Hash-based Distributed Public Key Infrastructure for Ubiquitous Environments

Atushi TAKEDA
Department of Intelligent Information System
Tohoku Bunka Gakuen University
Miyagi, JAPAN
Email: atushi@shiratori.riec.tohoku.ac.jp

Seiya NAKAYAMA
Graduate School of Computer Science
Tohoku University
Miyagi, JAPAN

Gen KITAGATA
Research Institute of Electrical Communication
Tohoku University
Miyagi, JAPAN

Debasish CHAKRABORTY
Research Institute of Electrical Communication
Tohoku University
Miyagi, JAPAN

Kazuo HASHIMOTO
Graduate School of Computer Science
Tohoku University
Miyagi, JAPAN

Norio SHIRATORI
Research Institute of Electrical Communication
Tohoku University
Miyagi, JAPAN

***Abstract*—In ubiquitous environments, there are a huge number of computer nodes which provide a lot of information services via the Internet. Therefore, a secure communication system is required for ubiquitous environments. In this paper, we propose a new architecture of distributed public key infrastructure, named as Hash-based Distributed Public Key Infrastructure (HDPKI). A HDPKI system achieves a scalable management of public key certificates by using Distributed Hash Table. Additionally, it is easy to use a HDPKI system, because we can register our public key certificates automatically. In this paper, we show scalability of HDPKI through computer simulation. According to these results, the total amount of required messages for certificate management in a HDPKI system is $O(\log n)$ where n is the number of nodes.**

***Keywords*-overlay network; peer-to-peer network; public key infrastructure; distributed hash table;**

I. Introduction

In ubiquitous environments, there are a huge number of computer nodes such as servers, routers, PDAs or sensors. These nodes provide a lot of information services via the Internet, and these services consists of a lot of private information such as personal schedule or health related matters[1]. Therefore, a secure communication system is required for ubiquitous environments. The most famous existing system, the Public Key Infrastructure (PKI), support secure communications in the Internet[2]. However, PKI can not work efficiently in ubiquitous environments. Because PKI does not assume an environment with a huge number of nodes. Additionally, it is not easy for almost all the users to use a PKI system, because public key certificates can not be registered on PKI systems automatically. Therefore, we require a new scalable public key infrastructure.

In this paper, we propose a new architecture of distributed public key infrastructure, named as Hash-based Distributed Public Key Infrastructure (HDPKI). A HDPKI system assumes a huge number of nodes, and it is easy to use a HDPKI system. A HDPKI system manages public key certificates under a simple rule. A HDPKI system manages only relationships between a node identification and its public key certificate, and A HDPKI system do not allow to overlap the certificates. Therefore, we can register our certificate on a HDPKI system if our node identification is not registered, and the HDPKI system ensures that we get a certificate of a specified node indentification because the certificate is related to the node identification in the HDPKI system. A HDPKI system do not have a cache mechanism in order to disallow overlapping of identification-certificate relationships. Therefore, HDPKI requires a distributed management mechanism. We solved this problem by using Distributed Hash Table. In this paper, we confirm scalability of HDPKI through the results of computer simulations. According to the simulation results, the total amount of required messages for certificate management in a HDPKI system is $O(\log n)$ where n is the number of nodes.

The remainder of the paper is organized as follows. We explain about the related works of our research in section II. We proposed a new scalable public key infrastructure, Hash-based Distributed PKI (HDPKI), is discussed in section III. Performance evaluation of HDPKI is given in section IV. We discuss about security of HDPKI in section V. Finally, we conclude our research and talk about our future works in section VI.

II. Related Work

A. Public Key Management

Nowadays, almost all secure communications in the Internet use public key cryptography such as RSA[3]. Security of public key cryptography is based on validation of public keys. Therefore, a management mechanism of public key certificates is required for getting a valid public key when public key cryptography is used.

The most famous management mechanism of public key certificates is Public Key Infrastructure (PKI)[2]. PKI manages public key certificates at servers, called Certificate Authority (CA). The security of PKI is based on social relationships between users of nodes and managers of CAs. Therefore, when users register their public key certificates on a PKI system, a manager of CA must check the users manually. This process needs a lot of time and a lot of money. Therefore, it is not easy for almost all the users to use the PKI system. Additionally, PKI has scalability problem, because a PKI system requires to manage a huge number of public keys at few servers.

PGP enables distributed management of public key certificates[4] In a PGP system, all the nodes manage public key certificates of each others, and nodes get a new public key certificate from trusted nodes. The security of PGP is based on the concept, called Web of Trust. This concept is useful for distributed management of public key certificates. However, PGP is not secure enough, because a PGP system allows overlapping of certificates. Additionally, PGP could not solve the scalable problem, because PGP does not have a efficient mechanism for searching a new public key certificate.

DNSSEC also enables distributed management of public key certificates[5]. In a DNSSEC system, name servers manage public key certificates, and nodes can get a public key certificate from name servers in the same way as Domain Name System. However, DNSSEC has a problem when many search messages flock to a root name server because DNSSEC uses tree topology networks. Each node in a DNSSEC system has caches of public keys in order to solve this problem, so it is difficult to manage uniqueness of each public key in a DNSSEC system.

There are some approaches for efficient distributed management of public key certificates[6], [7], [8]. These method search public keys efficiently by using routing information of computer networks. These method enables not only scalable distributed management of public key certificates but also efficient search of public key certificates. However, these method require specific network environment. Therefore, it is not easy to use them, because we must prepare specific network environments when we want to use them.

Secure communications in ubiquitous environments require an efficient distributed management mechanism of public key certificates which we can use in any computer networks. Our proposed HDPKI enables efficient distributed management of public key certificates by using Distributed Hash Table. And, we can use a HDPKI system in any computer networks because it does not require any specific network information.

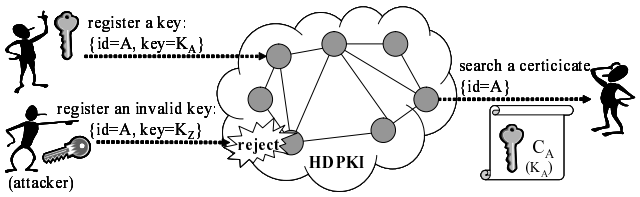

Figure 1. Requirements of HDPKI

B. Distributed Hash Table

There are many studies about Distributed Hash Table (DHT)[9], [10], [11]. DHT achieves distributed content management and scalable search. HDT systems are used for managing contents such as images, sounds or videos. Therefore, studies of HDT are mainly focused on efficiency of content management.

DHPKI aims to manage public keys for secure communications. And, in HDPKI systems, security is more important than efficiency. Therefore, we need a new secure algorithm and a new secure protocol for HDPKI systems.

III. Proposal: HDPKI

A. Basic Concept of HDPKI

In order to provide secure communications in ubiquitous environments, we require a new scalable public key infrastructure which can be used easily. In this paper, we propose a new scalable distributed public key infrastructure which is named Hash-based Distributed Public Key Infrastructure (HDPKI). HDPKI is not only scalable but also easy to use.

Fig.1 shows requirements of HDPKI systems. A HDPKI system looks like a data base of public key certificates which is constructed by a large number of nodes. Requirements of a HDPKI system are as follows.

- A node can register its node identification and its public key if a same identification has not been registered.
- Any nodes can not register its node identification if a same identification has been already registered.
- A node can update its public key. But, the other nodes can not update it.
- A node can search a public key certificate by using a node identification.

A HDPKI system manages relationships between node identifications and public keys. If a wrong node such as an attacker node try to update another node's public key, a HDPKI system rejects it. A HDPKI system enables a node to search a public key certificate, which contains a node identification and a public key. These mechanism

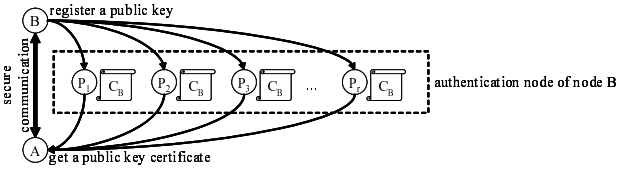

Figure 2. Public key management in HDPKI

of HDPKI guarantees that we can get a valid public key from a HDPKI system if the public key has been already registered. This mechanism of HDPKI enables an automatic registration of public keys, because HDPKI does not need to confirm validation of the node and the public key manually. Therefore, it is easy for all the nodes to use a HDPKI system.

Fig.2 shows a distributed management mechanism of public keys in a HDPKI system. In a situation shown in Fig.2, there are $r+2$ nodes (A, B, P_1, P_2, P_3, $\cdots$, P_r), and node A tries to get a public key of node B. Here, r is a HDPKI's parameter which means redundancy. In a HDPKI system, a public key is managed by r nodes which are called "authentication node". In a case shown in Fig.2, a public key of node B is managed by node P_1, node P_2, node P_3, $\cdots$, node P_r, which are authentication nodes of node B. Each node has r authentication nodes. A HDPKI system decides authority nodes under a single rule. So, we can search authentication nodes under the single rule, and we can get a public key certificate from the authentication nodes. In a case of Fig.2, node A gets a public key certificate of node B from authentication nodes of node B.

Technical issues of HDPKI are as follows.

- How to decide authentication nodes.
- How to search authentication nodes.
- How to add a node to a HDPKI network, and how to remove it from the network.
- How to protect a HDPKI system from the attackers.

In this section, we describe how to solve the technical issues of HDPKI. And, we explain that HDPKI is scalable.

B. Public Key Management and Searching Procedure

In a HDPKI system, each node manages public keys of each other. Public keys which a node manages are decided under a single rule which is similar to Chord, which is an existing Distributed Hash Table technology. Chord focuses on managing content data such as images, sounds or videos. However, HDPKI intends to manage public keys. Therefore, HDPKI requires a new algorithm and a new protocol which are more secure than Chord's ones.

Fig.3 shows a procedure of public key management. In a situation shown in Fig.3, there are node A, node B, node P_i, node S_i and node M, and C_k indicates a public key certificate of node k. In a HDPKI system, nodes are virtually located on a Hash-Ring. Hash-Ring is a circular ring indexed

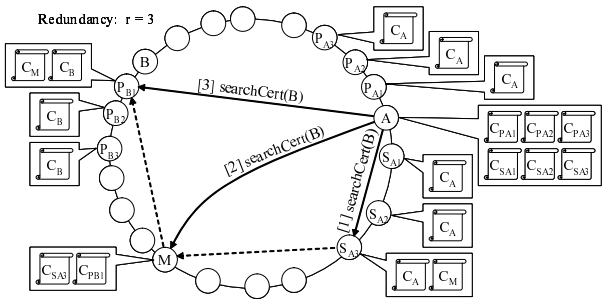

Figure 3. Procedure of searching a public key certificate

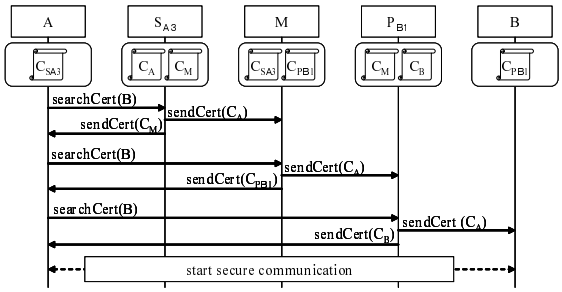

Figure 4. Protocol of searching a public key certificate

from 1 to N, and N is usually a huge number such as 2^{32}. A position of each node is derived from its hash value, which is calculated from the node identification by a one-way hash function. The position decides which public key certificate a node manages. A public key certificate of a node is managed by r predecessors of the node, called "authentication node". Here, r is a parameter of HDPKI, and it means redundancy of this system. In case of Fig.3, node P_{A1}, node P_{A2} and node P_{A3}, three predecessors of node A, manage a public key certificate of node A, and these nodes are authentication nodes of node A. Additionally, a public key of a node is sent to r successors of the node. In case of Fig.3, a public key of node A is sent to node S_{A1}, node S_{A2} and node S_{A3} which are the three successors of node A. So, a node manages $2r$ public keys of its neighbor nodes. In case of Fig.3, node A manages public keys of node P_{A1}, node P_{A2}, node P_{A3}, node S_{A1}, node S_{A2} and node S_{A3}. Furthermore, a node exchanges its public key with nodes which are placed in $2^m (m = 1, 2, \cdots)$ away from the node.

Fig.3 shows a procedure of searching a public key certificate, and Fig.4 shows a protocol for searching a public key certificate. In a HDPKI system, when a node wants to get a public key certificate of a target node, a node obtains new public key certificates from the trusted nodes with whom the node has exchanged public key. This process is as follows.

1) When a node N_s want to get a new public key certificate of a target node N_d, node N_s requests a new public key certificate of node N_d to the closest trusted

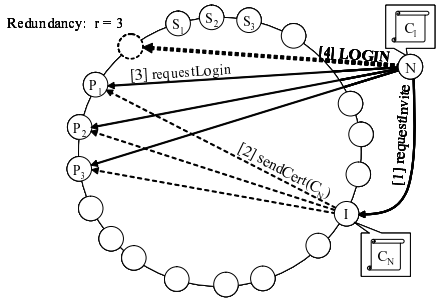

Figure 5. Procedure of joining to a network

node N_t to the target node N_d.

2) When node N_t has a public key certificate of node N_d, node N_t sends the public key certificate to node N_s, and node N_t sends a public key certificate of node N_s to node N_d. Then, node N_s and node N_d can do a secure communication by using public key encryption.
3) When node N_t does not have a public key certificate of node N_d, node N_t sends the public key certificate of node N_m to node N_s, and node N_t sends a public key certificate of node N_s to node N_m. Here, node N_m is the closest node to node N_d in trusted nodes of node N_t. After that, node N_s adds node N_m to trusted node list of node N_s, and searching process is repeated from process 1.

The node get a public key certificate of an authentication node of the target node through the above process, and the node get a target public key certificate from the authentication node. After that, the node can do a secure communication with the target node. All messages in this process have e-signature of public key encryption, and these messages could be encrypted if it is required. In this process, the number of required communication data is $O(\log n)$ where n is the number of nodes. This means that HDPKI is scalable.

C. Joining Procedure

A HDPKI system allows a node to join to its network. However, before the node joins the network, the node must exchange public keys with a node which has already joined the network. This is a requirements of joining to HDPKI networks. This exchange can be simply done by e-mail or face-to-face.

Fig.5 shows a procedure of joining to a HDPKI network, and Fig.6 shows a protocol for joining to a HDPKI network. Joining process is as follows.

1) A node N_n exchanges its public key with node N_i which has already joined to the HDPKI network. This exchange is realized by other systems. After that, node N_n requests node N_i to invite node N_n.
2) Node N_i exchanges public keys with nodes N_{p1}, N_{p2}, $\cdots$, N_{pr}, which are the authentication nodes of node N_n after node N_n joins to the network. After that, node N_i sends a public key certificate of node N_n to nodes N_{p1}, N_{p2}, $\cdots$, N_{pr}.
3) Node N_n sends a joining request to nodes N_{p1}, N_{p2}, $\cdots$, N_{pr}. If all of the authentication nodes N_{p1}, N_{p2}, $\cdots$, N_{pr} accept the joining request, node N_n can join to the HDPKI network. If some of the authentication nodes reject the joining request, node N_n can not join to the network.

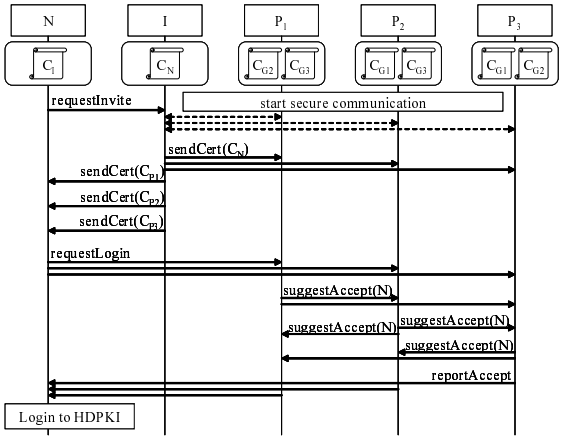

Figure 6. Protocol of joining to a network

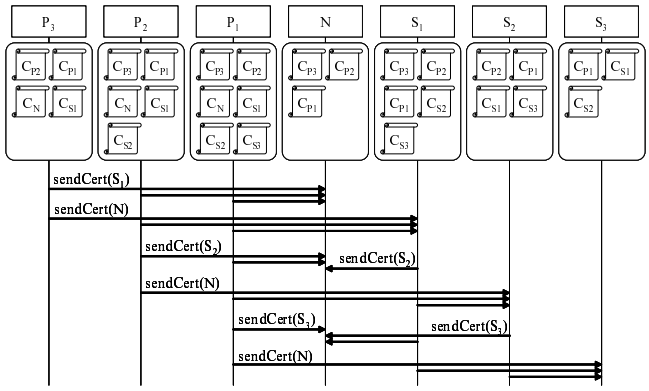

Figure 7. Maintenance protocol used when a node joins to a network

All messages in this process have e-signature of public key encryption, and these messages could be encrypted if it is required. In this process, the number of required communication data is $O(\log n)$ where n is the number of nodes. This implies that HDPKI is scalable.

Fig.6 shows a maintenance protocol which is performed after a node joins to a HDPKI network. All nodes in a HDPKI network communicate with neighboring nodes periodically in order to exchange their informations which contains neighbors list, trusted nodes list and managed public key certificates, etc. When a node joins to a HDPKI network, the node and its neighbors exchange their public

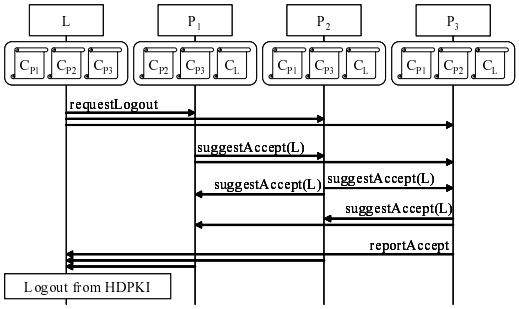

Figure 8. Protocol of leaving from a network

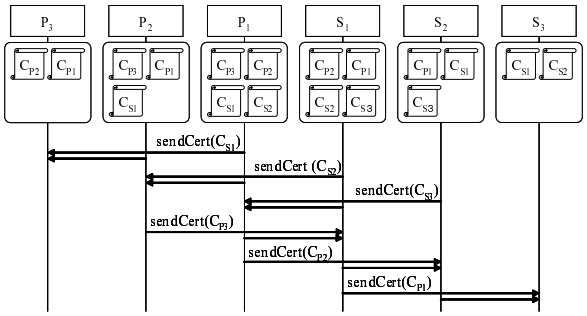

Figure 9. Maintenance protocol used when a node leaves from a network

keys with each r predecessors and each r successors. In this process, the node which joins to the network obtains public keys of its r successors, and its successors get a public key of the node. In this process, the number of required communication data is $O(r^2)$.

D. Leaving Procedure

A HDPKI system allows a node to leave from its network. Fig.8 shows a protocol for leaving from a network. When a node leaves from a HDPKI network, the node sends a leaving request to its authentication nodes. If all the authentication nodes accept the leaving request, the node is removed from the HDPKI network. If some of the authentication nodes reject the leaving request, the node is regarded as a active node until all authentication node recognize the leaving request. If a node leaves from a HDPKI network without this leaving procedure, instead of the leaving node, a neighbor node which has found the departing node sends a leaving request of the node. In this process, the number of required communication data is $O(r)$.

Fig.9 shows a maintainance protocol which is performed after a node joins to a HDPKI network. All the nodes in a HDPKI network communicate with neighbors periodically in order to exchange their informations. When a node leaves from a HDPKI network, its neighbors exchange their information, and they get the required public key certificates. In this process, the number of required communication data is $O(r^2)$.

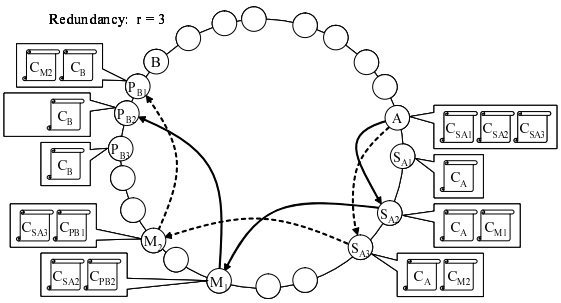

Figure 10. Multi-path for searching a public key certificate

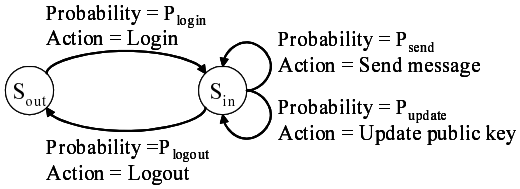

Figure 11. State transition diagram of node agents

E. Searching Procedure using Multi-Path

In a HDPKI system, a node obtain a new public key certificates through some relay nodes. If the relay nodes contain an attacker node, the node can not get a valid certificate. Therefore, HDPKI uses a multi-path mechanism in order to ensure the searching procedure.

Fig.10 shows an example of multi-path for searching a public key certificate. In this situation, even if node M_1 is the attacker, node A can get a valid public key certificate via node M_2. In this case, node A can be aware that an attacker node exists in its relay nodes. This multi-path searching mechanism makes HDPKI more secure.

IV. Performance Evaluation

A. Network Simulator

In order to evaluate performances of HDPKI, we developed a simulator which simulates operations of nodes in a network. This system is written in Java, and runs on Java Runtime Environment. In this simulator, all the operations of nodes are implemented in software agents called "node agent". The messages between the node agents simulates all the messages which are sent for participation, departure, updating public keys and sending messages.

Fig.11 is the state transition diagram of node agents. Node agents have two status. One is logout status (S_{out}) which means that the node is leaving the network. The other is login status (S_{in}) which means that the node is joining to the network. The probability of changing status S_{out} to status S_{in} is P_{login}, and the probability of changing status S_{in} to status S_{out} is P_{logout}. The probability of updating the public key of the node is P_{update}. And, the probability of sending

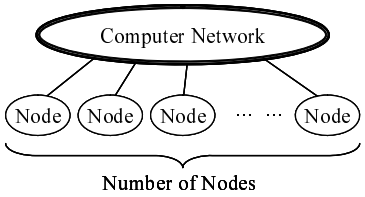

Figure 12. Network topology assumed in the simulation

a message to the randomly selected node is P_{send}. This message has an e-signature. When a node which receives this message, the node searches a public key certificate using a HDPKI system in order to validate the e-signature.

In the simulations, we set these parameters as follows: $P_{login} = 1.0$, $P_{logout} = 0.05$, $P_{update} = 0.05$, $P_{message} = 0.9$.

Fig.12 shows the network topology assumed in this simulator. In this simulator, all nodes are connected by some computer networks like the Internet, and can communicate with each others. Network failures such as packet loss are not assumed, and all communications are executed completely. In simulation results that follow, the number of nodes indicates the number of nodes participating in the computer network.

B. Performance Evaluation

In this section, we shows evaluations of HDPKI through simulation results. We simulated four types PKI system in this simulation, and its results indicate as follows.

PKI
: Performance of a Certificate Authority (CA) server in an conventional PKI system.

HDPKI-a
: Average performance of each node in a HDPKI system. A redundancy parameter r is 3. A node searches a public key certificate via a path.

HDPKI-b
: Average performance of each node in a HDPKI system. A redundancy parameter r is 3. A node searches a public key certificate via 3 paths.

HDPKI-c
: Average performance of each node in a HDPKI system. A redundancy parameter r is 5. A node searches a public key certificate via 5 paths.

We monitored amount of required messages for each system. Additionally, we monitored a relationship between security of a HDPKI system and its parameters.

1) Required Number of Messages for Searching a Certificate: Fig.13 shows amount of required messages for searching a public key certificate. A CA server of PKI requires no message to search public key certificates, because nodes of PKI uses a public key of CA in order to validate e-signatures. On the other hand, each node of HDPKI requires some messages to search public key certificates. However, fig.13 shows that the number of messages required for search in a HDPKI system is $O(\log n)$ where n is the number of nodes. This means that HDPKI is scalable.

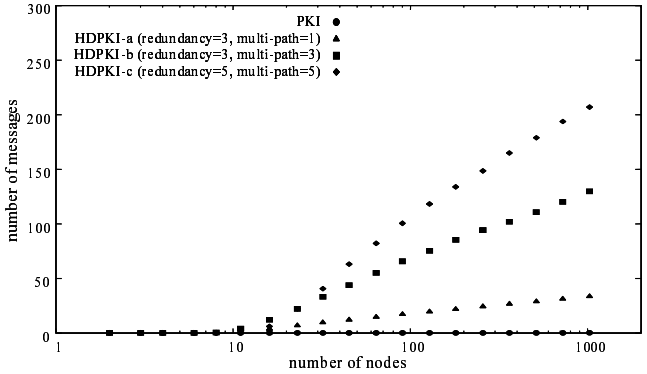

Figure 13. Number of messages for searching

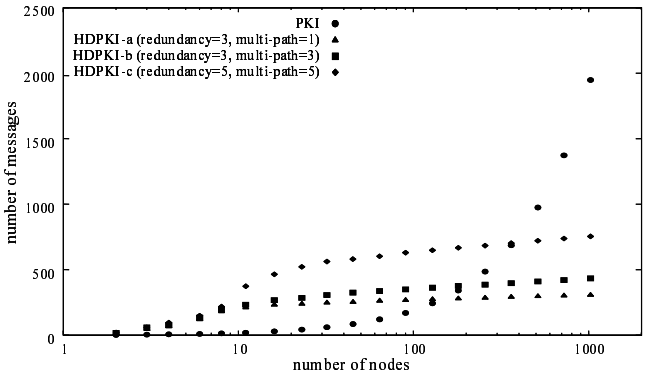

Figure 14. Number of messages for joining

2) Required Number of Messages for Joining to a HDPKI Network: Fig.14 shows amount of required messages for joining to a network. This process contains a registration procedure of a public key. In a PKI system, a CA server manages all public key certificates, so all registration message come to the CA server. On the other hand, a HDPKI system achives an efficient distributed management of public key certificates. Therefore, the required joining messages of HDPKI are less than required messages of PKI when the number of nodes is more than 500. Fig.14 shows that the number of messages required for joining to a HDPKI network is $O(\log n)$ where n is the number of nodes. This shows that HDPKI is scalable.

3) Required Number of Messages for Leaving from a HDPKI Network: Fig.15 shows amount of required messages for leaving from a network. This process contains a cancel procedure of a public key registration. In a PKI system, all the leaving nodes send messages to a CA server, because a CA server manages all the public key certificates.

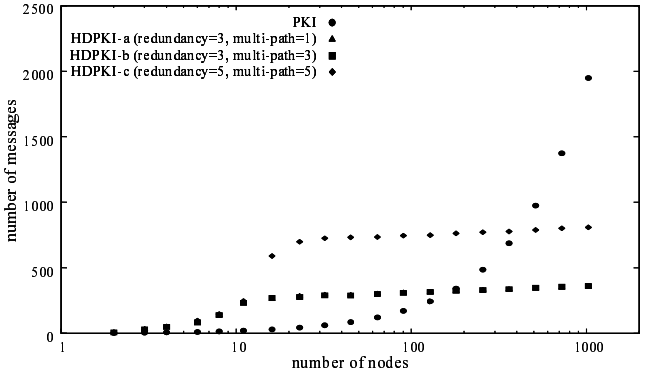

Figure 15. Number of messages for leaving

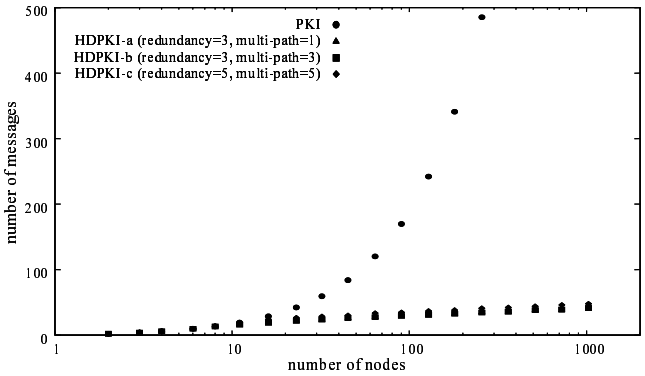

Figure 16. Number of messages for updating certificates

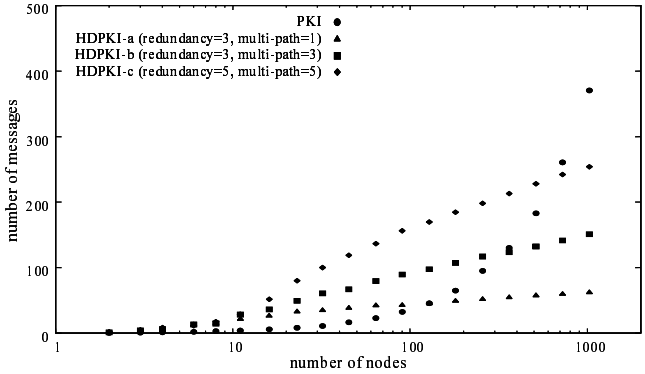

Figure 17. Number of total messages

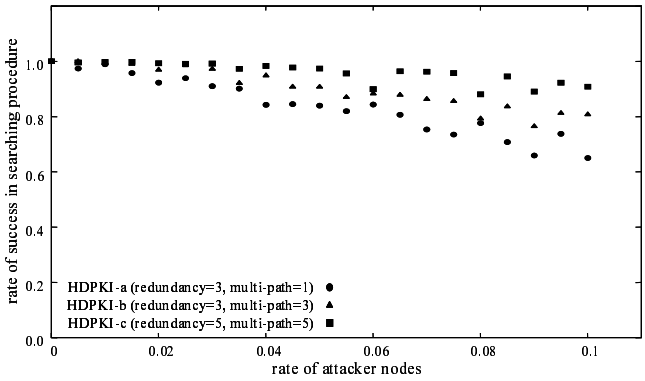

Figure 18. Rate of successful communication

On the other hand, leaving nodes of a HDPKI system send messages only to its neighbors. Therefore, the required number of leaving messages of HDPKI are less than the required number of messages of PKI when the number of nodes is more than 500. Fig.15 shows that the number of messages required for leaving from a HDPKI network is independent from the number of nodes. So, it also implies that HDPKI is scalable.

4) Required Number of Messages for Updating a Public Key: Fig.16 shows the amount of required number of messages for updating a public key. In a PKI system, a CA server manages the all public key certificates, so all updating message are sent to a CA server. On the other hand, in a HDPKI system, updating messages are sent only to neighboring nodes. Therefore, the required number of updating messages of HDPKI are much less than the required number of messages of PKI. Fig.16 shows that the number of messages required for updating in a HDPKI system is independent from the number of nodes. So, HDPKI can be said a scalable scheme.

5) Required Number of Total Messages: Fig.13 shows total amount of required messages. Node activities of this simulation is as described in IV-A. A PKI system requires no message for searching certificates, but a PKI system requires a lot of messages for joining, leaving and updating. The number of required messages in a PKI system is $O(n)$ when n is the number of nodes. On the other hand, a HDPKI system requires messages for searching certificates, but the number of messages required for joining, leaving and updating in a HDPKI system is less than the required messages in a PKI system. As a result, the number of required total messages in a HDPKI system is less than required messages in a PKI system, when the number of nodes is large enough. Fig.13 shows that the required number of total messages in a HDPKI system is $O(\log n)$ when n is the number of nodes.

6) Security of Searching Procedure: In a HDPKI system, nodes obtain public key certificates via some other nodes. Therefore, when there are some attackers in a HDPKI system, some nodes could get invalid public key certificates. In order to solve this problem, nodes in the HDPKI system get a public key certificate via multiple path. Fig.18 shows a success rate of searching a public key certificate under some multi-path parameters, which means a number of paths for searching a certificate. In this simulation, we assume stand-alone attacks, but we do not assume collusion attacks. If a node searches a public key certificate through a single-path,

the success rate of searching is low when some attackers exist in the system. On the other hand, if a node searches for a public key certificate via multiple paths, the success rate of this is more than the rate of single-path. This means that a multi-path mechanism of HDPKI makes this system more secure.

V. Discussion

A goal of HDPKI is to build a secure computer network for ubiquitous environments. In order to achieve this goal, a HDPKI system provides a secure management of public key certificates. The secure management in a HDPKI system means a secure management of relationships between node identifications and public key certifications. This rule is simpler than the conventional PKI systems. Therefore, HDPKI enables an automatic registration of public key certifications. On the other hand, it is difficult for a HDPKI system to protect the system from collusion attacks, such as Sybil Attak[12]. In order to prevent collusion attacks, HDPKI requires some administering rules. For example, a rule for node identifications makes the HDPKI system more secure. Because the rule can decide node locations in a Hash-Ring of HDPKI, and it makes collusion attacks more difficult. When we use a HDPKI system, we have to prepare administering rules. But, we do not indicate the rules, because the rules must be decided from the policies of an organization which uses a HDPKI system.

VI. Conclusion

In ubiquitous environments, there are a large number of computer nodes such as servers, routers, PDAs or sensors. These nodes provide a lot of information services via the Internet, and these services consists of a lot of private information such as personal schedule or health related matter. Therefore, new scalable public key infrastructure is required for ubiquitous environments in order to build secure communication networks. In this paper, we proposed a new architecture of distributed public key infrastructure: Hash-based Distributed Public Key Infrastructure (HDPKI). HDPKI manages relationships between node identifications and public key certificates, and this simple rule enables an automatic registration of public key certificates. Therefore, it is easy for all the users to use a HDPKI system. Additionally, it is scalable, because HDPKI manages public key certificates with a distributed hash table technology. In this paper, we showed performance evaluations of HDPKI systems through computer simulation. These simulation results indicates that a HDPKI system is more scalable than a conventional PKI system.

We are studying about a secure network which can be easily used. The secure network is required by ubiquitous environments, and we think that we can make it by using both HDPKI system and conventional PKI system. We will work to build a secure ubiquitous network architecture, and will study about its administering rules in the future.

Acknowledgment

This research was partly funded by National institute of Information and Communications Technology Japan, under the program of “Research and Development of Dynamic Network Technology” and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists, 20700069, 2008.

References

[1] H. Takahashi, S. Izumi, T. Suganuma, T. Kinoshita, and N. Shiratori, “An agent-based healthcare support system in ubiquitous computing environments,” *Lecture Notes In Computer Science*, vol. 5597, pp. 237–240, 2009.

[2] R. Housley, W. Polk, W. Ford, and D. Solo, “Rfc 3280: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile,” 2002.

[3] B. Kaliski, “Rfc 2313: Pkcs #1 : Rsa encryption version 1.5,” 1998.

[4] S. Garfinkel, *PGP : Pretty Good Privacy*. Oreilly and Associates Inc., 1994.

[5] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “Rfc 4033: Dns security introduction and requirements,” 2005.

[6] S. Capkun, L. Buttyan, and J.-P. Hubaux, “Self-organized public-key management for mobile ad hoc networks,” *IEEE Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52–64, 2003.

[7] Y. Kitada, A. Watanabe, I. Sasase, and K. Takemori, “On demand distributed public key management for wireless ad hoc networks,” *Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on*, pp. 454– 457, 2005.

[8] J. Goold and D. M. Clement, “Improving routing security using a decentralized public key distribution algorithm,” *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on*, 2007.

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[10] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, “Tapestry: A resilient global-scale overlay for service deployment,” *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, 2004.

[11] J. Aspnes and G. Shah, “Skip graphs,” *ACM Transactions on Algorithms*, vol. 3, no. 4, 2007.

[12] J. R. Douceur, “Secure group communications using key graphs,” *Proceedings of First International Workshop on Peer-to-peer Systems*, pp. 251–260, 2002.