情報処理学会論文誌 Vol.53 No.9 2234–2244 (Sep. 2012)

BPPM/AHES：準同型暗号系を用いた安全な自動トラスト交渉

青山 桃子[1,a)] 森 文宏[1] 八槇 博史[2]

受付日 2011年11月30日, 採録日 2012年6月1日

概要：ネットワーク上で互いに未知であるサービス提供者とサービス利用者の間で，信頼性評価に必要な証明書を，開示ポリシに反することなく交換する枠組みとして自動トラスト交渉（ATN）が研究されてきた．本研究では，加法的準同型暗号系を用いることで，ATN で問題であった不要な情報の開示を最小化する BPPM/AHES プロトコルを考案した．この方式に基づいて REST 型の通信により ATN を行う RATN システムを実装した．BPPM/AHES では，不要な情報開示をなくすことができる一方で，加法的準同型暗号系の処理速度の問題から，実行時間の点で課題が残る．

キーワード：自動トラスト交渉（ATN），加法的準同型暗号，ポリシマッチング

BPPM/AHES: A Privacy-preserving Automated Trust Negotiation Based on Homomorphic Encryption Systems

Momoko Aoyama[1,a)] Fumihiro Mori[1] Hirofumi Yamaki[2]

Received: November 30, 2011, Accepted: June 1, 2012

Abstract: Automated trust negotiation (ATN) is a framework where service providers and users exchange credentials for evaluating each other's trustworthiness. This paper presents BPPM/AHES, a protocol that minimizes unnecessary information disclosure by using additively homomorphic encryption systems. We implemented RATN, a system that performs ATNs based on BPPM/AHES. Performance issues associated with encryption systems still remain according to the experimantal results.

Keywords: Automated Trust Negotiation (ATN), additively homomorphic encryption, private policy matching

1. はじめに

インターネットのような開かれたネットワーク環境においては，相互に未知である相手との間でサービスや情報のやりとりを行う場面が頻繁にあらわれる．その前段階として，証明書（Credential）の交換を自動的に行うことで相手の信頼性を評価する枠組みとして自動トラスト交渉（Automated Trust Negotiation，ATN）[1] が提案されている．証明書には交渉者の名前や所属などアイデンティティを明らかにするような情報も含まれうるため，保持する証明書を無条件で開示することは望ましくない．ATN では，各証明書について，それを開示するために相手から開示を受けるべき証明書のリスト（開示ポリシ）の存在が前提とされており，この開示ポリシに反することのない開示手順を自動的に求めることが目標とされている．ATN における証明書交換手順の基本的な算出方法には Eager Strategy や Parsimonious Strategy [1] などが存在し，それらを実装した例としては，WS-ATN [2] や Trust Builder2 [3] などが存在する．これらに共通の問題として，交渉の過程で本来開示する必要のない余分な証明書まで開示したり，相手に知らせる必要のないポリシを知らせたりすることがある．

[1] 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University, Nagoya, Aichi 464–8601, Japan
[2] 名古屋大学情報基盤センター
Information Technology Center, Nagoya University, Nagoya, Aichi 464–8601, Japan
[a)] aoyama@net.itc.nagoya-u.ac.jp

本研究では，準同型暗号系を用いることにより，開示ポリシを暗号化したままの状態で証明書開示手順を求め，交渉が成功する手順が存在する場合にのみ復号することによって，開示ポリシまで含めて余分な情報開示のないATN プロトコルである BPPM/AHES を考案した．さらに BPPM/AHES を実装した ATN 基盤，RATN（Restful Automated Trust Negotiation）を開発し，動作実験と評価を行った．

本稿ではまず 2 章で ATN および提案手法で用いた準同型暗号系，単方向ポリシマッチングについて述べる．次の 3 章では BPPM/AHES のプロトコルと動作例を記述する．4 章では，RATN の実装について説明する．5 章では，提案手法についての評価と考察を行う．最後に 6 章で結論を述べる．

2. 準同型暗号系を用いた ATN

2.1 Automated Trust Negotiation

ATN では，サービス提供者（サーバ）とサービス利用者（クライアント）はそれぞれ証明書およびそれらの証明書を開示するための条件を記した開示ポリシを持つ．それに加えて，サービス提供者はサービスを利用するためにサービス利用者が満たすべきポリシ（Service Governance Policy，SGP）を持つ．証明書とは，提示者本人のものであると信頼された第三者（Trusted Third Party，TTP）により証明されたデータであり，氏名，所属など個人を特定する情報や機密性のある情報を含みうる．サーバ，クライアントともに複数の異なる証明書を保持する．

クライアント，サーバの開示ポリシの記述例を**表 1** に示す．開示ポリシは，$\leftarrow$ の左側（左辺）に記述された証明書を開示するための条件を $\leftarrow$ の右側（右辺）に記述する形で表現される．右辺が $true$ の場合，その証明書は無条件で開示可能であることを表す．右辺が $false$ の場合，その証明書はいかなる場合でも開示不可能であることを表す．開示ポリシは，開示するための条件を $\wedge$（かつ）や $\vee$（または）を用いて記述する．サーバの開示ポリシである SGP は左辺を R として記述する．サービス利用者により，SGP が満たされると交渉が成立してサービスが提供される．ATN は，以上のような状況において，証明書を開示ポリシに反することなく交換する手順を自動的に求める過程である．

表 1 開示ポリシの例
Table 1 Example of policies.

クライアントの開示ポリシ	サーバの開示ポリシ
$C_1 \leftarrow true$	$R \leftarrow (C_3 \wedge C_4) \vee C_6$
$C_2 \leftarrow true$	$S_1 \leftarrow true$
$C_3 \leftarrow S_1 \wedge S_2$	$S_2 \leftarrow (C_1 \wedge C_2) \vee C_3$
$C_4 \leftarrow S_3 \vee S_4$	$S_3 \leftarrow C_3 \vee C_4$
$C_5 \leftarrow S_2 \vee S_3$	$S_4 \leftarrow C_4$
$C_6 \leftarrow false$	$S_5 \leftarrow C_1 \wedge C_5$

そのための手法やプロトコルが多数 [4], [5], [6], [7], [8] 提案されている．以下に示す，Eager Strategy と Parsimonious Strategy は，Winsborough ら [1] によって提案されたものである．

2.1.1 Eager Strategy

Eager Strategy は，クライアントとサーバが交互にその時点で開示可能なすべての証明書を開示していき，SGP が満たされたとき，トラスト交渉が成立すると判定する手法である．

Eager Strategy は交渉プロセスが単純で，クライアントおよびサーバの開示ポリシが開示されないという特徴を持つ．しかし，Eager Strategy では，必ずしも開示する必要のない証明書（開示しても相手の証明書開示に影響しない証明書）まで開示してしまうという問題がある．また，交渉が成立することが分かってから証明書の開示を行うのではなく，実際に証明書を開示しながら交渉するため，交渉が成立しない場合でも証明書が開示されてしまうという欠点もある．

2.1.2 Parsimonious Strategy

Parsimonious Strategy は，クライアントとサーバが開示ポリシをもとに相手に証明書要求を行い開示手順を求め，サーバの SGP が満たされるような証明書開示手順を見つけた後に証明書の開示を行う手法である．開示手順は，両者のポリシから構成される探索空間内で開示手順に相当する解グラフを求めることで導出する．Parsimonious Strategy では，余分な証明書は開示されないが，証明書要求の過程において，最終的な証明書開示手順を求めるのに必要な開示ポリシ以外の開示ポリシが開示されるという欠点がある．ATN 研究において提案されてきた各種のプロトコルは，Parsimonious Strategy における探索手法を様々に改良したものが多くを占める．

2.2 加法的準同型暗号系

この節では，本研究で用いる準同型暗号系について述べる．

2.2.1 加法的準同型性

暗号化関数 E，秘密鍵 s に対応する公開鍵 p およびメッセージ m が与えられたとき，公開鍵 p によるメッセージ m の暗号文を $E_p(m)$ と記述する．暗号化関数 E はすべてのメッセージ m_1 と m_2 に対して次の関係が成り立つとき，加法的準同型性を持つ．

$$E_p(m_1)E_p(m_2) = E_p(m_1 + m_2)$$

また，準同型性暗号は次のような性質も持つ．

$$E(m)^c = E(cm)$$
$$E(m)E(0) = E(m)$$

2.2.2 El Gamal 暗号

El Gamal 暗号 [9] とは，離散対数問題の困難性を安全性の根拠とした公開鍵暗号である．加法的準同型性を持つ El Gamal 暗号は次のようなものである．位数 q が素数である巡回群を $G = \langle g \rangle$ とする．El Gamal 暗号の公開鍵は $h \in G$ であり，メッセージ $m \in \mathbb{Z}_p$ の暗号文は，$(a, b) = (g^r, h^r g^m)$，$r \in_r \mathbb{Z}_q$ である．秘密鍵は s は，$s = \log_g h$ である．秘密鍵 s が与えられたとき，暗号文 $(a, b) = (g^r, g^m h^r)$ の復号はまず $b/a^s = g^m$ を計算し，それから $m \in \mathbb{Z}_q$ を解くことにより実行される．なお本研究では $m \in \{-1, 1\}$ であり，m のとりうる値の範囲が小さい．ゆえに g^{q-1} および g^1 を求め，g^m と比較することによって m を決定している．

平文 $m_1, m_2 \in \mathbb{Z}_q$ の暗号文は，$(g^{r_1}, h^{r_1} g^{m_1})$，$(g^{r_2}, h^{r_2} g^{m_2})$ であり，2 つの暗号文を掛け合わせれば，$(g^{r_1+r_2}, h^{r_1+r_2} g^{m_1+m_2})$ となり，$m_1 + m_2$ の暗号文となることから加法的準同型性を持つことが分かる．

また，本研究では分散秘密鍵を用いて閾値復号 [10] を行う．復号手順の概要を以下に示す．サーバの分散鍵を s_s，クライアントの分散鍵を s_c とする．まずサーバが暗号文 (a, b) から $D_s = b/a^{s_s}$ を計算する．サーバは D_s をクライアントに送信し，クライアントは $D_s/a^{s_c} = g^m$ を計算する．これによってクライアントは g^m を得られる．サーバが結果を得たい場合は，クライアントとサーバの立場を逆にすることで実現可能である．

以下，暗号文 e を復号すると平文 m となることを $D(e) = m$，平文 m を暗号化すると暗号文 e となることを $E(m) = e$ と記述する．また閾値復号の途中で，サーバの分散鍵によって復号が行われた状態の暗号文を $D_s(m)$，クライアントの分散鍵によって復号が行われた状態の暗号文を $D_c(m)$ と記述する．

2.3 単方向ポリシマッチング

Kursawe ら [11] は，クライアントの開示可能な証明書群がサーバの証明書開示要求を満たすか否かを，加法的準同型暗号系を用いて証明書のリストを暗号化したまま決定する手法を提案した．本稿ではこの手法を単方向ポリシマッチングと呼び，それが満たされた場合の証明書集合をマッチングポリシと呼ぶ．定義は以下のようになる．

定義 2.1. 証明書集合を $\mathcal{C}$ とし，$\mathcal{S} = \{0, 1\}$ とする．クライアントが開示可能な証明書のリストは以下の関数 $f : 2^{\mathcal{C}} \to \mathcal{S}$ で表される．

$$f(C) = \begin{cases} 1 & C\text{がクライアントが開示してもよいと考える証明書の組合せ} \\ 0 & \text{otherwise} \end{cases}$$

サーバが要求する証明書のリストは以下の関数 $g : 2^{\mathcal{C}} \to \mathcal{S}$ で表される．

$$g(C) = \begin{cases} 1 & C\text{がサーバがサービス利用者に要求する証明書の組合せ} \\ 0 & \text{otherwise} \end{cases}$$

マッチング関数は，以下の関数 $M : 2^{\mathcal{C}} \times \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ で表される．

$$M(C, f(C), g(C)) = \begin{cases} 1 & f(C) = g(C) = 1 \\ 0 & \text{otherwise} \end{cases}$$

準同型暗号系を用いたポリシマッチングは，マッチングポリシが存在するかどうか，存在するならばどんなマッチングポリシであるかを求めるクライアントとサーバ間のプロトコルであり，それ以外の証明書のリストに関する情報が漏れることはない．

2.3.1 生成部分集合を用いた手法

クライアントの証明書リストを表す関数 f は単調減少ブール関数であり，$f(A) = 1$ かつ $B \subseteqq A \Rightarrow f(B) = 1$ が成り立つ．同様に，サーバの証明書リストを表す関数 g は単調増加ブール関数であり，$g(A) = 1$ かつ $B \supseteqq A \Rightarrow g(B) = 1$ が成り立つ．

定義 2.2（生成部分集合）．関数 $h : 2^{\mathcal{C}} \to \{0, 1\}$ を単調増加ブール関数とする．$\mathcal{H} = \{H_1, \ldots, H_a\} \subseteqq 2^{\mathcal{C}}$ は次の条件を満たすとき，関数 h の生成部分集合の集合である．

すべての $C \subseteqq \mathcal{C}$ に対して，

$$h(C) = 1 \iff \exists i \in \{1, \ldots, a\} : H_i \subseteqq C$$

クライアントの開示可能な証明書リストを表す関数 f は単調減少関数であるので，$\neg f$ は単調増加関数である．よって $\neg f$ の生成部分集合の集合は，$\mathcal{F} = \{F_1, \ldots, F_a\}$ と表される．本質的に，集合 $F_1, \ldots, F_a$ はクライアントが一緒に開示したくない証明書の組合せとなる．生成部分集合は $\{0, 1\}$ の列に符号化される．上記のように表 1 のクライアントの開示ポリシを符号化すると，生成部分集合は $\mathcal{F} = \{\{0, 0, 1, 0, 0, 0\}, \{0, 0, 0, 1, 0, 0\}, \{0, 0, 0, 0, 1, 0\}, \{0, 0, 0, 0, 0, 1\}\}$ と表される．同様に，サーバの証明書リストを表す関数 g の生成部分集合の集合は，$\mathcal{G} = \{G_1, \ldots, G_b\}$ と表される．集合 $G_1, \ldots, G_b$ は，サーバがサービスを提供する前にクライアントへ要求する証明書の組合せとなる．表 1 のサーバの開示ポリシのうち，R の開示ポリシを符号化すると，生成部分集合は $\mathcal{G} = \{\{0, 0, 1, 1, 0, 0\}, \{0, 0, 0, 0, 0, 1\}\}$ と表される．

つまりマッチングポリシを見つけることは，次の条件を満たすような証明書集合 $C \subseteqq \mathcal{C}$ を見つけることと等価となる．

$$\forall F_i \in \mathcal{F} : F_i \not\subseteqq C \text{ かつ } \exists G_j \in \mathcal{G} : G_j \subseteqq C$$

マッチングポリシが存在する場合，そのマッチングポリシはサーバの生成部分集合の中の 1 つである．したがっ

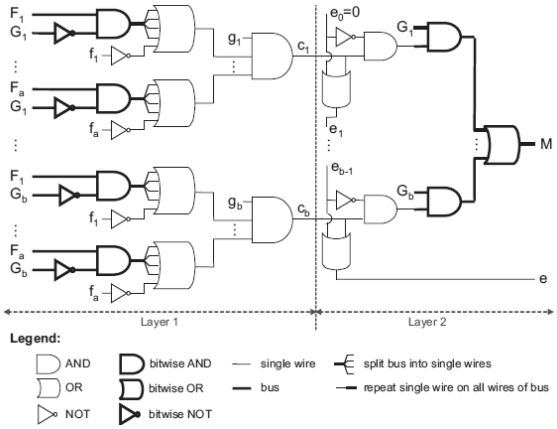

図 1 ポリシマッチング回路（文献 [11] より引用）
Fig. 1 Circuit for private policy matching.

て，マッチングポリシ C の条件はよりコンパクトに以下のように表現できる．

$$\forall F_i \in \mathcal{F} : F_i \nsubseteqq G_j \text{を満たす証明書集合} \; G_j \in \mathcal{G} \qquad (1)$$

式 (1) は，生成部分集合を**図 1** のポリシマッチング回路に入力したときの出力として得られる．回路中の計算暗号化した状態で行われるので，演算には Private Multiplier Gate と Conditional Gate [12] を用いる．Private Multiplier Gate は暗号化されていない入力ビットと暗号化されている入力ビットを乗算し，結果を暗号化されたビットとして出力する手法である．Conditional Gate は暗号化されている 2 つの入力ビットを乗算し，結果を暗号化されたビットとして出力する手法である．この 2 つの手法を用いることで，計算中に平文が明らかになることなく演算が可能となる．また ElGamal 暗号を使用する際に，生成部分集合中の 0 を $m = -1$，1 を $m = 1$ として扱う．暗号化した状態での AND，OR，NOT 回路は，ElGamal 暗号の加法的準同型性を用いて以下のように計算される．

暗号化された入力での AND 暗号化された入力ビット $E(x)$ と $E(y)$ に対して Conditional Gate を適用することで $E(x \wedge y)$ が得られる．

暗号化された入力と暗号化されていない入力との AND 暗号化されていない入力ビット x と暗号化された $E(y)$ に対して Private Multiplier Gate を適用すると $E(x \wedge y)$ が得られる．

暗号化された入力での OR 暗号化された入力ビット $E(x)$ と $E(y)$ に対して，まず Conditional Gate を適用する．準同型性暗号の性質を利用して $E(x \vee y) = E(x+y-xy)$ を計算する．

暗号化された入力と暗号化されていない入力との OR 暗号化されていない入力ビット x と暗号化された $E(y)$ に対して Private Multiplier Gate を適用して $E(xy)$ を得る．その後 $E(x \vee y) = E(x + y - xy)$ を計算する．

暗号化された入力での NOT 暗号化された入力ビット $E(x)$ に対して，準同型性暗号の性質を利用して $E(\neg x) = E(1 - x)$ を計算する．

2.3.2 単方向ポリシマッチングプロトコル

2.3.1 項の式 (1) を求める単方向ポリシマッチングプロトコルは次のようになる．

(1) クライアントとサーバは，あらかじめ分散鍵生成プロトコル [13] により，分散鍵を持っており，それぞれの分散鍵を用いて生成部分集合を表す文字列をビット単位で暗号化する．

(2) クライアントとサーバは，式 (1) を求める計算を暗号化した状態 [12] で実行する．式 (1) は，2.3.1 項のように表現された生成部分集合を，図 1 のポリシマッチング回路に入力することで求められる．出力は式 (1) を満たす証明書集合であるマッチングポリシが存在するか否かを示す値 e と，マッチングポリシ自体を表す値 M が，暗号化された状態で得られる．

(3) 得られた値 e を，閾値復号する．マッチングポリシが存在する場合は，さらに M を復号し，証明書集合を得る．

3. BPPM/AHES

2.3 節で示した方法では，クライアントがサーバを評価する双方向の自動トラスト交渉を行えない．そこで，これまでに述べてきたクライアントを開示する側，サーバを要求する側として行う単方向ポリシマッチングを正方向ポリシマッチング，逆にサーバを開示する側，クライアントを要求する側として行う単方向ポリシマッチングを逆方向のポリシマッチングと定義する．そして，正方向ポリシマッチングと逆方向ポリシマッチングを繰り返し行うことにより，双方向の ATN を実現する，BPPM/AHES（Bidirectional Private Policy Matching/Additive Homomorphic Encryption Systems）を提案する．

クライアントとサーバはそれぞれ**表 2**，**表 3** のような交渉表を持ち，この交渉表に交渉経過を書き込みながら交渉を進める．交渉表には，それぞれが持つ開示ポリシに加えて，開示フラグと MP（マッチングポリシ）という項目がある．開示フラグは，交渉の過程においてその証明書が開示済みなのかあるいは未開示なのかを記憶しておくためのものであり，開示済みなら 1，未開示なら 0 とする．交渉前の段階では，すべての証明書が未開示なので初期値として 0 を与える．MP にはポリシマッチングにより発見されたマッチングポリシを，暗号化したままの状態で保管する．交渉前の段階で右辺が *true* である証明書については，その MP の項目に *true* を記述する．

BPPM/AHES は，交渉プロトコルと証明書交換プロトコルの 2 つから構成されている．

表 2 クライアントの交渉表
Table 2 Client's negotiation table.

左辺	←	右辺	開示フラグ	MP
C_1	←	*true*	0	*true*
C_2	←	*true*	0	*true*
C_3	←	$S_1 \wedge S_2$	0	
C_4	←	$S_3 \vee S_4$	0	
C_5	←	$S_2 \vee S_3$	0	
C_6	←	*false*	0	

表 3 サーバの交渉表
Table 3 Server's negotiation table.

左辺	←	右辺	開示フラグ	MP
R	←	$(C_3 \wedge C_4) \vee C_6$		
S_1	←	*true*	0	*true*
S_2	←	$(C_1 \wedge C_2) \vee C_3$	0	
S_3	←	$C_3 \vee C_4$	0	
S_4	←	C_4	0	
S_5	←	$C_1 \wedge C_5$	0	

表 4 1 回目の各方向のポリシマッチング後のクライアントの交渉表
Table 4 Client's negotiation table after 1st policy matching.

左辺	←	右辺	開示フラグ	MP
C_1	←	*true*	1	*true*
C_2	←	*true*	1	*true*
C_3	←	~~$S_1 \wedge S_2$~~ *true*	0	E_{C_3}
C_4	←	$S_3 \vee S_4$	0	
C_5	←	~~$S_2 \vee S_3$~~ *true*	0	E_{C_5}
C_6	←	*false*	0	

表 5 1 回目の各方向のポリシマッチング後のサーバの交渉表
Table 5 Server's negotiation table after 1st policy matching.

左辺	←	右辺	開示フラグ	MP
R	←	$(C_3 \wedge C_4) \vee C_6$		
S_1	←	*true*	1	*true*
S_2	←	~~$(C_1 \wedge C_2) \vee C_3$~~ *true*	1	E_{S_2}
S_3	←	$C_3 \vee C_4$	0	
S_4	←	C_4	0	
S_5	←	$C_1 \wedge C_5$	0	

3.1 交渉プロトコル

交渉プロトコルでは，以下の手順で交渉の成否の決定を行う．

(1) クライアントがサーバに対してサービス要求をする．
(2) 要求を受け取ったサーバが開始シグナルを返す．
(3) クライアントとサーバは 2.3 節の方法で正方向ポリシマッチングを行う．
(4) 正方向ポリシマッチングが終了した時点で，サーバは自身の交渉表の R の右辺が *true* であるかを判定する．*true* ならば，証明書交換プロトコルに移る．*true* でない場合には，サーバはさらに自身の開示ポリシに右辺が *true* かつ開示フラグが 0 である証明書があるかを判定する．ある場合には，さらなる交渉の余地があるので，逆方向ポリシマッチング要求をクライアントへ送る．ない場合には，交渉の余地がないので交渉中止シグナルを送り，交渉中止となる．
(5) クライアントがサーバから逆方向ポリシマッチング要求を受け取ると，逆方向ポリシマッチングが開始される．
(6) 逆方向ポリシマッチングが終了した時点で，クライアントは自身の開示ポリシに右辺が *true* かつ開示フラグが 0 である証明書がないかを判定する．ある場合には，正方向ポリシマッチング要求をサーバへ送る．ない場合には，交渉中止シグナルを送り，交渉中止となる．

3.2 証明書交換プロトコル

証明書交換プロトコルでは，まず交渉成立となる証明書開示手順を求めてからその手順に則って実際に証明書を交換する．

(1) サーバは R の MP 欄の内容に従って証明書要求を生成し，クライアントに送信する．
(2) サーバの証明書要求が，クライアントが無条件で開示可能な証明書によって満たされた場合，証明書交換 (4) に移る．そうでない場合はクライアントは要求された証明書の MP 欄の内容に従って証明書要求を生成し，サーバに送信する．
(3) サーバは同様の判定を行う．条件を満たすまで，両者は証明書要求を交互に繰り返す．
(4) 以上の証明書要求を逆順にたどることで，証明書交換手順を得る．
(5) 証明書交換手順に従って，実際に証明書を交換する．

3.3 動作例

実際に本提案手法を実行したときの動作を各交渉段階での交渉表の記述の変化を見ながら順に追って説明する．クライアントとサーバの入力は表 1 の開示ポリシとする．

交渉プロトコルが実行されて，1 回目の正方向ポリシマッチングおよび 1 回目の逆方向ポリシマッチングが終了した時点のクライアントとサーバの交渉表を**表 4**，**表 5** に示す．

1 回目の正方向ポリシマッチングにおいて，証明書 C_1，C_2 が無条件で開示可能なため開示フラグが 1 となる．また S_2 が開示可能であることが判明することにより S_2 の右辺が *true* となり，そのときの暗号化されたマッチングポリシ E_{S_2} が S_2 の MP 欄に記載される．R の右辺が *true* でなく，右辺が *true* かつ開示フラグが 0 である証明書 S_1，

表 6 3 回目の正方向ポリシマッチング後のクライアントの交渉表
Table 6 Client's negotiation table after 3rd policy matching.

左辺	←	右辺	開示フラグ	MP
C_1	←	$true$	1	$true$
C_2	←	$true$	1	$true$
C_3	←	~~$S_1 \wedge S_2$~~ $true$	1	E_{C_3}
C_4	←	~~$S_3 \vee S_4$~~ $true$	1	E_{C_4}
C_5	←	~~$S_2 \vee S_3$~~ $true$	1	E_{C_5}
C_6	←	$false$	0	

表 7 3 回目の正方向ポリシマッチング後のサーバの交渉表
Table 7 Server's negotiation table after 3rd policy matching.

左辺	←	右辺	開示フラグ	MP
R	←	~~$(C_3 \wedge C_4) \vee C_6$~~ $true$		E_R
S_1	←	$true$	1	$true$
S_2	←	~~$(C_1 \wedge C_2) \vee C_3$~~ $true$	1	E_{S_2}
S_3	←	~~$C_3 \vee C_4$~~ $true$	1	E_{S_3}
S_4	←	~~C_4~~ $true$	0	E_{S_4}
S_5	←	~~$C_1 \wedge C_5$~~ $true$	1	E_{S_5}

S_2 が存在することから，サーバは交渉続行と判定し，逆方向マッチングが開始される．

続く 1 回目の逆方向ポリシマッチングでは同様にして，C_3，C_5 が新たに開示可能と判明し，それらの右辺が $true$ となり，対応する暗号化されたマッチングポリシ E_{C_3}，E_{C_5} が MP 欄に記載される．右辺が $true$ かつ開示フラグが 0 である証明書 C_3，C_5 が存在することから，クライアントは交渉続行と判定し，正方向マッチングへと移る．

以下同様の手順を繰り返し，3 回目の正方向ポリシマッチングが完了した時点の交渉表を**表 6**，**表 7** に示す．サーバの右辺が $true$ となっているため，サーバは成功する交渉手順が存在すると判断し，証明書交換プロトコルに移る．

証明書交換プロトコルでは，サーバはまず R の MP 欄の暗号化されたマッチングポリシ E_R をクライアントとともに閾値復号により復号し，$D(E_R) = \{C_3, C_4\}$ を得る．サーバは証明書要求 $C_3 \wedge C_4$ を生成し，クライアントに送る．証明書要求 $C_3 \wedge C_4$ を受け取ったクライアントは，証明書 C_3，C_4 の MP 欄の暗号化されたマッチングポリシ E_{C_3}，E_{C_4} をサーバとともに閾値復号により復号し，$D(E_{C_3}) = \{S_1, S_2\}$，$D(E_{C_4}) = \{S_3\}$ を得る．クライアントは証明書要求 $S_1 \wedge S_2 \wedge S_3$ を生成し，サーバに送る．以下同様にして，相手からの証明書要求が，無条件で開示可能な証明書で満たされるまで証明書要求を相互に繰り返す．その後，証明書要求を逆順にたどることで，上の例では $\{C_1, C_2\} \rightarrow \{S_1, S_2\} \rightarrow \{C_1, C_2, C_3\} \rightarrow \{S_1, S_2, S_3\} \rightarrow \{C_3, C_4\} \rightarrow \{R\}$ という証明書開示手順が得られる．この順序で証明書交換を行うことで，ATN が完了する．

3.4 交渉に不要なポリシ開示の抑制

交渉の成否決定前における情報開示量について，Eager Strategy と Parsimonious Strategy および BPPM/AHES を比較する．

Eager Strategy は証明書を開示しあいながら交渉を進める手法であるため，交渉の成否決定前に証明書が開示されてしまう．また，一番最初に開示する無条件で開示可能な証明書については必ず開示される．Parsimonious Strategy は証明書開示手順を求めてから証明書を交換する手法であるため，交渉の成否決定前に証明書が開示されることはないが，証明書要求の際に必要のない開示ポリシも開示する場合がある．BPPM/AHES はこれまでに述べてきたように，交渉の成否決定前における証明書の開示および開示ポリシの開示はいっさいない．

BPPM/AHES と Parsimonious Strategy は基本的に同じ流れであるが，証明書要求の生成の際に用いる条件のみが異なる．Parsimonious Strategy では開示ポリシそのものを用いるが，BPPM/AHES ではマッチングポリシを用いる．マッチングポリシとは，開示ポリシの十分条件であり，これを用いることで交渉プロトコルで成立することが分かった証明書開示手順を求めることが可能となる．Parsimonious Strategy において，開示ポリシの記述に論理和が含まれると，証明書要求の際に証明書開示手順に分岐ができてしまい，結果として解となる証明書開示手順を求めるのに必要となる開示ポリシ以外の開示ポリシが開示される．たとえば，$R \leftarrow (C_3 \wedge C_4) \vee C_6$ という開示ポリシを用いて交渉を行った場合を考える．R が開示されるために，Parsimonious Strategy の場合は $(C_3 \wedge C_4)$ と C_6 という 2 つの条件を持つポリシであることが相手に知られてしまう．BPPM/AHES の場合，解に必要となるどちらか一方の条件のみを開示することになるので，交渉に不要なもう一方の条件が相手に知られることはない．

4. REST による BPPM/AHES の実装

4.1 プロトコル

本研究では，最近の主な Web アプリケーションによくみられる，REST（REpresentational State Transfer）方式 [14] に基づく HTTP 通信を用いて実装を行った．

HTTP 通信下で交渉を行うにあたって，以下の点に注意する必要がある．HTTP 通信は，クライアントが要求を行い，サーバが返信するという 1 方向の通信であるが，自動トラスト交渉をするにあたって，二者が必要に応じて双方向に要求を出せる構造にしなければならない．またステートレスな通信方法であるため，何らかの方法で現在の交渉状態を保持しておけるようにする必要がある．

本研究では上記の課題を解決するために，次のような対策をとった．

(a) サーバ側では接続ごとに現在の交渉状況を保管し，手

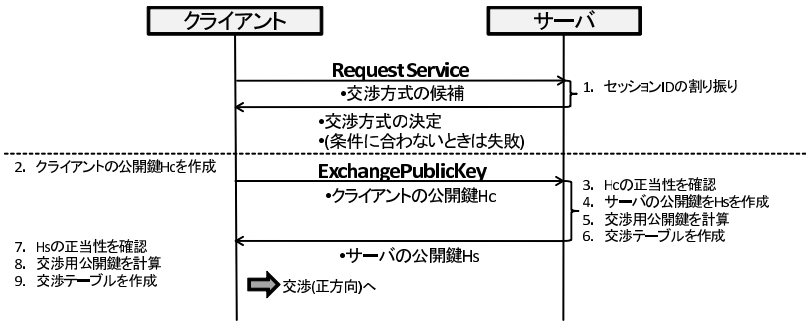

図 2 ハンドシェイク・鍵交換プロトコル
Fig. 2 Handshake and key exchange protocol.

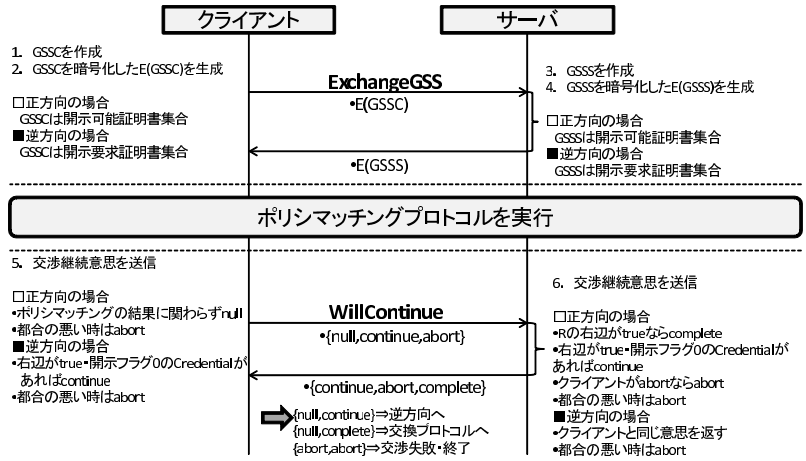

図 3 BPPM/AHES 交渉プロトコル
Fig. 3 BPPM/AHES negotiation protocol.

順番号を用いてサーバとクライアント間で整合をとる．

(b) サーバはクライアントに対して識別番号を割り振り，接続のたびに確認を行う．

(c) サーバが要求を出すべき状況になった場合，クライアントが要求を尋ねることによって，擬似的に双方向通信を行う．

以上の条件を満たすために，専用のプロトコルを考案した．

HTTP 用の BPPM/AHES プロトコルは，ハンドシェイク，鍵交換，BPPM/AHES 交渉，証明書交換の 4 つのプロトコルで構成され，これらを順に実行する．ハンドシェイクプロトコルと鍵交換プロトコルの流れを**図 2** に示す．ここでは，交渉に使用する証明書交換手順の算出方法と，識別番号の割振りおよび鍵交換を行う．BPPM/AHES 交渉プロトコルの流れを**図 3** に示す．ここでは，正方向ポリシマッチングと逆方向ポリシマッチングを繰り返し行い，交渉表を更新しながら交渉を行う．また図中の中央に記載されているポリシマッチングプロトコルについては**図 4** に示す．ポリシマッチングプロトコルは 2.3.1 項で示したポリシマッチング回路に生成部分集合を入力し，解が得られるまでのプロトコルである．手順 1 から手順 10 は，計算中に Conditinal Gate が出現するたびに，繰り返し実行される．この作業によって Conditinal Gate を用いた演算が実行され，最終的に $E(xs)$，$E(ys)$ から $E(xsys)$ が，$E(xc)$，$E(yc)$ から $E(xcyc)$ が導出される．なお，図中の復号処理に関しては 2.2.2 項に記述した閾値復号を行う．証明書交換プロトコルの流れを**図 5** に示す．ここでは，BPPM/AHES 交渉プロトコルによって求めた証明書交換手順の復号を行い，それに従って実際に証明書を交換する．

4.2 RATN システム

あるサービスが，クライアントとサーバを持つアプリケーションで提供されているとする．RATN システムはこのようなアプリケーションが ATN を取り入れる際の基盤となることを想定している．

交渉が必要となる場合には，サービス用のクライアントが RATN クライアントに対して交渉要求を出す．要求を受けると，RATN クライアントは証明書と開示ポリシを用いて RATN サーバと交渉を行う．交渉が成立すると，サービス用のクライアントはサービス用のサーバを利用できるようになる．

開発には，Oracle Corporation の Java Standard Edition Development Kit 6（version 1.6.0 update 19）を使用し

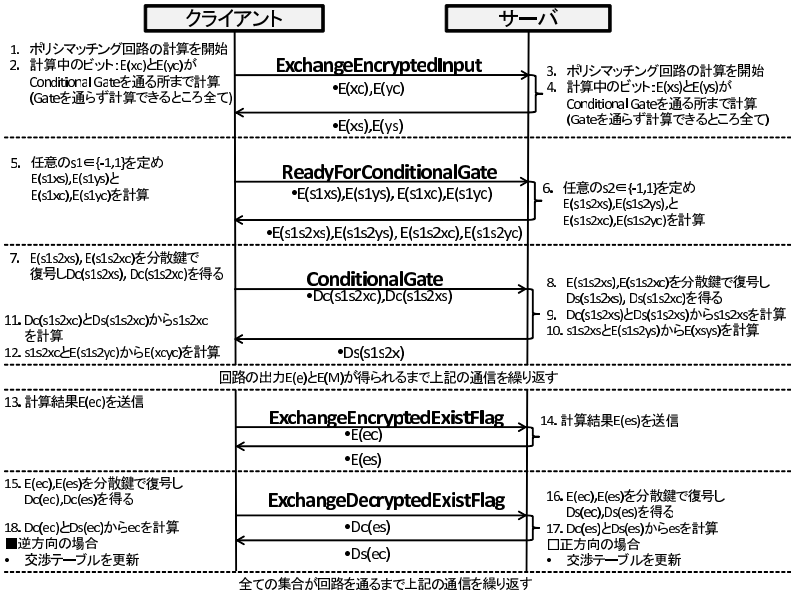

図 4 ポリシマッチングプロトコル
Fig. 4 Policy matching protocol.

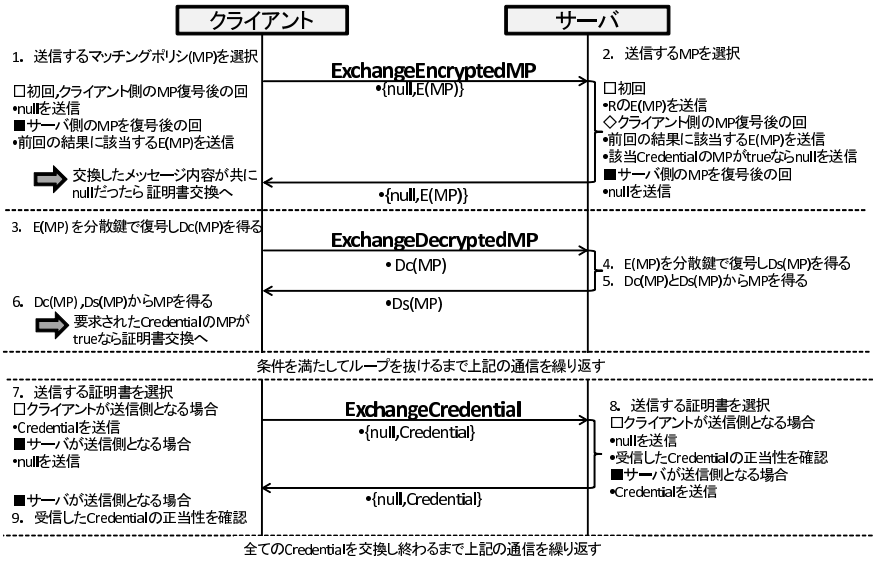

図 5 証明書交換プロトコル
Fig. 5 Credential exchange protocol.

た．REST のインタフェース用のライブラリとして，Noelios Technologies の RESTlet（version 2.0.4）を使用した．BPPM/AHES 用の El Gamal 暗号鍵作成のためのライブラリとして Bouncy Castle（version 1.4.1）を使用した．

5. 性能評価

5.1 RATN システムを用いた交渉実験

実装した RATN システムで BPPM/AHES プロトコルを用いて ATN を行い，実行時間を計測した．実行に使用した環境を以下に示す．クライアントは，OS が Windows 7 Professional (32 bit)，CPU が Intel(R) Core(TM)2 Duo L9400 1.87 GHz，メモリが 2.96 GB の計算機上で実行した．サーバは，OS が Windows 7 Professional (32 bit)，CPU が Intel(R) Core(TM)2 Duo E8500 3.16 GHz，メモリが 2.00 GB の計算機上で実行した．

表 8 サーバ，クライアントの開示ポリシの記述例
Table 8 Example of Server's and Client's policy.

クライアントの開示ポリシ	サーバの開示ポリシ
$C_1 \leftarrow true$	$R \leftarrow C_1 \wedge C_2$
$C_2 \leftarrow S_1$	$S_1 \leftarrow C_1 \vee C_3$
$C_3 \leftarrow S_1$	$S_2 \leftarrow C_1$

以下の各状況において RATN システムを用いて交渉を行い，交渉に要した時間と通信回数を計測した．交渉にはすべて**表 8** の開示ポリシを使用した．

(i) 生成部分集合のサイズを 4 ビットにした場合

(ii) 生成部分集合のサイズを 5 ビットにした場合
(iii) 生成部分集合のサイズを 50 ビットにした場合

実行時間の測定結果を**表 9** に示す．表中の通信時間は通信に関する処理を行った時間を示し，計算時間は通信時間を除いた，暗号処理を含むすべての処理を行った時間を示している．合計時間は，通信時間と計算時間を合わせたもので，合計時間の合計は交渉の全過程に要した時間を示している．また表中の交渉とは，鍵交換と BPPM/AHES 交渉の 2 つを合わせた処理を示している．

5.2 メモリ計算量

BPPM/AHES のメモリ計算量を以下に示す．クライアントの開示ポリシの生成部分集合の最大値を m_c とし，サーバの開示ポリシの生成部分集合の最大値を m_s とする．クライアントが所持する証明書の数を n_c とし，サーバが所持する証明書の数を n_s とする．

定理 5.1. *BPPM/AHES* のクライアント側のメモリ計算量は，$O(n_c^2+n_s^2+n_cn_s(m_c+m_s))$ である．*BPPM/AHES* のサーバ側のメモリ計算量は，$O(n_c^2+n_s^2+n_cn_s(m_c+m_s))$ である．

Proof. まず，クライアント側のメモリ計算量について考える．文献 [15] より，クライアントの開示ポリシを表現するためのメモリ計算量は $O(m_c(n_c+n_s))$ である．交渉表の開示フラグ，MP（マッチングポリシ）を記憶しておくためのメモリ計算量はそれぞれ $O(n_c)$，$O(n_cn_s)$ である．クライアントは交渉計算のためにすべての生成部分集合の集合を記憶する必要がある．正方向ポリシマッチングにおいてクライアントが生成する生成部分集合の集合のサイズは最大で n_c^2 である．正方向ポリシマッチングにおいてサーバが生成する生成部分集合の集合のサイズは最大で $n_cn_sm_s$ である．逆方向ポリシマッチングにおいてクライアントが生成する生成部分集合のサイズは最大で $n_cn_sm_c$ である．逆方向ポリシマッチングにおいてサーバが生成する生成部分集合の集合のサイズは最大で n_s^2 である．合計の生成部分集合の集合のサイズは最大で $n_c^2+n_cn_sm_s+n_cn_sm_c+n_s^2 = O(n_c^2+n_s^2+n_cn_s(m_c+m_s))$ である．証明書交換プロトコルにおいて，クライアントはサーバが開示した証明書を記憶しておく必要があり，そのためのメモリ量は証明書の数 n_s に依存する．よって，合計のメモリ消費量は最大で $(m_c(n_c+c_s))+n_c+n_cn_s+(n_c^2+n_cn_sm_s+n_cn_sm_c+n_s^2)+n_2$ である．したがって，クライアント側のメモリ計算量は $O(n_c^2+n_s^2+n_cn_s(m_c+m_s))$ である．同様にして，サーバ側のメモリ計算量は $O(n_c^2+n_s^2+n_cn_s(m_c+m_s))$ である． □

5.3 他方式との比較

本研究と同様に，暗号を用いて必要のない開示ポリシや証明書の開示を抑制する手法として文献 [16] が存在する．BPPM/AHES と文献 [16] の手法は，交渉を行う両者が開示可能な証明書や開示ポリシを暗号化した状態で見せあい，最終的に交渉が成立しうる解を求めるという点で共通している．ただし，文献 [16] の手法では最終的に満たすべ

表 9 交渉にかかった時間と通信回数
Table 9 The length of time and the number of HTTP sessions for negotiation.

		通信時間（ms）	計算時間（ms）	合計時間（ms）	通信回数（回）
	ハンドシェイク	530	702	1,232	1
	交渉	21,747	100,667	122,414	157
i	証明書交換	2,985	4,743	7,728	14
	合計	25,262	106,202	131,374	172
	全体に占める割合	0.192	0.808	-	-
			一回の通信あたりにかかった時間（ms/回）		764
		通信時間（ms）	計算時間（ms）	合計時間（ms）	通信回数（回）
	ハンドシェイク	530	702	1,232	1
	交渉	27,193	125,235	152,428	193
ii	証明書交換	2,323	7,583	9,906	14
	合計	30,046	133,520	163,566	208
	全体に占める割合	0.184	0.816	-	-
			一回の通信あたりにかかった時間（ms/回）		786
		通信時間（ms）	計算時間（ms）	合計時間（ms）	通信回数（回）
	ハンドシェイク	622	719	1,341	1
	交渉	231,781	5,357,629	5,589,410	1,813
iii	証明書交換	2,380	6,371	8,751	14
	合計	234,783	5,364,719	5,599,502	1,828
	全体に占める割合	0.0419	0.958	-	-
			一回の通信あたりにかかった時間（ms/回）		3,063

き R の開示ポリシから開始するトップダウンの探索を行うのに対し，BPPM/AHES では，現在開示可能な証明書によって開示可能な開示ポリシから順にボトムアップの探索を行っている．

このとき，メモリ計算量はクライアントの場合 $O(n_c^2 + n_s^2 + n_c n_s(m_c + m_s))$ となるが，m_c，m_s の値は証明書数 n_c，n_s の増加によって指数的に増加する．これは文献 [16] の手法の暗号文の計算量 $O((m/k)^{kh})$（$m = n_s = n_c$，k は次数，h は木の高さ）と比較しても，効率の面から見れば良くなったとはいえない．

また文献 [16] の手法では二分木に変換可能な開示ポリシを用いているのに対し，BPPM/AHES では積和標準形で記述可能なすべての論理式の開示ポリシを扱うため，より幅広い開示ポリシに対応することが可能である．

5.4 実用化への課題

BPPM/AHES は，暗号処理の都合上，サーバ・クライアント間の通信や計算に非常に時間がかかるプロトコルである．5.1 節から，特に生成部分集合のサイズが大きくなると，計算時間が増大し，パフォーマンスが低下することが明確になった．ポリシマッチングの仕様上，生成部分集合のサイズは，サーバとクライアントが所持する最大の証明書数より大きくなければならない．ゆえに，生成部分集合のサイズを減少させてパフォーマンスの向上を図ることは適切でない．

RATN におけるポリシマッチングの計算は，クライアントの計算が終了してから通信を行い，次にサーバが計算を行うという逐次実行である．パフォーマンスの向上を図るならば，サーバとクライアントのポリシマッチングの計算を，できる限り並行して行うべきである．

またサービスを利用したいときにすぐ利用できる状態にするには，事前にオフラインでの交渉を行っておくという方法がある．たとえばネットワーク上に代理プログラムを置き，自動的に交渉を行わせるという方法が考えられる．その際，ネットワーク上に証明書や開示ポリシを置いておく必要があるが，クラウドコンピューティングが普及した今日，自らのデータをネットワーク上に保管することは一般的となっており，現実的な実現方法であるといえる．

そのほかにも，ATN を用いた手法の本質的な問題点として，メモリ使用量が膨大になるということが文献 [15] でも述べられている．BPPM/AHES のメモリ計算量は 5.2 節で述べたとおりとなっており，生成部分集合の数や開示ポリシの数が増えると，メモリの処理能力を上回ってしまい，正常に動作しない恐れがある．実用化する場合は，開示ポリシの数を制限するなどの配慮が必要となる．

6. おわりに

本稿では，ATN における証明書交換手順の算出方法として BPPM/AHES を提案し，ATN 基盤への実装および評価を行った．実装した ATN 基盤，RATN は REST 方式に基づく HTTP 通信下で交渉を行うよう設計を行い，その専用プロトコルを考案した．

BPPM/AHES では，暗号化したままの状態での演算が可能な準同型暗号系を用いて交渉計算を行うポリシマッチングを正方向と逆方向に分けて定義し，2 つのポリシマッチングを繰り返し行うことにより，単方向ポリシマッチングでは実現できなかった双方向の自動トラスト交渉を実現した．本手法では，暗号化したままの状態で計算を行うことで，従来の手法の問題点であった交渉の成否決定前における証明書および開示ポリシの開示がなくなった．BPPM/AHES を実装した RATN システムを用いた実験により，El Gamal 暗号の計算時間や通信回数の面から交渉に要する時間が非常に長いことが判明し，実用性には課題が残った．

参考文献

[1] Winsborough, W., Seamons, K. and Jones, V.: Automated trust negotiation, *Proc. DARPA Information Survivability Conference and Exposition, 2000, DISCEX'00*, Vol.1, pp.88–102 (2000).

[2] Katugampala, I.，八槇博史，山口由紀子：Web サービス標準を用いた automated trust negotiation 基盤の開発，第 7 回情報科学技術フォーラム（FIT2008），第 4 分冊，pp.251–252 (2008).

[3] Lee, A., Winslett, M. and Perano, K.: Trustbuilder2: A reconfigurable framework for trust negotiation, *Trust Management III*, pp.176–195 (2009).

[4] Yarnaki, H., Fujii, M., Nakatsuka, K. and Ishida, T.: A dynamic programming approach to automated trust negotiation for multiagent systems, *Rational, Robust and Secure Negotiation Mechanisms in Multi-Agent Systems*, pp.55–65 (2005).

[5] Yu, T., Ma, X. and Winslett, M.: Prunes: An efficient and complete strategy for automated trust negotiation over the internet, *Proc. 7th ACM Conference on Computer and Communications Security*, pp.210–219 (2000).

[6] 中塚康介，石田　亨：分散 AND/OR 探索による Automated Trust Negotiation，情報処理学会論文誌，Vol.47, No.8, pp.2454–3463 (2006).

[7] Baselice, S., Bonatti, P. and Faella, M.: On interoperable trust negotiation strategies, *8th IEEE International Workshop on Policies for Distributed Systems and Networks, 2007, POLICY'07*, pp.39–50 (2007).

[8] Yu, T., Winslett, M. and Seamons, K.E.: Interoperable strategies in automated trust negotiation, *Proc. 8th ACM Conference on Computer and Communications Security, CCS'01*, pp.146–155, ACM, New York, NY, USA (2001).

[9] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms, *Advances in Cryptology*, pp.10–18 (1985).

[10] Desmedt, Y. and Frankel, Y.: Threshold cryptosystems, *Proc. Advances in Cryptology-CRYPTO'89*, pp.307–315 (1990).

[11] Kursawe, K., Neven, G. and Tuyls, P.: Private policy

negotiation, *Financial Cryptography and Data Security*, pp.81–95 (2006).
[12] Schoenmakers, B. and Tuyls, P.: Practical two-party computation based on the conditional gate, *Advances in Cryptology-ASIACRYPT 2004*, pp.129–145 (2004).
[13] Gennaro, R., Jarecki, S., Krawczyk, H. and Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems, *Advances in Cryptology-EUROCRYPT'99*, pp.925–963 (1999).
[14] Fielding, R.: Architectural Styles and the Dsign of Network-Based Software Architectures, Ph.D. thesis, University of California, Irvine (2000).
[15] Katugampala, I., Yamaki, H. and Yamaguchi, Y.: Memory Complexity of Automated Trust Negotiation Strategies, *Principles of Practice in Multi-Agent Systems*, pp.229–244 (2009).
[16] Tangtisanon, P. and Kikuchi, H.: Perfect privacy-preserving automated trust negotiation, 情報処理学会論文誌, Vol.52, No.9, pp.2697–2708 (2011).

青山 桃子

昭和 63 年生．平成 23 年名古屋大学工学部電気電子・情報工学科卒業．同年名古屋大学大学院情報科学研究科情報システム学専攻博士課程前期課程入学，在学中．トラスト形成に関する研究に従事．

森 文宏

昭和 61 年生．平成 22 年名古屋大学工学部電気電子・情報工学科卒業．同年名古屋大学大学院情報科学研究科情報システム学専攻博士課程前期課程入学．トラスト形成に関する研究に従事．平成 24 年名古屋大学大学院情報科学研究科情報システム学専攻博士課程前期課程修了．同年三菱電機メカトロニクスソフトウェア（株）入社．

八槇 博史 （正会員）

昭和 48 年生．平成 11 年京都大学大学院工学研究科情報工学専攻博士後期課程修了．博士（情報学）を取得．同年より京都大学大学院情報学研究科助手．平成 13 年より同研究科講師．平成 18 年より名古屋大学情報基盤センター准教授．ネットワーク応用，マルチエージェントシステム，ユビキタスコンピューティングに関する研究に従事．電子情報通信学会，人工知能学会，ACM，IEEE 各会員．