Efficient Key Management Scheme for Hierarchical Access Control in Mobile Agents

Hsing-Bai Chen\textsuperscript{a}, Chun-Wei Liao\textsuperscript{b}, and Chang-Kuo Yeh\textsuperscript{c*}

\textsuperscript{a}Department of Information Engineering and Computer Science, Feng Chia University
No. 100 Wenhwa Rd., Seatwen, Taichung, Taiwan 407, R.O.C.
\textsuperscript{b}Department of Information Management, Hsiuping Institute of Technology
No.11, Gongye Rd., Dali City, Taichung County, Taiwan 41280, R.O.C.
\textsuperscript{c*}Department of Information Management, National Tai-Chung Institute of Technology
129 Sanmin Rd., Sec. 3, Taichung, Taiwan 404, R.O.C.
\textsuperscript{c*}E-mail: yehlin@ntit.edu.tw

Abstract: - Mobile agents have great potential for increasing the realized benefit for a variety of e-commerce applications. However, enabling the mobile agent to safely travel over the open and uncontrollable Internet is necessary to protect the contents of a mobile agent. Recently, many agent structures that manage the keys needed to provide the access control mechanism for mobile agents have been developed. Nevertheless, these structures require either large amount of mobile codes or heavy computation loads. In this paper, a lightweight key management method for hierarchical access control in mobile agent environments is proposed. This method not only provides access control for mobile agents but also reduces the agent size as well as cuts down the computations cost.

Key-Words: - Access control, Information security, Key management, Mobile agent

1 Introduction

The evolution of computer systems have gradually changed from centralized monolithic computing devices into client-server environments. Even though the large scale proliferation of networking bandwidth is presented, complex forms of distributed computing may cause network jams and limitations. Traffic jams on the Internet have lead to serious performance failures for applications that required tight interactions between web software systems around the world [10]. The evolutionary path of allowing complete mobility of cooperating applications among supporting hosts to form a large-scale and loosely-coupled distributed system is a promising solution for overcoming this problem. The catalysts for this evolutionary path are mobile agents.

A mobile agent is a software program, often exhibiting learning capabilities, sent by agent’s owner to visit a series of hosts or act on the behalf of other software programs. It acts on the owner’s authority to work on these hosts autonomously toward a goal, to meet and to interact with other agents over the Internet via some communication paths. After that, the mobile agent returns the results to the agent owner. The scholars [11,15] summarize the capabilities of mobile agents as follow: Mobile agents should (1) be able to perform one or more goals automatically; (2) be capable of cloning itself and propagate accordingly; (3) collaborate and communicate with other software agents adequately; (4) be robust and competent; and (5) have some evolution states to record the computation status. Based on these capabilities, many researchers newly focused on mobile agent technologies and used them for modifying human business activities. For instance, in the e-commerce, mobile agents can be used for increasing the realized benefit for both bidders and sellers participating in on-line auctions. Agents can be generated by bidders to find designated items, to set up auctions, and to cast appropriate bids [7,8].

Because the environment where mobile agents work is open, when a mobile agent is executed, it comes into contact with a variety of hosts; some of which may be trustworthy, but others may be potentially malicious. Furthermore, the openness of the Internet environment is also a major concern about security because the information carried by the mobile agent is likely to be exposed. Therefore, security mechanism that manages the keys, needed to provide access control to the content of a mobile agent, is necessary to be deployed [4,6,9].
2 Related Works

Mobile agents roam among visited hosts and interact with other agents over an insecure Internet. The content of mobile agents may be modified and exposed. From the security perspective, the integrity and the confidentiality of mobile agents must be provided. In this section, a brief overview of recent schemes proposed by Volker-Mehrdad [17], Lin et al. [12], and Chen et al. [2] and their drawbacks are illustrated, respectively.

2.1 Volker and Mehrdad’s scheme

In 1998, Volker and Mehrdad [17] presented a tree-based key management structure to secure mobile agent against unauthorized accesses. Nevertheless, this scheme requires a large agent size and a heavy computation load due to its repetitious storing of cryptographic keys and its numerous costly public-key computations used to encrypt these keys [2,12]. The result of this excessive mobile code size is that it squeezes the bandwidth out of network communications and prevents useful activity from taking place, not to mention the heavy computational burdens it has on the mobile hosts. In 2004, Lin et al. [12] proposed a hierarchical key management scheme to eliminate the need of repeatedly storing cryptographic keys on a mobile agent. The difficulty of factoring problem [14] is employed to protect the key against any unauthorized use. However, the complicated computation puts burdens on both the agent owner and the visited host [2]. Recently, Chen et al. [2] showed an efficient key management mechanism in a hierarchy that employs lightweight operations including hash function and exclusive-or instead of time-consuming computations. Nevertheless, agent size is not tiny when relationships in the hierarchy are complicated and needed to be publicized, which will be illustrated in Section 2. Accordingly, the performance and mobile code size of all Volker and Mehrdad’s [17], Lin et al.’s [12], and Chen et al.’s [2] schemes are still not considerably satisfactory.

Because security, storage and computational efficiency are all necessary precondition for developing a mobile agent, in this paper, a scheme is proposed taking all three aspects into account. Besides security, the proposed scheme can balance storage and computational efficiency to be a candidate for efficiently managing keys and controlling access in a hierarchy of mobile agents.

2.1.1 Review of Volker and Mehrdad’s scheme

Volker and Mehrdad [17] designed a tree-based mobile agent structure to support authentication, access control, and key management. Their structure is divided into two branches. One of which is the static branch that contains all of the unchangeable data; data that remains the same during the lifetime of the agent such as class codes, security policies, etc. The other branch, the mutable branch, contains data that can vary during the lifetime of the agent; these data include the state of the agent and the collected data.

To guarantee the integrity of this content against any alteration and forgery, well-known signature techniques are applied for sealing the contents modified by the last visited host [3]. In order to keep the confidential contents in both the static and mutable branches secret, Volker and Mehrdad also proposed an access control and key management strategy. In this strategy, folders are created for each visited host within the static/sctx/acl folder and the mutable/sctx/acl folder, containing the corresponding decryption keys authorized to access the confidential static and mutable files, respectively. Assume that public key infrastructure is implemented. The contents of the folder are encrypted with the corresponding host’s public key to protect against disclosure. Only the specific host that possesses the corresponding and unique private key can obtain the content of its folder.

To give a clearer description, a simple example which only shows the access control and key management strategy for static branch of mobile agents is illustrated in Fig. 1. Initially, assume that the existence of four files in the “classes” folder: agent.zip, retrieval.zip, rule.zip, and bid.zip is generated by the agent owner to be shared with the corresponding auction sites. Furthermore, the rule.zip is the strategy of the agent owner’s auctioning. The bid.zip describes the bidding for specific items. Except for agent.zip, all other files are kept confidential and need to be encrypted with the decryption keys $K_4$, $K_5$, and $K_6$, respectively. Note that the keys $K_3$, $K_5$, and $K_6$ are generated by the agent owner to be shared with the corresponding auction sites. Furthermore, the folders $S_1$, $S_2$, and $S_3$ are set up in behalf of the hosts eBay.com, Amazon.com, and Rakuten.com, respectively. The host eBay.com holds $K_3$, $K_5$, and $K_6$ that signify the host’s ability to access the
specific files retrieval.zip, rule.zip, and bid.zip. The host Amazon.com has keys $K_4$ and $K_5$, which in turn, allows it to decrypt the files retrieval.zip and rule.zip. Similarly, Rakuten.com holds $K_5$ and $K_6$, and thus it can access the files rule.zip and bid.zip.

With this access control and key management strategy, the agent owner separately encrypts $K_4$, $K_5$, and $K_6$ stored in the folder `static/sctx/acl/S1` with the public key of the host eBay.com to grantee these decryption keys against unauthorized obtainment. Using this way, only the host eBay.com that possesses the corresponding private key can individually retrieve $K_4$, $K_5$, and $K_6$ stored in the folders `static/sctx/acl/S1` and `static/sctx/acl/S2` and $K_5$ and $K_6$ kept in the `static/sctx/acl/S3` are encrypted with the public keys of Amazon.com and Rakuten.com, respectively. Only the authorized hosts Amazon.com and Rakuten.com can retrieve the decryption keys $(K_4, K_5)$ and $(K_5, K_6)$, respectively.

### 2.1.2 The performance analysis
Volker and Mehrdad’s scheme provides the necessary security services for mobile agent systems. However, the drawbacks which make their scheme quite inefficient are illustrated as follows:

1. **Large agent size.** The same decryption key is encrypted and stored in different folders with repetition. According to the above example, $K_4$ is duplicated in $S_1$ and $S_2$; $K_5$ is duplicated in $S_1$, $S_2$, and $S_3$; $K_6$ is duplicated in $S_2$ and $S_3$. This redundancy increases the size of the mobile agent.

2. **More public-key computation.** Repeated decryption keys imply that the agent owner must use more public-key computation to encrypt these decryption keys. According to the above example, the mobile agent has to separately encrypt $K_4$ stored in the folders $S_1$ and $S_2$ two times; $K_5$ stored in the folders $S_1$, $S_2$, and $S_3$ three times; $K_6$ stored in the folders $S_2$ and $S_3$ two times. From the perspective of computational cost required in the host, eBay.com has to decrypt three times to gain $K_4$, $K_5$, and $K_6$; Amazon.com need two decryptions of public-key cryptosystems to gain $K_4$ and $K_5$; Rakuten.com requires two decryptions of public-key cryptosystems to gain $K_5$ and $K_6$. As the computational cost of public-key cryptosystems is relatively costly [14], Volker and Mehrdad’s scheme requires more computational resources to repeatedly en/decrypt the decryption keys.

![Diagram](image-url)

**Fig. 1.** An example of Volker and Mehrdad’s strategy
2.2 Lin et al.’s scheme

2.2.1 Review of Lin et al.’s scheme
Lin et al. proposed a scheme following the concept of hierarchy structure [1]. Fig. 2 illustrates Lin et al.’s scheme with the same example as shown in Volker and Mehrdad’s. In the hierarchy, the decryption keys housed in leaf nodes can be derived from the superkey located within its parent’s node.

Initially, assume that the agent owner chooses two large primes, \( p \) and \( q \), and publishes \( n = pq \). Then, the agent owner chooses a secure \( K \) and assigns the public primes \( e_1 \), \( e_2 \), and \( e_3 \) to the confidential files retrieval.zip, rule.zip, and bid.zip, where \( e_1 \), \( e_2 \), and \( e_3 \) are relatively prime to \((p-1)(q-1)\) and numbers in the range of \([2, n-1]\). To generate cryptographic keys, the agent owner computes decryption keys \( nKDK_i \mod d_i = 1, 1 \leq i \leq 3 \), for these three auction sites eBay.com, Amazon.com, and Rakuten.com. In Fig. 2, the agent owner respectively calculates the superkeys \( SK_i = K^d_i \mod n \), \( SK_2 = K^d_2 \mod n \), and \( SK_3 = K^d_3 \mod n \), where \( e_i \times d_i \equiv 1 \pmod{\phi(n)} \) for all internal nodes \( N_1 \), \( N_2 \), and \( N_3 \). To achieve the confidentiality, the superkeys \( SK_1 \), \( SK_2 \), and \( SK_3 \) are encrypted with public keys corresponding to eBay.com, Amazon.com, and Rakuten.com, respectively. In addition, the signature technique is employed to keep the integrity of the mobile agent.

From the key derivation perspective, the host eBay.com obtains \( SK_1 \) with its own private key and then can use it to derive \( DK_1 = SK_1^{e_1} \mod n \), \( DK_2 = SK_1^{e_2} \mod n \), and \( DK_3 = SK_1^{e_3} \mod n \), respectively. Similarly, the host Amazon.com holds \( SK_2 \), which can be used to derive \( DK_2 = SK_2^{e_2} \mod n \) and \( DK_3 = SK_2^{e_3} \mod n \). The host Rakuten.com gains \( SK_3 \) and then derives \( DK_2 = SK_3^{e_2} \mod n \) and \( DK_1 = SK_3^{e_3} \mod n \).

2.2.2 The performance analysis
Clearly, the number of public-key computation in the Lin et al.’s scheme is fewer than Volker and Mehrdad’s scheme. However, exponential operations are required both during key generation and derivation [2]. These exponential operations are generally too costly and may also become a performance bottleneck for the agent owners and the visited hosts when serving several of these agents at the same time.

![Fig. 2. An example of Lin et al.’s scheme in a hierarchy](image-url)
2.3 Chen et al.’s scheme

2.3.1 Review of Chen et al.’s scheme
Chen et al.’s scheme also follows the concept of hierarchy structure [1]. Fig. 3 illustrates an example the same as both Volker and Mehrdad’s and Lin et al.’s schemes.

Firstly, the agent owner computes the following relationships between each pair of parent and descendant nodes:

\[ R_{12} = h(K_1 || N_1 || N_2) \oplus K_2, \]
\[ R_{13} = h(K_1 || N_1 || N_3) \oplus K_3, \]
\[ R_{24} = h(K_2 || N_2 || N_4) \oplus K_4, \]
\[ R_{25} = h(K_2 || N_2 || N_5) \oplus K_5, \]
\[ R_{35} = h(K_3 || N_3 || N_5) \oplus K_5, \]
\[ R_{36} = h(K_3 || N_3 || N_6) \oplus K_6, \]

where \( K_1, K_2, K_3, K_4, K_5, \) and \( K_6 \) are the cryptographic key of nodes \( N_2, N_3, N_4, N_5, \) and \( N_6, \) respectively, \( h(. \) is a collision-free one-way hash function [13], and “||” denotes string concatenation. After that, the agent owner makes \( R_{12}, R_{13}, R_{24}, R_{25}, R_{35}, \) and \( R_{36} \) public. For the confidentiality, \( K_1, K_2, \) and \( K_3 \) are encrypted with the public keys of eBay.com, Amazon.com, and Rakuten.com, respectively. For the integrity, the agent is sealed with the agent owner’s private key.

To derive the decryption key, the host Amazon.com uses \( K_2 \) to derive \( K_4 = h(K_2 || N_2 || N_4) \oplus R_{24} \) and \( K_5 = h(K_2 || N_2 || N_5) \oplus R_{25} \). Similarly, the host Rakuten.com and eBay.com can use \( K_5 \) and \( K_1 \) to recover \((K_3, K_6)\) and \((K_4, K_5, K_6)\) with the related relationships, respectively.

2.2.2 The performance analysis
In Chen et al.’s scheme, it is clear to see that only lightweight operations consisting of one-way hash functions and bit-wise XOR operators are employed instead of complicated exponential computations. Furthermore, the number of public-key computations is fewer than Volker and Mehrdad’s scheme. Hence, it is more efficient than Volker and Mehrdad’s and Lin et al.’s schemes.

However, publicizing the relationship between each pair of parent and descendant nodes is required for recovery of descendant’s key. It implies that the agent size is huge when each descendant has numbers of parents in the worst case. Therefore, compared with Volker and Mehrdad’s and Lin et al.’s schemes, the agent size in Chen et al.’s is not economic.

![Diagram](Fig. 3. An example of Chen et al.’s scheme in a hierarchy)
3 Proposed Scheme
To take security, storage, and computational efficiency into account, we propose a secure and efficient key management for hierarchical access control scheme in mobile agents. Our scheme adopts the concept of the hierarchy structure [1] and the partially ordered hierarchical structure shown in Fig. 4. The root node in the hierarchy, named as \( N_1 \), represents the agent owner who possesses the superkey, \( K_1 \), used to derive all the cryptographic keys. The internal nodes represent the host who holds the corresponding keys used to derive the cryptographic keys that are located in its descendent nodes. The leaf nodes represent the decryption keys used to en/decrypt the confidential files within the mobile agent. When a key can be used to derive a valid decryption, this has an existence of relationship between each pair of these two nodes. This relationship is useful for key derivation in our scheme.

To demonstrate our scheme, firstly, key generation and derivation of the proposed hierarchical key management are present. Next, the e-auction scenario is adopted to clearly illustrate the hierarchical key management structure [17], Lin et al.’s [12], and Chen et al.’s [2].

3.1 Key generation
The agent owner carries out the following steps to generate the cryptographic keys.

Step 1) Depending on the access policy that decides which file of the mobile agent the visited hosts can access, construct the hierarchy, Fig. 4 for example. The construction method uses a top-down approach wherein each node \( N_i \), \( 1 \leq i \leq n \), corresponds to one of the host in the hierarchy. If the key stored in \( N_i \) can derive one stored in \( N_j \) and \( i < j \leq n \), this indicates that \( N_j \) is a descendent node of \( N_i \) and a relationship between \( N_i \) and \( N_j \) exists.

Step 2) Choose 256-bit \( K_I \) for each node or leaf as its assigned key, \( 1 \leq i \leq n \), respectively.

Step 3) Compute all the related parameter \( R_j \) of \( N_i \) and \( N_j \) in the hierarchy according to the following rules:

- If the number of \( N_j \)'s parent node \( N_i < 1 \), then \( R_j \) does not exist.
- If the number of \( N_j \)'s parent node \( N_i \geq 2 \), then
  \[ R_j = R_{-j}(x) = \prod_{i \in \psi} (x - h(K_i \| N_i)) + K_j \mod p, \]  
where \( R_{-j}(x) \) is a polynomial in finite field \( F_p[x] \) and \( \psi \) denotes the group of \( N_j \)'s parents.
- Else, \( R_j = h(K_i \| N_i \| N_j) \oplus K_j \),

where “\( \oplus \)” is a bit-wise exclusive-or operation.

Step 4) Store the all node identities and relationships into a public space of the mobile agent.

Finally, in order to guarantee the integrity of confidential files, well-known signature techniques [14] are applied.

3.2 Key derivation
If a host corresponds to a node \( N_i \) then that host can derive the entire cryptographic keys of its descendent nodes; i.e., all \( N_j \) such that \( i < j \leq n \). The host \( N_i \) will be able to decrypt the authorized files when it derives the decryption keys of the leaf nodes \( N_j \) with its assigned key \( K_i \).

With \( R_j \) constructed in Eq. (1) or Eq. (2), the host \( N_i \) derives the cryptographic key of its descendent node \( N_j \) according to the following rules:

- If the number of \( N_j \)'s parent node \( \geq 2 \), then
  \[ K_j = R_{-j}(x = h(K_i \| N_i)). \]  
- If the number of \( N_j \)'s parent node = 1, then
  \[ K_j = h(K_j \| N_j) \oplus R_j. \]

Fig. 4. The hierarchical key management structure
3.3 A simple example of the proposed scheme for mobile agents

To give a clearer description, only the key management of hierarchical access control strategy for static branch of mobile agents is shown as Fig. 5. Following our scheme will generate cryptographic key for each visited host. This key is only shared between the agent owner and the specific visited host. This implies that this key can guarantee the confidentiality of not only static branch but also mutable branch.

Fig. 5 illustrates a hierarchical structure of access control and key management. Consider the example in Fig. 5. The host eBay.com has the right to access the confidential files retrieval.zip, rule.zip, and bid.zip. The host Amazon.com is authorized to access the confidential files: retrieval.zip and rule.zip. The host Rakuten.com can access the confidential files: retrieval.zip and bid.zip. The host Amazon.com is authorized to access the confidential files: retrieval.zip and rule.zip. The host Rakuten.com can access the confidential files: rule.zip and bid.zip. To prepare for this scenario, the agent owner chooses \( K_1, K_2, K_3, K_4, K_5, \) and \( K_6 \), respectively. Afterwards, the agent owner uses them to compute the relationship following Eqs. (1) and (2) as

\[
\begin{align*}
R_{12} &= h(K_1 || N_1 || N_2) \oplus K_2, \\
R_{13} &= h(K_1 || N_1 || N_3) \oplus K_3, \\
R_{24} &= h(K_2 || N_2 || N_4) \oplus K_4, \\
R_{35} &= R_{35}, \\
R_{45}(x) &= (x - h(K_2 || N_2))(x - h(K_1 || N_1)) + K_5 \mod p, \\
R_{36} &= h(K_3 || N_3 || N_6) \oplus K_6,
\end{align*}
\]

where \( K_2, K_3, K_4, K_5, \) and \( K_6 \) are the cryptographic key of nodes \( N_2, N_3, N_4, N_5, \) and \( N_6 \), respectively. Because \( R_{35} \) and \( R_{35} \) are the same (equal to \( R_{45}(x) \)), the relationship is only presented as \( R_{45}(x) \). After that, the agent owner makes the five relationships including \( R_{12}, R_{13}, R_{24}, R_{35}(x), \) and \( R_{36} \) public. In addition, \( K_1, K_2, \) and \( K_3 \) are encrypted with the public keys of eBay.com, Amazon.com, and Rakuten.com, respectively.

The host Amazon.com can obtain \( K_2 \) with its own private key. To derive the decryption key, according to Eqs. (4) and (3), Amazon.com uses \( K_2 \) to derive \( K_4 = h(K_2 || N_2 || N_4) \oplus R_{24} \) and \( R_{45}(x) = (h(K_2 || N_2) - h(K_2 || N_1))(h(K_2 || N_2) - h(K_1 || N_1)) + K_5 \mod p = 0(h(K_2 || N_2) - h(K_1 || N_1)) + K_5 \mod p = K_5 \).

The host Rakuten.com uses its own private key to obtain \( K_3 \) and then utilizes it to derive \( K_6 = R_{45}(x = h(K_3 || N_3)) \) and \( K_6 = h(K_3 || N_3 || N_6) \oplus R_{36} \). The host eBay.com uses \( K_1 \) to derive \( K_3 = h(K_1 || N_1 || N_2) \oplus R_{12} \) and \( K_3 = h(K_1 || N_1 || N_3) \oplus R_{13} \). After that, eBay.com can use the derived \( K_2 \) and \( K_3 \) to compute \( K_4, K_5, \) and \( K_6 \) in the same Amazon.com and Rakuten.com.

In such a way, the key management mechanism can make eBay.com, Amazon.com, and Rakuten.com access the authorized files to achieve the access control.

Fig. 5. An example of the proposed scheme
4 Security and Performance Analysis
This section examines the security and the performance included storage and computational efficiency of the proposed scheme.

4.1 Security analysis
From the perspective of security, confidentialities of both the assigned keys and the protected files are necessary to assure that only the authorized host can recover the assigned key to obtain the specific file. Furthermore, the integrity of the mobile agent is required for guarantee of the expected purpose.

**Proposition 1.** Confidentiality of assigned keys: Except the specific host, no one can eavesdrop in the key assigned to the host by the agent owner.

**Proof.**
In the proposed scheme, the assigned key used to allow the specific host to derive all authorized decryption keys is encrypted with this host’s public key. In a public key cryptosystem [14], it is infeasible to obtain the message encrypted with a public key without the corresponding private key. Hence, no one can obtain the assigned key except the specific host. The confidentiality of assigned key is done.

**Proposition 2.** Confidentiality of mobile agents: Except the specific host, no one can derive the decryption key to obtain the corresponding content within the mobile agent.

**Proof.**
When an adversary obtains the public relationship \( R_i \), he/she wants to obtain the confidential file of the mobile agent, it must gain knowledge of the corresponding decryption key. This means that the adversary has to derive decryption key from Eq. (1) or Eq. (2). To do that, the adversary has to prepare the corresponding key assigned to a specific host. Based on Proposition 1, the adversary cannot perform Eq. (3) or Eq. (4) to derive a valid decryption key. Even though the adversary tries to guess the assigned key, it is still not feasible to compute the same hashed value due to random oracle of one-way hash functions [16]. This implies that it is difficult to guess a valid assigned key. Therefore, the adversary has no idea to obtain decryption key to attain unauthorized access to the confidential file of the mobile agent.

**Proposition 3.** Integrity of the mobile agent: Any alteration on the mobile agent will be detected by any verifier.

**Proof.**
In the proposed scheme, the mobile agent is sealed with a digital signature by the agent owner. Due to the capability of digital signature techniques [14,16], any alteration on the signature will cause the signature verification with the agent owner’s public key to get a failure. Hence, any verifier can check whether the integrity of the mobile agent is satisfied.

Accordingly, our hierarchical key management mechanism is secure enough to achieve the access control facility required by the mobile agents.

4.2 Performance analysis
In this subsection, the performance including the storage space required and the computational load demanded is evaluated. Assume that an agent will visit \( v \) hosts and carry \( u \) confidential files, let \( F_i \) be the number of files which the visited host \( i \) can access, where \( 1 \leq i \leq v \). In addition, assume that \( \omega \) is the total number of nodes involved in the hierarchy and \( \gamma_{j} \) is the total relationship number of the descendent node \( j \), \( 1 \leq j < \omega \). In other words, \( \gamma_{j} \) is the number of the descendent node \( j \)’s parents. Let \( \alpha \) is the number of the descendent node which has more than one parent, \( 0 \leq \alpha < \omega \).

**Proposition 4.** Reasonable agent size: The agent size does not follow the number of descendant node’s parents in a hierarchical access control.

**Proof.**
It is obvious to see that the number of relationship depends on the number of the descendnet node. Compared with Chen et al.’s scheme [2], the size of a mobile agent does not depend on the number of the descendnet node’s parents in our scheme. From the Fig. 3 and Fig. 5, the number of relationships required to be publicized are respectively 6 and 5 in Chen et al.’s scheme and our scheme. Hence, the size of a mobile agent can be reduced.

In the comparisons among the related works and our scheme, Volker and Mehrdad’s scheme [17] requires to store \( \sum_{i \leq \omega} F_i \) decryption keys. In the Lin et al.’s, Chen et al.’s, and ours schemes, only \( v \) keys need to be stored. It implies that the size of a mobile agent in Volker and Mehrdad’s scheme is the largest. The comparison is illustrated in Table 1.

In addition, in our scheme, the relationship between a specific descendnet node and its only one parent is generated through a one-way hash function. The hash function, such as SHA-1 and SHA-256 [13], can take an arbitrary-length input and return an output of fixed length, such as 256-bit
in SHA-256. Furthermore, the length of relationship indicated to a specific descendent node is 512-bit, in which \( p \) is 512-bit string. As the assumption, in the schemes Lin et al. [12] and Volker and Mehrdad [17], the length of every cryptographic key is 512-bit. At least 512\( v \) bits and 512\( \sum F_i \) bits are required in Lin et al.’s scheme and in Volker and Mehrdad’s, respectively. Compared with them, our scheme and Chen et al.’s scheme [2] only require 256\( v \) bits to store the keys, which indicate a smaller size of the stored cryptographic keys in the our scheme and Chen et al.’s scheme.

Compared with our scheme and Chen et al.’s scheme, the length of relationship stored in the mobile agent is the same such as 256-bit if the descendent node has only one parent. When the number of a descendant’s parent node is \( n \), 2 \( \leq n \), the size of a mobile agent will increase 512 bits and 256\( * n \) bits in our scheme and Chen et al.’s, respectively. Except the case that the number of a descendant’s parent node is 1 or 2, it is clear to know that the size of mobile agent is smaller in our scheme and Chen et al.’s scheme.

That is, the size stored in the mobile agent is (256(\( \omega-1+\alpha \)) \leq 256\( \sum k_{-j} \)) in the Chen et al.’s scheme in Table 1.

However, the total space for storing publicized relationships in our scheme and Chen et al.’s scheme may exceed that of Lin et al.’s when disorderliness appears in the hierarchy. For cryptographic key \( K_i \) of the node \( N_j \) whose degree is equal to one, we recommend adopting this equation \( K_i = h(SK_i \| N_i \| N_j) \) to eliminate the relationship. Careful formulation of the access policy can eliminate the agent size effectively.

**Proposition 5.** Tiny computation load: For speed up of the visited host’s dealing with the mobile agent, any time-consuming computations must be avoided.

**Proof.**
Since each host’s folder static/sctx/acl must be kept secret, it has to be encrypted using RSA with the corresponding public key [14,16]. Assume that each en/decryption procedure requires one exponential operation in a cryptosystem. Let \( T_{exp} \) be the computation cost of an exponential operation and \( T_{hash} \) be the computation cost of a hash function. Volker and Mehrdad’s scheme requires (2\( \sum F_i \))\( T_{exp} \) to en/decrypt the keys. But, our scheme, Lin et al.’s scheme, and Chen et al.’s scheme require only (2\( v \))\( T_{exp} \) to en/decrypt the keys. Hence, the performance of our scheme, Lin et al.’s scheme, and Chen et al.’s for en/decrypting the keys is higher than Volker and Mehrdad’s.

From the key generation and key derivation, it is clear to see that only light-weight one-way hashing and XOR operations are required. The logical XOR operations only require an extremely lightweight computation cost, and thus can be safely omitted without upsetting the overall performance evaluation. Furthermore, when the number of a descendant’s parent node is \( n \), 2 \( \leq n \), a modular polynomial is demanded in our scheme. From the perspective of key generation/derivation, our scheme requires (\( \omega-1-\sum k_{-j}+\sum k_{-j} \))\( T_{hash} \) to construct the relationship between all keys and (\( \omega-1 \))\( T_{hash} \) to derive all the used cryptographic keys. However, Lin et al.’s scheme requires one exponential operation to generate/derive the keys from its superkey thus it needs (\( \sum F_i \))\( T_{exp} \) to generate or derive all the used keys.

<table>
<thead>
<tr>
<th>Table 1. Performance comparisons among our scheme and related works</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our scheme</strong></td>
</tr>
<tr>
<td>Size for storing keys</td>
</tr>
<tr>
<td>Size of publicized relationship</td>
</tr>
<tr>
<td>Computation cost for en/decrypting keys</td>
</tr>
<tr>
<td>Computation cost for generating/deriving all keys</td>
</tr>
</tbody>
</table>
To measure the performance, we cite Crypt++ 5.2.1 Benchmarks [5] for cryptographic algorithms which are coded in C++, compiled with Microsoft Visual C++ .NET 2003, ran on a Pentium 4 2.1 GHz processor under Windows XP SP 1. According to [5], the speed benchmark of hash function SHA-256 is about 44 MB/s. This implies that 1-byte computation of SHA-256 spends 22.7 ns. Processing 256-bit output of SHA-256 only requires 5.8 µs. On other hand, the time spent on computing a 1024-bit exponential computation, such as RSA 1024, is 0.18 ms for encryption and 4.77 ms for decryption. Accordingly, the computation cost of SHA-256 is 31 times less than RSA 1024 encryption and 822 times less than RSA 1024 decryption, respectively. In other words, a total of $172 \times 10^3$ SHA-256 operations can be performed within one second and likewise $5.56 \times 10^3$ and 210 operations can be performed for RSA 1024 encryption and decryption, respectively. The Lin et al.’s scheme is RSA-based. Obviously, our scheme and Chen et al.’s scheme are more efficient.

In sum, our scheme generally has a smaller agent size and higher computation performance. Even though the computation performance may not good enough than Chen et al.’s scheme, our scheme has more economical agent size. Hence, our scheme can be treated as the balance to both the mobile agent size and the computation performance. We believe that the proposed scheme will be more acceptable than other related works and encouraging for a practical implementation in the real environment.

5 Conclusions
In this paper, we have designed an efficient key management scheme to provide hierarchical access control mechanism for the agent system. Besides security, the proposed scheme uses only lightweight hash functions and exclusive-or operations and polynomials instead of time-consuming modular exponential computations to generate/derive keys. Moreover, the agent size is effectively reduced regardless of the number of descendent node’s parents. Among the related works, the proposed scheme can balance storage and computational efficiency and is a candidate for efficiently managing keys and controlling access in a hierarchy of mobile agents. Hence, the computational bottleneck of visited hosts and the traffic jam will disappear in the proposed scheme, which is encouraging for a practical implementation in the real environment.

References:


