Authentication with controlled anonymity in P2P systems

Adam Wierzbicki
Polish-Japanese Institute
of Information Technology
Warsaw, Poland
adamw@pjwstk.edu.pl

Aneta Zwierko[1]
[1]Warsaw University
of Technology, Institute
of Telecommunication
Warsaw, Poland
azwierko@tele.pw.edu.pl

Zbigniew Kotulski[1,2]
[2]Polish Academy of Sciences
Institute of Fundamental
Technological Research
Warsaw, Poland
zkotulsk@ippt.gov.pl

Abstract

This paper describes a new protocol for authentication in Peer-to-Peer systems. The protocol has been designed to meet specialized requirements of P2P systems, such as lack of direct communication between peers or requirements for controlled anonymity. At the same time, a P2P authentication protocol must be resistant to spoofing, eavesdropping and playback, and man-in-the-middle attacks. The protocol is studied for a model P2P storage system that needs to implement file access rights.

1 Introduction

The requirement of peer authentication has been recognized by many designers of P2P systems. As new applications of the P2P model become available (such as P2P games or P2P auctions), the need for authentication becomes even more pressing. Many P2P applications rely on reputation systems that cannot operate without a form of authentication.

At the same time, P2P systems are more vulnerable to a number of well-known threats, such as identity theft (spoofing), violation of privacy, and the man-in-the-middle attack. The Sybil attack (identity cloning [3]) has become a concern in P2P systems that require reliable storage, use reputations or voting. All these threats are difficult to counter in an environment where membership is dynamic and the presence of central directories cannot be assumed.

In addition, many P2P systems have special anonymity requirements that cannot be easily reconciled with some forms of authentication known today. On the other hand, service providers that are bound by legal regulations have to be able to trace the actions of certain peers. Finding a reasonable trade-off between these two requirements is rather hard. In this paper, we use the term *controlled anonymity* for a system in which a peer cannot be identified to the outside world, but a trusted authority (superpeer) is provided with the possibility to identity actions performed by each peer.

These considerations lead to the conclusion that P2P systems can benefit from new, specialized methods of authentication. In this article, we combine two cryptographic techniques - Merkle's puzzles and zero-knowledge proofs - to develop a protocol for authentication in P2P systems. This protocol is resistant to man-in-the-middle and eavesdropping attacks and prevents identity theft. However, the protocol allows for controlled anonymity of peers and is adapted to the dynamic and decentralized nature of P2P systems.

We study the protocol for a model P2P storage system that needs to implement access rights to files. However, the applications of an authentication protocol in P2P systems can be wide, and our authentication protocol can be adapted to many other applications.

Organization of paper. In the next section, we review current state of art for anonymity and p2p networks. In section 4.1, we present and explain the cryptographic primitives used in our protocol. Section 4.2 presents the protocol, and concludes with an analysis of the protocol's security and efficiency. Section 5 concludes and discusses further work.

2 Related work

General P2P security architectures almost exclusively use public key cryptography (PKI or the Web of trust) [6]. These systems provide authentication without anonymity, and will be discussed in more details below.

FreeHaven uses a system described in [2] that allows for accountability. The approach is based on micropayments that utilize the concept of client puzzles. The puzzles used base on hash function and time-lock puzzles. In our work, we extend the mechanism of puzzles by combining it with

zero-knowledge protocols. In such a way, we achieve a novel authentication method.

Most systems that provide anonymity are not interested in allowing to trace the user under any circumstances. *Chaum mixing networks*, proxy servers, have not been designed to provide accountability. For P2P systems, approaches exists that provide unconditional anonymity, again without any accountability [10, 11].

Crowds ([9]) is a system that provides anonymity for Web browsing. This system hides the action of one user in actions of other users. All users are called *the crowd*, and a proxy server issues requests on behalf of its users. The content servers cannot determine which user requested which content. This system does not offer any authentication mechanism.

A Chaum mixing network, mentioned earlier, is a collection of special hosts (mixing nodes) that route user messages. Each node simply forwards an incoming messages to other nodes in the mixing network. The path (sequence of nodes) is chosen by the sender, and the message is put into envelopes (based on PKI infrastructure), one for each node on the path.

3 Discussed P2P model

Consider a storage system that uses the Peer-to-Peer model. The system makes it possible for peers to execute two operations: $STORE(k, f)$ and $RETRIEVE(k)$. The first operation stores a file, f, on one of the peers in the network, associating the stored file with a key, k. The second operation retrieves the file that has been associated with the given key. Note that a peer that executes the STORE operation need not to know which peer stores the file in the network.

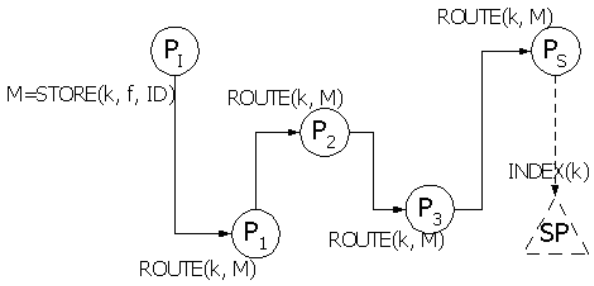


Figure 1. Routing of a *STORE* request in a P2P storage system

The described system is visualized on figure 1. The $STORE$ and $RETRIEVE$ requests are routed by the network using the $ROUTE$ operation. On the figure, the $STORE$ request is routed from peer P_I to peer P_S by the peers P_1, P_2, and P_3. The described P2P network is sufficiently general to model any structure P2P network (or Distributed Hash Table). We shall also assume that the key, k, is available to all peers in the system after the $STORE$ operation. This can be achieved using a superpeer that is notified by the storing peer that a new key is used in the system. However, this may not be necessary if the storage system uses keys that are universally known.

Consider now that the system wishes to enforce write permissions. For example, only the peer that has stored the file is authorized to modify the file. Everyone else has read permissions. In order to enforce the permissions, some form of authentication is required.

The first question is whether the peer identifier (ID) in the P2P network is sufficient for authentication. Is it possible for a malicious peer, P_M, to assume the ID of an innocent peer, P_I? Let us assume that P_I stores a file using his own ID as authentication information. The $STORE$ operation takes the form of $STORE(k, f, ID)$. However, P_M runs a DoS attack against P_I, forcing P_I to leave the network. After P_I has left, P_M joins the network assuming the ID of P_I. Next, P_M can execute the operation $STORE(k, f', ID)$.

What is needed to implement file access permissions? An authentication mechanism that allows the storing peer to authenticate the owner, does not use centralized control during authentication, provides controlled anonymity, is safe against playback attack, eavesdropping and man-in-the-middle attack.

4 Proposal

In this paper, we describe a new protocol for authentication for P2P storage systems. The protocol allows a peer to securely store a file on another peer and to change it (as described in section 3).

First, utilized cryptographic primitives are briefly introduced: the concept of zero-knowledge proofs and Merkle's puzzles. Then, we present the authentication protocol.

4.1 Cryptographic primitives

Our scheme involves two cryptographic primitives: Merkle's puzzles and zero-knowledge proofs. We describe them shortly below.

Merkle's puzzles. Ralph Merkle introduced his concept of cryptographic puzzles in [7]. The goal of this method was to enable secure communication between two parties: A and B, over an insecure channel. The assumptions were that the communication channel can be eavesdropped (by any third party, called E). Assume that A selected an encryption function (F). F is kept by A in secret. A and B agree on a second encryption function, called Gwhich is publicly

known. A will now create M puzzles (denoted as s_i, $0 \leq i \leq M$)in the following fashion:

$$s_i = G((K, X_i, F(X_i)), R_i)$$

K is simply a publicly known constant term, which remains the same for all messages. The X_i are selected by A at random. The R_i are the "puzzle" part, and are also selected at random from the range $(M \cdot (i-1), M \cdot i)$. B must guess R_i. For each message, there are N possible values of R_i. If B tries all of them, he is bound to chance upon the right key. This will allow B to recover the message within the puzzle: the triple $(K, X_i, F(X_i))$. B will know that he has correctly decoded the message because the constant part, K, provides enough redundancy to insure that all messages are not equally likely. Without this provision, B would have no way of knowing which decoded version was correct, for they would all be random bit strings. Once B has decoded the puzzle, he can transmit X_i in the clear. $F(X_i)$ can then be used as the encryption key in further communications. B knows $F(X_i)$ because it is in the message. A knows $F(X_i)$ because A knows X_i, which B transmitted in the clear, and also knows F, and so can compute $F(X_i)$. E cannot determine $F(X_i)$ because E does not know F, and so the value of X_i tells E nothing. E's only recourse is to solve all the N puzzles until he encounters the 1 puzzle that B solved. So for B it easy to solve one chosen puzzle, but for E is computationally hard to solve all N puzzles.

Zero-knowledge proofs. A zero knowledge proof system ([8], [4], [1]) is a protocol that enables one party to *prove* the possession or knowledge of a "secret" to another party, without revealing anything about the secret, in the information theoretical sense. These protocols are also known as minimum disclosure proofs. Zero knowledge proofs involve two parties: the prover who possesses a secret and wishes to convince the verifier, that he indeed has the secret. As mentioned before, the proof is conducted via an interaction between the parties. At the end of the protocol the verifier should be convinced only if the prover knows the secret. If, however, the prover does not know it, the verifier will be sure of it with an overwhelming probability.

4.2 The authentication protocol

The proposed protocol offers an authentication method for the model P2P storage system. The peer that wishes to store a file is equipped with a zero-knowledge value. After storage, this value will enable only the right peer to modify the previously stored file. Using the proposed protocol, the authentication information cannot be used by a peer that routes the message for its own purpose. A short overview is presented in this section and a detailed description in the next.

The proposed protocol has three phases. Initial, when a superpeer or bootstrap creates necessary values for authentication. 2^{nd}, storage of the file: the file is initially stored with additional zero-knowledge values that will enable the owner (and no one else) to modify the file and 3^{rd}, modification of the file: the owner uses a zero-knowledge proof and Merkle's puzzles to authenticate itself and to safely store the modified file.

The peer that owns the file is denoted as P_I, the storing peer as P_S and peers that route the message as $P_1, P_2, \ldots$, the file as f' (first version) and $f'', f''', \ldots$ (next versions). A is the authentication data.

In this basic scenario we assume that routing peers do not modify the data, just forward it correctly. Attacks: scenarios where these peers can modify or eavesdrop information are described in section 4.3.

Phase 1 - initial. This proposal is not directly based on zero-knowledge protocols, but on an identification system based on a zero-knowledge proof. We choose the GQ scheme ([5]) as the most convenient for our purposes. In this scheme, the superpeer or bootstrap has a pair of RSA-like keys: a public K_P and a private one k_p. The superpeer also computes public modulus $N = p \cdot q$, where p, q are RSA-like primes. The following equation has to be true:

$$K_P \times k_p \equiv 1 (\text{mod } (p-1) \cdot (q-1)).$$

The pair (K_P, N) is made public. The keys can be used for different purposes, not only for our system.

The superpeer computes a set of so-called identities, denoted by ID, and their equivalencies, denoted by J. It does not matter how J is obtained if it is obvious for all participants how to obtain J from ID. The pairs (ID, J) are generated for every peer that requests them. The identity is used to authenticate P_I during an attempt to modify the file. The superpeer also computes a secret value for each ID:

$$\sigma \equiv J^{-k_p} (\text{mod } N).$$

The secret σ is used by P_I to compute correct values for the GQ authentication scheme. P_I obtains the following information in the initial phase: ID (public) and σ (secret).

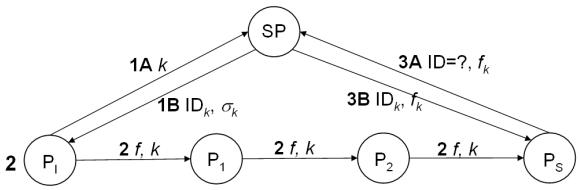
To preserve anonymity, peer P_I should request at least a few different pairs (ID, σ) or, if possible, obtain a new pair for each stored file (key).

Phase 2 - storage of the file. The purpose of this phase is to associate a proper ID with the file. Different methods may be used for that purpose, depending on the security and performance requirements of the system.

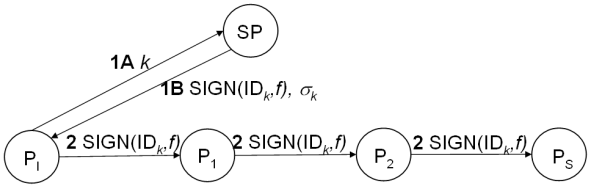
Here are some possibilities:

1. The peer P_I can simply send the ID with the file f in open text. In that situation, the peer P_I has to trust all other peers that they do not change neither file nor ID.

2. The peer P_I can ask the superpeer to store the files' key (k) and a ID value. After initial storage, P_S contacts the superpeer and obtains the proper ID.



3. A more secure way is to use the superpeer's keys for a different purpose, not only for the zero-knowledge protocol. After creation of an ID for peer P_I in the initial phase, the superpeer can sign the ID with his private key. In this case, the ID can be sent securely over multiple peers. After receiving the file, P_S can check the validity of the superpeer's signature and accept only a valid ID. To provide file integrity, the superpeer would have to sign a hash of the file ($h(f)$), as well. Another possibility is that the superpeer signs



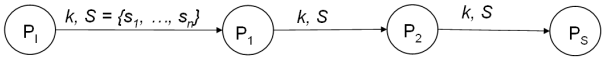
pairs of (ID_i, k_j): different IDs and different keys for files. Then, when P_I stores a file using key k_j, he will use ID_j as an identifier and send the signed pair to prevent modification by P_M.

Phase 3 - modification of the file.

1. The peer P_I creates a set of puzzles: $S = \{s_1, \ldots, s_n\}$. Each puzzle has a zero-knowledge challenge. This challenge is a number computed basing on a random value r, $r \in \{1, \ldots, N-1\}$. It is computed as following: $u = r^{K_P} \pmod N$.

   Creating a set of puzzles. Each puzzle used in the proposed scheme has a following form: $G(K, X_i, F(X_i), u), R_i)$, where K, X_i, R_i and F, are described in section 4.1 and it can contain a different u value (computed from r), which gives additional security.

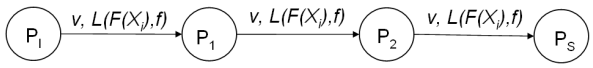
2. The peer P_I sends the whole set of puzzles to P_S.



3. P_S solves a chosen puzzle and chooses a random value $b \in \{1, \ldots, N\}$. P_S sends the puzzle's number (X_i) and b to P_I.

4. The P_I computes the next value in the GQ scheme, v. This values is based on the number b received from P_S and on the secret value σ of P_I: $v \equiv r \times \sigma^b \pmod N$.

5. P_I sends v and new version of the file (encrypted, using information from the puzzle). Some possible methods of securing the file are described below. The secured file has the form: $L(f', F(X_i))$.



6. P_S uses information extracted from the puzzle, ID, to obtain J and verify if v is the right value. To validate the response from P_I, P_S checks if $J^b \times v^{K_P} \equiv u \pmod N$. If the equation is satisfied, then the new version of file is accepted.

   Securing the new version of file. The value $F(X_i)$ is a secret known only to P_S and P_I. Thus, it can be used to establish a secure channel for the new version of the file. This can be used to provide either encryption - the file could be encrypted using $F(X_i)$ as a key for a symmetric cipher or integrity - the hash of the file could be encrypted using a symmetric cipher with key $F(X_i)$.

4.3 Security of proposed scheme

In this section, we are going to discuss only the security of phase 3 of the protocol, since the protocol offers several distinct possibilities in phase 2, each one with a different level of security.

Modyfing the file by an unauthorized user. Assume that one of the routing peers (P_M) wishes to modify the new version of the file or store its own version. P_M cannot store its own version (or masquerade as P_I), because P_M does not have the σ value to obtain the correct v for the authentication phase of the protocol. The values u, b or ID do not contain any information that would be useful in cheating P_S. This property is assured by the zero-knowledge protocol.

The file itself is secured by methods described in sec 6, using the $F(X_i)$ value. For any eavesdropper it is computationally infeasible to solve all puzzles to find the puzzle the with proper X_i (the one used by P_S) if the number of puzzles is large enough. E.g. if the function G would be DES, and R_i would be a key for a cipher with fixed 24 bits (so efficiently 32 bits long), then the number of computations

required to solve one puzzle is about 2^{31}. Now it is easy to estimate how many puzzles should be created by P_I.

Eavesdropping. An eavesdropping peer, P_M, can observe all values of the zero-knowledge protocol: u, b and r. This knowledge does not reveal anything about the secret σ and since u and b are random and change in every iteration of the protocol, that does not enable P_M to interfere and gain any important information. Also, if the number of puzzles is sufficiently large, solving all puzzles is infeasible in reasonable time and finding the puzzle that was used to secure the file is hard.

Play-back attack. Using our protocol, P_S chooses a random value b and then P_I has to compute the v value, which is later utilized by P_S to check if the authentication is successful. Therefore, previously used v, r (u) and b values are useless. Only P_I is able to create the proper v value for a random b.

Man-in-the-middle attack. The goal of this attack is to either change the new version of the file or to gain some information about σ by one of the intermediate peers (P_M). A property of zero-knowledge proofs used in our protocol is that gaining any information about u, b and v values does not reveal anything about the σ. Changing the file is also not possible since it is protected with the secret value $F(X_i)$, known only to P_I and P_S.

5 Conclusions

We have developed an authentication method that is secured against eavesdropping, man-in-the-middle, and play-back attacks in a P2P system, but does not require direct communication. The proposed method does not introduce significant computational or communication overheads. Also, the proposed method provides controlled anonymity that is not available when *PKI* is used.

The proposed system of authentication with controlled anonymity gives quite new possibilities for security solutions in p2p networks. First, it provides anonymity of the operating peer against other peers and any external users, except for the superpeer. Additionally, the system makes it possible to identify a peer's actions when cooperating with the superpeer. If practically implemented, the system can be controlled against a malicious peers trying violate the rules of a P2P application. The applications of this control can range from fairness in P2P games to the protection against the use of P2P storage systems by terrorists or criminals.

Future work. The form of anonymous authentication and controlled anonymous authentication should perhaps depend on the particular P2P application. Thus, the first possible extension of the results presented here is a precise analysis of requirements of chosen P2P applications. This problem will be the subject of future research.

Another extension of the presented results is to offer new security services. The first natural proposition is mutual authentication of peers, then non-repudiation of operations and finally combinations of all common security services applied to peers and the content of transferred files.

References

[1] Brandt J., Damgard I., Landrock P., Pedersen T.: Zero-Knowledge Authentication Scheme with Secret Key Exchange. *Journal of Cryptology 11 (3)*, 1998.

[2] DingledineR., Freedman M. J., Molnar D.: Accountability, Peer-to-Peer: *Harnessing the Power of Disruptive Technologies*, O'Reilly, 2001.

[3] Douceur, The Sybil Attack. In *Proc. of the IPTPS02 Worksho*p, Cambridge, MA (USA), March 2002

[4] Goldreich O.: *Foundations of Modern Cryptography*, 2001-2004, Cambridge University Press.

[5] Guillou L.C., Quisquater J-J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. *Lecture Notes in Computer Science on Advances in Cryptology-EUROCRYPT'88*, 1988. ISBN:0-387-50251-3.

[6] Kim W., Graupner S., and Sahai A.: A secure platform for peer-to-peer computing in the internet. In 35^{th} *Annual Hawaii International Conference on System Sciences*, January 2002.

[7] Merkle R.: Secure Communications over Insecure Channels. *Communications of the ACM*, April 1978 (pp. 294-299).

[8] Pieprzyk J., Hardjono T., Seberry J.: *Fundamentals of Computer Security*, Springer, Berlin 2003.

[9] Reiter M. K., Rubin A. D.: Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, Vol. 1, No. 1, November 1998, pp. 66-92.

[10] Scarlata V., Levine B., Shields C.: Responder Anonymity and Anonymous Peer-to-Peer File Sharing. *Proc. IEEE Intl. Conference on Network Protocols (ICNP)*, November 2001.

[11] Sirer E., Polte M., Robson M.: CliqueNet: A Self-Organizing, Scalable, Peer-to-Peer Anonymous Communication Substrate, White Paper, Cornell University, 2001.