Journal of Information Assurance and Security 1 (2008) 59-66

Hierarchical Multi-Party Key Establishment for Wireless Networks

Sigurd Eskeland Vladimir Oleshchuk
University of Agder
Grooseveien 36
N-4876 Grimstad, Norway
{sigurd.eskeland, vladimir.oleshchuk}@uia.no

Abstract: A common property of practically all multi-party key agreement protocols is that they are non-hierarchical, i.e., they do not support user hierarchies. However, in real life it is likely that members of groups and organizations differ in ranking according to their job positions. Thus, it is reasonable that participants of a certain ranking could access more privileged information than participants of lower rankings. Hierarchical access control (HAC) schemes address this issue, but in contrast to provide secure distribution of hierarchically-arranged short-term session keys, HAC schemes provide secure distribution of hierarchically-arranged long-term, predefined keys. In this paper, we present an efficient hierarchical multi-party key agreement protocol that is well-suitable for wireless networks. We also present a closely related hierarchical centralized key distribution protocol for totally-ordered and partially-ordered security classes.

Keywords: Cryptographic protocols, conference key agreement, hierarchical key distribution.

I. Introduction

Key establishment protocols can be categorized into key transfer protocols and key agreement protocols. Regarding key transfer protocols, one entity generates and transfer a secret session key securely to one or more participants over an insecure network. In key agreement protocols, the users actively participate in the establishment of the secret shared session keys by collaboratively contributing themselves to the values of the keys. Key agreement protocols for groups or teams of more than two participants are known as multi-party key agreement or conference key agreement protocols. Several conference key agreement protocols has been previously proposed, see e.g., [1, 2, 3, 8], and common for these is that they are non-hierarchical, which of course is suitable when the group members have same ranking. However, in real life, it is likely that there are circumstances and job situations where the participants have different rankings according to their job positions. Individuals of higher ranking are entrusted more privileges and thus more confidential information than the participants of lower ranking. For example, in the medical scenario, medical care is provided by medical teams that are composed of doctors and nurses where the doctors have a higher ranking than the nurses. Doctors are thus likely to be entrusted more confidential medical information than the nurses of the same team. Assuming that medical data is represented by means of electronic patient records, it is essential that the medical data is transferred securely to the legitimate team members according to their ranking. This could be achieved by means of encryption which requires secure establishment of shared secret keys according to the user hierarchy.

In this paper, we present a hierarchical multi-party key agreement protocol suitable for ad hoc wireless networks. It complies with an hierarchical arrangement of user classes (or security classes) where each user is associated with one security class that correspond to his or her job position. The protocol enables the participants to secretly establish one secret hierarchical session key for each security class – subsequently referred to as *class key*. Moreover, we present a closely related hierarchical centralized key distribution scheme, where class keys are originates from one party.

An essential security property is that the participants of any given security class can obtain the secret class keys that are established by their own and underlying security classes, while it is computationally infeasible to obtain class keys of overlying security classes.

II. Related work

The concept of hierarchical key agreement protocols seems to be somewhat absent in the literature. Hwang et al. [4] proposed a hierarchical key agreement protocol based on the multi-party key agreement key protocol in [7] that incorporates a hierarchy of classes that enables the participants to securely establish class keys for each class. Moreover, the participants of a given class can obtain the class keys of the underlying security classes, while it is prohibited that class keys of overlying security classes can be obtained. Unfortunately, the protocol provides no means to verify the user

Received January 14, 2008

levels which allows any user to pretend to have a higher ranking than his or her legitimate ranking. Thus, the protocol fails to provide secure and trustable user hierarchies.Another major disadvantage is that it is highly inefficient due to that the number of rounds equals the number of participants. Eskeland proposed an efficient hierarchical key agreement protocol that requires only two rounds of broadcasting [5]. An improved version with increased computational efficiency and improved user authentication was published at the Third International Symposium on Information Assurance and Security [6] whereof this is an extended paper.

Hierarchical key establishment is not to be confused with hierarchical access control schemes (HAC) and tree-based key management schemes (TBKM). Hierarchical access control is a class of cryptographic schemes that supports establishment and deduction of long-term predefined cryptographic keys that are hierarchically arranged, so that users of a given security class are able to securely compute such keys associated with their own and underlying security classes, while computation of keys associated with overlying security classes is prevented.

While the hierarchical key establishment schemes presented in this paper provide secure ad-hoc establishment of a hierarchy of "fresh" sessions keys, (referred to as class keys), HAC schemes do in contrast enable computation of predefined static keys from a key hierarchy. Providing computation of hierarchical predefined keys and not hierarchical sessions keys is reasonably a considerable limitation of the applicability and usefulness of such schemes.

However, due to access control purposes, many HAC schemes are compliant with user dynamics, i.e., inclusion and exclusion of users and corresponding renewal of hierarchical keys for the pertaining security classes. Examples of HAC schemes can be found in [14, 15, 16, 17].

Tree-based key management schemes can be regarded as centralized key distribution where the users of a group establish a *key tree* where the users are arranged as leaf nodes of the tree. Due to the to tree structure, it allows them to obtain a common key that is the root. Thus, such schemes are not hierarchical due to that the users obtain one shared secret key. Examples are Tree-based Group Diffie-Hellman agreement [18], and others in [19, 20]. An essential issue about HAC and TBKM schemes is support of group dynamics (joining and leaving of users) so that protocol re-run is not necessary for each user update.

III. Preliminaries

A. Security properties

The security properties of the hierarchical multi-party key agreement protocol presented in Section 4 are as follows:

- Class key confidentiality. Only legitimate participants (or insiders) must be able to establish and obtain the class keys (or hierarchical session keys). It must not be possible to deduce new class keys by means of former class keys of any security classes.
- User key confidentiality. It must be prevented that long-term secret user keys can be disclosed.
- User authentication. It must be securely established that each member is a genuine member of the claimed security class. Thus, user authentication must include certification of the pertaining security class of each user.
- Forward secrecy. Compromise of long-term secret user keys must not reveal formerly established class keys.
- Direction. The hierarchical multi-party protocol provides one secret class key for each level of the user hierarchy. It must be provided that each user of a given security class can only obtain class keys that are associated to his or her own and underlying security classes, and prohibited that class keys of overlying classes can be obtained.

The proposed schemes are broadcast-oriented and efficient, and are thus well-suitable for ad hoc wireless networks. Broadcasting efficiently distributes the key establishment messages from user user to the others, but allows an adversary to easily eavesdrop the key establishment messages. We can moreover assume that an adversary has been a former participant and may hold former keys. Consequently, the security properties must be satisfied in presence of passive adversaries with such capabilities.

An active adversary can modify (i.e., add, replace, replay) any broadcasted messages he or she wants. The adversary may, for example, attempt to impersonate any legitimate user by replaying old messages where the associated former class key is known. It must be infeasible to compromise the protocol without being detected or that the protocol does not terminate.

No online trusted party for is required establishment of class keys.

B. Hierarchical preliminaries

Let $\mathcal{U} = \{P_1, \ldots, P_m\}$ denote a team or group of *m* participants where each participant $P_i \in \mathcal{U}$ is associated with a hierarchy level and where L_i denotes the hierarchy level of P_i. We assume that each security level ℓ contains *one* security class $S_\ell \subseteq \mathcal{U}$ that includes all participants of that security level:

$$S_\ell = \{P_j \,|\, P_j \in \mathcal{U} \quad \text{and} \quad \ell = L_j\}$$

where $\ell \in \{1, \ldots, \lambda\}$ and λ denotes the top security level. We have that the security classes are partitions of $\mathcal{U}$ so that

$$\bigcup_{1 \le i \le \lambda} S_i = \mathcal{U} \qquad \text{and} \qquad S_i \cap S_j = \emptyset$$

where $i, j \in \{1, \ldots, \lambda\}$ and $i \neq j$. Thus, each participant $P_i \in \mathcal{U}$ is associated with *one* security class such that $P_i \in S_\ell$ for some $1 \le \ell \le \lambda$.

We denote the hierarchical ranking of the security classes according to the relation $\prec$ so that

$$S_i \prec S_j \quad \text{if} \quad i < j$$

which indicates that S_j has a higher ranking than S_i. The higher security level, the higher is the ranking in the hierarchy.

IV. The protocol

In this section, we present the hierarchical multi-party key establishment protocol. It allows any composition of hierarchically ranked participants to establish of a corresponding hierarchy of conference keys over ad hoc wireless networks. Authentication allows any participant to detect if the hierarchical user arrangement is compromised, i.e., if a participant pretends to have a higher ranking than his or her legitimate ranking. The protocol is based on [5, 6] which are partly based on the multi-party key agreement protocol in [1].

A. Initializations

A trusted third party (TTP) is required to provide the long-term secret user keys which are the basis for the user authentication of the protocol. The TTP has thus not a part in running the protocol. The TTP computes the composite modulus $n = p \cdot q$ of two distinct secret large prime numbers *p* and *q*. According to the RSA cryptosystem [9], the TTP selects a number *e* that is relative prime to $\phi(n) = (p-1) \cdot (q-1)$ and computes the secret key *d* so that $e \cdot d \equiv 1 \pmod{\phi(n)}$ where *e* and *n* are public.
For each user $P_i \in S_\ell$, the TTP computes an identifier as the hash of the concatenation of user identity ID_i and the pertaining user level L_i as $id_i = f(ID_i \,|\, L_i)$ where f is a secure one-way function. Based on id_i, the TTP computes the secret user key

$$s_i = id_i^d \pmod{n}$$

The secret long-term user keys are confidentially distributed to the respective users.

B. User arrangement

In agreement with the directions in Section 3.2, the users are arranged in increasing order according to their ranking. Moreover, we assume that within each security class and across the security classes, the users are linearly ordered. This means that the members of $\mathcal{U}$ form a sequence or string of users arranged in increasing order according to their ranking, where $P_1 \in S_1$ denotes the sequentially first user and $P_m \in S_\lambda$ denotes the sequentially last user. The users P_i and P_{i+1} are adjacent for $1 \le i < m$.
The sequential user arrangement implies that if $P_i \in S_\ell$ is sequentially positioned first in S_ℓ, he or she is adjacent with $P_{i-1} \in S_{\ell-1}$ of the underlying class $S_{\ell-1}$. Thus, $\ell = L_{i-1} + 1 = L_i$. Likewise, $P_i \in S_\ell$ is positioned at the end of S_ℓ if $\ell = L_{i+1} - 1 = L_i$.
The participants could, for instance, be ordered within each security class by sorting according to their identities. To ensure different user order for each session, the users within each security class could be ordered according to $f(ID_i, T)$ where f is a hash function and T is a timestamp. Moreover, a change in the hierarchy, inclusion of new participants or participants leaving requires protocol re-run.

C. The protocol

The following cryptosystem is certifiable according to the identity and the security level of each participant. Participants pretending to have a higher ranking than their legitimate ranking will be revealed according to the user authentication. The authentication is analogous to the authentication scheme of [10, 11]. We assume that the TTP makes n, e, α public where the element α is of maximal order in $\mathbb{Z}_n^*$. The protocol goes as follows:
Step 1. Each participant $P_i \in \mathcal{U}$, $1 \le i \le m$, generates a random secret number $r_i \in \mathbb{Z}_n$, and computes and broadcasts

$$x_i = \alpha^{e \cdot r_i} \pmod{n}$$

Step 2. Each participant P_i for $2 \le i \le m-1$ computes

$$v_i = k_{i-1,i} - k_{i,i+1}^2 \pmod{n}$$

where $k_{i-1,i} = x_{i-1}^{r_i} \pmod{n}$ and $k_{i,i+1} = x_{i+1}^{r_i} \pmod{n}$ are secretly established Diffie-Hellman keys that are shared between P_i and P_{i-1}, and P_i and P_{i+1}, respectively. The squaring of the second term of v_i is to ensure the one-way security property so that $P_i \in S_\ell$ cannot obtain class keys K_l for higher classes where $l > \ell$.
The participants P_1 and P_m, who do not have *two* adjacent users, compute a number linking to the current session, say, $v_j = c$ for $j \in \{1, m\}$ where $c = f(x_1|x_2|\ldots|x_m)$ represents the current session due to the concatenation of the session dependent numbers and f denotes a secure one-way function. For authentication, each user $P_i \in \mathcal{U}$ computes

$$w_i = s_i \cdot \alpha^{r_i \cdot f(x_i, v_i, c)} \pmod{n}$$

Each participant $P_i \in \mathcal{U}$ broadcasts (ID_j, L_j, v_i, w_i).
Step 3. Authentication. Each participant $P_i \in \mathcal{U}$ authenticates the other participants $P_j \in \mathcal{U}$, $i \neq j$, by verifying

$$w_j^e \stackrel{?}{\equiv} id_j \cdot x_j^{f(x_j, v_j, c)} \pmod{n}$$

where $id_j = f(ID_j \,|\, L_j)$.
Step 4. Class key establishment. We define the class key K_ℓ for a given class S_ℓ as the DH key $k_{j,j+1}$ established of the sequentially first participant $P_j \in S_\ell$ of that class and the adjacent participant P_{j+1}. Thus, $\ell = L_{j-1} + 1$ where $L_0 = 0$ is the initial condition.

Due to that the users are sequentially arranged in agreement to the increasing ordering of the security classes, each participant is able to deduce the DH-keys of the preceding participants, i.e., participants of underlying security classes. The converse is prohibited according to the one-way property.
Each participant $P_i \in S_\ell$ can compute the secret DH keys preceding participants $k_{j-1,j}$, $i > j$, according to the recurrence relation

$$k_{j-1,j} = v_j + k_{j,j+1}^2 \pmod n$$

Accordingly, $P_i \in S_\ell$ can compute the class key of his or her security class and underlying class keys K_γ, $1 \leq \gamma \leq \ell$, according to

$$K_\gamma = k_{j,j+1} = \alpha^{e r_j r_{j+1}} \pmod n$$

where P_j is the sequentially first participant of S_γ, i.e., $\gamma = L_{j-1} + 1$ where $L_0 = 0$.
Each participant $P_i \in \mathcal{U}$ can verify that preceding participants $P_j \in \mathcal{U}$, $j < i$, have computed v_j according to the protocol. This is done by checking that

$$v_j^2 \stackrel{?}{\equiv} \widehat{k}_{j-1,j}^2 - 2 \cdot \widehat{k}_{j-1,j} \cdot k_{j,j+1}^2 + k_{j,j+1}^4 \pmod n$$

holds where $\widehat{k}_{j-1,j}$ is the deduced candidate DH key.

D. Example

To illustrate the protocol and hierarchical user arrangements, here is an example of 7 members of two security classes, $S_1 = \{P_1, P_2, P_3\}$ and $S_2 = \{P_4, P_5, P_6, P_7\}$, where $S_1 \prec S_2$. In agreement with Step 4, $P_7 \in S_2$, holding $k_{6,7}$, computes the class key of S_2 according to

$$K_2 = v_5 + (v_6 + k_{6,7}^2)^2 = k_{4,5} = \alpha^{e r_4 r_5} \pmod n$$

Next, $P_7 \in S_2$ obtains the class key of the underlying security class S_1 by computing

$$K_1 = v_2 + \big(v_3 + (v_4 + k_{4,5}^2)^2\big)^2 = k_{1,2} = \alpha^{e r_1 r_2} \pmod n$$

V. Security analysis

Class key confidentiality. Key secrecy is based on the Diffie-Hellman computational problem, meaning that knowing $\alpha^x \pmod p$ and $\alpha^y \pmod p$ where p is a large prime, it is infeasible to find $\alpha^{xy} \pmod p$. Also note that due to the Discrete Logarithm Problem, it is computationally infeasible to deduce x given $\alpha^x \pmod p$. Accordingly, this holds for our scheme given where two participants P_i and P_{i-1} respectively hold the secrets r_i and r_{i-1}. Given the public numbers $\alpha^{e r_i} \pmod p$ and $\alpha^{e r_{i-1}} \pmod p$, the shared secret $\alpha^{e r_{i-1} r_i} \pmod p$ is protected due to the Diffie-Hellman computational problem.
User key confidentiality. Note that w_i is composed of two secret factors, s_i and $\alpha^{r_i \cdot f(x_i, v_i, c)}$. Due to the RSA assumption where $\phi(n)$ is unknown because of the unknown factorization of n; given e, it is computationally infeasible to obtain $e^{-1} = d \pmod{\phi(n)}$. This effectively prohibits that secret user keys can be obtained as $id_i^{(e^{-1})} \pmod n$ given id_i.
Accordingly, given x_i, it is computationally infeasible to obtain α^{r_i} since it is computationally infeasible to find $f(x_i, v_i, c)^{-1} \pmod{\phi(n)}$ due to the RSA assumption. Moreover, due to the Discrete Logarithm Problem, it is computationally infeasible to obtain r_i given x_i, which prevents establishment of α^{r_i}. This prevents deduction of the secret factor $\alpha^{r_i \cdot f(x_i, v_i, c)}$ which accordingly protects the secret user key s_i.
Direction. The one-way property prohibits that $P_i \in S_\ell$ can obtain K_l if $S_\ell \prec S_l$. This is ensured by squaring the last term of v_j. Since n is the product of two large secret primes, the value of $\phi(n)$ is unknown, and it is thus computationally infeasible to find roots in $\mathbb{Z}_n$. In order to obtain the succeeding secret DH key $k_{i+1,i+2}$, $P_i \in \mathcal{U}$, holding $k_{i,i+1}$, must solve

$$\begin{aligned} k_{i+1,i+2} &= \sqrt{k_{i+1,i+2} - v_{i+1}} \pmod n \\ &= \sqrt{k_{i,i+1} - (k_{i,i+1} - k_{i+1,i+2}^2)} \pmod n \\ &= \sqrt{k_{i+1,i+2}^2} \pmod n \end{aligned}$$

which is computationally infeasible since the factorization of n is unknown.
It is essential that each user $P_i \in \mathcal{U}$ computes v_i according to the protocol. The protocol could be subverted if a malicious user $P_i \in \mathcal{U}$ would broadcast $v_j = k_{j-1,j} - k_{j,j+1} \pmod n$ since this would break the directional property, allowing P_{i-1} to deduce $k_{i,i+1}$. Attempts of such violations will, however, be detected by an honest participant who will deduce the candidate DH key as $\widehat{k}_{j-1,j} = v_j + k_{j,j+1}^2 = k_{j-1,j} - k_{j,j+1} + k_{j,j+1}^2$. This will cause that the verification

$$\begin{aligned} v_j^2 &= (k_{j-1,j} - k_{j,j+1})^2 \\ &\neq \widehat{k}_{j-1,j}^2 - 2\,\widehat{k}_{j-1,j}\,k_{j,j+1} + k_{j,j+1}^4 \pmod n \end{aligned}$$

does not hold, and the protocol aborts.
User authentication. User authentication is analogous to that in [10, 11], and is achieved due to the user signature computed by each user $P_i \in \mathcal{U}$. The user signature is constituted by

$$\begin{aligned} x_i &= \alpha^{e \cdot r_i} \pmod n \\ w_i &= s_i \cdot \alpha^{r_i \cdot f(x_i, v_i, c)} \pmod n \end{aligned}$$

where r_i is secretly known only by $P_i \in \mathcal{U}$, and s_i is the secret long-term user key.
An adversary may attempt to forge a valid signature w_i by raising it to a power $a'_j = f(x'_j, v'_j, c')$ that represents another context according to

$$w'_j = w_i^{a'_j} \pmod n$$

This will fail since w'_j will correspond to $id'_j = id_i^{a'_j} \pmod n$ which means that the adversary must overcome the difficulty of reversing the hash function f by finding ID'_j and L'_j so that $id'_j = f(ID'_j \mid L'_j)$.
A similar authentication scheme is found in [12] where the verification is analogous to

$$y_j^e \stackrel{?}{\equiv} ID_j \cdot x_j^{f(T)} \pmod n$$

where $x_j = \alpha^{e \cdot r_j} \pmod n$, $y_j = s_i \cdot \alpha^{r_j \cdot f(T)} \pmod n$ and T is a timestamp. This scheme is not resistant to the Extended Euclidian Algorithm attack [13]. If e and $f(T)$ are relatively prime, we can find two integers u, v, so that $e \cdot u = 1 + f(T) \cdot v$. An adversary can thus pick a valid id_j, and compute $x_j = ID_j^v \pmod n$ and $y_j = ID_j^u \pmod n$. The attack succeeds in the scheme in [12] because

$$y_j^e = ID_j \cdot x_j^{f(T)} = (ID_j^u)^e = ID_j \cdot (ID_j^v)^{f(T)} \pmod n$$

This attack is thwarted in our scheme since x_j is included in certifying power $f(x_j, v_j, c)$.
Forward secrecy is defined as when a long-term key is compromised, class keys that were previously established using that long-term key should not be compromised too [8, p. 50]. Compromise of long-term user keys would enable an adversary to obtain $\alpha^{r_i \cdot f(x_i, v_i, c)}$ given w_i, $1 \leq i \leq m$. As noted, the secret r_i is protected due to the Discrete Logarithm Problem. Thus, the pertaining $k_{i,i+1}$ or $k_{i-1,i}$ cannot be deduced by means of long-term user keys, and forward secrecy is provided.

VI. Generalizing the protocol

The protocol presented in the previous section can be put in a more general form. In the general representation, any two-party key establishment scheme, user authentication scheme and one-way function could be used as building blocks, whereas the given security assumptions would consequently then rely on the actual security properties of the applied building blocks. Since user authentication may or may not be integrated to key establishment, we will here only consider the general class key establishment.
The same assumptions about user alignment apply for the generalized protocol so that the users are sequentially arranged in increasing order according to their ranking. Given any secure two-party key establishment protocol, each participant $P_i \in \mathcal{U}$ establishes the secret session keys $k_{i-1,i}$ (if $i > 1$) and $k_{i,i+1}$ (if $i < m$) shared respectively with $P_{i-1} \in \mathcal{U}$ and $P_{i+1} \in \mathcal{U}$.
Then $P_i \in \mathcal{U}$ computes and broadcasts

$$v_i = k_{i-1,i} - f(k_{i,i+1})$$

where f is a secure one-way function. Note that the computations could be modular or non-modular. This could in practise depend on the integer size of actual implementations. However, avoiding the modulus operator would obviously increase the computational efficiency.
A further generalization pertains operators which has so far been confined to subtraction and addition. Division (and subsequent multiplication for class key restoration) would work fine, but should be modular since computations involving real numbers should be avoided:

$$v_i = \frac{k_{i-1,i}}{f(k_{i,i+1})} \pmod p$$

Nevertheless, addition is more efficient than multiplication and would thus be more preferable. Lastly, the bitwise XOR operator could be applied for the best computational efficiency:

$$v_i = k_{i-1,i} \oplus f(k_{i,i+1})$$

Application of the bitwise XOR operator requires consequently that the number of bytes of the keys $k_{i-1,i}$ and the output of f are equal.
Computation of preceding keys and class keys would be in agreement with the respective recurrence relations

$$k_{j-1,j} = \begin{cases} v_j + f(k_{j,j+1}) \\ v_j \cdot f(k_{j,j+1}) \pmod p \\ v_j \oplus f(k_{j,j+1}) \end{cases}$$

where, in agreement with Step 4 in Section 4, $P_i \in S_\ell$ can compute $K_\gamma = k_{j,j+1}$ for $1 \leq \gamma \leq \ell$ if P_j is sequentially first in S_γ, i.e., $\gamma = L_{j-1} + 1$ where $L_0 = 0$.
Note that, for example, if the applied one-way function is a hash-function, the verification Step 4 in Section 4 would not work.

VII. Centralized key distribution

In this section, we present a centralized key distribution scheme that is based on the key establishment protocol in the previous section. By centralized we mean that one entity initiates the protocol by means of hierarchical public parameters of the pertaining group. This is in contrast to the previous protocol where all participants depend on the others in order to establish and deduce the secret hierarchical session keys.
As follows, the centralized key distribution scheme is presented respectively for both totally-ordered and partially-ordered security classes. Moreover, all participants of each security class share a long-term secret key that is associated to that class.

A. Security properties

In the previous protocol, the hierarchical session keys (or class keys) are established and deduced as a function of user inputs where each user contributes with session-dependant inputs. In the following centralized protocols, the class keys

are established as a function of the public parameters representing a team, the secret long-term hierarchical keys and a random number. Thus, the correct class key can only be established by means of the proper secret long-term hierarchical key.

An essential security property of our scheme is that although participants of a given security class may compute class keys of underlying security classes, it should be prevented that any given class may deduce secret *long-term hierarchical keys* of other classes, i.e., long-term hierarchical key confidentiality. A reason why this is essential, is that such keys could be used for other applications and purposes as well whereof participants of other security classes may not be involved with. Consequently, it must be ensured that long-term hierarchical keys remain undisclosed, even for participants of higher rankings.

Note that since any party could initiate key distribution without not necessarily possessing any pertaining long-term secret user keys, this party would be prevented from obtaining the corresponding hierarchical session keys. Since key transfer protocols allow one party to securely transfer a secret key to other parties, the following protocol qualifies as a key transfer protocol only if the initiating party possesses the pertaining long-term secret user keys.

The centralized protocols do not have explicit user authentication like the previous protocol. However, since user authentication is based on the assumption that only the legitimate users hold the pertaining secret long-term keys (whereof key correctness depends), user authentication is an implicit property of the protocol. Thus, the centralized protocols provide implicit user authentication, key secrecy and hierarchical one-way security property. The security properties comply with the previous protocol except that forward secrecy is not supported.

B. Totally-ordered centralized protocol

In this subsection, we present the totally-ordered centralized version of the protocol.

Initializations. Let $n = p \cdot q$ where *p* and *q* are two large distinct primes, and let α be an element of maximal order in $\mathbb{Z}_n^*$. The trusted third party (TTP) that sets up the scheme randomly generates λ secret long-term hierarchical user keys, $k_j \in \mathbb{Z}_{\phi(n)}$, $j \in \{1, \ldots, \lambda\}$, so that each user in the security class S_j is confidentially handed the pertaining k_j. The TTP computes for each $S_j \subseteq \mathcal{U}$ the public parameters

$$Y = \{y_j = \alpha^{k_j - 2k_{j+1}} \pmod n \mid 1 \leq j < \lambda\}$$

Key establishment. One particular party is required to initiate the protocol. We refer to this party as the registry. This could be any of the participants or an arbitrary outsider. The protocol goes through the following steps:

1. The registry selects a random number *r*, computes and broadcasts

$$z_j = y_j^r \pmod n \quad \text{and} \quad R = \alpha^r \pmod n$$

for each security class $S_j \subseteq \mathcal{U}$.

2. Each participant in S_ℓ computes the class key referring to his or her security class according to

$$K_\ell = R^{k_\ell} \pmod n$$

In general, each participant in S_ℓ computes the class keys K_i of his own and the underlying security classes S_i, $1 \leq i \leq \ell$ according to

$$\begin{aligned} K_i = R^{k_i} &= z_i \cdot z_{i+1}^2 \cdots z_{j-1}^{(2^{\ell-1})} \cdot R^{(2^{\ell} \cdot k_\ell)} \pmod n \\ &= R^{(2^{\ell} \cdot k_\ell)} \cdot \prod_{j=0}^{\ell-1} z_{i+j}^{(2^j)} \pmod n \end{aligned}$$

Example. Given a 4 level totally-ordered hierarchy, each participant in S_4 would compute $K_1 = z_1 \cdot z_2^2 \cdot z_3^4 \cdot R^{8 \cdot k_4} \pmod n$, etc.

C. Partially-ordered centralized protocol

The protocol for partially-ordered security classes is basically the same as the totally-ordered centralized protocol other from its more general hierarchical capabilities.

Definitions and notation. Let $\mathcal{U} = \{S_1, \ldots, S_\lambda\}$ be λ disjoint security classes. Let $H \subseteq \mathcal{U} \times \mathcal{U}$ be the binary relation where $(S_i, S_j) \in H$ *iff* S_i is an immediate predecessor of S_j and where the users in S_i have a higher security clearance than the users in S_j. A partially ordered set $(\mathcal{U}, H)$ can be represented by a Hasse diagram where an edge from S_i to S_j represents $(S_i, S_j) \in H$.

H^* denotes a reflexive transitive closure of *H*. Let $S_i \preceq S_j$ *iff* $(S_i, S_j) \in H^*$. This means that the users in S_i have a security clearance higher than or equal to the users in S_j. Fig. 1 shows an example of a partially ordered hierarchy defined by the relation $H = \{(S_6, S_5), (S_6, S_4), (S_4, S_3), (S_4, S_2), (S_3, S_1), (S_2, S_1)\}$. For example, $(S_6, S_4), (S_4, S_2) \in H$ implies that $(S_6, S_2) \in H^*$.

Initializations. The TTP generates long-term secret user keys as in the previous subsection, and computes the long-term public hierarchical parameters *Y* according partially-ordered hierarchical structure of the group $\mathcal{U}$.

For each edge $(S_i, S_j) \in H$, the TTP computes the public parameters

$$Y = \{y_{i,j} = \alpha^{k_i - 2k_j} \pmod n \mid (i,j) \in I_H\}$$

where $I_H = \big\{(i,j) \mid (S_i, S_j) \in H\big\}$. Thus, $|Y| = |H|$. The example in Fig. 1 corresponds to $Y = \{y_{6,5}, y_{6,4}, y_{4,3}, y_{4,2}, y_{3,1}, y_{2,1}\}$.

Key establishment.

1. The registry selects a random number *r*, computes and broadcasts

$$\begin{aligned} Z &= \{z_{i,j} = y_{i,j}^r \pmod n \mid y_{i,j} \in Y\} \quad \text{and} \\ R &= \alpha^r \pmod n \end{aligned}$$

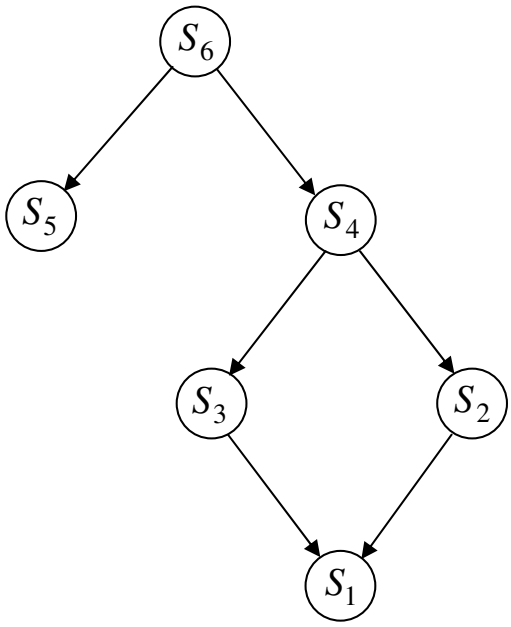

Figure. 1: Example of a partially ordered hierarchy

.

2. Each participant in S_ℓ computes the conference key referring to his or her security class according to

$$K_\ell = R^{k_\ell} \pmod{n}$$

If there exists an edge $(S_i, S_j) \in H$, then K_j can be computed if K_i is known:

$$K_j = z_{i,j} \cdot K_i^2 \pmod{n} \quad \text{where} \quad z_{i,j} \in Z$$

Thus, in general, if $S_{\ell,j} \in H^*$, then each participant in S_ℓ can recursively compute K_j.

VIII. Conclusion

In this paper, we have presented an efficient hierarchical multi-party key agreement protocol that enables an arbitrary number of users of λ security classes to establish a secret class key for each security class. It provides user authentication, and allows users in a given security class to obtain the secret class keys of the same and underlying security classes, while no user can obtain class keys of overlying security classes. It is broadcast-oriented and requires only two rounds of broadcasting, and is thus well-suitable for wireless networks.
We have moreover presented a centralized hierarchical key distribution scheme based on the former that supports totally-ordered and partially-ordered user hierarchies.

References

[1] M. Burmester, Y. Desmedt. A secure and efficient conference key distribution system. In proc. of Eurocrypt'94, LNCS, vol. 950, pp. 275 – 286, Springer-Verlag, 1994.

[2] M. Just, S. Vaudenay. Authenticated multi-party key agreement. In proc. of Asiacrypt'96, LNCS, vol. 1163, pp. 36 – 49, Springer-Verlag, 1996.

[3] S. Saeednia, R. Safavi-Naini. Efficient identity-based conference key distribution protocols. ACISP'98, LNCS 1438, pp. 320–221, Springer-Verlag, 1998.

[4] M. Hwang, W. Tzeng. A conference key distribution scheme in a totally-ordered hierarchy. ICOIN 2003, LNCS 2662, pp. 757 – 761, Springer-Verlag, 2003.

[5] S. Eskeland. Efficient Hierarchical Conference Key Establishment in Wireless Networks. IASTED International Conference on Communication, Network and Information Security '05, pp. 94 – 98, Acta Press, 2005.

[6] S. Eskeland, V. Oleshchuk. Hierarchical Multi-Party Key Agreement for Wireless Networks. Third International Symposium on Information Assurance and Security '07, pp. 39 – 43, IEEE Computer Society, 2007.

[7] K. Koyama, K. Otha. Identity-based conference key distribution system. Advances in Cryptology, Crypto'87, pp. 194 – 202, Springer-Verlag, 1987.

[8] C. Boyd, A. Mathuria. Protocols for Authentication and Key Establishment. ISBN 3-540-43107-1, Springer-Verlag, 2003.

[9] R. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Comm. of the ACM, Vol. 21, No. 2, pp. 120 – 126, 1978.

[10] K. Koyama, K. Ohta. Security if improved identity-based co0nference key distribution systems. Adv. in Cryptology - EuroCrypt '88, LNCS 330, pp. 11 – 19, Springer-Verlag, 1988.

[11] K. Koyoma. Secure conference key distribution schemes for conspiracy attack. Advances in Cryptology, Crypto '92, pp. 449–453, Springer-Verlag, 1992.

[12] W. Yang, S. Shieh. Password authentication schemes with smart cards. Computer & Security, vol. 18, no. 8, pp. 727 – 733, 1999.

[13] K. Chen, S. Zhong. Attacks on the (enhanced) Yang-Shieh authentication. Computer & Security, vol 22, no. 8, pp. 725 – 727, 2003.

[14] F. Kuo, V. Shen, T. Chen, F. Lai. Cryptographic key assignment scheme for dynamic access control in a user hierarchy. IEE Proc. Computers & Digital Techniques, Vol 146, No. 5, 1999, pp. 235 – 240.

[15] C. Lin. Dynamic key management schemes for access control in a hierarchy. Computer communications, Vol. 20, pp. 1381 – 85, 1997.

[16] C. Chang, C. Lin, W. Lee, P. Hwang. Secret sharing with access structures in a hierarchy. AINA, Vol. 2, pp. 31 – 34, 2004.

[17] X. Zou, B. Ramamurthy, S. Magliveras. Chinese Remainder Theorem based hierarchical access control for secure group communications. ICICS, LNCS, Vol. 2229, pp. 381 – 385, 2001.

[18] Y. Kim, A. Perrig, G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. Proc. of 7th ACM CCS, pp. 235 – 244, 2000.

[19] L. Dondeti, S. Mukherjee, A. Samal. DISEC: a distributed framework for scalable secure many-to-many communication. Proc. of 5th IEEE ISCC, pp. 693 – 698, 2000.

[20] A. Sherman, D. McGrew. Key establishment in large dynamic groups using one-way fucntion tree. IEEE transactions on Software Engineering, Vol. 29, No. 5, pp. 444 – 458, 2003.