A key agreement protocol based on Identity-Based Proxy Re-encryption

Adrian Atanasiu[1] **and Adela Mihaita**[1]
[1]Department of Computer Science, Faculty of Computer Science, University of Bucharest, Bucharest, Romania

Abstract— *In this paper, we present a problem and propose an elegant solution for it: a protocol that allows a manager to choose his team from a database of experts and then establish with them a common secret shared key. There are some restrictions on the protocol which led us to the use of proxy re-encryption: the list of experts chosen is not known outside the team and the secret key agreed on is known only by the team. The construction and security of the protocol is based on the concept of Identity-Based Proxy Re-encryption (IB-PRE). In the second part of this paper, we extend the IB-PRE scheme by combining it with an Identity-Based Time Specific Encryption (IB-TSE) scheme obtaining a time specific encryption scheme that allows not only encryption, but also re-encryption.*

Keywords: key agreement, proxy re-encryption, identity-based setting, time specific encryption, knapsack problem

1. Introduction

Let be a manager who wants to build a team of experts from a large database and establish together a shared communication secret key. We present here a possible solution which is secure, and which has as underlying concepts the knapsack vector problem and identity-based proxy re-encryption scheme.

In the last part of the paper, in order to limit the waiting times of acceptance, we combine IB-PRE and ID-TSE in an Identity-Based Time Specific Re-encryption Scheme.

The content of the paper is the following: section 2 introduces the problem which will be solved in this paper together with a provisory protocol; in section 3 we discuss Identity-Based Proxy Re-encryption which provides the encryption part of our protocol. Section 4 presents the construction of our protocol, while section 5 and 6 discuss some general considerations and security issues. Section 7 introduces the new scheme, IB-TSRE. The paper ends with conclusions.

2. Formatting Instructions

3. A simple key agreement protocol

We consider the following problem:

There is a database of experts. In order to evaluate a project, a manager (from the database) is established. He chooses his team from the members of the database. There are some initial conditions:

- No one outside knows exactly the list of experts chosen by the manager.
- The team members must agree on a communication key which only they know.

We first propose the following simple manner of adressing the problem:

Initial data

- the database $\mathcal{B} = \{P_1, ..., P_n\}$ is associated with a knapsack vector $A = (a_1, ..., a_n)$ and a large prime number $p > max_{1 \le i \le n}\{a_i\}$, both of them public.
- the knapsack problem is NP-complete for anyone outside the database. The members of the database can solve it in linear time.
- Each P_i has a public key e_i for encryption, a secret key for decryption d_i and a signing algorithm (sig_i, ver_i).

A simple key agreement protocol

Let's suppose that the manager $M \in \mathcal{B}$ wants to select his team $T_M = \{P_{i_1}, ..., P_{i_k}\}$ which will share the same secret key. We denote by $I = \{i_1, ...i_k\} \subseteq [1, n]$. The procedure is:

Algorithm A

1) M makes public
$$S = \sum_{i \in I} a_i \bmod p. \tag{1}$$
2) Each $P_i \in \mathcal{B}$ solves the knapsack problem (A, S) and checks if $i \in I$; if so, then he generates a random number α_i.
3) Each P_i sends to each P_j $(j \in I, j \neq i)$ the message
$$\{a_i, \alpha_i, sig_i(\alpha_i)\}_{e_j} \tag{2}$$
(we denoted by $\{w\}_e$ the encryption of w under key e)
4) Each $P_i \in T_M$:
    a) Decrypts the $k-1$ received messages
    b) Checks if
$$\sum_{j \in I} a_j = S \bmod p \tag{3}$$
    c) Checks if
$$ver_j(\alpha_j, sig_j(\alpha_j)) = True, \forall j \in I - \{i\} \tag{4}$$

d) If both conditions are satisfied, then he computes the secret shared key

$$K = \sum_{j \in I} \alpha_j (\bmod p) \quad (5)$$

This protocol looks simple, but it has certain disadvantages. Step 3 is followed by all members of $\mathcal{B}$ and it requires too many message exchanges between members of the team: $k(k-1)$. In order to reduce the number of sent messages, one can use a central authority, but here arises another problem since the CA doesn't have to know the team E_M nor the common key. Moreover, we would like to add some further conditions which are not met by the previous proposal:

- Only the members of the team should know that they were chosen; all the other experts should ignore this.
- One expert should have the possibility to reject the proposal of being part of an evaluation team; in this case, the manager must remove him from T_M and, eventually, replace him with another expert.

4. Preliminaries

For the construction of our protocol, we will make use of the concept of identity-based proxy re-encryption. Next we recall this notion.

Proxy re-encryption(PRE) allows a semi-trusted proxy to convert a ciphertext originally intented for Alice into one encrypting the same plaintext for Bob. The proxy needs for the conversion a re-encryption key issued by Alice and can not learn anything about the plaintext. An identity-based proxy re-encryption(IB-PRE) scheme [2] allows a proxy to translate an encryption under Alice's identity into one computed under Bob's identity. We will focus our attention on IB-PRE since we work in the identity-based setting.

An Identity-Based Proxy Re-Encryption scheme is an extension of Identity-Based Encryption scheme. Let's see a formal definition of IB-PRE scheme. An identity-based proxy re-encryption scheme [2] consists of the following algorithms:

- **Setup** takes as input the security parameter k and a value indicating the maximum number of consecutive re-encryptions permitted by the scheme and outputs the master public parameters which are distributed to the users and the master secret key (msk) which is kept private.
- **KeyGen** takes as input an identity id and the master secret key and outputs a private decryption key sk_{id} corresponding to that identity.
- **Enc** on input a set of public parameters, an identity and a plaintext, outputs the encryption of m under that identity.
- **RKGen** on input a secret key sk_{id_1} and identities id_1, id_2, outputs a *re-encryption key* $rk_{id_1 \leftarrow id_2}$.
- **Reenc** on input a ciphertext c_{id_1} under identity id_1, and a re-encryption key $rk_{id_1 \rightarrow id_2}$, outputs a re-encrypted ciphertext c_{id_2}.
- **Dec** decrypts the ciphertext c_{id} using the secret key sk_{id} and outputs message m of failure simbol $\perp$.

5. Our construction of the key agreement protocol

The protocol suggested in the first section has many vulnerabilities, but one of them is that the manager sends to every expert a solvable instance of the knapsak problem and therefore, any expert from the database knows the team composition. But this should be revealed only to the selected members of the team. So it is important that, in the first phase, the knapsack problem remains NP-complete for all the experts. Those who were selected to be part of the team will be sent an encrypted trapdoor for solving the knapsack problem. The protocol that we propose allows an expert to refuse the request of joining the team and so he can be replaced by another expert who accepts the request, if k, the number of members of the team is fixed.

For encryption of the trapdoor, we use the encryption algorithm from the identity-based proxy re-encryption scheme. The idea of this scheme allows re-encryption of a trapdoor under a key available to the recipients.

Now that we have seen the scheme we want to use for encryption, we can go on with the main issue of this paper, the key agreement protocol.

<u>Algorithm B</u>

1) M chooses a team $T_m = \{P_{i_1}, ..., P_{i_k}\}$ where $I = \{i_1, ..., i_k\}$ and a knapsack vector $A = (a_1, ..., a_n)$ which he makes public.
2) M computes and makes public the sum

$$S = \sum_{i \in I} a_i \bmod p. \quad (6)$$

This instance of the knapsack problem is NP-complete for all the experts.
3) M sends to each $P_j (j \in I)$ a nonce encrypted under his identity together with the trapdoor encrypted under M's identity (of course, in this case, none of the P_j is able to decrypt, since this is what we want for the moment).
4) Each P_j is able to decrypt the nonce and sends it back to M only if he accepts the proposal to be part of the team; note that P_j is not able yet to decrypt the trapdoor.
   a) If each P_j gives a positive answer, then M computes (using Reenc algorithm from the IB-PRE scheme) and makes public a vector of re-encryption keys

$$Rk = (rk_{id_{i_1}}, ..., rk_{id_{i_k}}), \quad (7)$$

where each rk_{id_j} corresponds to P_j, $\forall j \in I$. Then each P_j uses the key published for himself which allows him to re-encrypt the trapdoor received at step 3 under his own identity; once he obtains the trapdoor encrypted under his identity, he will use the private key associated to his identity and decrypt.

b) There might be experts who don't accept to join the team (they don't send back the answer); then M chooses other experts instead, and will repeat steps 3 and 4. If the new chosen experts accept, then M publishes re-encryption keys for each member, as in the step above, which allow them to obtain the encryption of the trapdoor under their own identity. After this, simply using their private keys, they will be able to decrypt the trapdoors. We must emphasize that before publishing the re-encryption keys, M will recompute

$$S = \sum_{i \in I} a_i \bmod p \tag{8}$$

as the sum of elements of the knapsack vector according to the new created team.

5) Each P_j is now in possesion of the trapdoor, so he is able to solve the knapsack vector problem and find out who are the other members of the team; then he generates a random number α_j and sends the message

$$\{a_j, \alpha_j, sig_j(\alpha_j)\}_{e_i} \tag{9}$$

to every P_i, where $i \in I, i \neq j$.

6) Every member P_j of the team follows the steps:
   a) Decrypts the $k-1$ received meesages;
   b) Checks if

$$\sum_{j \in I} a_j = S \bmod p; \tag{10}$$

   c) Checks if

$$ver_i(\alpha_i, sig_i(\alpha_i)) = True, \forall i \in I - \{j\}; \tag{11}$$

   d) If both conditions are satisfied, then he computes the secret shared key

$$K = \sum_{j \in I} \alpha_j (\bmod p) \tag{12}$$

7) The last step verifies if all the members of the team share the same key:
   a) Each $P_j, j \in I$, generates a random β_j and sends to M the message

$$\{\{\beta_j\}_{e_j}, a_j, sig_j(a_j)\}_K; \tag{13}$$

   b) M sends back to P_i the message

$$\{\beta_j, a_j - 1, sig_M(a_j - 1)\}_K. \tag{14}$$

6. Considerations about the protocol

We notice that even if we use the scheme of proxy re-encryption, there is no proxy in our protocol; the experts from the team play the role of the proxy and apply re-encryption algorithm. On the other hand, in order to reduce the number of messages sent during the protocol, M sends the nonce together with trapdoor, both encrypted, in a single step 3.

We mention that step 4.a) might be repeated more than once since the new chosen experts might as well refuse joining the team. In order to limit the waiting times of acceptance from step 4.b), we introduce in section 7 an Identity-Based Time Specific Re-Encryption scheme where precisely time is essential. Anyway, refusing experts are not a problem for the security of the protocol.

We remark that we assume from the begining that the members of the team are honest. It is very unlikely that an expert will want to fail the protocol, but still he can easily do this, for example, by sending different random numbers α_j to different members or by signing a different α_j in step 5. We have also omitted the situation when two or several managers want to create their own team from the same database of experts, in the same time. We leave this for future work together with the situation where each of the two managers M_1 and M_2 of two simultaneous teams, from the same database, is a member in the team of the other one.

We also recall that the knapsack problem used in our protocol is NP-complete. Many of the kanpsack cryptosystems have been proven to be weak against low-density attacks. In this paper, we propose for the construction of our protocol a knapsack cryptosyetm based on elliptic curve discrete logarithm [3]. We note that the cryptosystem from [3] has been broken by [1] who also proposes a simple solution in order to avoid their attack: in [3] instead of defining

$$C_{m_i} = \{k\alpha, P_{m_j} + ks_i\}, \tag{15}$$

one should define

$$C_{m_i} = \{ka_{\pi(i)}\alpha, P_{m_j} + ks_i\}. \tag{16}$$

This cryptosystem enjoys high-density and, therefore, avoids low-density attacks. The trapdoor for the knapsack vector is, as in the case of Merkle-Hellman cryptosystem, a super-increasing vector (which represents the private key) which allows linear solving of the problem. We refer the reader to [3] for more details on the construction of the knapsack cryptosystem suggested.

7. Security of the protocol

First of all, we notice that every expert chosen by M will receive the trapdoor encrypted, before he gives an answer. But this is not a problem, even if an expert doesn't accept participation, since the trapdoor is encrypted under M's identity, so nobody else, except him, is able to decrypt.

So, if an expert refuses joining the team, he can keep the trapdoor encrypted, but he won't be able to decrypt it, even if later M publishes re-encryption keys for the members of the team. Working in the identity-based setting enables decryption only for the intended recipients.

As we indicated at the begining, we use an identity-based proxy re-encryption scheme from [2], section 4, where Green and Ateniese present two non-interactive identity-based proxy re-encryption schemes which are secure under the Decisional Bilinear Diffie-Hellman Assumption (DBDH) in the random oracle model. The first, IBP1, is secure under chosen plaintext attack (in fact, it is IND-Pr-ID-CPA secure), while the second one, IBP2, presents stronger security under adaptive chosen ciphertext attack (IND-Pr-ID-CCA secure). Any of the two constructions presented in [2] based on biliniar pairings might be used for our protocol.

8. A Time-Specific Encryption scheme

In this section, we introduce an identity-based time-specific re-encryption scheme, starting from an IB-PRE scheme combined with Time Specific Encryption, a concept that we detail in the next subsection. This scheme can be used in order to limit waiting time at step 4, but we believe it might be useful also in some other applications where encryption and decryption are done in a timely manner. Briefly, the scheme allows re-encryption.

8.1 Identity-Based Time Specific Encryption

The cryptographic primitive Time Specific Encryption (TSE) was introduced in 2010 [4] and it's closely related to the concepts of Timed-Release Encryption (TRE) and Broadcast Encryption.

The idea behind TSE is allowing a user to specify during what time interval a ciphertext can be decrypted by the receiver. This is done in the following manner in TSE: a Time Server broadcasts a key, a Time Instant Key (TIK) k_t at the begining of each time unit, t. The TIK is available to all users. A sender, who wants to encrypt a message m to form a ciphertext c, can specify any interval $[t_0, t_1]$, with $t_0 \leq t_1$. In *Plain* TSE, a receiver can recover the message m only if he holds a TIK k_t for some $t \in [t_0, t_1]$.

Plain TSE was extended to public-key and identity-based settings. We remain in the identity-based setting (ID-TSE), where decryption requires also a private key coresponding to the identity of the receiver besides the appropriate TIK.

Formally, an ID-TSE scheme consists of the following algorithms[4]:

- **TS-Setup**. This algorithm is run by the Time Server, takes as input the security parameter k, T, the number of allowed time units and outputs the master public key TS-MPK and the master secret key TS-MSK.
- **ID-Setup**. This algorithm is run by the Trusted Authority (TA), takes as input the security parameter k and outputs the master public key ID-MPK and the master secret key ID-MSK.
- **TIK-Ext** This algorithm is run by the TS, takes as input TS-MPK, TS-MSK, t and outputs k_t which is broadcast by TS at time t.
- **ID.Key-Ext** This algorithm is run by the TA, takes as input ID-MPK, ID-MSK, an id and outputs the private key sk_{id} corresponding to id.
- **ID.Enc** This algorithm is run by the sender, takes as input TS-MPK, ID-MPK, a message m, a time interval $[t_0, t_1]$ and an identity id and outputs a ciphertext c.
- **ID.Dec** This algorithm is run by the receiver, takes as input TS-MPK, ID-MPK, a ciphertext c, a key k_t and a private key sk_{id} and outputs either a message m or a failure symbol $\perp$.

Paterson and Quaglia [4] use a binary tree for the construction of the TSE schemes. The leaves of the binary tree represent time instants. They also define two particular set of nodes. The idea is to view the nodes of the tree as identities and make use of identity-based encryption techniques to instantiate plain TSE. The number T of allowed time units will be of the form $T = 2^d$. The tree associated in [4] to the scheme has some properties:

1) The root of the tree has label $\emptyset$; the other nodes are labelled with binary strings of lengths between 1 and d. Therefore, each node has associated a binary string $t_0 t_1 ... t_{l-1}$, of length $l \leq d$. The leaves are labelled from 0...0 to 1...1 and each leaf will represent a time instant.

$$t = \sum_{i=0}^{d-1} t_i 2^{d-1-i}. \tag{17}$$

2) There are two particular set of nodes defined with respect to the tree:
   - $\mathcal{P}_t$ - the path to t. For a time instant

   $$t = \sum_{i=0}^{d-1} t_1 2^{d-1-i}, \tag{18}$$

   the following path $\mathcal{P}_t$ corresponding to t can be constructed in the tree:

   $$\emptyset, t_0, t_0 t_1, ... t_0 ... t_{d-1} \tag{19}$$

   - the set $\mathcal{S}_{[t_0,t_1]}$ which covers the interval $[t_0, t_1]$ - the minimal set of roots of subtrees that cover leaves representing time instants in $[t_0, t_1]$. The labels of the nodes in this set are computed in a particular order by running Algorithm 1 from [4].

The two sets $\mathcal{P}_t$ and $\mathcal{S}_{[t_0,t_1]}$ intersect in an unique node only if $t \in [t_0, t_1]$.

8.2 Identity-based Time Specific Re-encryption Scheme

We present here an identity-based time specific encryption scheme combined with identity based proxy re-encryption scheme; in fact, the aim of this time specific encryption scheme is to allow not only encryption, but also re-encryption. We start from an IB-PRE scheme $I = (Setup, KeyExt, Enc, RKeyExt, ReEnc, Dec)$ with message space $\{0, 1\}^l$ in order to derive an ID-TSE $X = (Plain.Setup, Plain.TIK-Ext, Plain.Enc, Plain.Dec)$ with the same message space. We call our scheme an Identity-Based Time Specific Re-Encryption scheme:

- **Setup(k,T).** Run Setup on input k to obtain a master public key TS-MPK and the secret key TS-MSK. We define $T = 2^d$ where d is the depth of the binary tree used in TSE, and T is the number of allowed time units.
- **ID-Setup(k,T).** Run by the TA (trusted authority), this algorithm generates the public key ID-MPK and the secret key ID-MSK.
- **TIK-Ext(TS-MPK,TS-MSK,t).** Construct the path $\mathcal{P}_t$ to obtain the list of nodes $\{0, p_1, ..., p_d\}$ on the path to t. Run Key-Ext algorithm for all nodes p din $\mathcal{P}_t$ to obtain a set of private keys
$$\mathcal{D}_t = \{d_p : p \in \mathcal{P}_t\}. \qquad (20)$$
Return $\mathcal{D}_t$ which represents the key k_t broadcasted at moment t.
- **RKGen** $(\mathcal{D}_t, [t'_0, t'_1].)$ This algorithm returns a set of re-encryption keys for messages that were initially encrypted under interval $[t_0, t_1]$ to encrypt them under another interval $[t'_0, t'_1]$. $\mathcal{D}_t$ represents the set of private keys associated to the identities from the set $\mathcal{P}_t$, where $t \in [t_0, t_1]$. For every $d_p \in \mathcal{D}_t$, run $RKeyExt(TS - MPK, d_p, [t'_0, t'_1])$, and obtain the set
$$Rk_{[t_0,t_1]\to[t'_0,t'_1]} = \{rk_{d_p} : d_p \in \mathcal{D}_t\}. \qquad (21)$$
- **Encryption** $(TS - MPK, m, [t_0, t_1])$. Run Algorithm 1 from [4] on input $[t_0, t_1]$ to compute a list of nodes $\mathcal{S}_{[t_0,t_1]}$. For each $s \in \mathcal{S}_{[t_0,t_1]}$ run Enc$(TS - MPK, m, s)$ to obtain a list of ciphertexts
$$\mathcal{CT}_{[t_0,t_1]} = \{c_p : p \in \mathcal{S}_{[t_0,t_1]}\}. \qquad (22)$$
Then, each ciphertext obtained is encrypted under the identity of the recipient.
- **Re-Enc** $(TS - MPK, \mathcal{CT}_{[t_0,t_1]}, Rk_{[t_0,t_1]\to[t'_0,t'_1]})$. For each $c_p \in \mathcal{CT}_{[t_0,t_1]}$ and each corresponding $rk_{d_p} \in Rk_{[t_0,t_1]\to[t'_0,t'_1]}$, run $ReEnc(params, rk_{d_p}, c_p)$, and obtain a set of ciphertexts encrypted under interval $[t'_0, t'_1]$.
- **Decryption** $(TS - MPK, C, \mathcal{D}_t)$. Here $C = (\mathcal{CT}, [t_0, t_1])$ represents a list of ciphertexts together with a time interval. If $t \notin [t_0, t_1]$, then decryption can not be applied. Otherwise run Algorithm 1 from [4] to generate an ordered list of nodes $\mathcal{S}_{[t_0,t_1]}$ and generate the set $\mathcal{P}_t$. The intersection of these sets is the unique node p. Obtain the key d_p corresponding to p from $\mathcal{D}_t$. Run $Dec(TS - MPK, c_p, d_p)$, where $c_p \in \mathcal{CT}$ is in the same position in the list $\mathcal{CT}$ as p is in $\mathcal{S}_{[t_0,t_1]}$ and obtain either the message m or a failure symbol.

8.3 Security of the scheme

Paterson and Quaglia [4] concentrate on achieving IND-CPA security for ID-TSE and even IND-CCA security for the *Plain TSE*, but they didn't manage to achieve IND-CCA security for ID-TSE. We already discussed in section 6 the security of the IB-PRE scheme.

9. Conclusions

We suggested in this paper a problem for which we built a protocol based on the idea of proxy re-enecryption. We first proposed a simple solution improved later by using re-encryption. The protocol's security relies on the identity-based setting in which we work and on the security of the IB-PRE scheme used.

We also built an IB-TSRE scheme in order to limit waiting time at step 4 in algorithm B, which allows time specific encryption and proxy re-encryption in the same time. We think that this scheme is valuable in certain situations since it extends the notion of time specific encryption.

References

[1] Jingguo Bi, Xianmeng Meng, and Lidong Han. Cryptanalysis of two knapsack public-key cryptosystems. [online]. Available: http://eprint.iacr.org/2009/537.pdf

[2] M. Green, G. Ateniese. "Identity-Based Proxy Re-encryption" in *Applied Cryptography and Network Security (ACNS '07)*, 2007. [online]. Available: http://eprint.iacr.org/2006/473.pdf

[3] Min-Shiang Hwang, Cheng-Chi Lee, and Shiang-Feng Tzeng. "A knapsack public-key cryptosystem based on elliptic curve discrete logarithm", *Applied Mathematics and Computation*, vol. 168, Issue 1, September 2005, pp 40-46

[4] K.G. Paterson and E.A. Quaglia. *Time Specific Encryption*, In J. Garay and R. De Prisco (eds.), SCN 2010, Lecture Notes in Computer Science Vol. 6280, pp. 1-16, Springer, 2010.