RL-006

# BPPM/AHES に基づく自動トラスト交渉基盤の開発
# An Automated Trust Negotiation Framework based on BPPM/AHES

**青山桃子** † **森文宏** † **八槇博史** ‡
Momoko Aoyama Fumihiro Mori Hirofumi Yamaki

## 1. はじめに

サービスの利用者と提供者が互いに未知である場合に，互いの信頼を確立することをトラスト形成という．インターネットを介してサービスを利用または提供する場合には，サービスの利用者と提供者の間でトラスト形成を行う必要がある．

インターネット上でトラスト形成を行う仕組みとして，自動トラスト交渉（ATN:Automated Trust Negotiation） [8] が提案されている．本研究では，通信に REST（REpresentational State Transfer）方式 [3] を採用した ATN 基盤, RATN（Restful Automated Trust Negotiation）の開発を行った．

ATN によってトラスト形成を自動化するためには，ポリシに反することなく証明書を交換する手順の算出方法が必要となる．ATN における証明書交換手順の基本的な算出方法である Eager Strategy [8] や Parsimonious Strategy [8] を実装した例としては，WS-ATN [4] や Trust Builder2 [1] などが存在する．本研究では既存の算出方法の中でも，優れた特徴を持つ BPPM/AHES（Bidirectional Private Policy Matching based on Additively Homomorphic Encryption Systems） [9] を使用した．

BPPM/AHES では，ポリシや開示可能な証明書の情報を，加法的準同型性を持つ ElGamal 暗号を用いて暗号化し，その状態のままでポリシを満たすかどうかの判定を行う．このため，情報の開示を必要最低限に抑えることができる．RATN では，算出方法として BPPM/AHES のみを実装しているが，その他の方法を容易に追加できる構造になっている．

## 2.Automated Trust Negotiation

ATN とは，サービスの利用者と提供者が互いの証明書を，証明書の開示条件であるポリシに反することなく交換して信頼を確立する作業を，計算機によって自動で行うための仕組みである．

### 2.1.ATN の定義

本論文では，[7] に示されている記述形式を用いて，ATN を定義する．

ATN における交渉とは，サービスの利用者と提供者が，互いのポリシに反することなく証明書を交換し，目的とするリソースを得る為の過程を指す．ここでは交渉によって最終的に得られるリソースをサービス，サービスの利用者をクライアント，提供者をサーバと呼ぶこととする．このとき，サービスを R，クライアントの証明書を $C1, C2, ..., Cn_c$，サーバの証明書を $S1, S2, ..., Sn_s$ と記述する．ただし $n_c$ はクライアントの持つ証明書数，$n_s$ はサーバの持つ証明書数を示す．証明書 $C$ を開示するためのポリシは $C \leftarrow F_c(S1, S2, ..., Sk)$ と記述する．ポリシの右辺となる $F_c(S1, S2, ..., Sk)$ は，相手が開示しなければならない証明書 $S1, S2, ..., Sk$ と論理記号 $\wedge$ および $\vee$，必要に応じた括弧 ( および ) からなる論理式である．相手が既に開示した証明書を示す述語記号に $true$ を代入した後，論理式 $F_c(S1, S2, ..., Sk)$ が $true$ になるなら，証明書 $C$ は開示可能であるという．

証明書 C のポリシは積和標準形で $C \leftarrow D_1 \vee ... \vee D_l$ と記述できる．ただし $D_i = S_{i1} \wedge ... \wedge S_{ik_i}$ で，$S_{ij}(1 \leq i \leq l, 1 \leq j \leq k_i)$ は相手が開示しなければならない証明書である．上記のような $Di$ は項と呼ぶ．相手が開示した証明書を表す述語記号に $true$ を代入した際，$D_1 \vee ... \vee D_l$ のうち少なくとも一つの項が $true$ であるなら $C$ のポリシを $C \leftarrow true$ と記述する．一方，証明書 $C$ がどんな場合でも開示できないのであれば，$C$ のポリシを $C \leftarrow false$ と記述する．

サービス自体は実際の証明書ではないが，サービスのアクセス制御ポリシを上記の方法と同様に記述出来る．交渉の結果として，サービス $R$ が開示可能になるなら，クライアントがサービスを利用可能になり，交渉が成功したという．そうでなければ交渉が失敗したという．

### 2.2.ATN の例

インターネット上であるレンタルサービスが提供されているとする．サービス利用者である Alice は，免許証とそれを開示するためのポリシを持っている．サービス提供者であるレンタル店は，企業証明書と提供するレンタルサービス，それらを開示するためのポリシを持っている．このとき，$C1$ = 免許証，$C2$ = 保険証，$S1$ = 企業証明書，$R$ = レンタルサービスとすると，ポリシは表 1 のように表すことができる．これを用いて，図 1 に示す手順でトラスト形成を行う．

表 1: Alice とレンタル店が持つ証明書とポリシ

| Alice のポリシ | レンタル店のポリシ |
|---|---|
| $C1 \leftarrow S1$ | $R \leftarrow C1 \vee C2$<br>$S1 \leftarrow true$ |

ATN を用いることで，このような任意の証明書とポリシの組み合わせの整合性を自動的に判定することができる．ATN によってトラスト形成を自動化するためには，ポリシに反することなく証明書を交換する手順の算出方法が必要となる．本論文では，証明書交換手順の算出方法のことをプロトコルと記述する．プロトコルは，[8] や [7] といったいくつかの論文で提案されている．

†名古屋大学大学院情報科学研究科
‡名古屋大学情報基盤センター

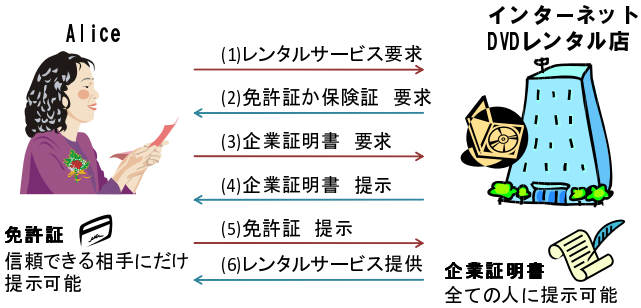


図 1: トラスト交渉の例

### 2.3. ポリシマッチング

既存のプロトコルとして，Kursawe ら [5] の提案した Private Policy Negotiation が存在する．Private Policy Negotiation ではポリシマッチングという手法を用いて交渉を行う．

ATN では証明書やポリシといった個人情報を扱うため，トラスト形成に必要のない証明書やポリシの開示が無いことが望ましく，これを privacy preserving であるという．ポリシマッチングを用いると，privacy preserving に交渉を行うことができる．以下にその概要を述べる．

ポリシマッチングとは，一方が開示してもよいと考える証明書と，その交渉相手が開示を要求する証明書が一致するかどうかを見つけ出すというプロセスである．このポリシマッチングを行った結果，上記が一致したときの証明書集合をマッチングポリシと呼ぶ．ATN ではクライアント，サーバともに開示ポリシを持つ双方向のトラスト形成を考えるが，Kursawe らによる手法ではサーバ側のみが開示ポリシ（SGP）を持つ単方向の状況を前提としている．

計算には図 2 のポリシマッチング回路を用いる．ポリシマッチング回路の入力は，開示不可能証明書集合 $\mathbb{F} = \{F_1, ..., F_a\}$ と，開示要求証明書集合 $\mathbb{G} = \{G_1, ..., G_b\}$ である．$\mathbb{F}$ の要素である $F_1, ..., F_a$ は，開示不可能である証明書がどれであるかを示す集合である．また $\mathbb{G}$ の要素である $G_1, ..., G_b$ は，積和標準系で表現したポリシの項 $D_1, ..., D_b$ を示す集合であり，これらを生成部分集合という．出力は，ポリシを満たす条件が見つかるかどうかのフラグ $e$ と，満たされた条件（マッチングポリシ）$M$ となる．

加法的準同型性を持つ ElGamal 暗号 [6] を用いると，入力を暗号化した状態のままで，ポリシマッチング回路の出力を得ることができる．この場合，回路中の計算には Private Multiplier Gate と Conditional Gate [2] を用いる．なお，Conditional Gate の通過には複数回の通信を伴う．暗号化した状態での AND，OR，NOT 回路は以下のように定義されている．

**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．

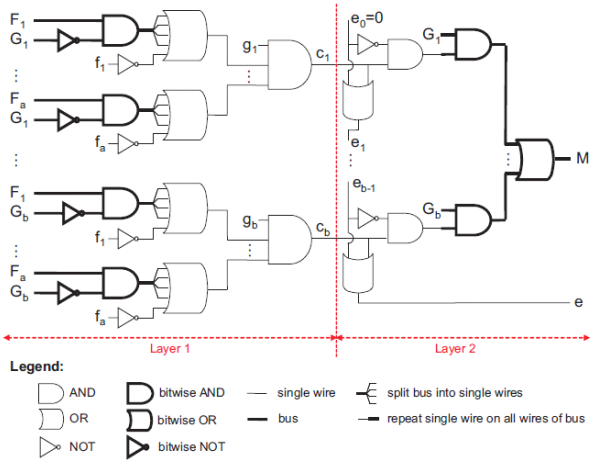


図 2: ポリシマッチング回路（[5] より引用）

**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．

### 2.4.BPPM/AHES

BPPM/AHES [9] は，Private Policy Negotiation を双方向のトラスト形成を行うように拡張したプロトコルである．

サーバ，クライアント間でポリシマッチングを繰り返し行い，開示と要求の役割を両者が交互に担うことで，双方向に信頼を確立することができる．両者はポリシマッチングの結果によって各自の持つ交渉表を更新し，更なる交渉の余地があった場合は役割を交代して，再度ポリシマッチングを行う．クライアントが開示，サーバが要求を行う時のポリシマッチングを正方向ポリシマッチングと呼び，クライアントが要求，サーバが開示を行う時のポリシマッチングを逆方向ポリシマッチングと呼ぶ．

ポリシマッチングは双方向に拡張をしても，privacy preserving であるという性質は失われない．また，証明書の交換は交渉が成立すると判明した後なので，交渉が失敗した場合には証明書は一切，開示されない．具

体的な交渉の手順については，3 章で REST による実装とともに後述する．

## 3.REST による BPPM/AHES

本研究では，既存のサービスアプリケーションとの親和性が高い REST（REpresentational State Transfer）方式 [3] を通信に用いることを前提として開発を行った．

**課題** BPPM/AHES を REST 方式のシステムとして実装するためには，次のような課題がある．

(a). BPPM/AHES では交渉を何度かの通信に分けて行うため，サーバは現在の状況を何らかの形で次の通信の時に取得できるようにしなければならない．

(b). 同時に複数の利用者がサーバを利用する場合を考え，その利用者を区別する方法を用意しておかなければならない．

(c). REST はクライアントが要求を行い，サーバが返信するという一方向の通信であるが，トラスト形成をするにあたって，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．

**対策** 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). サーバ側では接続毎に現在の交渉状況を保管し，手順番号を用いてサーバとクライアント間で整合性を取る．

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

(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事によって，擬似的に双方向通信を行う．

(c) を満たす為に，専用のプロトコルを考案した．本章ではそのプロトコルの詳細について記述する．なお，(a)，(b) を満たす実装についても以下のプロトコル内で言及している．

REST 用の BPPM/AHES プロトコルは，大きく分けて以下の四つの部分を順に実行している．

1. ハンドシェイク
2. 鍵交換
3. BPPM/AHES 交渉
4. 証明書交換

### 3.1. ハンドシェイクプロトコル

プロトコルの流れを図 3 の上半分に示す．`RequestService` メッセージによって，交渉に使用する証明書交換手順の計算手法の決定と，識別番号の割り振りを行う（図 3 の 1）．ここで割り振られた識別番号は，通信の度に現在の交渉状況を示す手順番号と一緒に渡され，交渉中にサーバが通信相手を識別するのに使用される．これは課題 (a)，(b) の対策である．

### 3.2. 鍵交換プロトコル

プロトコルの流れを図 3 の下半分に示す．`ExchangePublicKey` メッセージによって，BPPM/AHES で使用する ElGamal 暗号鍵の作成を行う（図 3 の 2 から 9）．サーバとクライアントで同じパラメータを使用した ElGamal 暗号鍵を作成し，それらを利用した交渉用公開鍵を作成する．交渉用公開鍵はサーバとクライアントで共通である．それぞれが作成した ElGamal 暗号鍵は，交渉用公開鍵の分散鍵となる．

### 3.3.BPPM/AHES 交渉プロトコル

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．プロトコルの流れを図 4 に示す．

まずは `ExchangeGSS` メッセージによって，ポリシマッチングに使用する互いの生成部分集合の集合を交換する（図 4 の 1 から 4）．ここでポリシマッチングを行い，結果が得られたら `WillContinue` メッセージによって互いの交渉継続意思を確認する（図 4 の 5 から 6）．その結果，交渉を継続する場合は，再び図 4 の手順に従って反対方向のポリシマッチングを行う．交渉が成立すると判明した場合には，証明書交換プロトコルへ進む．交渉が成立し得ないと判明した場合には，交渉失敗となりプロトコルを終了する．

ポリシマッチング中のプロトコルを図 5 に示す．ポリシマッチングは 2.3 節に示した手法に従って行う．この際，回路中の AND ゲートと OR ゲートを通過するためには Conditional Gate の通過が必要となる．Conditional Gate の通過が必要となった時，サーバとクライアントは `ExchangeEncryptedInput` メッセージによって，Conditional Gate を通したい入力を互いに交換する（図 5 の 1 から 4）．その後，`ReadyForConditionalGate` メッセージと `ConditionalGate` メッセージによって，互いの入力を Conditional Gate に通した結果を計算する（図 5 の 5 から 12）．

上記の通信を繰り返し，ポリシマッチング回路を通過すると，サーバとクライアントはそれぞれ出力 $E(e)$ と $E(M)$ を得る．$E(e)$ と $E(M)$ はポリシマッチング回路の出力が暗号化されたものである．両者は `ExchangeEncryptedExistFlag` メッセージによって，それぞれの $E(e)$ を交換する（図 5 の 13 から 14）．その後，`ExchangeDecryptedExistFlag` メッセージによって，$E(e)$ を復号する（図 5 の 15 から 18）．両者は $e$ の値によって，適宜交渉表を更新し，与えられた全ての生成部分集合が回路を通過するまで，図 5 の手順に従って計算を行う．

### 3.4. 証明書交換プロトコル

証明書交換プロトコルでは，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．

プロトコルの流れを図 6 に示す．

まず，`ExchangeEncryptedMP` メッセージによって，復号したいマッチングポリシを送信する．この際，マッチングポリシはどちらか片方だけが送信する（図 6 の

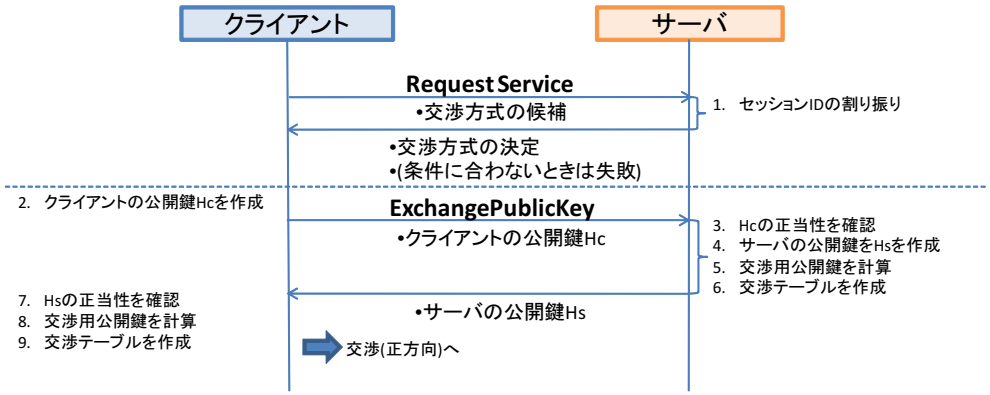


図 3: ハンドシェイク・鍵交換プロトコル

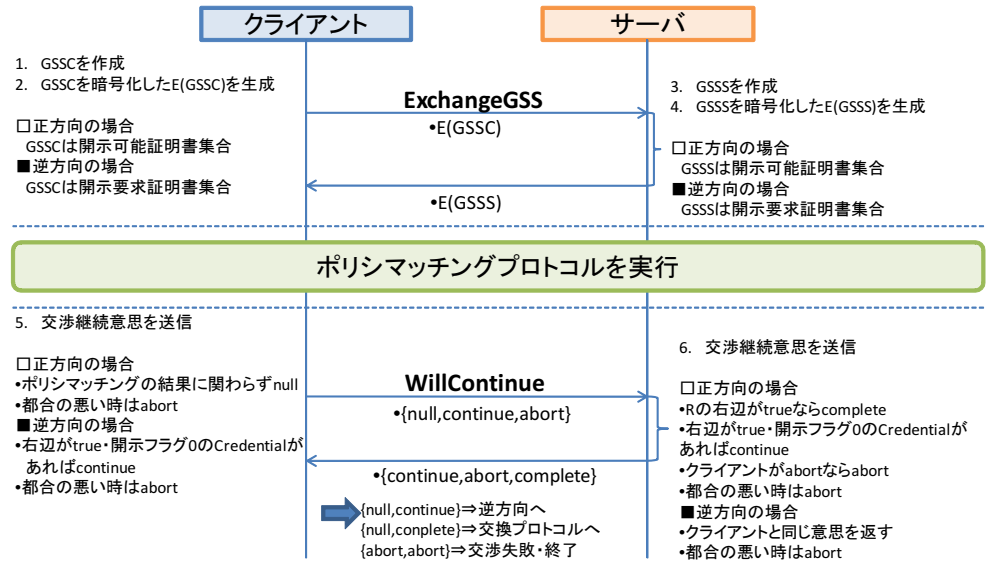


図 4: BPPM/AHES 交渉プロトコル

1 から 2)．最初はサーバがサービスリソース $R$ のマッチングポリシを送信し，その後は復号結果に従って交互に送信する．次に ExchangeDecryptedMP メッセージによってマッチングポリシを復号する（図 6 の 3 から 6)．上記の通信を繰り返すと，最適な証明書交換手順が復号される．

復号した証明書交換手順に従い，Exchange Credential メッセージによって，該当する証明書を交換する（図 6 の 7 から 9)．最後に，クライアントがサービスリソース $R$ を得たところで全ての通信が終了する．

## 4.RATN システム

### 4.1. 概要

あるサービスの利用と提供が，クライアントとサーバを持つアプリケーションで実装されているとする．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．RATN システムの構造とサービスが提供されるまでの流れを，図 7 に示す．

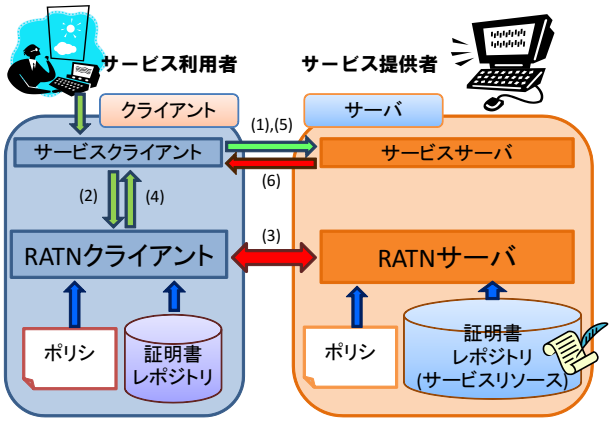


図 7: RATN における ATN の手順

システムは，サーバとクライアントを持つ．サーバには，サービスを提供するアプリケーションのサーバと，ATN を行う RATN サーバが存在する．クライア

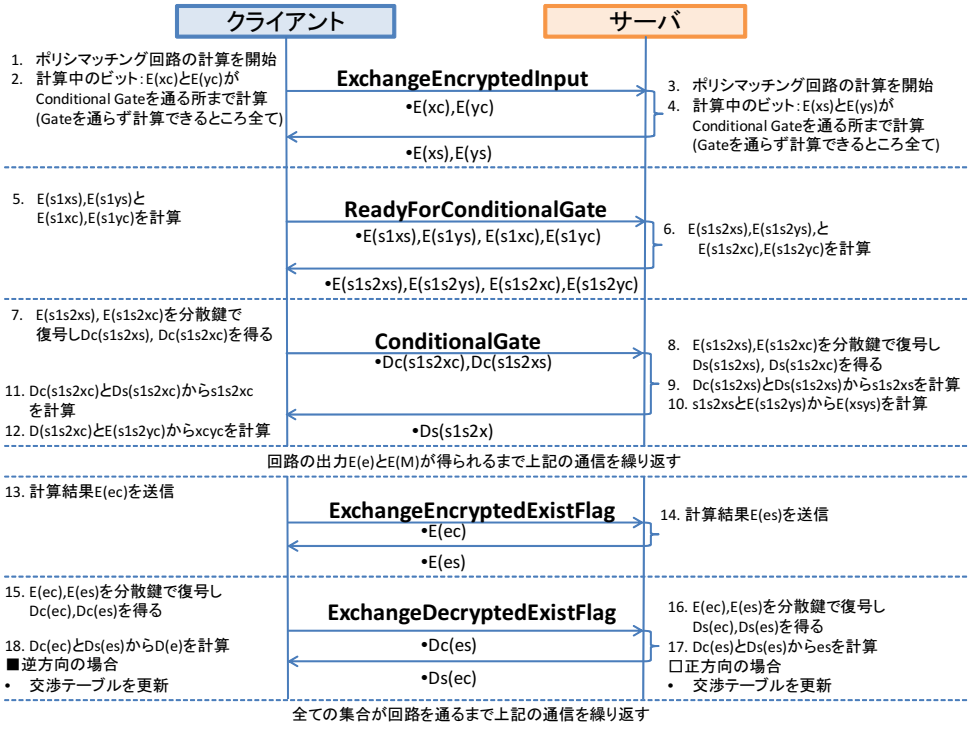


図 5: ポリシマッチングプロトコル

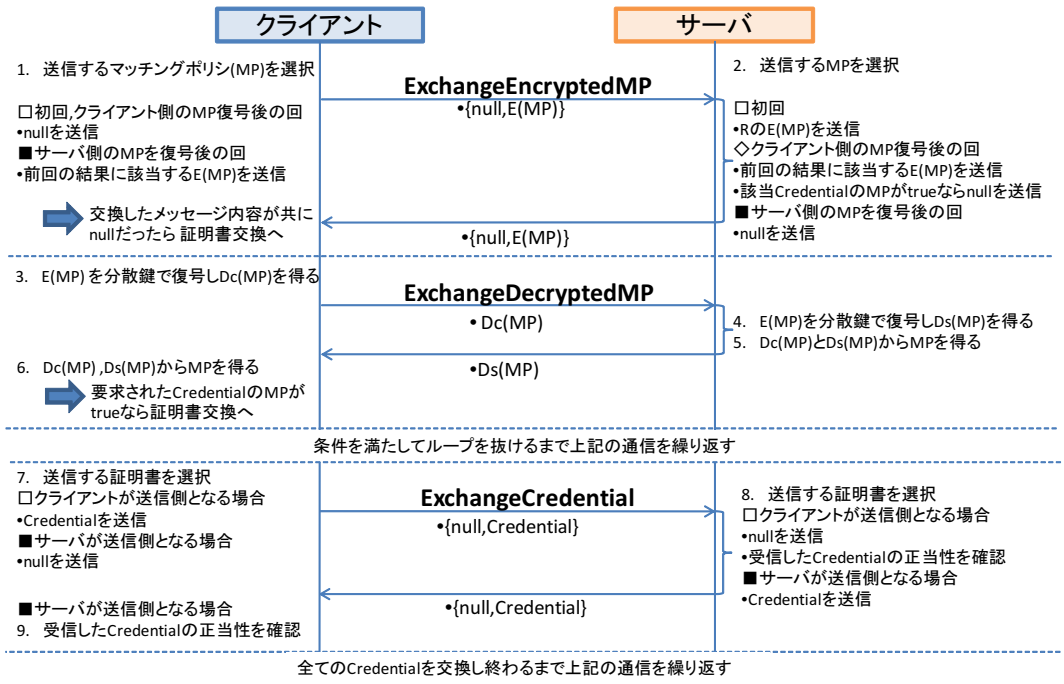


図 6: 証明書交換プロトコル

ントにも同様に，サービスの受け口となるアプリケーションのクライアントと，ATN を行う RATN クライアントが存在する．本研究ではサービスを提供するアプリケーションのサーバのことをサービスサーバ，サービスの受け口となるアプリケーションのクライアントのことをサービスクライアントと呼ぶ．

サービス利用者がクライアントを使用するときにはまずサービスクライアントにアクセスする．サービスクライアントは，サーバ側のサービスサーバにサービスリソースを提示することで，サービスを享受することができる．

サービスクライアントがサービスを利用する条件を

満たしていない場合，サービスサーバは，ATN によってサービスリソースを取得するよう求める（図 7(1)）．このとき，サービスクライアントは RATN クライアントを呼び出し，該当サービスのサービスリソースを取得するよう要求を出す（図 7(2)）．RATN クライアントは RATN サーバとトラスト交渉を行い，成功すれば RATN サーバは RATN クライアントにサービスリソースを渡す（図 7(3)）．RATN クライアントは，得られたサービスリソースをサービスクライアントに渡す（図 7(4)）．このサービスリソースの提示によって，サービスクライアントがサービスを利用する条件が満たされるので，利用者はサービスを受けることができる（図 7(5)，(6)）．サービス用のアプリケーションと RATN は独立しているので，一度サービスリソースを得てしまえば，二度目以降は RATN を呼び出すことなくサービスを受けられるという利点がある．

### 4.2. 実装

開発には，Oracle Corporation の Java Standard Edition Development Kit 6（version 1.6.0 update 19）を使用した．REST のインター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 ElGamal 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．

交渉に利用するポリシを記述したポリシファイルの例を表 2 に示す．ポリシは積和標準形で記述する．ポリシファイルの記述では，一行が一つの証明書とポリシを表現している．証明書に対して割り振られた索引番号を使用して，{ 証明書の索引番号 } : { ポリシ } のように記述する．索引番号はあらかじめ定められており，サーバとクライアントの両者にとって既知である．サーバのポリシファイルでは，証明書の索引番号が 0 である証明書はサービスリソースを指し，そのときのポリシが SGP である．

ポリシは，相手の証明書の索引番号を使用して記述する．ファイル中の& は ∧ を表現し，| は ∨ を表現している．その証明書のポリシが $true$ または $false$ である場合はポリシの部分にそのまま $true$ または $false$ を記述する．

表 2: ポリシファイルの記述例

| ポリシ | ポリシファイルの記述 |
|---|---|
| $R \leftarrow (C3 \wedge C4) \vee C6$ | 0 : 3&4 \| 6 |
| $S1 \leftarrow true$ | 1 : $true$ |
| $S2 \leftarrow (C1 \wedge C2) \vee C3$ | 2 : 1&2 \| 3 |
| $S3 \leftarrow C3 \vee C4$ | 3 : 3 \| 4 |
| $S4 \leftarrow C4$ | 4 : 4 |
| $S5 \leftarrow C1 \wedge C5$ | 5 : 1&5 |

交渉結果はサービスクライアントに用意されている GUI と，コンソールに出力される．ただし，サービスクライアントは提供されるサービスの内容によって様々な形式になることが想定される．動作実験に使用したサービスクライアントを図 8 に示す．RATN クライアントからサービスクライアントに送られる交渉結果は，ハンドシェイクの成否，BPPM/AHES 交渉の成否，証明書交換の成否の三つである．コンソールには交渉結果だけでなく，交渉表の状態や導出された証明書交換手順，実際に交換された証明書の順番も表示される．コンソール出力の例を図 9 に示す．交渉中に交わされるメッセージ例を図 10 に示す．

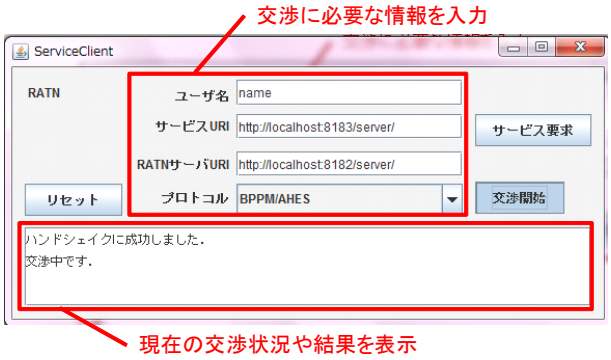


図 8: サービスクライアントの例

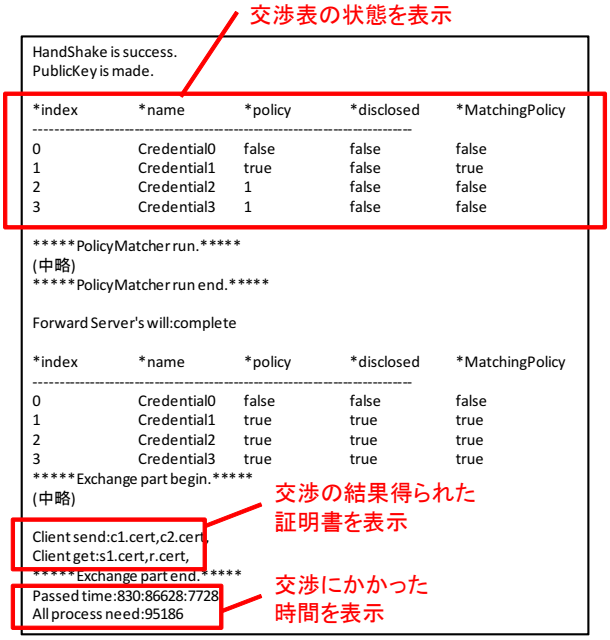


図 9: RATN クライアントの例

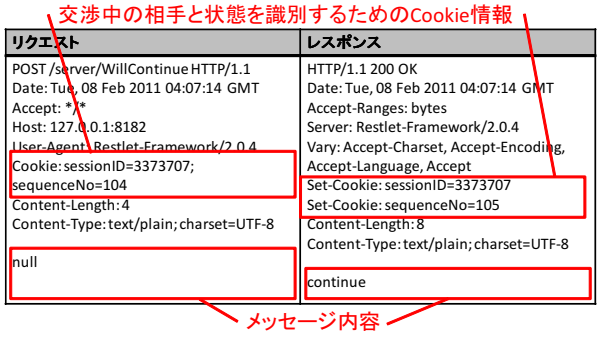


図 10: 実際に交わされるメッセージ例

表 3: RATN システムを実行した環境

| | クライアント |
|---|---|
| OS | Windows 7 Professional(32bit) |
| CPU | Intel(R) Core(TM)2 Duo L9400 1.87GHz |
| メモリ | 2.96GB |

| | サーバ |
|---|---|
| OS | Windows 7 Professional(32bit) |
| CPU | Intel(R) Core(TM)2 Duo E8500 3.16GHz |
| メモリ | 2.00GB |

表 4: サーバとクライアントのポリシ

| クライアントのポリシ | サーバのポリシ |
|---|---|
| $C1 \leftarrow true$ | $R \leftarrow C1 \wedge C2$ |
| $C2 \leftarrow S1$ | $S1 \leftarrow C1 \vee C3$ |
| $C3 \leftarrow S1$ | $S2 \leftarrow C1$ |

### 4.3.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表 3 に示す．

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．交渉には全て表 4 のポリシを使用した．

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 5 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，暗号処理を含む全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 交渉の二つを合わせた処理を示している．

i と ii の結果から分かるように，生成部分集合のサイズを 1 ビット大きくすると，交渉の全過程に要した時間が約 30 秒長くなっている．iii の例では，ポリシマッチングに非常に長い時間を要している．このことから，生成部分集合のサイズはパフォーマンスに大きく影響していると分かる．

また iii のように生成部分集合が大きい時には，全過程における計算時間の割合が約 95% と非常に高いことが分かる．このことから，計算時間の短縮が，パフォーマンスの大きな向上に繋がると考えられる．

## 5. 考察

### 5.1. 既存の ATN 基盤との比較

RATN と，既存の ATN 基盤である WS-ATN および Trust Builder2 との比較を表 6 に示す．比較は，実装に用いた通信方式と実装プロトコルの観点から行った．現在，REST で通信を行うサービスは主流となりつつあるので，RATN はそういったサービスの基盤として導入が容易であるという利点がある．また，REST のサービスは必ずしも独自のクライアントを必要とする訳ではない．その観点から見ても，RATN は多種多様なサービスへの導入が容易である．

実装プロトコルについては，WS-ATN と Trust-Builder2 で用いたプロトコルはほぼ同じである．それらは不要な証明書やポリシを開示してしまうという問題を持っている．それに対して，RATN で実装したプロトコルは不要な情報を一切開示しない．つまり RATN は，他の二つが達成していない privacy preserving を達成している．ただし計算時間の観点から比較すると，他の二つに劣っている．ゆえに，実用化を考えるならば低いパフォーマンスを補う方法を用意する必要がある．

### 5.2. 暗号の処理コスト

4.3 節から，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で行うポリシマッチングの計算を，並行して行うべきである．並行に計算を行うことで，計算時間をほぼ半分に減らすことができる．これを実装する為には，3 章で提案したプロトコルの改善が必要である．

またポリシマッチング回路の計算方法を分析し，更に少ない通信回数で回路を通過することができるプロトコルを考案することができれば，かなりのパフォーマンスの向上が見込める．

### 5.3. 代理によるトラスト交渉

BPPM/AHES はポリシマッチングの仕様上，どうしても通信回数や通信時間が大きくなってしまうという欠点がある．

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．例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言える．

## 6. おわりに

本研究では，BPPM/AHES による自動トラスト交渉を REST 環境で実装する ATN 基盤，RATN システムの開発を行った．ATN 基盤を利用するアプリケーションは，サービスを利用するためのクライアントと，サービスを提供するためのサーバを持つ．RATN サーバと RATN クライアントは，それらのクライアントとサーバに代わって ATN を行う役割を担う．本研究では，まず BPPM/AHES を REST 方式の通信で実装するためのプロトコルを考案した．それに基づき，RATN サー

表 5: 交渉にかかった時間と通信回数

| | | 通信時間 (ms) | 計算時間 (ms) | 合計時間 (ms) | 通信回数 (回) |
|---|---|---|---|---|---|
| i | ハンドシェイク | 530 | 702 | 1,232 | 1 |
| | 交渉 | 21,747 | 100,667 | 122,414 | 157 |
| | 証明書交換 | 2,985 | 4,743 | 7,728 | 14 |
| | 合計 | 24,428 | 109,202 | 133,630 | 172 |
| | 全体に占める割合 | 0.183 | 0.827 | - | - |
| | 一回の通信あたりにかかった時間 (ms/回) | | | | 777 |
| | | 通信時間 (ms) | 計算時間 (ms) | 合計時間 (ms) | 通信回数 (回) |
| ii | ハンドシェイク | 530 | 702 | 1,232 | 1 |
| | 交渉 | 27,193 | 125,235 | 152,428 | 193 |
| | 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) | |
| iii | ハンドシェイク | 622 | 719 | 1,341 | 1 |
| | 交渉 | 231,781 | 5,357,629 | 5,589,410 | 1,813 |
| | 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 |

表 6: 既存の ATN 基盤との比較

| | RATN | WS-ATN | Trust Builder2 |
|---|---|---|---|
| 通信方式 | REST | SOAP | Socket |
| 実装プロトコル | BPPM/AHES | Eager Strategy, Parsimonious Strategy | TrustBuilder1-Relevant Strategy |
| 不要な情報開示 | 無 | 有 | 有 |
| パフォーマンス | × | ○ | ○ |

バと RATN クライアントを実装し，サービス用のアプリケーションがそれらを利用して ATN を行う仕組みを実現した．

今回実装した BPPM/AHES は，加法的準同型性暗号を用いたプロトコルである．このプロトコルは他の手法に比べて，不要な証明書やポリシの開示を抑えることができるという利点がある．しかし，BPPM/AHES は非常に通信回数が多く，計算時間が非常に長くなることが分かった．

今後の課題として，プロトコルの改善によるパフォーマンスの向上や，データをクラウド上にいて代理プログラムによってあらかじめ計算を行っておき，長時間の計算を補うなどの対処方法の実装が挙げられる．

## 参考文献


[1] A.J.Lee, M.Winslett, and K.J.Perano. Trustbuilder2: A recon gurable framework for trust negotiation. Trust Management III, Vol. 300/2009, pp. 176–195, 2009.

[2] B.Schoenmakers and P.Tuyls. Practical two-party computation based on the conditional gate. Lecture Notes in Computer Science, Vol. 3329/2004, pp. 129–145, 2004.

[3] R.T. Fielding. Architectural Styles and the Dsign of Network-Based Software Architectures. PhD thesis, University of California, Irvine, 2000.

[4] I.H. Katugampala, 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.

[5] K.Kursawe, G.Neven, and P.Tuyls. Private policy negotiation. Financial Cryptography and Data Security, Vol. 4107/2006, pp. 81–95, 2006.

[6] T.ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. Information Theory, IEEE Transactions on, Vol. 31, pp. 469–472, 1985.

[7] T.Yu, X.Ma, and M.Winslett. Prunes: an efficient and complete strategy for automated trust negotiation over the internet. CCS '00 Proceedings of the 7th ACM conference on Computer and communications security, 2000.

[8] W.H. Winsborough, K.E. Seamons, and V.E. Jones. Automated trust negotiation. DARPA Information Survivability Conference and Exposition, Vol. 1, pp. 88–102, 2000.

[9] 森文宏, 八槇博史. 準同型暗号系を用いた双方向ポリシマッチング. 電子情報通信学会技術研究報告, Vol. 110, No. 304, pp. 7–12, 2010.