A CA-based Scheme of User Authentication over Content-Centric Networking

Kai Lei*, Zhongjie Wang
The Shenzhen Key Lab for Cloud Computing Technology and Applications (SPCCTA),
Shenzhen Graduate School, Peking University, University Town, Shenzhen 518055, P. R. China
*leik@pkusz.edu.cn, zj.wang@pku.edu.cn

Abstract—**Content-Centric Networking (CCN) is a predominant substitute of the current TCP/IP networking and it is proposed to be the next generation Internet foundation. The evident characteristic of this network architecture is caching and indexing contents by the inner nodes – the routers, so as to reduce the redundant transmission and thereby shorten the distance between user and content. In this paper, we propose and implement a user authentication scheme over CCN. We adopt the trust model based on certificate authority (CA) to provide the service of binding certificate with user's identity, and help user determine the authenticity and reliability of the publisher of the network content. Also the specialized CA we designed for CCN takes advantage of the decentralization characteristic and cache mechanism of CCN to distribute the certificates and certificate revocation list (CRL) into the network, and it reduces the load of the CA central server when retrieving and verifying certificates. Besides, we propose a timeline-based method to segment the CRL with certificate issue date, thereby making the retrieval of CRL more effective.**

Keywords-Certificate Authority, user authentication, Content-Centric Networking

I. INTRODUCTION

The scale of data on the Internet is becoming increasingly large with the continuous growth of the Internet users and the rapid development of the mobile Internet in recent years. And the data transmission pattern is gradually shifting from one-to-one to one-to-many. A white paper published by Cisco VNI in 2011 showed that the global IP traffic is growing at a compound annual growth rate of 32 percent from 2010 to 2015, and will increase fourfold in 5 years. The broadcast contents (most are videos) will account for about 90 percent of global consumer traffic by 2015. However, the existing TCP/IP network has some inherent deficiencies in dealing with broadcast contents transmission. Although there're some remedial methods like CDN and P2P, all of them only can reduce the traffic out from the servers, but can't solve the issue of data redundant transmission in the network fundamentally.

In order to solve this problem, Van Jacobson et al. from PARC proposed the concept of Content-Centric Networking (CCN) [1]. In this new network environment, the user requires content by expressing a broadcast interest message containing the name of the content into the network, and the routers will forward this interest message to the nodes may have that content. The nearest node along the dissemination path will respond with the demanded content, and any other responds from other nodes will be discarded. A marked difference between CCN routers and traditional routers is the CCN router contains a Content Store (CS) which caches all the contents passing by. By doing this, users can fetch the cached contents from the routers instead of retrieving them from the content sources, so the distance between user and content is minimized.

According to the features of CCN, the sources of the contents become quite uncertain. Contents can be retrieved from either trusted sources or untrusted sources. Thus we need a mechanism to guarantee the content security. The security problems were taken into consideration at the early stage of CCN. Each content block has a digital signature generated by its publisher signing over the combination of the name and the content [2]. The signature bits and publisher's information is stored in the header of the content object. Users can get the publisher's public key or certificate from the network to verify the signature, but there's not yet an effective approach to verify the certificate itself.

In modern information security, public key infrastructure (PKI) is the most widely used certificate management architecture. Certificate Authority (CA), as the most important part of PKI, is responsible for the generation, management, storage, distribution, verification and revocation of certificates. Besides the CA model, there're some other models to authenticate user identity, like Web of Trust [3] or SPKI/SDSI [4, 5].

The CA model has the highest security level among them. It has been extensively used for authentication in the areas of E-commerce, business emails, enterprise internal networks, etc. Comparatively speaking, due to lack of a unique certificate management authority, the Web of Trust model allows misjudgments on user authentication and may rectify itself with time, so it has a relatively low security level and is not suitable for the business applications. In the CA model, the rights of trust are kept by the authority, so the administrator can manage all the certificates universally. In the Web of Trust model, users have the rights to determine whether to trust a counterpart or not. The CA model is considered to be centralized while the Web of Trust model is considered to be quasi-decentralized. However, there're still some problems exist in the CA model. For example, users need to trust untrusted CAs in server-defined certificate chain, which implies accepting transitive trust.

In this paper, we will introduce our solution of user authentication over CCN. We have adopted the CA model in PKI, and leveraged the decentralized characteristics of CCN to optimize the service in certificate storage, usage and revocation. Our user authentication scheme has single level CA running in standalone mode, so we can focus on the primary functions of CA over CCN. To test our design and implementation in practical environment, we have implemented a simple network file sharing system in CCN environment.

In Section 2 of this paper, we shall briefly introduce some background knowledge about related data structures in CCN and the signing and issuing process of certificates of CA. Then we shall elaborate the design of certificate authority over CCN in Section 3. The implementation of our system and a further discussion of applying the service in practical circumstances will be brought up in Section 4. Finally we shall make a conclusion in Section 5.

II. BACKGROUND

A. Data Format and Signature in CCN

In CCN, there're two types of messages in the network: the Interest message and the ContentObject message. When a user wants to require his desired content, he first needs to broadcast an Interest message into the network. The Interest message mainly contains the name of the content, the publisher information, exclusion information, selector, source type, etc. Some of the key fields of Interest message are listed here:

- **Name**: A globally unique CCN name consists of several components, usually including a version number (timestamp). Segmentation numbers are also included for large contents. An example of CCN name is "/ccnx.org/publications/papers/ccn.pdf%FD%04%EB%15%C5%10%00".
- **PublisherPublicKeyDigest**: The digest of content publisher's public key.
- **AnswerOriginKind**: This field indicates whether the content should be newly generated or existing one.
- **InterestLifeTime**: The expiration time of the Interest message. The routers will automatically drop the message after this period.

Corresponding to the Interest message, the returned content is wrapped in the ContentObject message. The ContentObject message mainly contains four elements—Signature, Name, SignedInfo and Content. The digest algorithm and signature bits are in the Signature element. And the publisher's public key or certificate is in the SignedInfo element, otherwise there's a key locator indicating where to find the key or certificate. The Content element represents the encoded binary data of the content. Some of the key fields in Signature element are listed here:

- **DigestAlgorithm**: The digest algorithm used in the signing process.
- **Witness**: A field used to store extra information for the aggregated signing on multiple blocks.
- **SignatureBits**: The signature binary data, which is generated by signing over the conjunction of the Name, SignedInfo and Content elements.

Some of the key fields in SignedInfo element are:

- **PublisherPublicKeyDigest**: The digest of the publisher's public key, used to verify the content source.
- **Timestamp**: The timestamp when the content is published.
- **Type**: Type of the content, including DATA, ENCR, GONE, KEY/, LINK, NACK.
- **FreshnessSeconds**: Used to judge if the content is stale or not.
- **KeyLocator**: May be the actual key or certificate of the publisher, or a link to the key or certificate.

The content is signed when the first time it's being published into the network. Currently CCN supports two kinds of signing: individual block signing and aggregated signing [6].

For individual block signing, the publisher generates a standard digital signature using PKCS#1. A digest will be calculated on the concatenation of Name, SignedInfo, and Content portion of the encoded ContentObjects, and then the publisher signs the digest with his own private key.

For aggregated signing, CCN uses the Merkle hash trees. First it arranges the set of content blocks as the leaf nodes in a n-ary tree, and calculate the digest of each leaf node as it does for individual block signing. Then it calculates the digest of parent nodes by concatenating the digests of all its child nodes and compute with the digest algorithm again. This process is iterated up the tree, and finally we can get the digest of the root node. The publisher signs this digest with his own private key and put the result into the signature bits field. The path information of Merkle hash tree is stored in the Witness field.

B. Certificate Authority

In PKI architectures, the main functionality of a CA is binding the entities in physical world with digital certificates in the network. We adopt the ITU-T X.509 v3 certificate standard which is most widely used. The subject and issuer of a certificate are identified by distinguished names (DN). A DN is unique within the range of CA, but the same DN may refer to different entities in different CAs. So it's necessary to specify both the DN and CA to uniquely identify an entity. Because DN is only used as a unique identifier, it can be named by CA with its own naming convention. In our system, we use user's identifier (UID) as the user's DN.

Usually the registration process of a certificate needs a cautious approval before a detailed investigation on applicant's information. The policies in different CAs are separately made by their administrators, so there isn't a unified standard or specification. Therefore we would not discuss on the detail about the certificate registration procedure. Seeing that our

Identify applicable sponsor/s here. *(sponsors)*

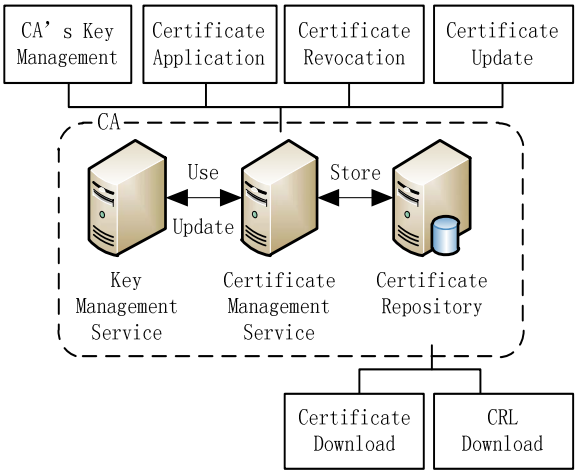

Figure 1. Certificate Authority in CCN environment

service is applied in a networking file sharing system, we choose a simple certificate registration policy, which allows the users to create their own certificates freely and the CA has the right to revoke the certificates when necessary.

In addition, we have a user information service in our system which records users' identifiers and the detail information binding to the UIDs. The information is provided by user during the registration procedure, and needs to be verified with some interactive process, e.g. email or SMS.

III. DESIGN AND IMPLEMENTATION

The certificate authority we designed mainly provides the functionalities of certificate registration, signing and issuing, storage, revocation, validation and secret key management. It consists of three main parts as showed in Fig. 1: (1) Certificate Management Service (CMS), (2) Certificate Repository (CR), and (3) Key Management Service (KMS).

The CMS has some human-computer interactive interfaces to the users and administrators. The CR is used to store all the digital certificates and certificate revocation lists (CRLs) signed and issued by CA. For the sake of protecting the private key of CA, we put it on a separate server in the intranet and only can be accessed by the Certificate Management Service. The KMS doesn't provide any service other than using or maintaining the CA's private key.

A. Certificate Registration

When a new user has entered the system, he needs to apply for a user certificate binding to his identifier by sending a registration request to the CMS. The user should always provide his UID and personal information to the server and the server will examine the information with the policies predefined by the administrator. If the server accepts the information, it will send a verification code to the user's private address, e.g., email address or cell phone. The user needs to send back the code to prove the authenticity of his identity. The detailed procedure is described as follows:

(1) The user applying for a certificate first sends an Interest message to the CMS.

(2) When the server receives the request, it returns an Interest message to acquire the user's information and public key.

(3) The user generates the public key pair locally, and then sends its public key along with his personal information wrapped in an encrypted ContentObject message to the server. The encryption is 128 bits AES, and the secret key is randomly generated by the user and encrypted with CA's public key.

(4) When the server receives the ContentObject message, it uses its private key to get the AES key and then the user's public key and personal information. If the server approves the user's request, it will send the verification code to the user by email. Afterward the server will send an Interest message to the user to inquire the verification code.

(5) The user encrypts the verification code with AES key and sends it back to the server.

(6) The server checks the correctness of the verification code returned by the user. If it's correct, the server will sign and issue a certificate for the user; otherwise it will return an error message.

The user's public key pair is generated on user's local machine instead of on the server so as to protect the key from man-in-the-middle attacks. All communications between user and server are encrypted with AES, so the information can't be sniffed by a third party. The user who initiates the session needs to generate the AES key and create the secure channel. An important premise of this method is that the user needs to hold the correct public key of CA locally. The public key can be fetched along with the client side software, and the user should check it while installation.

Because CCN doesn't support sending Interest messages with application specified parameters, the server needs to send back an Interest message to the user, and then the user can reply with his personal information in a ContentObject message. In CCN, the Interest messages are broadcast into the network, so it's necessary to set the PublisherPublicKeyDigest field to specify a certain destination for the message. After sending an Interest message, the user may register a filter locally to handle the returned ContentObject message from the network.

B. Certificate Generation and Signing

When the CMS has validated a user's identity, it generates a certificate for that user and signs the certificate by sending it to the KMS. For the signing of certificates comes down to the usage of CA's private key, it's strongly suggested to be performed on a device or server which is highly secure and only takes charge of the usage and maintenance of CA's private key.

First, the CMS fills some necessary fields when generating a new certificate, including serial number, validity, issuer, subject, etc. The serial number is a positive integer which is

unique in the range of CA. We choose to let the serial number be incremental starting from 1. Also the validity must be specified to indicate when the certificate will be expired and needs to be reissued. Usually the identity of the certificate owner needs to be rechecked when reissuing the certificate. In our system, there isn't a complex censorship during certificate registration, so the validation period can be relatively longer. We use the CA's name as the issuer's distinguished name and UID as the subject's distinguished name.

After filling the prerequisite fields, the server encodes the certificate into binary bytes with ANS.1 DER, and then computes the hash digest with the algorithm specified in the Signature Algorithm field. To sign the certificate, the server sends the hash digest to the KMS, and the latter will sign the digest with CA's private key. Finally the result will be filled into the Signature field. All the certificates issued by CA are stored on both server side and user side for other to download.

C. Certificate Storage

Like other content objects, the CCN certificates can be cached in the network. After certificates being published into the network, anyone can obtain them from the CA or some others who have downloaded them. This is not similar to the traditional CA, under which the certificates could only be acquired from the servers. This decentralized way of storing certificates has mitigated the server load, and the certificates still can be achieved even the servers are unavailable.

All of the certificates issued by CA are stored in the repository we called CR. The name of a CCN certificates is hierarchical, which includes the DN (i.e. UID) of the certificate owner and the version number. We put the certificates under the namespace of the current application or under a specified global namespace. For example, ccnx/apps/ccn-maze/certs/<UID>/<Version>, or /ccnx/CA/<region>/certs/<UID>/<Version>.

D. Certificate Revocation

Certificate revocation means during the validation period, the administrators manually invalidate the certificate when the owner of the certificate no longer has the right to use it. The certificate revocation is implemented through certificate revocation list (CRL). For a revoked certificate, its serial number, time and reason of revocation are all detailed in the CRL. The CRL is signed with CA's private key so its data integrity is assured. Because the size of the CRL increases over time, so the CRL is usually split into delta CRLs. In our system, we segment the entire CRL by the issuing date of certificate. Thus the users can check the revocation status of a certificate by downloading a small piece of CRL chunk file. Each CRL chunk file contains the revocation records of certificates issued in a certain period, e.g. a day or a month. Users can easily retrieve the CRL chunk files by the issuing dates of certificates. For example, the name of the CRL chunk file containing certificates issued on Sep 30, 2011 is /ccnx/apps/ccn-maze/CRL/20110930/<Version>.

In the same way, the CRL chunk files could be aggregated to extend its range to a month or a year. The CRL chunk files are stored in the CR together with the certificates. According to the feature of CCN, the CRL can be fetched from the CA or from other users who have downloaded them.

E. Certificate Verification

Before being used, certificates need to be verified for its validity. The aspects of verification include: (1) Authenticity, which is to check if the certificate is issued by an existing valid CA; (2) Identity, which is to check if the publisher of the content is the same as the owner of the certificate; (3) Validity, which is to check whether the certificate is expired or not and if it is revoked, i.e., specified in the CRL.

We assume that the users have preserved CA's certificate locally, so the public key in the CA's certificate could be used to verify the integrity of other users' certificates. Also, we compare the PublisherPublicKeyDigest in ContentObject and the hash digest computed by the public key in the user's certificate to examine the identity. Some additional information about the certificate owner is available from the User Server of the system. A combination of the validation period in the certificate and the CRL is used to check the validity of the certificate.

A user may request the CRL file through the network after knowing the issuing date of the certificate. The CRL file may not be the latest one, because it could be gotten from other users. CCN has defined the FreshnessSeconds in the ContentObject to indicate if the content is stale or not, and this value for CRL file should be rather short due to timeliness. Another way to ensure we get the latest CRL file is sending the Interest message directly to the CA. The CA will respond with a LINK type ContentObject pointing to another CCN name which contains a version number, and then the user will send another Interest message to get that CRL file. For example, the user sends an initial request with the CCN name /ccnx/apps/ccn-maze/CRL/20110930/latest, and the server will return a link pointing to /ccnx/apps/ccn-maze/CRL/20110930/<Version>. The user then could get the CRL file with the latter CCN name.

All of the CRL files downloaded are stored in user's local repository, so it's possible to be reused and shared with other users.

F. CA's Key Management and Update

The security of CA's private key is the foundation of the CA technology. Once the CA's private key is compromised, the attacker can pretend to be any other legal user, even the CA itself. Even worse he can replace the existing CA's certificate with his own certificate, and it will be a disaster for the entire CA. So the CA's private key must be well protected. Usually it's stored in a specialized device which is physically isolated. To realize this, we put the CA's private key on a specialized sever in the intranet which only provides the functionalities of key creation, update and usage. In our design, the certificate issuing process doesn't need a physical operation by any administrators, so all services can be accessed through the network. For some other application, the human interaction can be involved.

There exist two situations for CA's private key update: key leakage and key expiration. In the first situation, the administrators need to regenerate the CA's public key pair and certificate at once, and sign all the previous issued contents (certificates and CRLs) again with the new private key. To update a self-signed certificate (CA's certificate), three kinds of certificates should be published: the OldWithNew certificate, the NewWithOld certificate, and the NewWithNew certificate [7]. The OldWithNew certificate contains the old public key signed with the new private key and is used to authenticate the old certificate with the new one. The NewWithOld certificate contains the new public key signed with the old private key and is used to authenticate the new certificate with the old one. The NewWithNew certificate is the new self-signed certificate used to replace the old one and it contains the new public key signed with the new private key. The CA will broadcast an Interest message to notify all the users for its certificate update. For those who are not online, they will receive a certificate update message next time when they are online.

In the other situation, the CA updates its key due to expiration. The contents previously issued by CA don't need to be signed again, and only the CA's key and certification need to be regenerated. The contents signed by the old key are still valid until they are expired.

G. Implementation

At the current stage, we have implemented the three basic parts of CA service over CCN and a client-side library based on the Java API provided by the CCNx project. All of our source code is written in Java. Our test environment includes 1 certificate server running the CMS and the CR, 1 key server running the KMS, and 10 clients. The OS of servers and clients is Ubuntu 10.04 and ccnx 0.4.2 is installed on all the machines. Except the KMS, each machine is running a ccnd process and a ccn_repo process.

IV. DISCUSSION

A. Importance of CA

As we have described before, user authentication is an uncharted territory to be explored in CCN. Any application relies on secure communication or data integrity examination needs to be built on an infrastructure which can provide user authentication. Usually, it's thought common users have no discrimination against the legitimacy of contents in the network. A user decides whether to trust a content object only by the identity of the human related to that object. In the traditional Internet, we care about where we get the content, e.g. whether they're from a credible domain secured with SSL. In CCN, the provenance of the content is what we don't care about, but we do care about the publisher of the content. An ideal solution is to have a global unique authority which specially provides the user authentication service. It records all users' information in detail which is used to build the trust profiles, and it also acts as a guarantor for user's behavior. Binding the certificates with identities makes it possible to trace to the person in the physical world by a digital certificate. So people have to take the responsibility for their actions on the Internet.

B. CA vs. Web of Trust vs. SDSI

Just like the TCP/IP architecture, the trust mechanism hasn't been built into the infrastructure of the CCN architecture. But obviously it's especially important for CCN. Currently there isn't a trust mechanism like the PKI/CA mechanism can be widely used in the CCN network. The signature mechanism in CCN determines that many trust models, including CA, Web of Trust and SDSI, can be used to manage the trust relationships. Each of them has its advantages and disadvantages. However from the aspects of authority and credibility, CA is regarded as the most applicable one.

Though the CA model is usually considered a centralized one, which seems contrary to the concept of decentralization in CCN, we still can partition the governance. We let the local CA manage a small scope of users, which is similar to the multi-level CA in PKI. In this model, the scopes of CAs are mapped to the namespaces in CCN. Another problem worrying people is the transitive trust in the CA model. We define that the trust is not biased against the subjects. CA as a public observer, it judges the possibility a user may do harm to the others. In order to prevent the CA being compromised, more than one CA are used to administer a region. So the transitive trust can be acceptable in such case.

The Web of Trust model has eliminated the transitive trust, but when building the trust web, the users choose the introducers mostly by their subjective judgments, as well as set the trust level for introducers. This may cause inaccurate judgments on credibility of introducers, and will further make the whole trust web more untrustworthy. But the advantage of Web of Trust is the high flexibility, which makes it can incorporate with the CA as fully-trusted introducers to enhance its reliability.

As a distributed trust model, the SDSI model allows groups of users autonomously manage their trust and therefore discards the centralized management. It's relatively easy to be implemented in the CCN network. The advantage of SDSI is simplicity and flexibility, and it also has some global "distinguished root" principles quite similar to the root CA in PKI. But due to the lack of a consolidated authentication standard, the users from different local name spaces have limited knowledge to establish links to each other. In this case global organizations are still needed as introducers.

To sum up, the CA model can provide a more precise and reliable user authentication service in a security-demanding situation and it's more convenient for global certificate management. By leading the CA model into CCN, we can build a trust profile for each user and make it public. It will result in a fair and open community to restrict the users to doing harms.

C. Trust Evaluation

With a small scale system test in the practical environment, we found there're still some problems existing in our system. For example, after introducing user certificate authority, we can easily tell the identity of content publisher. But to a newly registered user, we know nothing about its reliability. In the physical world, different user has different credibility, and the credibility may vary with time. From this perspective, we need

to build a whole trust evaluation mechanism. Abdul-Rahman [8] and Ziegler [9] has proposed some very valuable methods on quantifying the trust values in the Web of Trust model. We may first create the initial trust for a user after verifying his legality by the CA, and then further classify him by trustworthiness into a certain class. For example, we can divide the users into four trust classes—unknown, untrustworthy, marginal and fully-trusted. This is to help the users better recognize the credibility of a certain user. Because in our system there are explicit relations of friendship among users, just like the links in social network, it's possible to introduce the mechanism of recommendation. We may leverage the user friendship to recommend new reliable users for current users to build new trust relationships. Similar to the voting mechanism in the Web of Trust, the users in our system can rate other users. For a user who has been rated for many times and has a relative high grade is considered to be more trustworthy. The grade determines which trust class the user belongs to. This method has the users endorsed by the CA and also supervised by the other users, so it makes the trust more comprehensive.

V. CONCLUSION

In recent years, the content security problems have been emerging with the increasingly hot topic of information-centric networking which is attracting more and more discussions. Designing a security paradigm for this new kind of networking to ensure the privacy, integrity and security of network contents is always an issue concerned by people and needs to be solved. In this paper, we proposed and implemented a user authentication scheme over CCN which has a good actual result in particular CCN application. The certificate management mechanism in it leverages the decentralization characteristics of CCN, so it has some advantages over traditional CA when stores and distributes certificates. The scheme embodies the thought of applying the CA trust model into the CCN network, and it can be further applied to the entire CCN network as a common model for user authentication. We also made a discussion on the idea of incorporating CA, Web of Trust and SDSI to make the user authentication more decentralized in CCN and evaluate the trust of users with a quantitative method.

ACKNOWLEDGMENT

This work was supported by NSFC (No:61103027), GuangDong Gov Project (2011A090200063) and Shenzhen Gov Project (JC201104210107A and JC2011042 10117A) and PKUSZ Dean's Student Research Project (No.2010004).

We would like to express our gratitude to Prof. Xiaoming Li and Dr. Beichuan Zhang, who had provided us so much help and guidance on CCN. We also would like to thank Prof. Lixia Zhang and Van Jacobson who take charge of the CCN project and held the community conference. We had a deeper insight into the CCN principles through the communications.

REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," ACM CoNEXT'09, Rome, Dec. 2009.

[2] D. Smetters and V. Jacobson, "Securing Network Content," PARC Tech Report, October 2009.

[3] A. Abdul-Rahman. "The PGP Trust Model," EDI-Forum: the Journal of Electronic Commerce, Apr. 1997.

[4] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI certificate theory," IETF Network Working Group RFC 1693, September 1999.

[5] R. L. Rivest and B. Lampson. "SDSI - A Simple Distributed Security Infrastructure," Technical report, MIT, 1996.

[6] D. Smetters, P. Golle, and J. Thornton, "CCNx Access Control Specifications," PARC, 2010.

[7] I. Jeun, J. Park, T. Choi, S. Park, B. Park, B. Lee, and Y. Shin, "A Best Practice for Root CA Key Update in PKI," In ACNS, pages 278–291, 2004.

[8] A. Abdul-Rahman and S. Hailes, "A Distributed Trust Model," In ACM New Security Paradigms Workshop, 1997.

[9] C. Ziegler and G. Lausen, "Propagation models for trust and distrust in social network," Information Systems Frontiers, Vol. 7, No. 4-5, Dec. 2005, pp. 337–358.