A Self-organized Public-Key Certificate System in P2P Network

Zhen Luo and Zhishu Li
School of Computer Science, Sichuan University, Chengdu, Sichuan, P. R. China, 610065
Email: Austin.luo.sh@gmail.com

Biao Cai
Network School, Chengdu University of Technology, Chengdu, Sichuan, P. R. China, 610059

Abstract—P2P network is open, anonymous and dynamical in contrast with conventional network. It’s full of dishonesty, deceit, selfishness. It’s necessary to build a robust security infrastructure system. The one important module of security infrastructure is to identify the users in transactions. This paper will propose an identification solution based on PKI. In our algorithm, the peer generates their public-private key pairs by themselves, issues certificates, and performs authentication without any centralized services. It can realize the identification, meanwhile keep peers anonymous. Moreover, it can limit the quantity of ID that a user applies for; alleviate the risk of Sybil attack.

Index Terms—Peer-to-Peer, security, key authentication, public-key cryptography, PKI, DHT

I. INTRODUCTION

In transaction system, Identification and authentication are the essential infrastructure; it’s the same in a variety of p2p applications. Trustor needs to identify the trustee user before making deal with it.

In this paper, we propose a public-key management system based on DHT (discrete hash table) that allows users to create, distribute, and revoke their public keys without the help of any trusted authority or fixed server. Moreover, there are no specific missions to a subset of nodes; all the nodes take the same role.

How to publish and verify users’ public-key is the main problem of public-key management. The well known approach to the public-key management problem is based on public-key certificates. A public-key certificate is a data structure in which a public key is bound to an identity and possibly to some other attributes by the digital signature of the issuer of the certificate. But it’s difficult for the absence of centralized center in P2P network.

In our proposal, we build up a DHT circle to manage user’s public-key or certificate which is created by user itself.

The paper is organized as follows: In Section 2, we present an overview of the existing proposals for public-key management in P2P networks. In Section 3, we describe the basic self-organized public-key management of our proposal. In Section 4, we present how to cope with malicious actions in our proposal. In Section 5, we conclude and give some remarks on our future work.

II. STATE OF THE ART

Some solutions have been proposed for public-key management in P2P network or similar mobile ad hoc networks.

A distributed public-key management service [1] called partial distributed CA method is proposed for ad hoc networks. It has a public/private key pair K/k that is used to verify/sign public-key certificates of the network nodes. In the proposal, assume that all nodes in the system know the public-key K and trust any certificates signed using the corresponding private key k. The private key k is divided into n shares using an (n, t) threshold cryptography scheme and the shares are assigned to n arbitrarily chosen nodes, called servers or super node. When issuing a certificate, each server generates a partial signature for the certificate using its private key share and submits the partial signature to a combiner that computes the signature from the partial signatures. The application of threshold cryptography ensures that the system can tolerate a certain number t (t < n) of compromised servers in the sense that at least t+1 partial signatures are needed to compute a correct signature. In the proposal, it’s vulnerable to Sybil attack when collusive fraud peers reach to a certain scale.

Another approach [2] called distributed CA method is similar. The main difference between them is that it provides a more fair distribution of the burden by allowing any node to carry a share of the private key of the certificate service. The advantage is increased availability since any t+1 node in the local neighborhood of the requesting node can issue or renew a certificate. The defect is similar with the former approach but it’s more vulnerable to collusive attack than the former.

Both of the former 2 proposals are not self-organized and not suitable for P2P network. Meanwhile they are dependent on TTP (Trusted Third Party) or offline TTP, and once the private key k is disclosed, TTP needs to be involved and initiate the process again. When malicious peers reach to a certain degree, the certificate management will become not secure any more. Finally their certificate management process is relatively complex.
In Certificate Chaining (also named by Web of Trust) proposal [7], key authentication is performed via chains of public-key certificates. When a user u wants to obtain the public key of another user v, she acquires a chain of valid public-key certificates to v. each user is her own authority and issues public-key certificates to other users. Certificates are stored and distributed by the nodes and each node maintains a local certificate repository that contains a limited number of certificates selected by the node according to an appropriate algorithm. Key authentication is performed via chains of certificates. When user u wants to verify the authenticity of the public key of another user v, they merge their local certificate repositories and u evaluates the authenticity of K, based on the certificates contained in the merged repository. The detection of false certificates is enabled through the certificate exchange scheme that allows nodes to detect any conflicting certificates. The proposal’s advantage is it is independent of TTP or offline TTP. However, there are 2 defects in the approach. Firstly once user logouts, it’s easy for malicious peer to impersonate it. So user is vulnerable to lose its ID. Secondly it’s easy for malicious user to create several IDs for network attack.

ID-KPC (Identifier-Based Public Key Cryptography) takes user id or IP address as public key, it doesn’t need PKI and certificate repository, because the public key just is the user’s identity. The ID-PKC requires a trusted authority, and key master guardian, called Private Key Generator (PKG). The PKG can impersonate any user at any moment since secret keys are calculated by it, so it has key escrow, then it is impossible to guarantee non-repudiation.

III. BASIC OPERATION OF OUR SOLUTION

A. Peer Organizing

In our proposal, we organize peers on DHT as Chord; the discrete hash function assigns every peer an m-bit identifier using a general hash function such as SHA-1. A Chord like peer’s identifier is chosen by the peer’s IP address and port number. We use the term ‘key’ to refer the hashed value of the unique identifier under the hash function. The identifier length of m must be large enough to make the probability of two peers hashing to the same key is negligible. All peers are organized in a circle according to their keys with ascendant order. Peers organized in such circle are showed in Figure 1.

After we organize the peer in such circle, we can store some information in that circle. The approach is to calculate the key of the information by the same Hash function or the other which can map the date into the same hash value space. K stands for the key of information here; the information will be stored in the successor of its key K in the circle just like Figure 1.

Every peer in the circle needs to maintain 3 kinds of information table: routing table (finger table), certificate register table, IP record tables.

a) Routing table (finger table)

Routing table stores m routing information of which K is \( N + 2 \) \( \mod 2^n \) \( 0 \leq i \leq m - 1 \)

\[ K_i = \text{N} + 2 \mod 2^n \]

b) Certificate register table

For example, in Table 1, for peer N0, its routing table is:

<table>
<thead>
<tr>
<th>i</th>
<th>K_i</th>
<th>N_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N0+1</td>
<td>N6</td>
</tr>
<tr>
<td>2</td>
<td>N0+2</td>
<td>N6</td>
</tr>
<tr>
<td>3</td>
<td>N0+4</td>
<td>N6</td>
</tr>
<tr>
<td>4</td>
<td>N0+8</td>
<td>N12</td>
</tr>
<tr>
<td>5</td>
<td>N0+16</td>
<td>N23</td>
</tr>
<tr>
<td>6</td>
<td>N0+32</td>
<td>N32</td>
</tr>
</tbody>
</table>

According to the routing mechanism, we can seek a target peer in finite hops. Information is stores in related peer, so we can locate the information through the approach. Figure 2 shows an example of routing.

b) Certificate register table

Every peer will store some certificate information which corresponds with it according our approach in its certificate register table. A user calculates its username’s hash value by predefined hash function, and then stores its certificate in corresponding peer. Every peer needs to maintain the table during its life cycle. It will be described in more detail in the sections that follow.

c) IP record table

To alleviate the risk of Sybil attack, we limit the quantity of username which are registered or activated.
B. Certificate management

To maintain certificate management of peers based on DHT in the P2P network, each peer will be assigned \( r \) continuous certificate managers to manage certification information of its own. The certificate manager of peer \( i \) is assigned as follows: if peer \( j \) is a certain successor of \( h_{ID(i)} \), where \( h_{ID(i)} \) is the hash value of username or ID of peer \( i \) hashed by the predefined hash function, and then allocate the peer \( j \) and the next \( r-1 \) successors of peer \( j \) as the certificate manager of peer \( i \). If other peers want to get certificate of a peer \( i \), they may issue a lookup on the manager and obtain the certificate from the manager.

To make sure the certificate safe, we have to use \( r \) redundant certificate managers. Redundancy is to avoid the harm in the following cases:

(a) Malicious peer

If the certificate manager is a malicious peer, it may change the certificate of target peer or replace the user’s certificate with its own certificate in case the target peer accumulates a high or good reputation. It would disguise itself as the target peer to do some malicious actions taking advantage of target peer’s good reputation.

(b) Peer leaving

Sometimes peer maybe would crash or leave network accidentally without notification, and then the information on it would be lost. For routing information, according to Chord natural mechanism, it will be rebuilt but the other information will be lost.

To resolve the both issues, we give every peer \( r \) certificate managers. When certificate conflict happens, the certificate is decided by \( r \) certificate managers together. The result of certificate is the most value of them.

In Chord cycle, the distributions of neighbor peers are well balanced in real network. The probability that these neighbor peers get into trouble together is very low, for example they are controlled by malicious peer or drop down together. Moreover, neighbor is easy to locate. So we adopt \( r \) continuous peer as a peer’s certificate managers.

These certificate managers need to check on schedule if the other managers dropped out, if so, they will update their information with new partners.

Under such mechanism, the probability that malicious peers can change someone’s certification is very little. To do that, malicious peers must own more than \( r/2 \) relevant certificate managers.

In our Chord-circle, every peer get its ID according to its IP address and port number, it cannot select its ID arbitrarily. So in our case, to own most of relevant certificate managers for a user, malicious peers must depend on good luck.

For a \( m \) bit Chord circle, assume that there are \( N \) peers in the circle, the probability that malicious peers take control more than \( r/2 \) relevant peers is:

\[
P = \frac{C_r^{N-[r/2]}}{C_r^N} / C_r^N
\]  

(1)

When \( N \) is enough large, the probability malicious peers take control becomes very little, we can neglect it. For dishonest certificate manager, we will give it penalty; even exclude it from network community.

The whole model consists in four parts: the creation of public/private key pairs, the issuing of certificates, the certificate authentication, and the certificate revoke.

(a) The creation of public/private key and public-key certificate

In our model, the public key and the corresponding private key of each user is created locally by the user herself. The public-key is for other user to authenticate user. The private-key is used to sign. Username is identifier of user and it is also created locally. When a peer joins a P2P network, the peer will find and take its position by DHT rule firstly, the process just likes in Chord network, then it will declare its username and issue public-key certificate for itself. Every username must be unique; it’s the user’s identifier. Username is created by user itself in public/private creation and verified if it’s unique during the certificate register.

A certificate contains the following content:

- **Owner ID:** hash value of username, it records the owner of certificate
- **Version:** certificate version number
- **Release Date:** date of certificate release
- **Expire Date:** date of certificate expiration

© 2011 ACADEMY PUBLISHER
Certificates are issued with a limited valid period $T$, and each certificate contains its issuing and expiration times. For safety, to avoid some users issue certificate with unreasonable long valid period. We setup a threshold value $T_0$. If $T > T_0$, We think $T$ is invalid, we will take $T_0$ as default.

The public key is issued with the certificate. The private key is kept by user itself. If the private key is disclosed, the peer has to change the public/private pair key. It will revoke the certificate firstly, then issue new certificate. Other peers can decrypt the signature with public-key to check the certificate. If the decrypted value is equal to the hash value of certificate content and the valid period doesn’t expire, the certificate is valid, or it’s invalid. The user should drop it and ask target user to update its certificate.

b) Issue of certificate

After coming into position in DHT circle, the first thing the peer will do is declaring its user identifier bound with a public-key certificate.

It will locate the certificate manager according to $h_{iD(i)}$ and send certificate register message to the manager, the message includes related certificate. When the expected manager receives the message, firstly it will check if the certificate is correct. It will decrypt the signature with the public-key in the certificate, and verify the data. If it’s not correct, the manager will reject the request. If it’s correct, the manager will search in its certificate register table. If the $h_{iD(i)}$ has been registered. The certificate manager will reject the request, the requester has to recreate the username and related public/private key, redo the process again. If the $h_{iD(i)}$ is available, the certificate still need to check if the peer applied or activated too many username to avoid user makes use of many IDs to commit malicious action for example Sybil attack. We use another hash function $h'$ based on the same value space to calculate the key of the requester’s IP address, and according to $h'$ to locate where the quantity of the username requested from a certain IP is stored. We can replace $h_i$ with $h_i + h_c \mod 2^m$, here $c$ is constant value. It’s also effective. We have a threshold value $N$ for every IP, if the number of registered user from an IP is larger than $N$, the certificate manager will reject the request, or it will increase the number by 1 and add a record in IP record table in IP record peer, then accept the request and register $h_{iD(i)}$ in certificate register table of certificate manager.

c) Certificate authenticate

When a peer $A$ wants to transact with the other peer $B$, they need to authenticate each other by certificate or public-key. If peer $A$ has no public-key of $B$, it will issue a searching action according to the username of $B$, first it calculate $h_{iD(i)}$, then locate the certificate manager $C$ of $B$ corresponding to $h_{iD(i)}$. The approach is similar with which mentioned above. If $C$ has a certificate for $B$, it will return it to $B$. If not, it means $B$ is not registered yet and not an authentic peer.

When $A$ gets certificate of $B$, firstly $A$ check if the certificate is correct and valid, then it will challenge $B$ by sending random data to $B$. After $B$ receive the random data, $B$ send the signed content back with its private-key. $A$ check the content with $B$’s public-key, if correct, the $B$ is the true one, or it’s fake.

Vice versa, $B$ can authenticate $A$ in the same way. A needn’t to get $B$’s certificate from certificate manager if $A$ has $B$’s certificate (they transacted before) and the certificate doesn’t expire.

Under 3 cases, peer needs to get the other peer’s certificate from the other peer’s certificate manager.

1) It has not the certificate of the other peer
2) The related certificate expires
3) Authentication fails with the old certificate, maybe the peer updated/revoked its certificate

After they get certificate of each other, they can start the transaction. There are 2 authentication methods of mutual and unilateral. The basic step which two peers authenticate each other at first time is as follow:

1) Mutual authentication
   
   User $B$→User $A$: $R_A$;
   User $A$→User $B$: $R_B||[R_A||B]||\text{Sig}_A(R_A||R_B||B)$;
   User $B$: get certificate of $A$ from its certificate manager and authenticate $A$. 
   User $B$→User $A$: $R_A||[R_A||A]||\text{Sig}_A(R_A||R_A||A)$;
   User $A$: get certificate of $B$ from its certificate manager and authenticate $B$.

$R_A$ and $R_B$ are random value, produced by $A$ and $B$ respectively. We include the context in the message to avoid Wiener attack.

2) Unilateral Authentication
   
   User $B$→User $A$: $R_B$;
   User $A$→User $B$: $\text{Sig}_A(M, R_B)$;

In both cases, the receiver authenticates the sender though decrypting the message using the public key of target user and comparing the value with the original value, if equal, the receiver accept the authentication, or reject it.

To enhance security, we can add timestamp in the communication protocol. Message receiver can compare timestamp $T$ with local time, if the difference between them is small enough, it will think the data is fresh. The mechanism depends on clock synchronization. If adopting timestamp, unilateral authentication can be simplified as:

   User $A$→User $B$: $\text{Sig}_A(M, T_A)$;

After users authenticate each other, they start making deal. During the transaction, we still can use public/private key mechanism to encrypt or verify the transmitted data.

d) The certificate revoke and update

Every user can revoke or update a certificate that it issued if it believes that the user-key binding expressed in
that certificate is no longer valid. Moreover, if a user believes that its own private key is disclosed, it can revoke its corresponding public key and update the certificate. There are two certificate revocation approaches: explicit and implicit. In the explicit revocation approach, to revoke a certificate that the peer issued, the user issues an explicit revocation message to its certificate manager. The certificate manager will authenticate if the peer is the true one by challenge approach. After that it will revoke the certificate according to the message.

The implicit revocation approach is based on the expiration time of the certificates. Specifically, each certificate is implicitly revoked after its expiration time. As we already described, each certificate contains its issuing time and expiration time. After this period elapses, the certificate is not considered valid anymore.

The process is the inverse of certificate issuing. The certificate manager will delete the certificate record when the expiration time comes, and decrease certificate record of related IP.

In some application cases, we will have a threshold period $T_{\text{max}}$. During a period $T_{\text{max}}$, if there are no any action on the user’s certificate, for example certificate searching, updating, the certificate manager will think the user has left the P2P network and never come back, will revoke its certificate. So the $T_{\text{max}}$ should be enough long to avoid revoking wrongly.

These approaches allow each user to react to any detected misbehavior by issuing a revocation message. In addition, key revocation enables users to perform authentication with a higher confidence in the validity of the certificates, but also in the correctness of the user-key bindings contained in the certificates because of their limited validity period.

Key revocation is based on the same approach as for certificate revocation: If a user believes that its private key has been disclosed, it revokes its corresponding public key in question by sending message to its certificate manager.

Apparently everyone has strong incentives to keep their certificate valid and updated, in order to provide sufficient proof of the authenticity of their public keys to other users and to be able to correctly authenticate.

IV. COPING WITH MALICIOUS ACTIONS

Firstly, there are some defects to Use IP and port as identifier. IP address and port are user’s network attributes; they are easy to be faked by malicious users. User’s status is dynamical, user maybe use different IP address during different term, so you cannot identify user according to IP address and port. Moreover, except user ID identification, we need to keep data secure, integrated, and consistent, and transaction non-repudiation in P2P system, only dependence on IP mechanism can’t resolve these issues. In our proposal, we use username as long term unique identifier for user. With our mechanism, a user can only gain limited quantity of IDs (username) according its IP address, It alleviates the risk of Sybil attack. We can change the mechanism a little to let only one or certain user IDs activation from an IP address at the same time. It’ll help to keep the network health.

Secondly, the certificate of a user is stored by r consecutive certificate managers. So it’s difficult for a certificate manager to issue false certificates and try to deceive a given user that the certificates are correct. Moreover, as described, the probability which collusive malicious take control of certificate searching is very small.

Thirdly, if a user leaves the P2P network temporarily for several days, its ID and certificate are reserved by its certificate manager. It’s still impossible for malicious user to fake false one during the term; the fake one cannot pass the authentication.

Fourthly, because of attribute of Chord-like network, authentication efforts are balanced well, every peer will take similar role, no centralized center or supper node to take specific task and it’s more robust and defends DOS attack better.

Finally, the authentication is based on ID (username) not on IP network level; it is convenient for user and is not easy for malicious user to fake by using the same IP address or faking the IP address. It’s an application layer authentication, not a transportation layer authentication. It makes it possible to evaluate a user for long term that uses a p2p network application.

V. SOLUTION PERFORMANCE

A. Certificate searching

As described before, in our solution, we store certificate in Chord structure. Every peer has a finger table to record routing information. For a $2^m$ Chord circle, the routing record has m routing records. When we are searching a user’s certificate, the distance to object peer will decrease at least a half when searching passes every routing peer. So the routing complexity is $O \log N$ if there are N peers in relevant network.

B. Structure maintain

When peers join or leave network, the routing information related with them will be incorrect. In such cases, we must make sure every peer has correct successor peer firstly and information including certificate must be stored in correct successor secondly. For example, when a new peer joins the network, we must finish the following step:

1) Initiate its predecessor and successor list and finger table
2) Update the information of related peers
3) Transfer the information which the new peer should store from other peers to the new peer.

When a new peer join network, there are often $O \log N$ peers which need to update. Before updating them, we have to searching them. So the operation complexity is $O \log^2 N$.

C. Storage cost

Every peer needs to store three tables: finger table, certificate table, IP record table. In a N peers’ network, the scale of finger table is around $O \log N$. Because in
Chord network, every peer stores information in same probability with distribution hash function, the storage for IP table, certificate table stay low level. Moreover, every peer needs to store certificate of other peers which have transacted with it. That cost some storage.

D. Security

Three trust levels were proposed to evaluate system authority in [6]:

- Level 1, the authority knows (or can easily compute) users’ secret keys and, therefore, can impersonate any user at any time without being detected.
- Level 2, the authority doesn’t know (or cannot easily compute) users’ secret keys. Nevertheless, the authority can still impersonate a user by generating false guarantees (e.g. false certificates).
- Level 3, the frauds of the authority are detectable. It cannot compute users’ secret keys, and if it creates another user’s keys, it is possible to prove that authority generated false guarantees.

According to it, our proposal is at 3 trust level, the highest level. The frauds of the authority are detectable. It cannot compute users’ secret keys, and if it creates another user’s keys, it is possible to prove that authority generated false guarantees.

The Table 2 is a comparing table between some authentication system and our solution. It’s apparent our solution has good properties. Full independence on TTP makes our proposal more robust and secure, meanwhile it make proposal more flexible and scalable. So it is feasible for large scale P2P application. Because our proposal is based PKI, it has some PKI-like attribute, for example Non-repudiation. Otherwise, our proposal has some own peculiarity. Except independence on TTP, it well balances the burden of certificate management. Users have more power in our solution than conventional PKI. Users issue/revoke its own certificate in certificate management, the certificate manager’s function is only to store and verify the certificate. Compared with other proposals mentioned previously, user certificate can be stored and verified for long term in our proposal. Even if the user logouts, the system still can prevent malicious peers from impersonating the user and even occupy the user’s ID. Moreover through IP limited mechanism, certificate quantity which users can apply for from an IP is restricted to a certain number. These mechanisms enhance network secure. Or malicious actions will damage network health largely. Our proposal alleviates these risks.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced and discussed related proposals in P2P identification and authentication, these proposal include partial distributed CA method, distributed CA method, Web of trust and ID-KPC. Based on these studies, we proposed a full self-organized public-key management scheme that does not rely on any TTP or fixed server, including in the initialization phase. In our approach, each user is its own authority and issues public-key certificates. Certificates are stored in the related certificate manager according to an appropriate algorithm. Key authentication is performed via getting certificate from certificate manager. Because of Chord like model, it has good load-balance. As discussed, It also has the highest trust level and can avoid sorts of malicious actions in our approach, including Sybil attack.

Our future work includes further exploration of mechanisms for improvement of the certificate models, security, efficiency and performance.

REFERENCES