Regular Paper

Perfect Privacy-preserving Automated Trust Negotiation[⋆1]

Tangtisanon Pikulkaew[†1] and Hiroaki Kikuchi[†1]

Recently, Automated Trust Negotiation (ATN) has played an important role for two participants who want to automatically establish a trust relationship between each other in an open system so that they can receive some services or information. For example, when a person wants to buy a product from a website he will need to know that the website can be trusted or not. In this scheme, both parties (i.e., the person and the website, in this example) exchange their credentials and access control policies to each other automatically to ensure that the required policies of each party are met; following which the trust negotiation is established. Our proposed scheme allows both parties to learn whether or not, they agree to establish a trust relationship. After the scheme was performed, no policy was disclosed to each other. In this paper, we provide some building blocks used to construct our proposed scheme and describe the basic ideas for hiding access control policies and for implementing a conditional transfer. We also define the steps of how our protocol works with a numerical example. Moreover, we evaluate our scheme in terms of the computation cost by a mathematical analysis and the implementation using binary tree model of credentials and policies. Finally, we show that our scheme can be securely performed.

1. Introduction

Automated trust negotiation (ATN) aims to allow two parties to securely exchange digital credentials in X.509 format [1)] that contains sensitive information such as the name, the address, the birthday and memberships, as well as access control decisions (what credentials are acceptable). Both parties wish to minimize the information to be disclosed to the other party in order to learn the minimal agreement of both private policies.

Winsborough, et al. propose the first scheme for ATN, classified into two extreme strategies, called, *parsimonious* and *eager* strategies in Refs. 2)–4). The eager strategy requires parties to disclose credentials as soon as their access control policy is satisfied, while in the parsimonious strategy, the parties disclose credentials only after a successful outcome is ensured through negotiations. The former strategy has a serious disadvantage that it discloses many non-required credentials. Moreover, in both schemes, two parties need to reveal their partial policies and credentials gradually. Hence, no privacy is preserved. The time complexity of the negotiation depends on how many credentials the two parties have. Hongwei, et al. present a highly efficient strategy for ATN called *Deterministic Finite Automation Negotiation Strategy* (*DFANS*) where only relevant credentials are disclosed and the communication complexity is very low since it depends on the number of the requested credentials in Ref. 5). This scheme also can solve the problem of cyclic dependencies with lower time than the existing scheme.

A number of cryptographic protocols have been proposed so far to address secure and private ATN. Li, et al. propose an oblivious signature based envelop in which a user sends her credentials to a server who jointly computes with the user such that the user sees the requested resource if and only if both policies are consistent in Ref. 6). Nakatsuka and Ishida present a scheme to minimize the sum of costs for disclosure of credentials in Ref. 7). Seamons, Yu and Winslett show two ATN strategies, the relevant credentials set and the all relevant policies strategies, that can protect access control policies in Ref. 8). The relevant credentials set strategy runs very fast, but like the eager strategy, it may disclose unnecessary credentials. The all relevant policies discloses only necessary access control policies. The advantage of the all relevant policies strategy is that it discloses fewer credentials than the relevant credentials set strategy.

Our Contribution

In this paper, we present a new automated trust negotiation scheme that ensures perfect privacy preserving which satisfies the following properties;

(1) No policy is revealed after the performance of the protocol. No certificate is known to either party.

(2) Both parties can learn whether their access control policies have agreement, or not with respect to a target resource.

Although the above requirements sound infeasible, because of the redundancy of a logical formula of access control policy, an agreement of policies can be

⋆1 The original version of this paper is accepted to be published on Advanced Information Networking and Applications (AINA 2011).

†1 School of Information Technology, Tokai University

ensured without disclosure of policies. Using our proposed scheme, both parties are able to securely make sure that their access control policies are satisfied or not before they initiate a transaction, without learning what polices and credentials are used. For example, Alice has a policy "open student id (credential) c if Bob has either official certificate s_1 or s_2" Bob has certificates s_1 and s_2. After the negotiation, Alice does not learn which certificate Bob used to unlock her credential c. Our proposed protocol is the first ATN protocol that ensures the perfect privacy preserving.

In order to fulfill the requirement of a perfect privacy preserving protocol, we combine two cryptographical primitives for a private set intersection protocol proposed by Freedman, et al. in Ref. 9) and the secure multiparty protocol for set operations due to Kissner, et al. in Ref. 10).

The rest of our paper is organized as follows. In Section 2, we give fundamental definitions used in ATN and review the representative schemes of ATN. Section 3 provides some building blocks used to construct the proposed schemes. We use two cryptographical protocols for our purpose. In Section 4, after we describe the basic ideas for hiding access control policies and for implementing a conditional transfer, the proposed scheme is described with a numerical example. In Section 5, we evaluate our proposed protocol based on the construction of the policies in the binary tree model and show that our protocol can be performed securely. Section 6 gives a conclusion of our study and a discussion of future studies.

2. Trust Negotiation

2.1 Fundamental Definition

A *policy* is a set of logic formula consisting of certificates.

Figure 1 shows an example of policies owned by Alice who acts as a client and Bob who acts as a server, where R is a target service. The situation is that Alice wants to buy a beer from Bob while Bob is the owner of a Liquor Store Franchise. This example shows credentials and policies of the two parties, Liquor Store and a customer, Alice.

Alice holds four credentials. Credential (c_1) issued by Japan Automobile Federation (JAF) is her digital driver license. Credential (c_2) issued by a bank is her credit card. Both credential (c_4) and credential (c_5) are issued by the store to verify that Alice is one of the members of this store and she has a point card, respectively. Alice considers her driver license as sensitive. In order to disclose it, she needs a CashierID issued by the store. Since the Credit card is also a sensitive-credential, she will disclose it only to those who have a security certificate issued by Trusted Certificate Authority (TCA) and a good credit record issued by National Credit Bureau (NCB). Her member card is so sensitive that she is willing to disclose only to those who have a branch identity credential that is issued by the head office of the store.

	Alice (Customer)
q_1 :	JAF.DriverLicense (c_1) $\leftarrow$ Store.CashierID (s_1)
q_2 :	BANK.CreditCard (c_2) $\leftarrow$ TCA.SecurityCertificate (s_2)$\wedge$ NCB.GoodCredit(s_3)
q_3 :	Store.MemberCard (c_4) $\leftarrow$ Store.BranchID(s_4)
q_4 :	Store.PointCard (c_5)
	Bob (Liquor Store)
p_1 :	Store.BuyBeer (R) $\leftarrow$ JAF.DriverLicense(c_1)$\vee$ BANK.CreditCard (c_2)
p_2 :	Store.CashierID (s_1) $\leftarrow$ RD.StaffCard (c_3)$\vee$ Store.MemberCard (c_4)
p_3 :	TCA.SecurityCertificate (s_2) $\leftarrow$ Store.MemberCard (c_4)$\vee$ Store.PointCard (c_5)
p_4 :	NCB.GoodCredit (s_3)
p_5 :	MainStore.BranchID (s_4)

Fig. 1 Example of trust policies.

Liquor Store holds four credentials: Credential (s_1) issued by the store, Credential (s_2) issued by TCA to ensure that all of the transactions process securely, Credential (s_3) issued by NCB to guarantee that the store has no bad credit record, Credential (s_4) issued by the main store for the entire branches. Among its credential, the store considers the CashierID and the security certificate as sensitive credentials. To disclose the former, the store needs a staff card issued by Revenue Department (RD) or a member card issued by the store and for the latter, the store needs a member card or a point card issued by the store itself.

The logical relationship between Alice and Bob can be represented in a single trust target graph, shown in **Fig. 2**

The negotiation begins when Alice urges for service R from Bob. At the first round, after receiving a request from Alice, Bob checks his policies and requests

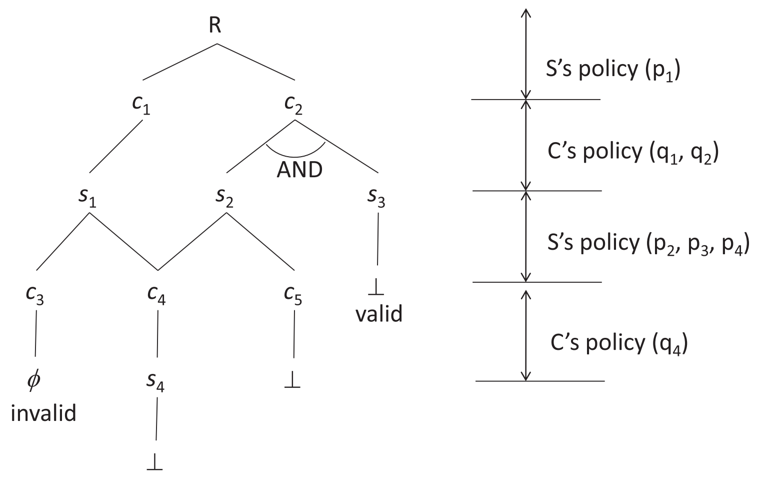


Fig. 2 Example of trust target graph.

either c_1 or c_2 from Alice [⋆1].

In the second round, Alice checks her policies and requests back $s_1 \vee (s_2 \wedge s_3)$. In the third round, Bob checks policies and requests for $(c_3 \vee c_4) \vee (c_4 \vee c_5)$. In the fourth round, Alice requests for s_4 and gives c_5 to Bob. At the final round, Bob matches c_5 with his policy since it is satisfied by c_5, so the negotiation is successful.

2.2 Two Extreme Strategies

In Ref. 2) Winsborough, et al. proposed two extreme strategies for negotiation, an *eager strategy* in which both parties disclose each policy immediately after the condition of policy is satisfied, and a *parsimonious strategy* in which policies are gradually disclosed only after a sufficient policy is ensured.

Definition 2.1 A *policy disclosure rate* is a ratio of the number of disclosed policies over the whole policies, denoted by η. A *round of negotiation* is a number of transmissions of messages between two parties, denoted by ρ.

⋆1 Note that a condition of policy is not specified with the payload of credential (c_1) but with the issuer name of credential (JAF). Namely, Bob requests Alice to show if her credential is issued by JAF, which is denoted by a predicate JAF.DriverLicense (c_1). See Appendix. A.1 for a definition of attributes "issuer".

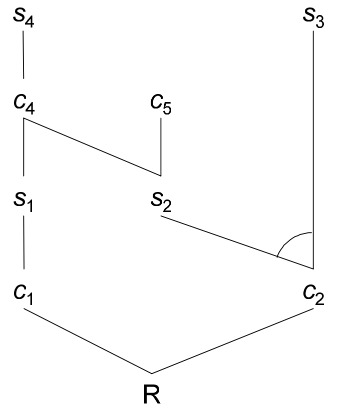


Fig. 3 Eager strategy.

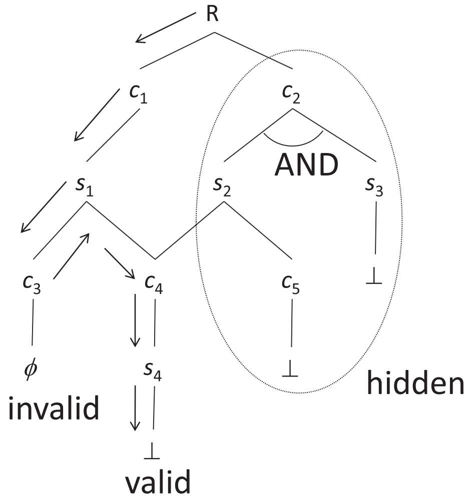


Fig. 4 Parsimonious strategy.

For instance of Fig. 1, the negotiation results in eager and parsimonious strategy are shown in the **Figs. 3** and **4**, respectively. As shown in the figures, some portions of the tree is revealed and even part of one policy is disclosed. For example, in Fig. 4, the policy $p_3 \rightarrow c_4 \vee c_5$ is split into disclosed and hidden parts.

Hence, we divide disjunctive policies into multiple equivalent policies, such as, $p_3 \to c_4$ and $p_3 \to c_5$. Hence, the example with 4 policies for Alice $(q_1, \ldots, q_4)$ plus 5 policies for Bob $(p_1, \ldots, p_5)$ and 3 disjunctive policies (p_1, p_2, p_3) gives 12 in total.

From the above policies, 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 $\eta_{eager} = 11/12 = 0.92$ since after the negotiation was performed Alice and Bob exchange 11 policies to each other from a total of 12 policies and the negotiation successfully ends in $\rho_{eager} = 4$ rounds. In this protocol, both parties must disclose non-required credentials such as c_5 so the discloser rate is very high but it can be performed with high speed. While the parsimonious strategy discloses all possible (authorized to access) policies in Fig. 4, $\eta_{parsimonious} = 6/12 = 0.5$ in $\rho_{parsimonious} = 7$ 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). The parsimonious strategy discloses only necessary credentials so the discloser rate is lower than the eager strategy but requires more time to process.

In these traditional ATNs, a part of policy must be disclosed, i.e., the disclosure rate is minimized, but it never becomes zero. While, our protocol ensures that the discloser rate is 0. After the negotiation, Alice and Bob know only whether they agree or disagree to establish trust while Bob could not figure out which credentials Alice has. Also, Alice could not learn what policy Bob has. Hence, no policy or credential is disclosed. We give a definition of a perfect ATN as follows.

Definition 2.2 An ATN protocol is called *perfect* if and only if its disclosure rate is 0.

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,

$$E[M_1]E[M_2] = E[M_1 + M_2], \quad (1)$$
$$E[M_1]^{M_2} = E[M_1 M_2].$$

For instance, the Paillier cryptosystem [12)] and the modified ElGamal cryptosystem are widely used. Both allow us to get key generation and decryption processes distributed among semi-trusted authorities sharing a private key.

The Paillier cryptosystem [12)] consists of three algorithms, key generation, encryption, and decryption.

- *Key generation:* Let N be pq, a multiplication of large primes p and q, $g \in Z^*_{N^2}$ be a generator whose order divides N. Compute $\lambda = \mathrm{LCM}(p-1, q-1)$ and $\mu = (L(g^\lambda \pmod{n^2}))^{-1}$, where L is defined by $L(u) = (u-1)/n$. The public key is (N, g) and the private key is (λ, μ).
- *Encryption:* A ciphertext of M is defined with a randomly chosen $r \in Z^*_{n^2}$ as,
  $$E(M) = g^M r^N \pmod{n^2}.$$
- *Decryption:* Given ciphertext c, plaintext M is computed by $L(c^\lambda \pmod{n^2})$.

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 solving the discrete logarithm problem. We implement the Paillier cryptosystem for performance evaluation since the single computational cost for encryption is more significant for our proposed protocol, mentioned in a later section.

3.2 Private Matching [9)]

Freedman, et al. presents a cryptographical protocol for a secure set intersection in Ref. 9).

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

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

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 computes

$$\left(\prod_{i=0}^{k_c}(E(\ell_i))^{y^i}\right) E(y) = E(P(y))^r E(y)$$
$$= E(rP(y) + y)$$

and sends k_s ciphertexts to C in a random order, where r is a 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 Ref. 10), 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 to be disclosed to the other party even after their negotiation is completed.

We wish to make a party to send a policy only if the corresponding logical condition is satisfied. To do so, we combine the secure protocol for a set intersection[9] and the set operation protocol[10]. For example, the first policy in Fig. 1,

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

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

$$P_1(x) = r_1(x - c_1)(x - c_2).$$

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

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

can be formed in

$$P_4(y_1, y_2) = r_2(y_1 - s_2) + r_3(y_2 - s_3),$$

which becomes 0 only when $y_1 = s_2$ and $y_2 = s_3$ where r_1, r_2, r_3 are random coefficients.

For preserving the privacy of a policy, we have these polynomials encrypted with a public key of the other party. For example, the ciphertext of polynomial $P_1(x) = x^2 - (c_1 + c_2)x + c_1c_2 = ax^2 + bx + c$ is a tuple $(E(a), E(b), E(c))$, which we denote by $E(P_1(x))$ 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 the candidates of a certificate only if the condition is met but without revealing which certificate is sent. The other party in turns sends a new candidate policy whose condition has been satisfied with the previously sent policy. These interactions are proceeded with preserving the privacy until a requested party verifies if the condition of the target is satisfied. What both parties learn eventually from the communication is just a boolean value.

To make it possible for the conditional transfer, we introduce a new trick based on the Freedman's protocol. Suppose that a client having a policy $c_1 \leftarrow s_1$ receives an encrypted polynomial $E_S(P(x)) = E_S(r_1(x - c_1)(x - c_2)) = (E_0, E_1, E_2)$. 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 the server the condition of c_1 as

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

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. Note that the multiple encryption E_C of E_S is not always possible since modulus n_C can be greater than modulus n_S. The overflow problem will happen when the client tries to compute $E_S(rP(c_1) + E_C(Q(y))$ since the client's bitlength is more than the server's one. Hence, we embed a temporary symmetric key k into the ciphertext since a symmetric key's size can be less than an asymmetric key , and send the corresponding appropriate symmetric ciphertext $\mathcal{E}_k()$ 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 a hybrid encryption for a simplification reason.

The client attempts to send each of her policies $(Q(y))$ one by one in this manner since she does not know which policy is satisfied. In the example in Fig. 1, the client (Alice) sends four ciphertexts,

$$B_1 = E_S(r_1P_1(c_1) + E_C(Q_3(y)),$$
$$B_2 = E_S(r_2P_1(c_2) + E_C(Q_4(y_1, y_2)),$$
$$B_3 = E_S(r_3P_1(c_4) + E_C(Q_9(y)),$$
$$B_4 = E_S(r_4P_1(c_5)),$$

where with only B_1 and B_2 the server (Bob) succeeds to decrypt and extracts the encoded polynomials Q_3 and Q_4. The correspondence between the subscripts and the credential is shown in **Table 1**.

4.3 Proposed Scheme

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

(1) A server sends to a client an encrypted polynomial for a target condition, $A_1 = E_S(P_0(x))$.

(2) The client evaluates the encrypted polynomial with her certificates c_i with a policy $c_i \leftarrow f_i(s_1, \ldots, s_{n_s})$ in her policy set Q as

$$B_i = E_S(rP_i(c_i) + E_C(Q(y_1, y_2, ...))$$

for $i = 1, \ldots, n_C$, and sends to the server $B_1, \ldots, B_{n_C}$ in a random order. Note that r is a random number uniformly chosen for every policy. For simplification, we drop the index.

(3) The server decrypts $B_1, \ldots, B_{n_C}$ with his private key, wishing to get $D_S(B_i) = 0$, 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 messages. For a 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

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

where r is a 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, he needs attempting the evaluation for all size-m, $\binom{n_s}{m}$ combinations of his 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 is $D(A) = 0$, implying "Satisfied Negotiation". If the number of iterations is more than the number of policies (n_S or n_C), then terminate declaring "Negotiation Failure".

Table 1 The correspondence between the ciphertexts and credentials.

ciphertext	credential
B_2, B_5, B_9	Q_1
B_3, B_6, B_{10}	Q_2
B_4, B_7, B_{11}	Q_4
A_1	P_0
A_4, A_7, A_9, A_{11}	P_1
A_5, A_8, A_{10}, A_{11}	P_2

4.4 Example

Table 3 illustrates the sequence of messages sent from a client and a server having policies in Fig. 1. In 5 rounds, the protocol is terminated successfully with the decryption being zero and hence the server learns that their policies have an agreement to provide the requested service.

5. Evaluation

5.1 Computation Cost based on Implementation

We show a performance comparison of negotiation strategies in terms of the degree of privacy to be preserved (disclosure rate), and the communication overhead in **Table 2**. The eager strategy runs the fastest but reveals many policies, while the parsimonious takes a longer time in negotiation with a smaller disclosure rate. Our protocol achieves the best privacy with an intermediate number

Table 2 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	11/12	0
rounds ρ	7	4	5

⋆1 We assume that the integrity of a message can be tested by a predetermined format of valid message so that we easily see if an attempt of decryption is successful or not. The chance that the ciphertexts might be accidentally decrypted as "false success" is negligibly small with the size of modulus. When we perform in 1024 bits a probability to get a false success is $1/2^{1024}$.

Table 3 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	$R: P_0(x) = (x - c_1)(x - c_2)$ $s_1: P_1(x) = (x - c_3)(x - c_4)$ $s_2: P_2(x) = (x - c_4)(x - c_5)$ s_3, s_4
1	$\longleftarrow$	$A_1 = E_S(P_0(x))$
2	$B_1 = E_S(rP_0(c_5))$ $B_2 = E_S(rP_0(c_1) + E_C(Q_1(y)))$ $B_3 = E_S(rP_0(c_2) + E_C(Q_2(y_1, y_2)))$ $B_4 = E_S(rP_0(c_4) + E_C(Q_4(y)))$	$\longrightarrow$
3		decrypt $B_1, \ldots, B_4$ and see $D_S(B_1) \neq 0,\ D_S(B_4) \neq 0$, $D_S(B_2) = E_C(Q_1(y)),\quad D_S(B_3) = E_C(Q_2(y_1, y_2))$,
	$\longleftarrow$	$A_2 = E_C(rQ_1(s_3))$ $A_3 = E_C(rQ_1(s_4))$ $A_4 = E_C(rQ_1(s_1) + E_S(P_1(x)))$ $A_5 = E_C(rQ_1(s_2) + E_S(P_2(x)))$ $A_6 = E_C(rQ_2(s_3, s_4))$ $A_7 = E_C(rQ_2(s_3, s_1) + E_S(P_1(x)))$ $A_8 = E_C(rQ_2(s_3, s_2) + E_S(P_2(x)))$ $A_9 = E_C(rQ_2(s_4, s_1) + E_S(P_1(x)))$ $A_{10} = E_C(rQ_2(s_4, s_2) + E_S(P_2(x)))$ $A_{11} = E_C(rQ_2(s_1, s_2) + E_S(P_1(x)) + E_S(P_2(x)))$
4	decrypt $A_2, \ldots, A_{11}$ and gets valid $E_S(P_1) = D_C(A_4)$, and $E_S(P_2) = D_C(A_8)$	
	$B_5 = E_S(P_1(c_1) + E_C(Q_1(y)))$ $B_6 = E_S(P_1(c_2) + E_C(Q_2(y_1, y_2)))$ $B_7 = E_S(P_1(c_4) + E_C(Q_4(y)))$ $B_8 = E_S(P_1(c_5))$ $B_9 = E_S(P_2(c_1) + E_C(Q_1(y)))$ $B_{10} = E_S(P_2(c_2) + E_C(Q_2(y_1, y_2)))$ $B_{11} = E_S(P_2(c_4) + E_C(Q_4(y)))$ $B_{12} = E_S(P_2(c_5))$	$\longrightarrow$
5		decrypt $B_5, \ldots, B_{12}$ and gets $D_S(B_7) = E_C(Q_4(y)),\ D_S(B_{11}) = E_C(Q_4(y))$ and $D_S(B_{12}) = 0$, hence ends "Successfully".

of rounds.

In order to evaluate the computation cost, we implement our protocol using J2SE (Tomcat apache as a server) and perform two experiments in a platform (Window7 64-bit, CPU core i5, 2.27 GHz, 4 GB). The Encryption and decryption perform the Paillier cryptosystem with 1024 bits modulus and 1024 bits random number. The negotiation time is measured in elapsed time.

(1) **Fundamental case**

We evaluate the performance per one round negotiation in a case that the client has 10 to 50 credentials $c_1, \ldots, c_{50}$, with no condition. While, the server has policies as follows

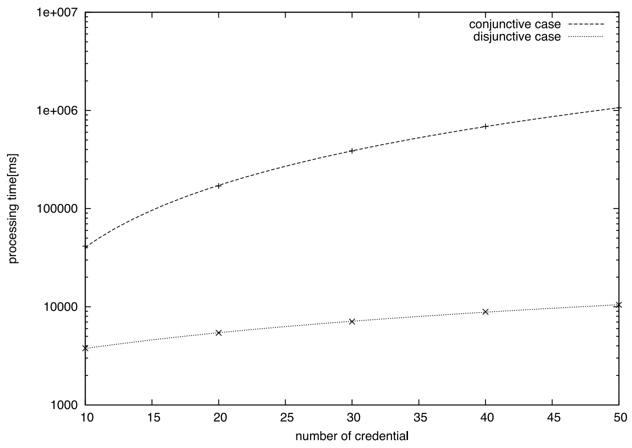


Fig. 5 Processing time with respect to the number of credentials in conjunctive and disjunctive case.

(a) $p_1 : R \leftarrow c_1 \vee c_2$,

(b) $p'_1 : R \leftarrow c_1 \wedge c_2$.

The result of the fundamental case is shown in **Fig. 5**, where the server has one disjunctive policy p_1 that contains two credentials (a) and the client has n_c policies without condition. A total number of credentials up to 150 are used to test. In disjunctive policies (a), in average it requires 3.7 s (elapsed time) to perform the negotiation with 10 credentials. Roughly speaking, a negotiation of disjunctive policy takes 0.37 s per credential. (b) In conjunctive policies, in average it requires 41,439 ms (elapsed time) to perform the task. The experimental result shows that our protocol works effectively since the computation time increases in linear. Unfortunately, in the case of a conjunctive policy, the protocol runs in $O(n_c^2)$ since $\binom{n_c}{2}$ assignments arise to evaluate a two-variable polynomial.

(2) **Advance case**

Suppose credentials and policies of the client and the server are in a balanced binary tree [13]. For example, **Fig. 6** is the binary tree with height $h = 3$, order $k = 2$, a client with 2 policies and a server with 4 policies. The server and the client have the conjunctive policies shown in **Fig. 7**.

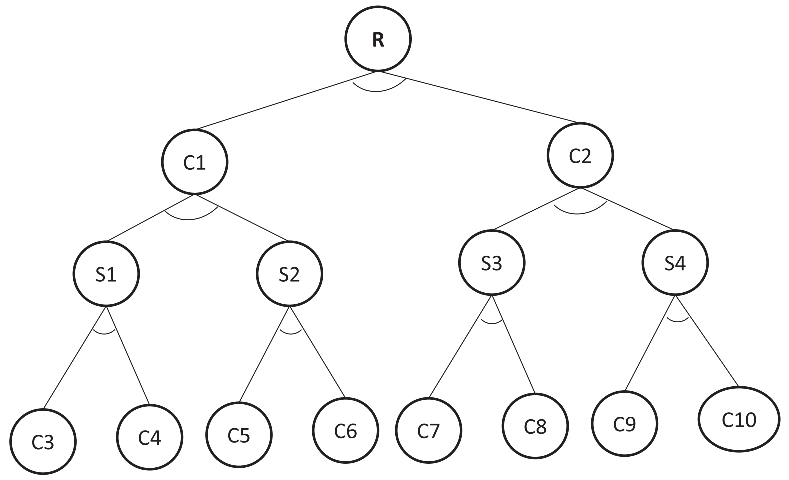


Fig. 6 Example of conjunctive policies based on binary tree.

client		server	
$q_1 :$	$c_1 \leftarrow s_1 \wedge s_2$	$p_1 :$	$R \leftarrow c_1 \wedge c_2$
$q_2 :$	$c_2 \leftarrow s_3 \wedge s_4$	$p_2 :$	$s_1 \leftarrow c_3 \wedge c_4$
$q_3 :$	c_3	$p_3 :$	$s_2 \leftarrow c_5 \wedge c_6$
$q_4 :$	c_4	$p_4 :$	$s_3 \leftarrow c_7 \wedge c_8$
		$p_5 :$	$s_4 \leftarrow c_9 \wedge c_{10}$

Fig. 7 Example of trust policies 2.

(a) All-conjunctive case: Every policy in both parties contains only "a conjunctive" condition as shown in Fig. 6. In order to reach a target credential R, the server requires the client to have both certificates c_1 and c_2.

(b) All-disjunctive case: Every policy in both parties contains only "a disjunctive" condition. For example, in order to reach a target credential R, the server requires the client to have either certificate c_1 or c_2.

(c) Mixed case: Policies contain both conjunctive and disjunctive. The example of policies tree with height = 3 is shown in **Fig. 8**. The

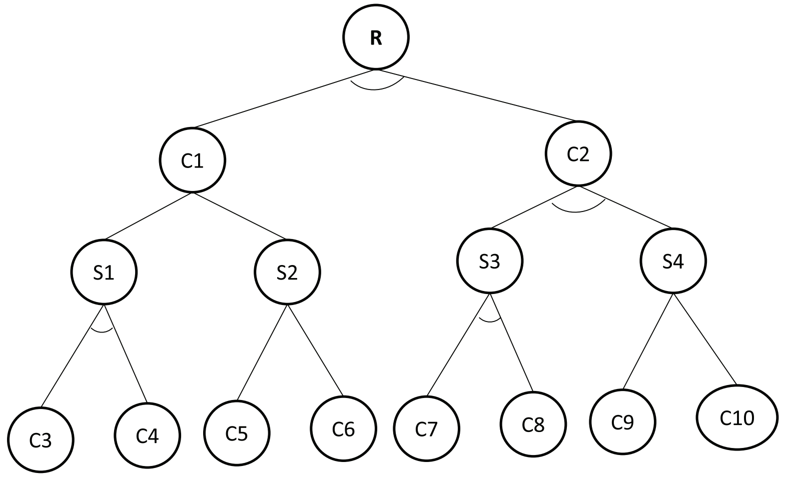


Fig. 8 Example of binary tree with mixed policies.

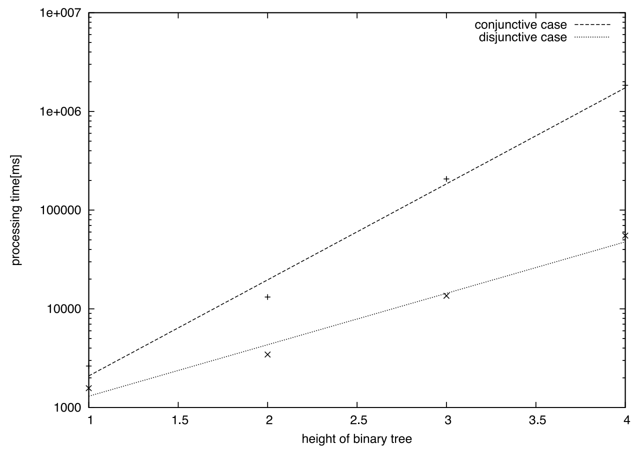


Fig. 9 Processing time with respect to the height of a tree (rounds).

Table 4 Comparison of strategies based on mix-policies binary tree.

strategy	parsimonious (top down)	eager (bottom up)	proposed (top down)
disclosure rate η	$2^h + (\frac{3(2^h-1)}{2})$	1	0
rounds ρ	h+1	h+1	h+1

Table 5 Performance of the proposed protocol.

	policies (Fig. 1)	complexity
# of Ciphertexts	79	$O((\frac{m}{k})^{kh})$
# of En/Decryptions	84	$O(k^h(\frac{m}{k})^{kh})$
# of Rounds	5	1+h

performance of the proposed protocol shown in **Table 4**.

The result of the experiment is shown in **Fig. 9**. In all-disjunctive policies case, the average processing time is increasing about 17,871 ms per round. In all-conjunctive policies, the processing time increases about 612,016 ms per round in average. Our protocol performs efficiently in disjunctive case as it is shown that the computation time is a linear pattern. The conjunctive policy also works well in the case that height = 1 but the more rounds are involved the greater process time is required since in the conjunctive case the total of k out of n_c or n_s combinations are attempted.

5.2 Computation Cost based on Analysis

The cost of computation and communication of our proposed protocol depends on the number of policies that the two parties hold and the complexity of the policies, i.e., the number of credentials as condition of policies.

Suppose that both the client and the server have policies followed by the complete binary tree with a degree of k and a height of h as shown in Fig. 6 and the number of credentials of the client is equal to that of the server side, i.e. $n_c = n_s = m$. The performance number of ciphertexts and En/Decryption are shown in **Table 5**, where we show the upper bounds of the number of ciphertexts using the Stirling's approximation of binomial coefficients [13].

5.3 Security Evaluation

We examine the security of our protocol by assuming a semi-honest model in which both parties must follow the proposed scheme, using their real credentials and policies, but they may collect information received during the negotiation and try to learn the other party's credentials or policies later on.

Suppose a server has a target policy $p_1 : R \leftarrow c_1 \vee c_2 \vee \ldots \vee c_n$.

At Step1 in Section 4.3, the server sends to a client,

$A_1 = E_S(P_0(x)) = E_{s_0}, E_{s_1}, E_{s_2}, \ldots, E_{s_n}$.

An adversary who gets ciphertext A_1, still could not know the server's policies since the policies are encrypted by the server's public key. They need λ and μ to decrypt it. Unfortunately, it is too difficult to compute since the semantic security of Paillier is proved under *the decisional composite residuosity assumption* in Ref. 14). Similarly, the client is not able to learn what policy the server has.

After steps 2 and 3, the server may get decrypted messages. If the server's policy is satisfied then the decryption of B_i becomes 0. Any random number multiplied by zero is equal to zero from multiplicative properties. If $P_i(c_i)$ is not equal to zero then the server gets a uniformly distributed random number with no meaning.

The advantage of the proposed protocol is that after the negotiation was done, no credential or policies was revealed to each other as shown in Table 2 and Table 4.

In this experiment, we test with the assumption that the client and the server act as semi-honest model. Unfortunately, the critical weakness of ATN is that no matter the prevention tool that is applied to it; there is a chance that the client could try the guessing attack with the server. Anyway, the server could apply "Adaptive Oblivious Transfer [16)]", so one target can be queried adaptively in limited times. Also, there are many works that solve the colluding problem. For example, Ref. 15) proposed private inference control (PIC) using computational symmetrically private information retrieval (PIR), homomorphic encryption [12)], non-malleable encryption and zero-knowledge. This method ensures that users are prevented from making undesired inference while privately accessing the database, even when users collude.

6. 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 aggregated without revealing any piece of private information.

Our proposed protocol performs with a low performance in the conjunctive case, a high computation cost which is n out of k complexity. We leave this problem to be solved in the future.

Acknowledgments We thank the anonymous reviewers for their constructive suggestions.

References

1) Housley, R., Ford, W., Polk, T. and Solo, D.: Internet X.509 public key infrastructure certificate and CRL profile, IETF RFC 2459 (1999).
2) Winsborough, W.H., Seamons, K.E. and Jones, V.E.: Automated trust negotiation, *DARPA Information Survivability Conference and Exposition*, Vol.I, pp.88–102, IEEE Press (2000).
3) Winsborough, W.H. and Li, N.: Towards practical automated trust negotiation, *Proc. 3rd International Workshop on Policies for Distributed Systems and Networks* (*POLICY02*), pp.92–103 (June 2002).
4) Li, J., Li, N. and Winsborough, W.H.: Automated trust negotiation using cryptographic credentials, *ACM Trans. Inf. Syst. Secur.*, Vol.13, Issue 1, pp.1–35 (2009).
5) Hongwei, L. and Bailing, L.: DFANS: A highly efficient strategy for automated trust negotiation, Computers and Security, Vol.28, Issue 7, pp.557–565 (2009).
6) Li, N., Du, W. and Boneh, D.: Oblivious signature-based envelope, *Proc. 22nd ACM Symposium on Principles of Distributed Computing* (*PODC*), ACM Press (2003).
7) Nakatsuka, K. and Ishida, T.: Distributed AND/OR Search for Automated Trust Negotiation, Journal of Information Processing (JIP), Vol.47, No.8, pp.2454–2463, IPSJ (2006).
8) Kent, E.S., Marianne, W. and Ting, Y.: Limiting the discloser of access control policies during automated trust negotiation, *Proc. Symposium on Network and Distributed System Security* (*NDSS01*), (Feb. 2001).
9) Freedman, M.J., Nissim, K. and Pinkas, B.: Efficient Private Matching and Set Intersection, *Eurocrypt 2004*, pp.1–19, Springer LNCS (2004).
10) Kissner, L. and Song, D.: Privacy-Preserving Set Operations, *Crypto '05*, pp.200–212, Springer LNCS (2005).
11) Kikuchi, H., Kizawa, H. and Tada, M.: Privacy-Preserving Collaborative Filtering Schemes, *International Conference on Availability, Reliability and Security* (*ARES*), pp.911–916 (2009).
12) Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, *Eurocrypt 1999*, pp.223–238, Springer LNCS (1999).
13) Thomas H.C., Charles E.L., Ronald L.R. and Clifford, S.: *Introduction to algo-*

rithms, second edition, The Massachusetts Institute of Technology (2003).

14) Catalano, D., Gennaro, R., Howgrave-Graham, N. and Nguyen, P.Q.: Paillier's Cryptosystem Revisited, *Proc. 8th Association for Computing Machinery Conference (ACM) on Computer and Communications Security (CCS)*, pp.206–214 (Nov. 2001).

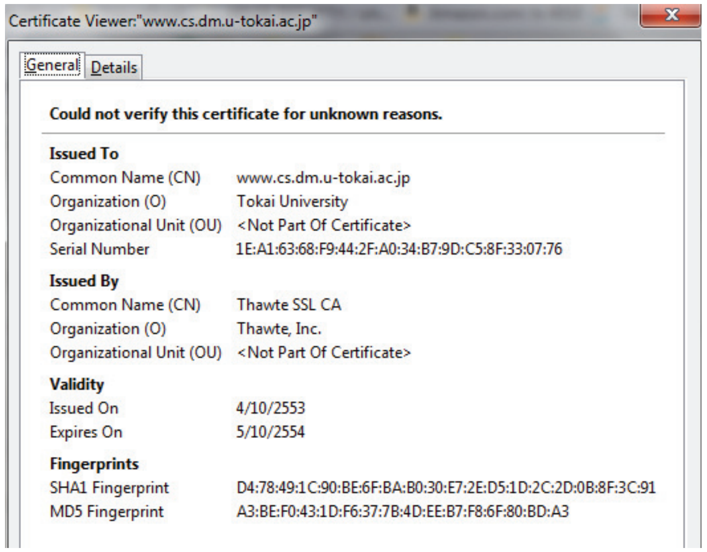
15) Woodruff, D. and Staddon, J. Private Inference Control, *Proc. 11th ACM Conference on Computer and Communications Security (CCS 2004)*, pp.188–197 (Oct. 2004).

16) Naor, M. and Pinkas, B.: Oblivious transfer with adaptive queries, *Proc. 19th Annual International Cryptology Conference on Advances in Cryptology*, pp.573–590 (1999).

Appendix

A.1 Definition of certificate

A public key certificate is an electronic document used to validate the owner's authorization. **Figure 10** shows the sample of a public key certificate of an email's server located in Tokai University issued by thawte, the first certificate authority to issue SSL certificates to public entities outside of the United States. The document contains attributes specified by X.509, including issuer, serial number and so on.


Certificate Viewer:"www.cs.dm.u-tokai.ac.jp"

General Details

Could not verify this certificate for unknown reasons.

Issued To

Common Name (CN)	www.cs.dm.u-tokai.ac.jp
Organization (O)	Tokai University
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	1E:A1:63:68:F9:44:2F:A0:34:B7:9D:C5:8F:33:07:76

Issued By

Common Name (CN)	Thawte SSL CA
Organization (O)	Thawte, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Validity

Issued On	4/10/2553
Expires On	5/10/2554

Fingerprints

SHA1 Fingerprint	D4:78:49:1C:90:BE:6F:BA:B0:30:E7:2E:D5:1D:2C:2D:0B:8F:3C:91
MD5 Fingerprint	A3:BE:F0:43:1D:F6:37:7B:4D:EE:B7:F8:6F:80:BD:A3

Fig. 10 Example of certificate.

Table 6 The summary of terminologies.

Attribute	private	public
Serial Number	√	
Issuer (CA)		√
Subject (Common Name)	√	
Validity	√	
Public Key	√	
Signature	√	

The attributes specified in the certificate are listed in **Table 6**, where the attributes are classified into either public or private to the other party. Attribute "issuer" is classified as public since the list of trusted issuers of certificate is commonly known. For example, anyone could know the list of major credit card companies, VISA, MASTER, and AMEX. The web browser has the list of trusted Certification Authorities. On the other hand, the sensitive information is an ownership of the certificate. Hence, server does not know which CA issued the client's certificate.

The server can compute P_0 while it does not have to know the client certificate. In the negotiation process, the server only asks if the client has the certificates that satisfy its policies or not without verifying the ownership in this process (the certificate contains only public information). After the negotiation succeeds, the ownership can be sent to the server if necessary[⋆1].

The private information part will be used later by Alice to verify the ownership of the certificate when the negotiation is successful.

(Received December 1, 2010)
(Accepted June 3, 2011)
(Released September 7, 2011)

⋆1 Our protocol focuses on matching between clients and servers. After the client consents to appropriate one, the ownership can be revealed to the server only.

Tangtisanon Pikulkaew received her B.E., and M.E. degrees from King Mongkut's Institute of Technology Ladkrabang in 2003 and 2005. After she worked in ITONE Company from 2003 through 2004, she has joined King Mongkut's Institute of Technology Ladkrabang in 2004 as a research assistant. Since 2005, she has been with Information Engineering department at King Mongkut's Institute of Technology Ladkrabang, Thailand, where she is presently a lecturer. Now, she is Ph.D. candidate at Tokai University. Her recent research interests include Cryptography, Network Security and Web Application.

Hiroaki Kikuchi was born in Japan. He received his B.E., M.E. and Ph.D. degrees from Meiji University in 1988, 1990 and 1994. After he working in Fujitsu Laboratories Ltd. from 1990 through 1993, he joined Tokai University in 1994. He is currently a Professor in the department of communication and network Engineering, school of information and telecommunication Engineering, Tokai University. He was a visiting researcher of the school of computer science, Carnegie Mellon University in 1997. His main research interests are fuzzy logic, cryptographical protocol, and network security. He is a member of IEICE, the Japan Society for Fuzzy Theory and Systems (SOFT), IEEE and ACM. He is a fellow of IPSJ.