long. We use $(u_1 \ldots u_i \ldots u_k)$, where $u_i \in \{0,1\}$, to represent $u$.

(*b*) (*Request Verification*) $R$ computes $u' = h_4(x, PI_R, T_R, g^a)$, then verifies whether $u = u'$. If the verification holds, peer $R$ goes to next step (*c*), otherwise it rejects this authentication request.

(*c*) Peer $R$ checks whether $PI_I = h_3(Seed_I, n_I)$ holds. If not, peer $R$ terminates authentication. Otherwise, peer $R$ computes $\{v_{j_i}\}^k$ of peer $I$, $v_{j_i} = h_2(Seed_I, j_i, n_I)$, $j \in J_I$, $i = 1 \ldots k$.

(*d*) (*Challenge*) $R$ sends a random binary vector $e_j = (e_{j_1}, \ldots, e_{j_k}) \in \{0,1\}^k$ to $I$.

(*e*) (*Proof generation*) Peer $I$ sends the following to peer $R$:

$$y = c(\prod_{i=1}^{k} s_{j_i}^{e_{j_i} + u_i} \pmod{n_I}), s_{j_i} \in \{s_{j_i}\}_I^k, i = 1 \ldots k$$

(*f*) (*Verification*) Peer $R$ checks (note here $R$ uses its own $PI_R$, $T_R$ to compute the following result):

$$y^2 = x(\prod_{i=1}^{k} v_{j_i}^{e_{j_i} + u_i} \pmod{n_I}), v_{j_i} \in \{v_{j_i}\}_I^k, i = 1 \ldots k$$

Peer $R$ accepts $I$'s proof if the equality holds, otherwise it rejects the proof. After peer $R$ verifies $I$, $I$ verifies $R$. The above interactive communication is anonymous, since the messages are relayed through onion paths from $I$ or $R$ to their tail nodes $T_I$ or $T_R$.

The session key exchange scheme here deserves some discussion. When peer $R$ executes step (*a*) on its side, it picks a random number $b \in [1, Q)$ simultaneously, and keeps $b$ as secret. When the authentication is successfully completed, peer $I$ computes $K = (g^b)^a \bmod P$, and peer $R$ computes $K' = (g^a)^b \bmod P$. Clearly, we have $K = K' = g^{ab} \bmod P$, and therefore, peer $I$ and peer $R$ use $K$ as their session key for the subsequent file transmissions.

In PT, (1) the length of $PI$, (2) the length of $m$, (3) the large integer $n$, and (4) the number of quadratic residue $k$ are four key parameters. The selections of $m$, $n$ and $k$ are essential to the security degree of PT, on which we have more discussions in Section 4.

## 4. Security Analysis

PT is designed to achieve three security goals: anonymity of peer real identity, authentication among peers, and resistance against impersonation and Man-In-Middle-Attacks (MIMA).

### 4.1 Degree of anonymity

The anonymity of a peer's identity comes directly from the one-way property of cryptographic hash functions. Let $h(\cdot)$ be a hash function with $m$-bit-long hash values, and assume it is well designed and has no structural drawback for cryptanalysis. In cryptograph terminology, $h(\cdot)$ takes advantage of a pre-image-resistance property, i.e., for any given hash value $y$, it is computationally infeasible to find an $x$ such that $h(x) = y$. Here "infeasible" means we need at least $2^{m-1}$ calculations of hash evaluation in general to find such an $x$ [21].

A malicious peer may launch advanced attacks, such as finding two different but real identities so that the two identities have the same $PI$. It might then use one of the two identities to impersonate the peer with the other identity. However, this kind of attack is withstood by the collision-resistance of hash functions: it is computationally infeasible to find a pair $(x, y)$ such that $h(x) = h(y)$, say a collision. An adversary needs $2^{m/2}$ calculations to find a collision with probability 1/2, which is infeasible for $m \geq 128$, see §18 in [21].

For $m = 64$, finding the pre-image of this hash value needs 600,000 mips-year, while finding the collision of this hash value only needs $2^{32}$ calculations, which can be executed in 1 mips-hour (more details can be found in § 7.2 in [21]). Hence, the length of hash values should be more than 64 bits. In the PT design, proper hash functions include SHA-1 ($m = 160$) and its offspring. If we choose SHA-1, a malicious peer must take $2^{159}$ calculations to compute the identity from the PI, and $2^{80}$ calculations to find a collision with probability 1/2. These are computationally infeasible. Thus, PT achieves anonymity for peers in networks. Malicious peers cannot deduce a real identity from a $PI$. In other words, PT provides a secure conversion from real identities to anonymous $PI$s.

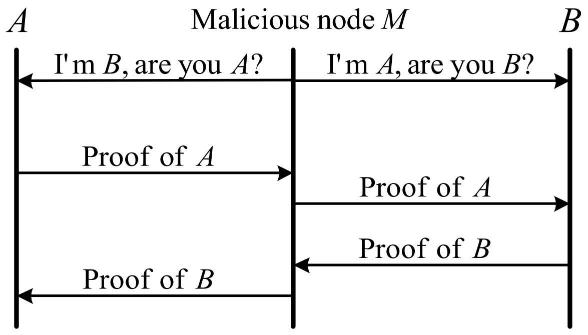

**Figure 3. MIMA attack. *M* cheats *A* or *B* that *M* is the real opposing party.**

## 4.2 Security discussion

In PT, the authentication of peers is employed through an underlying ZKP protocol (see Section 3.4). A ZKP protocol has the characteristic that even if multiple collaborating adversaries collect interactive information about former executions between two participants, they do not benefit more from launching impersonation attacks on participants of the protocol than in the situation of observing nothing. Due to the limitation of pages, in the security analysis below, we only give the analysis of impersonation and man-in-middle attacks. In our full version paper [15], we analyze the security goals of PT against replay, collaborating, and denial of service attacks. We also prove that *PI* is secure when it is issued in unsafe public sites. At last, we compare PT with PKI-based authentication protocols. From the comparison, we show that PT outperforms PKI-based authentication approaches.

*Impersonation*: To successfully defend against impersonations, PT should have the properties of completeness and soundness. In cryptology, completeness is defined as the ability of the verifier to accept true statements by the prover, while soundness asserts that the verifier cannot be "tricked" into accepting an invalid statement from a false prover. We prove these two properties of the PT design via the following lemmas.

**Lemma 1** (*Completeness*) If peers *I* and *R* properly follow the authentication procedure, then *R* always accepts the *PI* of *I* as valid.

**Lemma 2** (*Soundness*) Assume it is computationally infeasible for factoring *n,* and a malicious peer *M* does not have any partial knowledge of the initiator's secret $\{s_{j_i}\}$, $i$ = 1...$k$. Suppose *M* interacts the ZKP protocol with responder *R* to impersonate initiator *I* and convince *R* that it is *I.* Then the probability that *M* succeeds is $\max(2^{-k}, 2^{-t})$, where $k$ is the length of a challenge message $e$ (in PT, $k$ = 80, on which we have more discussions later), and $t$ is an integer dependent on $n$ only. When $n$ is of 1024 bit length, $t$ = 87.

The proofs of above lemmas can be found in our full version paper [15].

According to Lemma 2, the probability of a successful impersonation decreases when $k$ grows (here $k$ is the bit number of $e$). In this design, we believe it is safer to choose $k$ = 80 to get high overall security.

## 4.3 Man-In-Middle-Attack

A Man-In-Middle-Attack (MIMA) is an attack in which an intruder *M* is able to arbitrarily access and modify messages between two parties without either party knowing that the link between them has been compromised [15]. As a result, *M* can successfully impersonate the initiator to the responder, or vice versa. To PT users, intruders can modify and relay the forged authentication messages to participants and try to convince peer *I* or peer *R* that *M* is the opposing party. We define a MIMA to be successful if a malicious peer *M* is able to convince peer *I* or peer *R* that $T_M$, which is actually the tail node of *M*, is $T_I$ or $T_R$, a tail node of peer *I* or peer *R*. We also assume *M* is able to intercept, replace, and modify the messages arbitrarily.

As shown in Fig. 3, *M* impersonates two victims simultaneously, which is challenging to defend against, and a key issue in our discussion. Such a MIMA is based on two possible instances. *Instance* 1: *R* does not receive *I*'s query *q*. *Instance* 2: *R* receives *I*'s *q*.

For *Instance* 1, since *R* does not receive *q*, *R* does not respond. In this case, to cheat *R*, *M* has to (1) forward peer *I*'s query *q* directly to *R*, or (2) send a forged *q*, *q'*, to *R*.

For (1), *M* operates as other relaying nodes in the transmission. Since *M* does not modify anything, *R* connects with $T_I$ through $T_R$ directly. Thus, *M* cannot cheat anyone.

For (2), the possible modification on *q* by *M* leads to two sub-cases: (a) *M* modifies or just replaces the $PIC_I$ with its $PIC_M$ in *q* such that *R* considers *M* as an initiator. This is useless for *M*'s attack because it would fail in the later verification without a valid $PI_I$. Otherwise, *M* may hope to find a valid $PIC_M$ with the same hash value as $PI_I$, which is computationally infeasible as we discussed in Theorem 1. (b) *M* modifies *q* into $q' = (PIC_I, T_M, f)$, as the point ① shown in Fig. 4. The following discussion is based on this situation.

After receiving *q'*, *R* replies to *I* with $(PIC_R, T_R, f)$. *M* intercepts this reply, modifies this message to $(PIC_R, T_M, f)$, and delivers it to *I*. Note that *M* has to modify $T_R$ to $T_M$, otherwise *I* would ask $T_I$ to contact $T_R$. Thus, $T_M$ is not involved in the authentication and the attack fails. See point ② in Fig. 4.

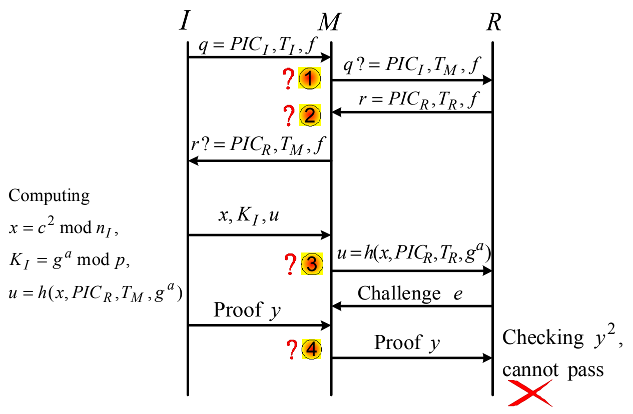

**Figure 4. Resistance from the Man-in-middle-attack.**

Following step (*a*) in the authentication procedure, peer *I* randomly chooses *c* and *a* and computes the commitment $x = c^2 \bmod n_I$, $g^a \bmod P$, and $u=h_4(x, PI_R, T_M, g^a)$. Then *I* sends them back to $T_M$.

Upon intercepting this message, *M* has only two choices of how to continue its intruding actions:

i) *M* relays this message to *R* without modification. Then *R* computes the $u' = h_4(x, PI_R, T_R, g^a)$ and checks it with the *u* peer *R* just received. According to the pseudo-random feature of the hash function, we have $u \neq u'$. *R* terminates the authentication procedure, and the attack fails. See point ③ in Fig. 4.

ii) *M* computes $u'' = h_4(x, PI_R, T_R, g^a)$ and sends it to *R*. In such a case, $u'' = u'$. *R* continues authentication. *R* then sends a challenge *e* to $T_M$. *M* cannot know *e* in advance and the best choice for *M* is to deliver the challenge to *I*. Otherwise *M* can only guess *e* with a probability of $2^{-k}$.

In ii), *M* cannot modify the *x* and $g^a$ to make a $u = u'$. This is because that according to the collision-resistance property of well-designed hash functions, finding such $x'$ and $g^{a'}$ that makes $h_4(x, PI_R, T_M, g^a) = h_4(x', PI_R, T_R, g^{a'})$ is computationally infeasible.

*I* and *R* continue the authentication procedure following the PT protocol until step (*e*). At this point, *I* generates a proof $y = c(\prod_{i=1}^{k} s_{j_i}^{e_{j_i}+u_i} (\bmod n_I))$, and sends it back. Since the secret $\{s_{j_i}\}^k$ of *I* is unknown to *M*, *M* cannot forge a proof to pass *R*'s verification. If *M* changes *e* so as to pass the verification, it must guess the value of *e* before *R* generates *e*, and change the value of *y* accordingly. Since the probability of such a successful guess is $2^{-k}$, it is infeasible. See point ④ in Fig. 4.

In step (*f*), *R* checks whether $y^2 = x\prod_{i=1}^{k} v_{j_i}^{e_{j_i}+u'_i} \bmod n_I$ holds. According to the previous discussion, $u = h_4(x, PI_R, T_M, g^a)$, but $u' = h_4(x, PI_R, T_R, g^a)$. It is clear that $u \neq u'$. Thus, *R* stops the authentication.

If *M* attempts to continue cheating *I* by impersonating *R*, since *M* does not know the secret $\{s_{j_i}\}^k$ of *R*, it cannot pass the verification on *I*'s side. Thus, MIMA attempts made by intruder *M* in *Instance* 1 fail.

For *Instance* 2, *R* receives *I*'s query *q*. In this case, *R* has multiple queries containing an identical *PI* with different tail nodes. Aware of being under attack, *R* can simply discard the query, or randomly select one of them to initiate the authentication procedure. The remaining analysis is similar to the case in *Instance* 1.

The key point of the authentication technique in PT is that we bind the commitment, the tail node's information, and the key exchange data together with a peer's *PI*. With this design, any attempts to modify the identity messages cannot pass the verification of genuine protocol participants. Thus, MIMA fails to attack our proposed PT protocol.

## 5. Performance Analysis

We first evaluate the PT design by trace-driven simulations, in which the P2P topologies are obtained from the DSS Clip2 trace [1]. Our simulations are performed on those traces in a variety of network sizes ranging from hundreds to thousands. For each simulation, we take the average result from 1,000 runs. The results are consistent with traces of different days and here we show the representative results.

### 5.1 Response time

Of all latencies in a P2P system, the response time from query issuance to the start of the download is of greatest concern, as it has a significant bearing on the system usability. Figure 5 plots the simulation results of the response time, where we show the accumulative percentage of returned responses versus time for PT, overt Gnutella protocol, and APFS. Note that we have only included the latency of sending queries and responses in the APFS protocol, as we do not want the other components of APFS to influence our results.

The comparison in Fig. 5 shows that the response time of APFS is approximately 3 times that of overt Gnutella, while PT is around 7 times that of overt Gnutella. In APFS, users need one onion path plus a flooding procedure to send a query out, one TCP link to deliver the response between tail nodes, and two onion paths to send the response anonymously. In PT's two-phase authentication procedures between two parties, the numbers of used onion paths and TCP links are 12 and 6 to follow APFS, respectively. According to the design of

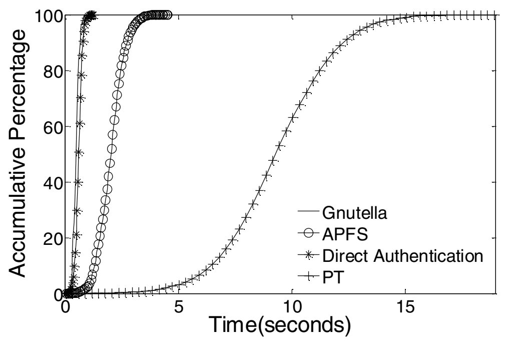

**Figure 5. Response time.**

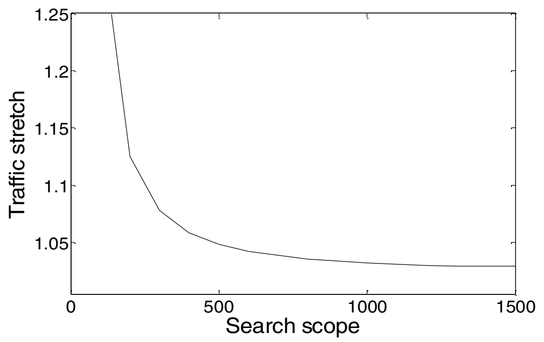

**Figure 6. Traffic stretch.**

PT, the authentication messages pass through the TCP connections between two tail nodes 6 times in a mutual authentication procedure. We also simulate a pure mutual authentication procedure without overt Gnutella and APFS, shown as the star line in Fig. 5. Our observation shows that the average response time of normal query flooding, direct authentication, APFS, and PT are about 493ms, 600ms, 2031ms, 9296ms, respectively. Note that the time consumed in anonymous paths of PT constitutes a major part of the whole latency, which is four times more than that of APFS. Therefore, the time consumption of authentication is indeed trivial. Our later offline implementations also support this summary.

### 5.2 Traffic overhead

In our next experiment, we test the extra traffic cost brought about by authentication procedures. We define the traffic stretch as the traffic cost ratio between PT plus Gnutella, and Gnutella only. As shown in Fig. 6, traffic stretch decreases when search scope increases. The traffic stretch is lower than 1.03 when the search scope reaches 1,400 peers, which means less than 3% additional traffic is incurred by PT.

The extra traffic cost is mainly incurred by anonymous communications and authentication interactions among peers. These connections comprise two anonymous sessions and a TCP link. Therefore, the scale of extra traffic cost mostly depends on the sum of those connection lengths. In fact, we observe that the average distance between two random nodes tends to be constant with the growing size of the P2P overlay. As a result, the extra traffic cost caused by the PT authentication also slightly fluctuates around a constant. In our experiments, the traffic stretch first decreases sharply and then tends to be constant. As a reflection, Fig. 6 illustrates this tendency.

## 6. Prototype Implementation

To better evaluate PT, we implemented a prototype in our labs at the Chinese Academy of Sciences (CAS), the campus of Beihang University and Hong Kong University of Science and Technology, and others sites. We launched two-part experiments.

The first set focuses on the extra computation overhead caused by PT, and test the computing capabilities of normal PCs running this protocol. The second set tests the overall latency of pseudo identity authentication procedures in the Internet environment.

### 6.1 Offline experiments

We first examine the performance of PT in an offline environment by conducting experiments on four different desktop PCs with the following configurations: PIII450M/128M, PIV1.8G/256M, PIV2.6G/256M, and P-M1.4G/256M. Figure 7 plots the time consumption of PIC generation (including the PI generation) on the above four machines with a 1024-bits moduli, which is a default selection, providing enough security to PT. The average time to generate a PIC increases linearly when the amount of quadratic residue, *k*, grows. In previous discussions, we show it is safe when *k* is no less than 80. For a current popular PC configuration, such a generation needs less than 8 seconds, which is acceptable because PIC generation is a one-time job, necessary only when a peer joins the P2P community.

Figures 8 and 9 plot the time used for the proving and verifying operations on the four machines. We can see that with 1024-bit moduli, for a popular machine, proof generation requires less than 0.01 seconds and verification only needs 0.002 seconds when *k* is 80. Even for poorly configured PCs such as PIII450M/128M, the processing time is still acceptable.

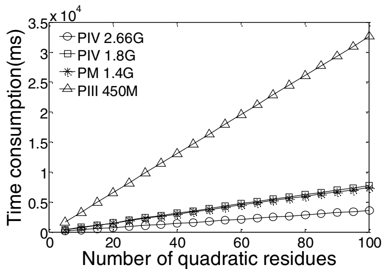

**Figure 7. *PIC* generation (1024bit moduli).**

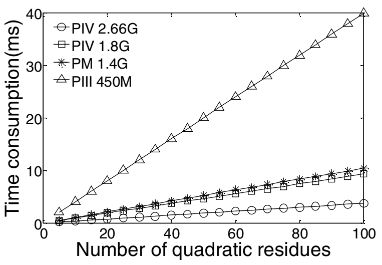

**Figure 8. Proof generation (1024bit moduli).**

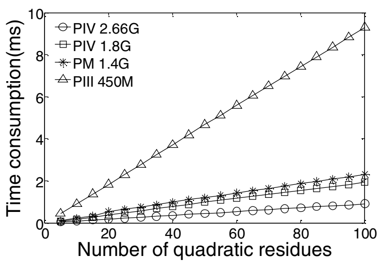

**Figure 9. Verification (1024bit moduli).**

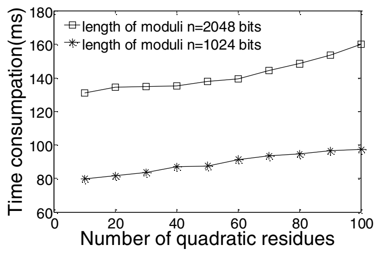

**Figure 10. CAN test.**

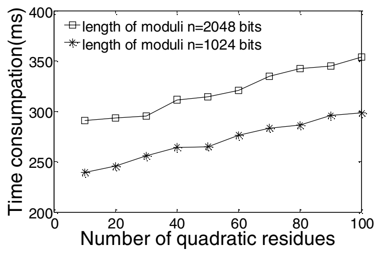

**Figure 11. MAN test.**

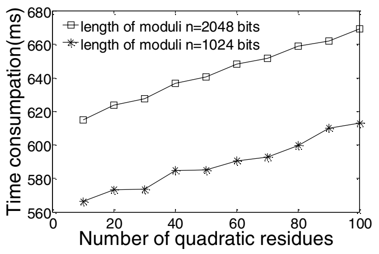

**Figure 12. WAN test.**

### 6.2 Implementation in real internet environments

We then implement our PT prototype in the internet via a P2P overlay comprising of fifty desktop PCs at the labs in Beijing and Hong Kong. The LANs of the lab are interconnected with D-Link switches 3624i, and connected with the campus network with Cisco routers Catalyst 2950. All LAN bandwidths are 100Mbps. To better evaluate PT, in this implementation, we ignore the time consumed by the APFS protocol. Figures 10-12 show experimental results in the Campus Area Network (CAN), Metropolitan Area Network (MAN), and Wide Area Network (WAN). The number of quadratic residues range from 10 to 100 and we use 1024 and 2048 bits as moduli sizes, respectively.

In the campus network of HKUST, the ping time of the biggest packet of PT messages is less than 2ms. This latency mainly comprises the time consumption for the proving and verifying operations of PT. In WAN, the distance between two parties becomes the main factor of latency. As shown in Fig. 12, sometimes the irregular variety of communication channels influences the curve of latency and causes the curves to plunge down or rise into revulsions.

We employ ping tests with packets of the same size as PT messages over the two involved computers between Hong Kong and Beijing in WAN. The results range from 0.07 seconds to 0.12 seconds. Therefore, the overall latency of PT is less than 0.67 seconds, which is relatively small for an authentication procedure. Note that the extra latency of PT in implementation results is much shorter than that in the simulation results because the time consumed by the Gnutella and APFS protocols is included in the simulation, but not in the prototype implementation. The package of PT prototype is available at [15].

## 7. Conclusion

Due to the inherent tradeoff between trust and anonymity, existing attractive identity-based trust management schemes cannot be directly employed in anonymous P2P systems. We propose an anonymous zero-knowledge authentication protocol in this paper, called Pseudo Trust. In this design, a ZKP-based authentication scheme is designed to support trust management in anonymous P2P systems, so that peers may use unforgeable and verifiable pseudonyms instead of their real identities in P2P communities.

We prove that the probability of a successful impersonation is computationally infeasible, even if the adversaries have collected all of the previous authentication messages. We also manage to address man-in-middle-attacks in the PT design. The results of trace-driven simulations show that PT has perfect scalability in both static and dynamic environments. We also implement a prototype of PT and evaluate its performance by experiments. We believe that wide deployment of Pseudo Trust will provide better privacy and security for P2P users.

## Acknowledgements

This work was supported in part by the NSFC grants No. 60573053 and No.90412011, the National Grand

Fundamental Research 973 Program of China under Grant No. 2005CB321803, the NSFC Funds for Distinguished Young Scholar under grant No. 60525209, and the Hong Kong RGC grants HKUST6264/04E and HKUST6152/06E.

## References

[1] The Gnutella Protocol Specification v4.0. http ://www.clip2.com/GnutellaProtocol04.pdf.

[2] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information Systems. In Proceedings of *the international conference on Information and knowledge management*, 2001.

[3] T. Beth. Efficient Zero-Knowledge Identification Scheme for Smart Cards. In Proceedings of *EUROCRYPT*, 1988.

[4] J. Brandt, I. B. Damgard, P. Landrock, and T. Pedersen. Zero-Knowledge Authentication Scheme with Secret Key Exchange. In Proceedings of *Advances in Cryptology*, 1990.

[5] F. Cornelli, E. Damiani, S. D. C. d. Vimercati, S. Paraboschi, and P. Samarati. Choosing Reputable Servents in a P2P Network. In Proceedings of *the international WWW conference*, 2002.

[6] E. Damiani, S. D. C. d. Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In Proceedings of *ACM CCS*, 2002.

[7] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976.

[8] U. Fiege, A. Fiat, and A. Shamir. Zero Knowledge Proofs of Identity. In Proceedings of *ACM Symposium on Theory of Computing(STOC)*, 1987.

[9] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge: Cambridge University Press, 2001.

[10] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 1989.

[11] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In Proceedings of *the international WWW conference*, 2003.

[12] Z. Kim and K. Kim. Provably-Secure Identification Scheme based on Braid Group. In Proceedings of *the international conference on soft computing and intelligent systems(SCIS)*, 2004.

[13] P. P. C. Lee, J. C. S. Lui, and D. K. Y. Yau. Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups. *IEEE/ACM Transactions on Networking*, 2006.

[14] S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative Peer Groups in NICE. In Proceedings of *IEEE INFOCOM*, 2003.

[15] L. Lu, Y. Liu, L. Hu, J. Han, and L. M. Ni. Pseudo Trust: Zero-Knowledge Authentication in Anonymous Peer-to-Peer Protocols. http://www.cse.ust.hk/~liu/PseudoTrust050608.htm.

[16] Y. Lu, W. Wang, D. Xu, and B. Bhargava. Trust-Based Privacy Preservation for Peer-to-peer Data Sharing. In Proceedings of *the NSF/ NSA/ AFRL workshop on secure knowledge management (SKM)*, 2004.

[17] W. Mao. *Modern Cryptography: Theory and Practice*, 2004.

[18] D. H. Nyang and J. S. Song. Knowledge-Proof Based Versatile Smart Card Verification Protocol. *ACM SIGCOMM Computer Communication Review*, 2000.

[19] K. Ohta and T. Okamoto. A Modification of the Fiat-Shamir Scheme. In Proceedings of *Advances in Cryptology*, 1990.

[20] V. Scarlata, B. N. Levine, and C. Shields. Responder Anonymity and Anonymous Peer-to-Peer File Sharing. In Proceedings of *IEEE ICNP*, 2001.

[21] B. Schneier. *Applied Cryptography - Protocols, Algorithms, and Source Code in C*, Second ed: John Wiley & Sons, Inc, 1996.

[22] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. $P^5$: A Protocol for Scalable Anonymous Communication. In Proceedings of *IEEE Symposium on Security and Privacy*, 2002.

[23] G. J. Simmons and G. B. Purdy. Zero-Knowledge Proofs of Identity And Veracity of Transaction Receipts. In Proceedings of *EUROCRYPT*, 1988.

[24] A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In Proceedings of *the 3rd International IEEE Conference on Peer-to-Peer Computing*, 2003.

[25] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. In Proceedings of *IEEE Symposium on Security and Privacy*, 1997.