ポリシーを秘匿した自動トラスト交渉

菊池浩明　　　Tangtisanon Pikulkaew

東海大学大学院工学研究科，259-1292 神奈川県平塚市北金目 1117，kikn@tokai.ac.jp

あらまし　自動トラスト交渉とは，証明書から成るポリシーを持った 2 者間で互いのポリシーの漏洩を最小限にして，安全に同意を取る問題である．本稿は，両者が要求した証明書の譲渡されるかどうかの判定結果だけを知るプロトコルを提案する．プロトコル実行後でもいかなるポリシーも漏れず，どのような証明書を持っているかも互いに知られない．

Privacy-Preserving Automated Trust Negotiation

Hiroaki Kikuchi　　　Tangtisanon Pikulkaew

Graduate School of Engineering, Tokai University,
1117, Kitakaname, Hiratsuka, Kanagawa, Japan, 259-1292,

Abstract The Automated Trust Negotiation aims to securely identify the consensus between two sets of policies consisting of certificates, with minimal disclosure of policies to each other. The paper proposes a new scheme that allows both parties to learn whether or not, both parties agree to transfer a given target certificate to the requesting party. No policy is revealed after performance of the protocol. No certificate is known to each other.

1 Introduction

Automated trust negotiation (ATN) aims to allow two parties to exchange digital credentials in X.509 format that contain sensitive information such as name, address, birthday and memberships, as well as access control decisions (what credentials are acceptable). Both parties wish to minimize information to disclose to other party in order to learn the minimal agreement of both private policies.

A number of cryptographic protocols have been proposed so far to address secure and private ATN. Winsborough et al. proposes the first scheme for ATN, classified into two extreme strategies, called, *parsimonious* and *eager* strategies in [1]. In both schemes, two parties need to reveal their partial policies gradually and hence no privacy is preserved. Li, Du and Boneh proposes an oblivious signature based envelop in which a user send her credentials to a sever who jointly compute with the user such that she sees the requested resource if and only if both policies are consistent in [2]. Nakatsuka and Ishida presents a scheme to minimize the sum of costs for disclosure of credentials in [3].

In this paper, we present a new scheme combining two cryptographical protocol for secure set operations, e.g., union and intersection, [4] and [5]. Our scheme allows both parties to learn whether or not, both parties agree to transfer a given target certificate to the requesting party. No policy is revealed after performance of the protocol. No certificate is known to each other.

2 Trust Negotiation

2.1 Definition

A policy is a set of logic formula consisting of certificates. Figure 1 shows an example of policies owned by a client and a server, where

client	
$q_3:$	$c_1 \leftarrow s_1$
$q_4:$	$c_2 \leftarrow s_2 \wedge s_4$
$q_9:$	$c_4 \leftarrow s_4$
$q_{10}:$	c_5

server	
$p_1:$	$R \leftarrow c_1 \vee c_2$
$p_3:$	$s_1 \leftarrow c_3 \vee c_4$
$p_4:$	$s_2 \leftarrow c_4 \vee c_5$
$p_5:$	s_3
$p_6:$	s_4

Figure 1: Example of Trust Policies

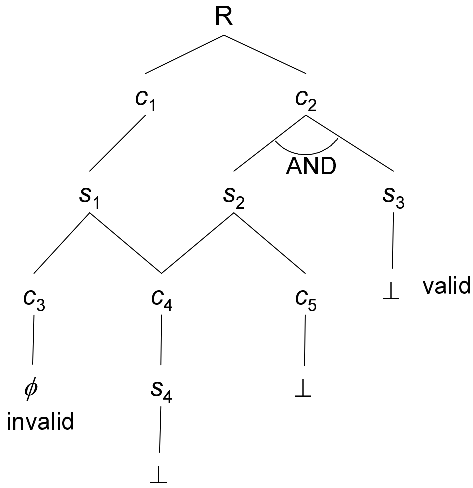

Figure 2: Example of trust target graph

R is a target service.

The logical relationship between client and server can be represented in a single trust target graph, shown in Fig. 2

2.2 Two Extreme Strategies

In [1] Winsborough et. al proposed two extreme strategies for negotiation, an *eager strategy* in which both party disclose each policy immediately after the condition of policy is satisfied, and a parsimonious strategy in which policies are gradually disclosed only after sufficient policy is ensured.

A *policy disclosure rate* is a ratio of disclosed policies over the whole policies, denoted by η. A *round of negotiation* is a number of transmissions of message between two parties, denoted by ρ. For instance, the eager strategy gives the consensus in the sequence shown in Fig. 3, yielding R, c_1, s_1, c_4, s_4. The disclosure rate is $\eta_{eager} = 6/12 = 0.5$ and the negotiation ends in $\rho_{eager} = 7$ rounds. While, the parsimonious strategy discloses all possible (au-

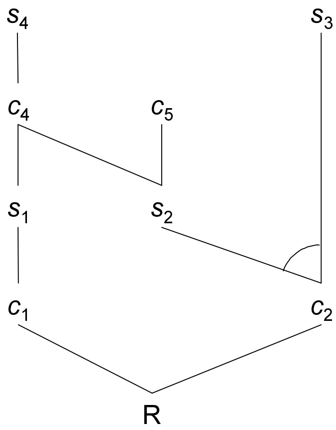

Figure 3: Eager Strategy

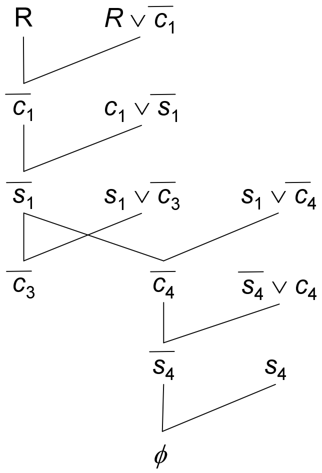

Figure 4: Parsimonious Strategy

thorized to access) policies in Fig. 4, which $\eta_{parsimonious} = 12/12 = 1.0$ in $\rho_{parsimonious} =$ 4 steps. Both parties have three paths to the given target, (s_4, c_4, s_1, c_1, R), (c_5, s_2, c_2, R) and (s_3, c_2, c_5, s_2, R).

3 Preliminary

3.1 Additive Homomorphic Public-key Encryption

To preserve the privacy of users, we use a public-key cryptosystem E which satisfies an additive homomorphic property, i.e., taking message M_1, M_2,

$$
\begin{aligned}
E[M_1]E[M_2] &= E[M_1 + M_2], \qquad (1)\\
E[M_1]^{M_2} &= E[M_1 M_2].
\end{aligned}
$$

For instance, the Paillier cryptosystem[7] and the modified ElGamal cryptosystem are widely used. Both allow us to get key generation and decryption processes distributed among semi-trusted authorities sharing private key.

The Paillier is more efficient than the ElGamal in the sense of decryption overhead, while the latter requires a sort of brute force technique (in the limited domain) for decrypting candidates of messages. We implement the Paillier cryptosystem for performance evaluation since the single computational cost for encryption is more significant for our proposed protocol.

3.2 Private Matching[4]

Freedman et. al presents a cryptographical protocol for secure set intersection in [4].

Let C and S be sets of secret $X = \{x_1, x_2, \ldots, x_{k_c}\}$ for client C and $Y = \{y_1, y_2, \ldots, y_{k_s}\}$ for server S. User C uses a polynomial having elements of X as its root defined as

$$\begin{aligned} P(x) &= (x - x_1)(x - x_2) \cdots (x - x_{k_c}) \\ &= \ell_{k-1} x^{k-1} + \cdots + \ell_0 \end{aligned}$$

to encode X and then send to S a sequence of ciphertexts $E(\ell_0), \ldots, E(\ell_{k_c})$ for all coefficients $\ell 0, \ldots, \ell_{k_c}$ of P.

For y, server S computers

$$\begin{aligned} E(rP(y) + y) &= E(P(y))^r E(y) \\ &= \prod_{i=0}^{k_c} (\ell_i) y^i \end{aligned}$$

and sends ks ciphertexts to C in random order, where r is uniform random number.

Finally, client C decrypts the ciphertexts to obtain the elements of the intersection $X \cap Y$ without learning any other element.

3.3 Secure Set Operations

In [5], Kissner and Song extends Freedman's protocol so that multiple parties can perform union of each set in addition to intersection.

4 Proposed Scheme

4.1 Hidden Policy

Neither of the parsimonious or the eager strategies preserves the privacy of policies. We aim to minimize the policies disclosed to other even after their negotiation completed.

We wish to make party to send policy only if the corresponding logical condition is satisfied. To do so, we combine the secure protocol for set intersection [4] and the set operation protocol [5]. For example, the first policy in Fig. 1,

$$p_1 : \; R \rightarrow c_1 \vee c_2$$

is represented by a 2-order polynomial $P_1(x)$ contains the conditional certificates c_1 and c_2 as its root, e.g.,

$$P_1(x) = (x - c_1)(x - c_2).$$

In the same way, we represent a conjunction of certificates in the form of multi-variable polynomial. For instance, client 's second policy

$$p_4 : \; c_2 \rightarrow s_2 \wedge s_3$$

can be formed in

$$P_4(y_1, y_2) = (y_1 - s_2) + (y_2 - s_3),$$

which becomes 0 only when $y_1 = s2$ and $y_2 = s3$.

For preserving privacy of policy, we have these polynomials encrypted with a public key of the other party. For example, the ciphertext of polynomial $P_1(x) = x^2 - (c_1 + c_2)x + c_1 c_2 = x^2 + ax + b$ is a tuple $(E(a), E(b))$, which we denote by $E(P_1)$ for simplification. Note that the additive homomorphic property allows any party to evaluate the polynomial at an arbitrary point without revealing the plaintext.

4.2 Conditional Transfer

A party wishes to send all candidates of certificate only if the condition is met but without revealing which certificate is sent. The other party in turns send a new candidate policy whose condition has been satisfied with the previously sent policy. These interactions are processed with preserving privacy until a requested party verifies if the condition of the target is satisfied. What both party learn eventually from communication is just a boolean value.

To make it possible for the conditional transfer, we introduce a new trick based on the Fredman's protocol. Suppose that client having a policy $c_1 \to s_1$ receives a encypted polinomial $E_S(P(x)) = E_S((x-c_1)(x-c_2)) = (E_0, E_1, E_2)$. He obliviously evaluates $P(c_1)$ as $E_0 E_1^{c_1} E_2^{c-1^2} = E(P(c_1))$ and choosing a random number r sends back to server the condition of c_1 as

$$E_S(P(c_1))^r (E_C(Q(y)) = E_S(rP(c_1) + E_C(Q(y)) \qquad (2)$$

where $Q(y) = (y - s_1)$ is a polynomial hiding his condition s_1 and E_C is an encryption with the client's public key. Whether or not the domain of E_C is greater than that of E_S, the ciphertext of polynomial $E_C(Q(y))$ is a multiple of its modulus. Hence, we embed a temporary symmetric key k instead of $E_C(Q(y))$ itself into the ciphertext, and send the corresponding appropriate symmetric ciphertext in conjunction to the asymmetric ciphertext as

$$E_S(rP(c_1) + k); \mathcal{E}_k(E_C(Q(y)),$$

but we often write the two ciphertexts in the notation in Eq. (2) implicitly using hybrid encryption for simplification reason.

The client attempts to send each of his policies one by one in this manner since he does not know which policy is satisfied. In the example in Fig. 1, the client sends four ciphertexts,

$$\begin{aligned} B_1 &= E_S(r_1 P_1(c_1) + E_C(Q_3(y)), \\ B_2 &= E_S(r_2 P_1(c_2) + E_C(Q_4(y_1, y_2), \\ B_3 &= E_S(r_3 P_1(c_4) + E_C(Q_9(y), \\ B_4 &= E_S(r_4 P_1(c_5)), \end{aligned}$$

where with only B_1 and B_2 the server succeeds to decrypt and extract the encoded polynomial Q_3 and Q_4.

4.3 Proposed Scheme

A client and a server have set of policies $P = \{p_1, \ldots, p_{n_C}\}$ and $Q = \{q_1, \ldots, q_{n_S}\}$, respectively. Let E_C, D_S and E_S, D_S be public-key encryption and decryption algorithms for client and server, respectively.

1. A server sends to a client an encrypted polynomial for a target condition, $A_1 = E_S(P_1(x))$.

2. The client evaluates the encrypted polynomial with encrypted for each certificate c_i of a policy $c_i \leftarrow f_i(s_1, \ldots, s_{n_s})$ in his policy set Q as

   $$B_i = E_S(rP_i(c_i) + E_C(Q_i(y))$$

   for $i = 1, \ldots, n_C$, and sends to the server $B_1, \ldots, B_{n_C}$ in random order.

3. The server decrypts $B_1, \ldots, B_{n_C}$ with his private key, wishing have $D_S(B_i) = 0$, which implies that the condition for the target has been satisfied in their negotiation, and then terminates processing of the protocol.

4. Otherwise, the server retrieves an encrypted polynomial from successfully[1] decrypted messages, say $E_C(Q_{i_1}), \ldots, E_C(Q_{i_k})$, where k is the number of successfully decrypted message. For valid polynomial Q_i ($i = 1, \ldots, n_C$), the server securely evaluates polynomials for each of his policies, $\{s_1 \leftarrow g_1(c_1, \ldots, c_{n_C}), \ldots, s_{n_S} \leftarrow g_{n_S}(c_1, \ldots, c_{n_C})$ as

   $$A_j = E_C(r_j Q_i(s_j) + E_S(P_j(x)),$$

   where r_j is uniformly chosen random number and P_j is the corresponding polynomial defined from the j-th policy. If the polynomial has multiple, say m, variables, she needs attempting evaluation for all size-m, $\binom{n_s}{m}$ combinations of her certificates. Finally, the server sends to the client $A_1, \ldots, A_{n_S}, \ldots, A_{n_s k}$.

5. Go to Step 2 until either of them successfully decrypts *null* ciphertext, which

[1] We assume that the integrity of message can be tested by predetermined format of valid message so that we easily see if an attempt of decryption is successful or not.

is $D(A) = 0$, implying "Satisfied Negotiation". If the number of iteration is more than the number of policies (n_S or n_C), then terminates declaring "Negotiation Failure"

4.4 Example

Table 2 illustrates the sequence of messages sent from client and server having policies in Fig. 1. In 5 rounds, the protocol is terminated successfully with decryption being zero and hence the server learn that their policies have an agreement to provide the requested service.

4.5 Evaluation

We show a performance comparison of negotiation strategies in terms of degree of privacy to be preserved (disclosure rate), and the communication overhead in Table 1.

5 Conclusions

We have proposed a new cryptographical protocol for trust negotiation with full privacy preserved. Our protocol allows parties with private policies to learn if their policies can be aggraded without revealing any piece of private information.

References

[1] W. H. Winsborough, K. E. Seamons, and V. E. Jones. "Automated trust negotiation", In DARPA Information Survivability Conference and Exposition, volume I, pp. 88 - 102, *IEEE Press*, 2000.

[2] N. Li, W. Du and D. Boneh, "Oblivious signature-based envelope", In proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC), *ACM Press*, 2003.

[3] 中塚康介, 石田亨, "分散 AND/OR 探索による Automated Trust Negotiation", 情報処理学会誌, Vol. 47, No. 8, pp. 2454-2463, 2006.

[4] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection", Eurocrypt 2004, pp. 1-19, *Springer LNCS*, 2004.

[5] L. Kissner and D. Song, "Privacy-Preserving Set Operations", Crypto'05, pp. 200-212, *Springer LNCS*, 2005.

[6] 木澤, 磯崎, 菊池, 秘匿積集合プロトコルを利用したプライバシ協調フィルタリングの提案, SCIS2009,2009.

[7] P. Paillier: "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes", Proc. *EUROCRYPT'99*, LNCS 1592, pp. 223-238, 1999.

Table 1: Comparison of strategies

strategy	parsimonious (top down)	eager (bottom up)	proposed (top down)
trust path	R, c_1, s_1, c_4, s_4 (s_4, c_4, s_1, c_1, R), (c_5, s_2, c_2, R), (s_3, c_2, c_5, s_2, R)	R, c_2, s_2, s_3, c_5	
disclosure rate η	6/12	12/12	0
rounds ρ	7	4	5
communication	light	large	$\rho(2n_C n_s)$

Table 2: Sample Negotiation Processing

	client	server
0	$c_1:\ Q_1(y) = (y - s_1)$ $c_2:\ Q_2(y_1, y_2)) = (y_1 - s_2) + (y_2 - s_3)$ $c_4:\ Q_4(y) = (y - s_4)$ c_5	$R:\ P_0(x) = (x - c_1)(x - c_2)$ $s_1:\ P_1(x) = (x - c_4)(x - c_5)$ $s_2:\ P_7(x) = (x - c_4)(x - c_5)$ $s_3,\ s_4$
1	$\longleftarrow$	$A_1 = E_S(P_0(x))$
2	$B_1 = E_S(r_1 P_1(c_5))$ $B_2 = E_S(r_1 P_1(c_1) + E_C(Q_1(y))$ $B_3 = E_S(r_1 P_1(c_2) + E_C(Q_2(y_1, y_2))$ $B_4 = E_S(r_1 P_1(c_4) + E_C(Q_4(y))$	$\longrightarrow$
3	$\longleftarrow$	decrypt $B_1, \ldots, B_4$ and see $D_S(B_1) \neq 0,\ D_S(B_4) \neq 0$, $D_S(B_2) = E_C(Q_1(y)),\quad D_S(B_3) = E_C(Q_2(y_1, y_2))$, $A_2 = E_C(rQ_1(s_3))$ $A_3 = E_C(rQ_1(s_4))$ $A_4 = E_C(rQ_1(s_1) + E_S(P_1(x))$ $A_5 = E_C(rQ_1(s_2) + E_S(P_2(x))$ $A_6 = E_C(rQ_2(s_3, s_4))$ $A_7 = E_C(rQ_2(s_3, s_1) + E_S(P_1(x))$ $A_8 = E_C(rQ_2(s_3, s_2) + E_S(P_2(x))$ $A_9 = E_C(rQ_2(s_4, s_1) + E_S(P_1(x))$ $A_{10} = E_C(rQ_2(s_4, s_2) + E_S(P_2(x))$ $A_{11} = E_C(rQ_2(s_1, s_2) + E_S(P_1(x)) + E_S(P_2(x))$
4	decrypt $A_2, \ldots, A_{11}$ and gets valid $E_S(P_1) = D_C(A_4)$, and $E_S(P_2) = D_C(A_8)$ $B_5 = E(P_1(c_1)) + E_S(Q_1(y))$ $B_6 = E(P_1(c_2)) + E_S(Q_2(y_1, y_2))$ $B_7 = E(P_1(c_4)) + E_S(Q_4(y))$ $B_8 = E(P_1(c_5))$ $B_9 = E(P_2(c_1)) + E_S(Q_1(y))$ $B_{10} = E(P_2(c_2)) + E_S(Q_2(y_1, y_2))$ $B_{11} = E(P_2(c_4)) + E_S(Q_4(y))$ $B_{12} = E(P_2(c_5))$	$\longrightarrow$
5		decrypt $B_5, \ldots, B_{12}$ and gets $E_C(Q_4(y)) = D_S(B_7)$, $E_C(Q_4(y)) = D_S(B_{11})$ and $D_S(B_{12}) = 0$, hence ends "Successfully".