Pseudo Trust: Zero-Knowledge Authentication in Anonymous P2Ps

Li Lu, *Member, IEEE*, Jinsong Han, *Member, IEEE*, Yunhao Liu, *Senior Member, IEEE*, Lei Hu, Jinpeng Huai, *Member, IEEE*, Lionel M. Ni, *Fellow, IEEE*, Jian Ma

Abstract—Most 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 that 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 security and the anonymity in PT, and evaluate its performance using trace-driven simulations and a prototype PT-enabled P2P network. The strengths of our design include 1) no need for a centralized trusted party or CA, 2) high scalability and security, 3) low traffic and cryptography processing overheads, and 4) man-in-middle-attacks resistance.

Index Terms—Peer-to-Peer, Authentication, Mutual Anonymity, Trust, Zero-Knowledge Proof.

1 INTRODUCTION

As an emerging model of communication and computation, Peer-to-Peer (P2P) networking has recently gained significant acceptance [1-3]. Most widely-deployed P2P systems today, such as 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 used 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 user privacy. The IP addresses of object requesters and providers can easily be discovered and translated into user names or postal addresses. Hence, many studies such as P^5 [4] and APFS [5] 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 [6-8]. Most trust models, however, 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 and forgery, 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 (ZKP) to help unfamiliar peers successfully complete authentication procedures during transactions. The salient features of Pseudo Trust include:

(1) Achieving anonymity for authentication. PT enables pseudonym-based trust management so that the real identities of peers are protected during the authentication. PT can adopt current anonymous systems for anonymizing communications.

(2) Realizing distribute trust. Previous trust management systems usually need a centralized trusted party, e.g.

- *L. Lu, J. Han, Y. Liu, and L. M. Ni are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, P.R. China. Email: luli@cse.ust.hk, jasonhan@cse.ust.hk, liu@cse.ust.hk, ni@cse.ust.hk.*
- *L. Hu is with the State Key Laboratory of Information Security, the Graduate University of Chinese Academy of Sciences, Beijing 100049, P.R. China Email: hu@is.ac.cn.*
- *J. Huai is with the School of Computer Science, Beihang University, Beijing 100083, P.R. China. Email: huaijp@buaa.edu.cn.*
- *J. Ma is now with Nokia Research Center, Beijing 100083, P.R. China. Enail:Jian.J.Ma@nokia.com.*

Manuscript received (insert date of submission if desired). Please note that all acknowledgments should be placed at the end of the paper, before the bibliography.

Certificate Authority (CA), to provide the trust for users. All users have to fully trust such an entity and share some secret with it. It risks leaking the user's private information and increases the system overhead. In contrast, PT allows users to generate their pseudo name individually and users do not depend on any third party to authenticate with each other.

(3) Resisting man-in-the-middle-attacks. If we remove the centralized authority from the system, Man-In-The-Middle-Attack (MIMA), may severely threaten the authentication in distributed systems. A MIMA attacker can arbitrarily intercept, modify, or forge the message between two authentication parties so that the attacker can impersonate either of them to another party. PT, however, employs the one-way hash function to bind users' pseudonyms and the authentication paths together. Using the one-way hash function plus ZKP, users in PT not only authenticate the pseudo identities, but also detect the authenticity of anonymous agents and check the integrity of messages. Our theoretical analysis shows the security of PT under MIMA attacks.

We discuss the implementation choices that were made for security and efficiency reasons, and use the traces from DSS Clip2 to conduct trace-driven simulations to evaluate the parameter selections and the performance of our design. We also implement a PT prototype within a 50-machine overlay across the Internet in our labs in Beijing, Hong Kong, and other sites. Both our theoretical analyses and experimental results show that PT is effective, scalable, and completely decentralized with no need of a CA.

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 security degree of PT. Section 5 presents our trace driven simulations and the performance evaluation. We introduce the PT prototype implementation 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 Schemes

Abdul-Rahman et al. proposes a trust model and a recommendation protocol [9], focusing on decentralized systems. A prime and clear definition of trust is also provided. XREP [7] enables peers to evaluate and share other peer reputations by introducing a distributed polling algorithm. XREP also employs confirmation voting procedures among randomly chosen peers in order to prevent collusive cheating from cliques of malicious peers. The P-Grid focuses on an efficient data management technique to construct a scalable trust model for decentralized applications. EigenTrust [8] builds a virtual global matrix to represent individual reputations. NICE [10] provides a platform to implement distributed cooperative applications. Based on trust chains, NICE computes a user reputation in a PGP-like model. XenoTrust [11] provides a public infrastructure for a wide-area trust computing environment. Generally, most P2P trust designs are identity-based, where one peer does not trust another before knowing its identity.

2.2 Anonymity

Privacy has become an increasingly salient issue, and considerable progress has been made with anonymous communications [4, 5]. Several solutions achieve mutual anonymity for both initiators and responders in P2P systems, which generally aim at concealing the real identities of users during transactions. For example, in APFS [5], peers construct an anonymous path with tail nodes using an onion technique, providing complete and mutual anonymity for peers. Recent research has attempted to introduce reputation value into anonymous P2P systems, or construct a trust management based on proxy techniques. However, failure to support authentication makes these approaches vulnerable to impersonation and man-in-the-middle-attacks. Therefore, it is argued that providing privacy to peers increases the difficulties of authenticity and security. Obviously, there is a 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 the knowledge during the verification process (zero-knowledge). The concept of ZKP was first introduced by Goldwasser et al. in [12], and has since been employed in many authentication and identification protocols. Loosely speaking, a ZKP is an interactive proof system which is comprised of 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 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 [12] or computing a discrete logarithm problem. Some have been improved to employ mutual authentication and key exchanges [13]. However, since almost all ZKP-based identification schemes are dependent on a trusted third party (such as a CA) as an authorized central server, they are not directly adopted by this design.

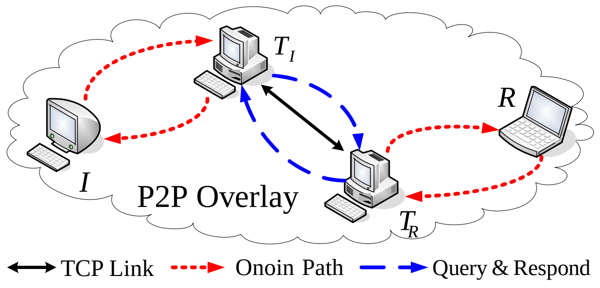

Fig. 1. PT query issuance and authentication.

TABLE 1
NOTATIONS OF VARIABLES

Notation	Specification
I	Initiator of a query
T_A	Tail node of peer *A* [5]
R	Responder
f	Index of the requested file
M	Malicious peer
PI_A	The pseudo identity of peer *A*.
PIC_A	Pseudo identity certificate of peer *A*
K	Session Key
h_i	Hash function
e	Challenge message
k	Bit number of *e*

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 real IDs to deal with other peers in a P2P society, can peers use pseudonyms to interact with others and accumulate their reputations?

Clearly, if we would like 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 pseudonym it claims.

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

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 reverse of the query path.

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 [5]. Other selections of anonymous protocol designs are possible, but such changes are out of the scope of this discussion.

Before going into the details of PT, we list the notations we use in later discussions in Table 1.

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. 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.

However, a naive *PI* generation can lead to malicious impersonation. Thus, a *PIC* is generated to authenticate the *PI* holder. Whenever a *PI* is issued, a *PIC* follows. A peer with a *PIC* is able to verify whether the other party is the one it claims to be. The *PI* and *PIC* generation mechanism is described as follows. Terms not defined here can be found in [14].

Let $ID \in \{0,1\}^*$ 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: Peer *A* randomly chooses two large primes p_1 and p_2, and calculates the integer $n = p_1 \times p_2$. PT then adopts a hash function, such as SHA-1. We slightly modify the prototype SHA-1 and name the revisions h_i to fit the different inputs and outputs. Here, PT 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. Peer *A* then computes its $Seed_A$ by $Seed_A = h_1(ID, p_1, p_2) \in \{0,1\}^m$

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

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

(2) Compute the smallest square root s_j of $v_{j_i} \pmod n$. 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, where $i = 1 \ldots k$. 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 [15].

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

After the above operation, peer *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 authentic-

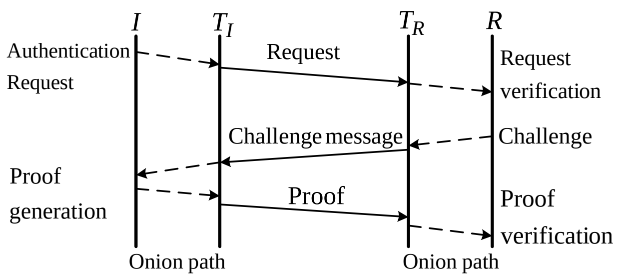

Fig. 2. Procedure of zero-knowledge proof.

ity 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. For a detailed analysis on this part, see Subsection 4.7.

3.3 New Peer Initialization

After joining the P2P system, peer A constructs anonymous sessions with existing peers using the APFS protocol. In this design, anonymous sessions are onion routes (OR) comprised of chosen peers. A tail node T_A acts as an agent relaying a message for peer 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, T_A constructs the following OR:

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

Meanwhile, peer A builds another onion path OR_{T_A} for sending message to T_A anonymously.

$$OR_{T_A} = \{X, \{...T_A, \{mix\}K_{T_A}, ...\}K_X\}$$

After the anonymous session construction (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_{T_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, it replies to this query through its tail node with a response including its tail node and the reputation record.

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

3.4 Authentication and Session Key Exchange

PT employs a modified ZKP of Identity scheme. To adopt it into decentralized P2P networks, we remove the central authority servers in [16]. This scheme is based on the assumption that factoring a large integer is computationally infeasible.

When the query initiator, peer I (note that I is the pseudo identity of the initiator), receives multiple responses, it selects one or more peers with high reputations as potential downloader or responder. Without loss of generality, suppose R is selected as one of the responders. Peer I can initiate the authentication procedure to verify that the peer claiming to be the holder of R is not lying. Peer I sends an authentication request to R through the anonymous path, $I \to T_I \to T_R \to R$ (note $I \to T_I$ and $T_R \to R$ are onion paths OR_{T_I} and OR_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_I.

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

Having the PIC_R of R, peer I initiates the *authentication procedure* as illustrated in Fig. 2. The authentication procedure includes two main phases: (1) peer I acts as a prover to prove its validation to R, and (2) peer R proves its authenticity to peer I accordingly. 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 Section 4.

To provide confidentiality and integrity to data exchanges after authentication, we embed a Diffie-Hellman Key Exchange protocol into the authentication procedure to generate a session key held by I and R only. For efficiency purposes, we adopt a digital signature standard group (DSS, see [17]) to simplify the key exchange protocol. 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. 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 a secret. Here, h_4 is a hash function h_4: $Z_n^* \times \{0,1\}^m \times \{0,1\}^* \times Z_P^* \to \{0,1\}^k$ to generate u, and u is a k bits long. We use $(u_1...u_i...u_k)$, where $u_i \in \{0,1\}$, to represent u.

(*b*) (*Request Verification*) Peer R computes $u' = h_4(x, PI_R, T_R, g^a)$, then verifies whether $u = u'$. If the verification holds, peer R goes on to the next authentication step, 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...k.

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

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

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

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

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

Peer *R* accepts peer *I*'s proof if the equality holds, otherwise it rejects the proof. After peer *R* verifies peer *I*, peer *I* verifies *R*. Note that the above interactive communication is anonymous and the messages are relayed through onion paths and the TCP connection between the tail nodes T_I and 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 a 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, peers *I* and *R* use *K* as their session key for the subsequent file transmissions.

Here, (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. Based on our analyses and experimental results, we choose $m \geq 64$, *n* as a number being 1024 bits long, and $k = 80$.

3.5 File Delivery

After completing the authentication procedure, the two parties obtain a shared session key $K = g^{ab} \bmod P$. For confidentiality, PT employs a symmetric cipher algorithm such as AES to encrypt the content of a desired file using the key *K*. The integrity mechanism are used to prevent the data from being forged, replaced, or modified during transmission without being detected, while the authenticity indicates that the receiver can be assured that the sender really did send that message (non-repudiation). In this design, peers use a SHA-1 hash function to generate a Message Authentication Code (MAC) as a warrant to convince the opposing party that the file is valid and guarantee the integrity of the data. For example, *R* computes a MAC, $h(F||K)$, using hash function $h(\cdot)$, then encrypts the file *F* attached with the MAC into ciphertext *C*, and delivers *C* to *I*, where || denotes the concatenation.

$$R \xrightarrow{C = Encrypt_K(F || h(F||K))} T_R \rightarrow T_I \rightarrow I$$

When peer *I* receives *C*, it decrypts *F'* from this cipher and checks whether $h(F'||K)=h(F||K)$. If they are equal, peer *I* can be sure that *F* is indeed the file sent from *R*, and *F* has not been modified during transmission. If the file does not need to be encrypted, *R* can simply compute $h(F||K)$ and attach it with the file. A peer can make a flexible choice depending on its security requirements.

3.6 Trust and Reputation Management

After completing the authentication procedure, peer *I* can download files from peer *R*. Similar to the authentication message exchange, the file can be delivered through anonymous sessions. A peer can either use the session key to encrypt the file or only encrypt the MAC of the file according the users' security requirements. Since only peer *I* and peer *R* know their session keys, other peers, even the tail nodes, cannot forge the file to deceive the initiator during the data transmissions.

After downloading a file, peer *I* can evaluate the file and provide comments to peers who provide resources such that most existing trust management mechanisms, such as EigenTrust [8] and XenoTrust [11], become applicable. The only difference after employing PT in these systems is that the reputation of peers will be connected with peer pseudo identities (*PI*s) instead of their real IDs or IP addresses.

3.7 PT in Structured P2P System

PT can also be adopted in structured P2P systems. In structured P2Ps, such as CAN and Chord, each peer's IP address and resource can be mapped to a point in the ID space using distributed hashing tables (DHT). Each peer stores a fraction of the entire ID space, for example a *zone* in CAN. Lookup processes are handled as follows. On one hand, the resource index is mapped to a point in the ID space via DHT by the resource holder, for example the Insert(key, value) process in CAN. The index, with the IP address of resource holder, will be maintained by the peer whose zone covers the point. On the other hand, a requester computes the key of desired resource and issues it to DHT, for example the Lookup(key) process in Chord, to obtain the index of desired resource from the peer holding this information. Then the requester will directly contact the provider for retrieving the resource. In this procedure, there is no anonymity protection for peers.

PT can be applied in structured P2P systems. First, the core technique of structured P2P systems is the consistent hashing function, which has the features including the uniform distribution of outputs and resilience to collisions. These features are also essential properties of cryptographic hash functions[1] [14], which have been employed in the PT for *PI* generation. The typical consistent hash function used in structured P2P systems is MD-5 or SHA-1. In PT, the *PI*s of users are also generated by using SHA-1 or MD-5, which enables *PI*s of PT to be used in structured P2P systems with slightly modification. In addition, adopting PT in structured P2P system can enhance the anonymity of peers. Structured P2Ps such as CAN or Chord generate the peer's identity through DHTs but are 'limited' to achieve anonymity. In most structured P2Ps, a peer's ID is derived by hashing the peer's IP address. Peers do not input any unique secret into the hash function to generate the IDs. Therefore, any peer who knows a given IP address can easily generate the corresponding ID. This process makes peers in CAN or Chord suffering impersonation. It thereby degrades the anonymity and security, and cannot guarantee the validation of authentication. In PT, peers use their unique secrets as an input of ID generation. It makes the pseudo ID verifiable and invulnerable to impersonation. By using PT authentication

[1] Loosely speaking, the cryptographic hash function is defined as the hash function with three properties: one-wayness, uniform and pseudo-random outputs, and collision resistance.

mechanism, the validation of a pseudo ID is guaranteed. Thus, replacing the original ID with a *PI* via PT protocol will enhance the anonymity and the security in structured P2P systems. Second, instead of embedding IP address in the resource index, the resource provider can pre-construct an onion path, which points to itself, and attach it to the resource index. Be that do, any requester can utilize the onion path to contact the resource provider, while knowing nothing about the provider's identity. Of cause, the initiator also employs onion path and anonymous agent for issuing the query to DHT, communicating with the peer holding the resource index, and retrieving the resource from the provider. In this way, PT's authentication mechanism can be adopted in structured P2Ps.

Some previous works have been done to anonymize structured P2P systems. Similar to unstructured P2Ps, anonymous approaches in structured P2Ps achieve anonymity via secured paths. The main principle of those schemes is to construct an anonymous path, predetermined by the initiator [18] or randomly probed during the message delivery [19]. For example, Salas [20] presents an anonymous communication system based on DHT. To enhance the anonymity, each user constructs a circuit using Tor, which is in fact an anonymous path, to contact a proxy node. The proxy node performs lookup for the initiator, and hence, this proxy node is equivalent to a tail node, which is an anonymous agent as we mentioned in the Subsection 3.4. In other anonymous structured systems, most of them have such kind of anonymous paths, such that the nodes at the end of those paths can serve as the anonymous agents. Once assigning the anonymous agents, PT can be easily conducted over those systems. Therefore, structured P2P systems can seamlessly adopt PT.

4 Security Analysis

We first analyze the anonymity degree of PT, and then discuss attack resilience of PT, such as the impersonation and man-in-the-middle-attack (MIMA).

4.1 Anonymity

PT uses a cryptographic hash function such as SHA-1 to generate *PI*s from real identities for two reasons. First, the hash function is efficient and easy to use. And second, the hash function is publicly available for all peers in all networks with no need to exchange confidential information shared among them. These two properties are extremely appropriate for open P2P environments. We show the computation cost of *PI* generation in Section 6.

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 should do at least 2^{m-1} calculations of hash evaluation in general to find such an x. See §18.1 in [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 $h(x) = h(y)$. For a hash function with m bit hash values, $2^{m/2}$ calculations are required to find a collision with probability 1/2, which is infeasible for $m \geq 128$. See §18.1 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 hash value length should be more than 64 bits. In the PT design, for properly executing the ZKP (see Lemma 2) and obtaining high overall security, we suggest $m \geq 80$. Proper hash functions for our design include SHA-1 (m=160) and its offspring. If we choose SHA-1, a malicious peer must take 2^{159} calculations to work out the identity from the *PI*, and 2^{80} calculations to find a pair of inputs with the same output. These are obviously infeasible computations. Thus, PT achieves complete anonymity for peers in networks. Malicious peers cannot deduce a real identity from a *PI*. In other words, the PT scheme provides a secure conversion from real identities to anonymous *PIs*.

Note that crypto-analysis is making surprisingly excellent progress in cracking widely-used hash functions. However, the recent common view is that these functions are still cryptographically secure for applications. In PT, we employ the hash function SHA-1.

4.2 Impersonation

PT can effectively defend against impersonation. If PT is executed successfully between the initiator I and responder R, I (or R) does not leak any secret $\{s_j\}^k$ used in the authentication procedure to the opposite side and any adversary (i.e. PT is zero-knowledge). Without the secret, none can impersonate others, i.e. PT would not be subject to impersonation. We prove this feature below.

Theorem 1. Assuming factoring a composite number n is intractable, if initiator I and responder R properly follow the PT protocol, then R always accepts *PI* of I as valid, and vise versa. Meanwhile, I or R would not leak any knowledge of the secret $\{s_j\}^k$ to the opposite side and any malicious peer M. If a malicious node M plays an impersonation, the successful probability 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. Especially, when $|n|$ = 1024 bits, t = 87.

Proof: First we prove PT has the properties of completeness and soundness. In cryptology, loosely speaking, 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.

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

Proof: According the authentication procedure steps in Section 3.4, if peer I owns the correct secrets $\{s_j\}^k$, it calculates a valid answer using the data c received from peer R.

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

Peer R verifies the answer y as follows:

$$y^2 = (c\prod_{i=1}^{k} s_{j_i}^{e_{j_i}+u_i})^2 = c^2\prod_{i=1}^{k}(s_{j_i}^2)^{e_{j_i}+u_i} = x\prod_{i=1}^{k} v_{j_i}^{e_{j_i}+u_i} \pmod{n_I}.$$

Thus, peer R always accepts the y generated by I if peer I has the proper secrets $\{s_{j_i}\}^k$. □

Similarly, peer I always accepts the *PI* of peer R as valid, if I and R properly follow the authentication procedure. The proof is same as above.

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 $\{s_{j_i}\}$, $i = 1 \ldots k$. Suppose M interacts the PT protocol with responder R to impersonate initiator I and convince R that it is I. Then the probability that M succeeds is $\max(2^{-k}, 2^{-t})$.

Proof: The ZKP protocol is based on the intractability of finding the square roots of modulo n. According to the number theory, finding such roots is equivalent to factor n (Fact 3.46 in [22]).

Meanwhile, according to recent progress on number theory, the complexity of the best algorithm for factoring a generic number n is $O(2^t)$ [23], where $t = a(\ln n)^{1/3}(\ln\ln n)^{2/3}/\ln 2$ and $a \approx 1.93$. When $n \approx 2^{1024}$, $t = 87$.

On the other hand, M can cheat R by first guessing the binary vector e_j and then sending $x = c^2\prod_{i=1}^{k} v_{j_i}^{e_{j_i}+u_i}$ (mod n_I) as well as $y = c$ (mod n_I) to R to impersonate I. The success probability of guessing e_j is 2^{-k} since the vector is k-dimensional. M may try to increase this probability as follows. M chooses an x such that for a non-negligible probability it can compute the square roots y'/y'' of $x/(\prod_{i=1}^{k} v_{j_i}^{e_{j_i}+u_i})$ (mod n_I) for two vectors e' and e''. Then the value y'/y'' is calculated as follows:

$$y'/y'' = \left(\frac{x/(\prod_{i=1}^{k} v_{j_i}^{e'_{j_i}+u_i})}{x/(\prod_{i=1}^{k} v_{j_i}^{e''_{j_i}+u_i})}\right)^{\frac{1}{2}} \pmod{n_I}$$
$$= \prod_{i=1}^{k} s_{j_i}^{e''_{j_i}+e'_{j_i}} \pmod{n_I}$$

y'/y'' (mod n_I) is the square root of $\prod_{i=1}^{k} v_{j_i}^{e''_{j_i}+e'_{j_i}}$ (mod n_I). So M can compute the square roots in $Z_{n_I}^*$ within a polynomial time. As mentioned before, finding square roots in $Z_{n_I}^*$ is exactly equivalent to factor n_I, product of two primes. Therefore, we now have a method to solve the NP problem of factoring a composite number, which is obviously intractable in a polynomial time.

In summary, the probability that M successfully convinces peer R that it is peer I is $\max(2^{-k}, 2^{-t})$. □

Similarly, if M interacts the PT protocol with initiator I to impersonate responder R and convince I that M is R, then the probability that M succeeds is $\max(2^{-k}, 2^{-t})$. The proof is same as above.

Now we prove that PT would not leak any knowledge of the initiator and responder in the authentication procedure. In other words, PT is a zero-knowledge proof protocol.

Lemma 3 (*Zero-knowledge*) PT is a zero-knowledge proof protocol.

Proof (Sketch): Due to the page limitation, we just give a non-rigorous proof sketch. In PT, no information is revealed whatsoever I's secret $\{s_j\}$ is. This is because the random number x consists of random squares, and I's proof y contains an independent random variable which masks the values of $\{s_j\}$. Hence, all the messages sent from I to R are in the pattern of random numbers indistinguishable with the uniform distributions. That is, R cannot get any information about I's secret s_j. Which means PT is a zero-knowledge proof protocol. For formal and detailed proof, we refer readers to the reference [16]. □

In summery, PT successfully performs authentication between honest initiator I and responder R, and effectively defends against the impersonation attack. Therefore, we have proved Theorem 1, which claims that the PT is provably secure under the impersonation attack. □

4.3 Replay Attack

For a replay attack, malicious peers collect some previous proofs of an initiator, and resend these proofs to the responder. To convince the responder, malicious peers must guess the challenge message e (see Section 3.4) generated by the responder completely and correctly. According to the proof of Lemma 2, the probability of a malicious peer's guessing correctly a challenge message e chosen by the responder is 2^{-k}. Thus, the success probability of a replay attack is 2^{-k}.

4.4 Man-In-The-Middle-Attack

A Man-In-the-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. 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 indeed 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: peer R does not receive peer I's query q. Instance 2: R receives I's q.

For **Instance 1**, since R does not receive q, R does not

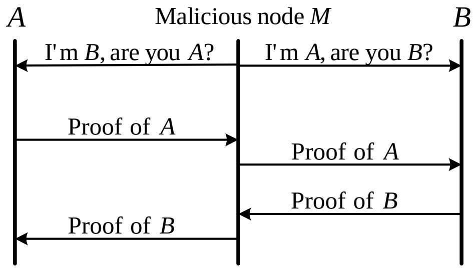

Fig. 3. MIMA attack. M cheats A or B that M is the real opposing party.

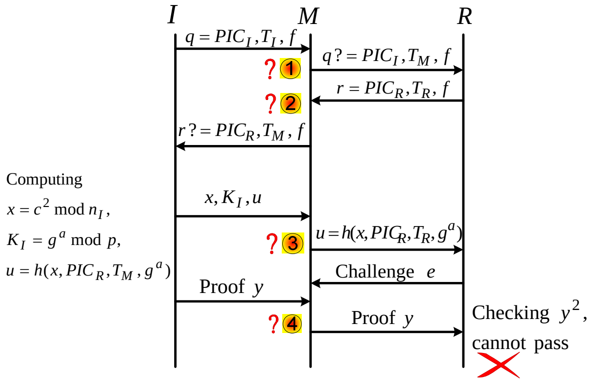

Fig. 4. Resistance from the Man-In-the-Middle-Attack.

respond. In this case, to cheat R, M has to (1) forward peer I's query q directly to R, or (2) forge a q' and send it to R.

For (1), M acts like 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. That is, T_M is not involved in the authentication and the attack fails. See point ② in Fig. 4.

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 $u = u'$. Due 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.

Peers I and R continue the authentication procedure following the PT protocol until step (e). At this point, peer I generates a proof $y = c(\prod_{i=1}^{k} s_{j_i}^{e_{j_i}+u_i}) \pmod{n_I}$, 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 $y^2 = x\prod_{i=1}^{k} v_{j_i}^{e_{j_i}+u_i'} \pmod{n_I}$ 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 proof, 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.

4.5 Collaborated Attack

In practical P2P systems, subverted nodes may collaborate to enact *identity fraud* activities. Combined with the reputation auditing design, PT is also effective in defending against such collaborated attacks.

There are two extremely challenging collaborative attacks: (1) I (or R) collaborates with intruder M and (2) the tail node T_I (or T_R) collaborates with M.

When I (or R) is a co-conspirator that assists the malicious peer M, it provides its secret $\{s_{j_i}\}^k$ to M. M can impersonate I (or R). However, after authentication, I (or R) must be responsible for its bad behavior. Therefore, collaboration with M leads to I (or R) losing reputation.

Another possible situation is when the tail node T_I (or

T_R) acts as a collaborator to help the intruder to provide a forged file. In this case, T_I (or T_R) acts in a normal manner when I and R authenticate each other. This would relay all messages for I and R without modifying them until the authentication procedure completes. It would, however, replace the original file delivered from R to I with a forged one. Since PT combines a session key exchange procedure with the authentication phase, the confidentiality and integrity of the file can be protected by the session key $K = g^{ab} \bmod P$. Therefore, the tail nodes cannot substitute a forged file to cheat peer I.

4.6 PI Security in Unsafe Public Sites

As mentioned in our paper, the pseudo identity *PI* and pseudo identity certificate *PIC* of each peer are published on some well-known web sites. There exists a security problem which may jeopardize the publisher. If a powerful adversary cracks a web site which publishes the *PI*s and *PIC*s of some peers, the adversary would change some *PI*'s moduli n to another one of which he holds the factors. Thus, the adversary can forge the values used in the authentication procedure to impersonate these cracked *PI*s. Fortunately, the technique used in the *PI* generation can resist this attack. PT allows a pseudo identity *PI* to be generated from the moduli n and a seed through a hash function. Based upon the collision-resistance property of the hash function, *PI* becomes the message authentication code (MAC) of moduli n, and protects moduli n from being forged.

When an adversary forges the moduli n of a *PI*, he has two choices: 1) computing $PI'=h_3(Seed, n')$ and replacing (PI, n) with (PI', n') or 2) just replacing n with n'.

Case 1) Since our scheme is pseudo identity based, this attack would not affect the owner of *PI*. In fact, this attack is equivalent to publishing a new pseudo identity of the adversary.

Case 2) According to the generation of *PI*, a pair (n, n') satisfying $PI=h_3(Seed, n)=h_3(Seed, n')$ is called a collision of h_3. Due to the collision-resistance property of the hash function (see Section 4.1), the probability of finding such n' is $2^{-m/2}$ (where m is the length of *PI*).

In summary, we manage to defend against the forging of moduli n by binding *PI* with n through a hash function.

4.7 Comparison to PKI-Based Authentication Protocols

In some applications, authentication systems are implemented based on an asymmetric cryptographic algorithm, such as RSA. There are two application models in the Public Key Infrastructure (PKI): CA based and non-CA based. The majority of PKI systems need centralized trust servers. In those applications, a commonly trusted third party, namely Certification Authority (CA), is required to ensure the validity of the binding of a user public key and her identity. By doing so, users usually provide signatures as well as their certificates to pass one another's verification. However, this CA based model is not suitable for decentralized P2P environments.

To adopt PKI, anonymous systems remove CA. Users simply use their public keys as the pseudonyms. Further, self-signed certificate are used to authenticate the reality of a peer's pseudonym. This technique can provide anonymity for vender and vendee in a "face to face" scenario. In the complicate decentralized networks, however, this technique cannot resist the MIMA. For example, when Alice is going to download a file *f*, she first performs a search in the network. As we presented in the Subsection 4.4, an attacker in the middle of communication between two transaction participants can impersonate them without being disclosed. Due to the lack of the knowledge of communication channels, current anonymous systems are subject to the MIMA attack. The main reason is that they do not bind the knowledge of communication channels with the authentication messages. Therefore, an MIMA attacker can arbitrarily relay the messages and impersonate the transaction participants.

Compared to them, PT handles the MIMA attack better, as we discussed in the Subsection 4.4. Both the initiator and responder embed the knowledge of the path in the exchanged messages. After verifying the combined message's validation, transaction participants can detect the impersonation if it happens.

4.8 Denial of Service Attack

Denial of service (Dos) attack has gained increasing concern in distributed networks. Typically, this attack renders networks, hosts, and other victim systems unusable by consuming the bandwidth of victim networks or deluging them with a huge number of requests to overload their systems. In the design of PT, the system may suffer Dos attacks during authentication interactions. Since I chooses a desired responder from the returned responses to start an authentication procedure, I does not suffer from Dos attacks. Thus, we simply focus on the possibility that responders are under Dos attacks. Indeed, the authentication sequence of PT is well designed to defend against Dos attacks. In the PT design, R asks I to verify PI_I first, and then I asks R to verify PI_R. If we transpose this sequence, attackers may send a lot of authentication requests to the targeted R with forged *PI*s, each of whom asks R to provide a (x, u). R has to answer each request by computing a (x, u) and sending it back. R then waits for challenges from these peers, which will never come. This kind of Dos attack may devour the computational resources or available network connections of R and T_R, and render them unable to answer legitimate requests until the attack ends. To avoid this possible drawback, *PT* orders that I must prove itself to R first, and then R conducts the verification of I. Therefore, if attackers conduct Dos attacks on R by propagating numerous forged authentication requests, they must generate the same amount of valid *PI*s in advance. Otherwise, forged *PI*s would fail to pass the verification of R. For each verification request, R does nothing but generate a random number e for each I before it completes the proving step. While I has to own a valid *PI* first and computes (x, u), or if possible, provides the answer y to R before asking R to prove its PI_R. That is, the attackers should not only try to generate enough valid *PI*s, but also finish the proving phase on their side. Since the

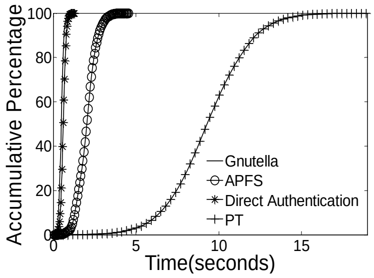

Fig. 5. Response time.

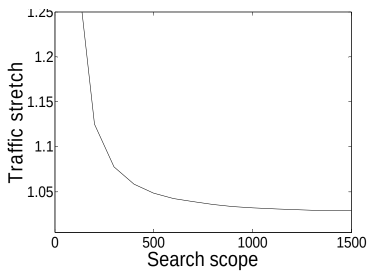

Fig. 6. Traffic stretch.

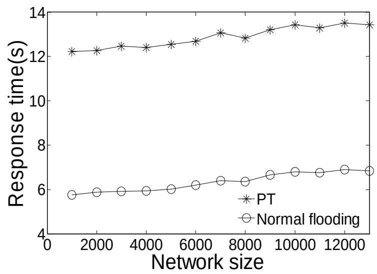

Fig. 7. Scalability of PT.

computational overhead in the verification process is relatively small compared to that in the proving procedure, a Dos attack hardly has an effective influence on *R*. Moreover, valid *PI*s used to perform Dos attacks would degrade the reputations of those pseudo identities. To continue Dos attacks, adversaries have to generate more valid *PI*s. Even if adversaries form a clique to boost each other to corrupt the reputation system, the overhead of Dos attacks is quite expensive and unacceptable. Thus, the design of PT effectively defends against Dos attacks.

5 Performance Evaluation

We first evaluate the PT design by trace-driven simulations, in which the P2P topologies are obtained from the DSS Clip2 trace. 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.

P2P networks are highly dynamic, with peers frequently joining and leaving. We simulate the joining and leaving behavior of peers by turning on and off logical peers, respectively. In our simulation, each node issues 0.3 queries per minute. When a peer joins, a lifetime in seconds is assigned to the peer. The lifetime of a peer is defined as the time period that peer stays in the system. The lifetime is generated according to the distribution observed in [24]. The mean of the distribution is chosen to be 10 minutes.

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 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 also 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.

5.3 Scalability

As the size of the P2P overlay grows, PT has good scalability in the response time and the traffic overhead. This is due to the fact that (1) no central server or CA is involved during our authentication procedures, (2) the design is efficient and the extra computation needed is relatively trivial, and (3) the underlying anonymous protocol we have selected, APFS, has good scalability.

We increase the size of the P2P overlay from 500 to

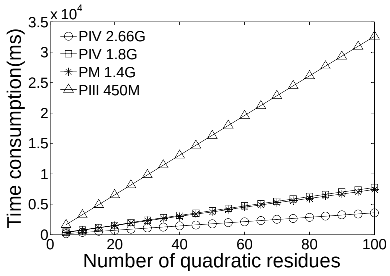

Fig. 8. Time consumption of *PIC* generation.

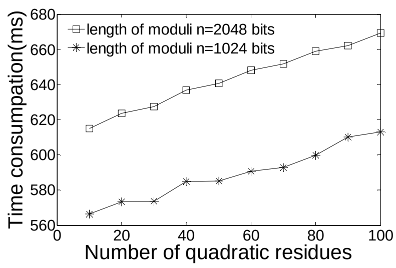

Fig. 9. Time consumption of authentication in Internet.

13,000, and observe the average response time of 10,000 queries. Results show that the growth of the P2P overlay has little impact on the PT performance, as shown in Fig. 7. In fact, PT employs an authentication procedure through the direct TCP connection and the delay is mainly related to the average distance between two nodes in the physical Internet layer.

6 Prototype Implementation

We implement a prototype in our labs at the Chinese Academy of Sciences (CAS) of Beijing, the campus of 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 tests 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 Overview

We develop PT prototype programs over VC6 and WinXP SP2 platforms. To shorten the development cycle, we use Victor Shoup's NTL [25], a number theory library which has gained wide acceptance in the cryptography community for large integer operating.

The prototype P2P servant modifies Gnutella 0.6 by adding three major components: *System initialization*, *Prover*, and *Verifier toolkit*. The System initialization deals with new peers joining, including the *PI* and *PIC* generation and issuance, parameters setup, anonymous path construction, and so on. The main authentication algorithms and their operations are conducted by the Prover and Verifier toolkit. To generate high-quality random and nondeterministic numbers, PT allows users to grab the system clock and mouse movements as the random seed resource.

6.2 Offline Experiments

We first examine the running capacity necessary for 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 8 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 *PIC* generation time grows 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 generation needs less than 8 seconds, which is acceptable as *PIC* generation is a one-time job, necessary only when a peer joins the P2P community.

We also test the time consumption of the proving and verifying operations. The experiment demonstrates that the computation overhead of the authentication is trivial.

6.3 Implementation in Internet Environments

We then implement our PT prototype in an overlay comprising of fifty desktop PCs at the labs of CAS, the campus network of HKUST, and other sites. The representative configuration of each machine includes a PIV1.8G CPU, 256MByte of memory, and a 1000M Ethernet card.

To better evaluate PT, in this implementation, we ignore the time consumed by the APFS protocol. To easily adopt PT in current P2P systems, we have included some reliable modulus in API. The primary version of the PT package is available on our public web site [26].

Figure 9 shows experimental results in the Internet. The number of quadratic residues range from 10 to 100 and we use 1024 and 2048 bits as moduli sizes, respectively.

We employ ping tests with packets of the same size as PT messages over the two involved computers between Hong Kong and Beijing in Internet. 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.

We also perform experiments in the campus area network and metropolitan area network. The experimental results can be found in our technical report [26]. To repeat our experiment, readers can find the package of PT prototype at [26].

7 Conclusions

Due to the inherent tradeoff between trust and anonymity, identity-based trust management schemes cannot be directly employed in anonymous P2P systems. We propose an anonymous zero-knowledge authentication protocol, called Pseudo Trust. In this work, a ZKP-based authentication scheme is designed to support trust management in anonymous environments, 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 defend against man-in-the-middle-attacks. The results of trace-driven simulations show that PT is scalable in both static and dynamic environments. We also implement a prototype of PT and evaluate its performance through comprehensive experiments. We believe that wide deployment of this design will provide better privacy and security for P2P users.

ACKNOWLEDGMENTS

Some preliminary results of this paper were presented in IEEE IPDPS 2007. This research was supported in part by NSF China grants No. 60573053 and 60673179, NSF China Key Project 60736016, NSF China Distinguished Young Scholar grant No. 60525209, China 973 Fundamental R&D Program No. 2005CB321803, China 863 National High Technology R&D Program No. 2007AA01Z18, Hong Kong RGC grants N_HKUST614/07 and HKUST 6152/06E, Microsoft Research Asia, and Nokia APAC research grant.

REFERENCES

[1] D. Qiu, and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *Proc. ACM Sigcomm*, 2004.

[2] W. W. Terpstra, J. Kangasharju, C. Leng, A. P. Buchmann, "BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search," *Proc. ACM Sigcomm*, 2007.

[3] Y. Liu, Li Xiao, and Lionel M Ni, "Building a Scalable Bipartite P2P Overlay Network," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, pp. 1296-1306, 2007.

[4] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, "P5: A Protocol for Scalable Anonymous Communication," *Proc. IEEE Symp. Security & Privacy*, 2002.

[5] V. Scarlata, B. N. Levine, and C. Shields, "Responder Anonymity and Anonymous Peer-to-Peer File Sharing," *Proc. International Conf. Network Protocol*, 2001.

[6] P. P. C. Lee, J. C. S. Lui, and D. K. Y. Yau, "Distributed Collabora-tive Key Agreement and Authentication Protocols for Dynamic Peer Groups," *IEEE/ACM Trans. Networking*, vol. 14, pp. 263-276, 2006.

[7] E. Damiani, D. C. d. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A Reputation-Based Approach for Choosing Reliable Re-sources in Peer-to-Peer Networks," *Proc. ACM Conf. Computer & Communication Security*, 2002.

[8] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigen-Trust Algorithm for Reputation Management in P2P Networks," *Proc. International Conf. World Wide Web*, 2003.

[9] A. Abdul-Rahman and S. Halles, "A Distributed Trust Model," *Proc. New Security Paradigms Workshop*, 1997.

[10] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative Peer Groups in NICE," *Proc. IEEE INFOCOM*, 2003.

[11] B. Dragovic, E. Kotsovinos, S. Hand, and P. Pietzuch, "XenoTrust: Event-based Distributed Trust Management," *Proc. IEEE Trust and Privacy in Digital Business Workshop*, 2003.

[12] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complex-ity of Interactive Proof Systems," *SIAM J. Computing*, vol. 18, pp. 186 - 208, 1989.

[13] J. Brandt, I. B. Damgard, P. Landrock, and T. Pedersen, "Zero-knowledge Authentication Scheme with Secret Key Exchange," *Proc. Advances in Cryptology*, 1990.

[14] O. Goldreich, "Foundations of Cryptography", Cambridge University Press, 2001.

[15] W. Mao, "Modern Cryptography: Theory and Practice", Prentice Hall, 2004.

[16] U. Fiege, A. Fiat, and A. Shamir, "Zero Knowledge Proofs of Iden-tity," *Proc. ACM conf. Theory of Computing*, 1987.

[17] Digital Signature Standard, FIPS PUB 186, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf, 2007.

[18] G. Ciaccio, "Improving Sender Anonymity in a Structured Overlay with Imprecise Routing," *Proc. Privacy Enhancing Technologies Workshop*, 2006.

[19] N. Borisov, "Anonymous Routing in Structured Peer-to-Peer Overlays," Ph.D dissertation, University of California, Berkeley, CA, 2005.

[20] A. Nambiar and M. Wright, "Salsa: A Structured Approach to Lar-geScale Anonymity," *Proc. ACM Conf. Computer and Communica-tions Security*, 2006.

[21] B. Schneier, "Applied Cryptography - Protocols, algorithms, and source coed in C, Second Ed.", John Wiley & Sons, Inc, 1996.

[22] J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[23] K. Lenstra and E. R. Verheul, "Selecting Cryptographic Key Sizes," *J. Cryptology*, vol. 14, pp. 255-293, 2001.

[24] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking*, 2002.

[25] NTL, http://shoup.net/ntl/, 2007.

[26] L. Lu, L. Hu, J. Li, J. Han, and Y. Liu, "Anonymity and Security Analysis of Pseudo Trust," Technique Report, http://www.cs.ust.hk/~liu/PseudoTrust.htm, 2007.

Li Lu received his BS and MS degrees at Zhejiang University in 2000 and 2003, respectively. He earned his Ph.D. degree in Information Secrity from Chinese Academy of Sciences in 2007. He is now a Post-doctoral Fellow in the Department of Computer Secience and Engineering at Hong Kong University of Science and Technology. His research interests include applied cryptography, network security, pervasive computing and sensor networks. He is a member of the IEEE Computer Society.

Jinsong Han (M'04) received his BS degree in Computer Science Department from Shandong University of Technology, China, in 1997, an MEng degree in Computer Science Department from Shandong University, China, in 2000, and a Ph.D. degree in Computer Science and Engineering at Hong Kong University of Science and Technology, in 2007. He is now a postdoctoral fellow in the Department of Computer Science and Engineering at Hong Kong University of Science and Technology (HKUST). His research interests include peer-to-peer computing, anonymity, network security, pervasive computing, and high-speed networking. He is a member of the IEEE Computer Society, and a member of ACM.

Yunhao Liu (SM'06) received his BS degree in Automation Department from Tsinghua University, China, in 1995, and an MA degree in Beijing Foreign Studies University, China, in 1997, and an MS and a Ph.D. degree in Computer Science and Engineering at Michigan State University in 2003 and 2004, respectively. He is now with the Department of Computer Science and Engineering at Hong Kong University of Science and Technology (HKUST). He is also an adjunct professor of Xi'an Jiaotong University. His research interests include peer-to-peer computing, pervasive computing and sensor networks. He is a senior member of the IEEE Computer Society, and a member of ACM.

Lei Hu received the B. S. degree and the M. S. degree from Peking University, Beijing, China, in 1988 and 1991, respectively, and received the Ph. D degree from the Chinese Academy of Sciences in 1994. Since 2002 he has been a professor at the Graduate School of the Chinese Academy of Sciences. His research interest includes cryptology and information security.

Jinpeng Huai is a Professor and Vice President of Beihang University. Dr. Huai serves the Steering Committee for Information Tech., the National High-Tech Program (863) as the Chief Scientist. He has published over 100 papers. His research interests include service-oriented computing and middleware, distributed computing, trustworthiness and security.

Lionel M. Ni earned his Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1980. He is Chair Professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). He is also Director of the HKUST China Ministry of Education/Microsoft Research Asia IT Key Lab and Director of the HKUST Digital Life Research Center. A fellow of IEEE, Dr. Ni has chaired many professional conferences and has received a number of awards for authoring outstanding papers.

Jian Ma received his B.Sc. and MS.C respectively in 1982 and 1987 from Beijing University of Posts and Telecommunications, and his Ph.D. in 1994 from the Department of Electronics Engineering in Helsinki University of Technology. He is now Principal Member of Research Staff, in Nokia Research Center, Beijing, responsible for university cooperation in China and other APACs, Nokia Postdoc Working Station, and Nokia-Tsinghua Joint Research Lab. He has 11 patents granted and tens of patent applications, and is author or co-author of more than 150 publications in journals and conferences, and couple of books and book chapters. He is also adjunct professor of Beijing Posts and Telecommunications, Institute of Computing Technology in Chinese Academy of Science, Graduation University of Chinese Academy of Science, Tongji University, and Beihang University. His current research interests are consumer services on wireless sensor networks, mobile Internet applications, pervasive computing, ad-hoc and P2P networking, etc.