SCIS 2014 The 31st Symposium on
Cryptography and Information Security
Kagoshima, Japan, Jan. 21 - 24, 2014
The Institute of Electronics,
Information and Communication Engineers

個人情報保護に向けたプログラム変換方法の検討と評価
Program Conversion to Protect User's Personal Information and its Evaluation

工藤 邦晃* 高橋 健一 * 川村 尚生 * 菅原 一孔 *
Kuniaki Kuto Kenichi Takahashi Takao Kawamura Kazunori Sugahara

あらまし 近年，インターネットの普及により，ショッピング，ホテルの予約など様々なサービスがインターネット上で提案されている．これらのサービスの一部では個人情報の提供を要求する．しかし，利用者はサービス提供者に提供した個人情報の利用方法を確かめることができない．その結果，利用者がサービス提供者に不安を感じても，サービスを受けるには個人情報を提供しなければならない．このことを解決するために，利用者が個人情報の処理方法を制御できる仕組みを提案する．個人情報はサービス提供者の持つプログラムで処理される．そのため，そのプログラムに利用者の意図を反映させることで，利用者に個人情報の処理方法を制御する仕組みを実現する．

キーワード プライバシー，プログラム変換，個人情報，セキュリティ

1 はじめに

現在，誰もが簡単にインターネットを利用できる．それに伴い，インターネット上では，オンラインショッピングやオンラインバンキングなどのサービスが公開されている．これらのサービスは，利用者に名前，住所，電話番号などの個人情報を要求する．しかし，利用者には実際に個人情報がどのように利用されているかわからない．また，近年では，情報漏洩事件 [1] や不正利用，フィッシングサイトによる被害 [2] が多発している．その対策として，これまでに認証プロトコルや電子署名，SSL などのセキュリティ技術が開発されてきた．しかし，これらの技術を用いるかの決定権はサービス提供者にあり，利用者に個人情報の処理方法の決定権がない．そこで，我々は利用者が個人情報の処理方法を指定できるような仕組みを提案する．

利用者の個人情報はサービス提供者の持つプログラムで処理される．そのため，提案モデルでは，このプログラムを利用者が指定した個人情報の処理方法に書き換える．サービス提供者は書き換えられたプログラムで個人情報を処理する．これにより，個人情報は利用者が指定した処理方法で処理されるため，利用者はサービス提供者による個人情報の悪用を妨げることができる．その結果，利用者は安心して個人情報をサービス提供者に提供できる．

本稿では，提案モデルを実現するために，サービス提供者が持つプログラムに利用者が指定した処理方法を適用する方法を検討する．

2 関連研究

企業のウェブページの多くは，プライバシーの考え方（Yahoo! Japan）やプライバシー（IBM Japan）といった個人情報の取り扱い方を示したページを持つ．これらのページでは，サイトが収集した情報をどのような目的のためにどのように取り扱うのかを示す．また，サイトが収集する情報の取り扱い方を利用者に提示するためのフォーマットとして P3P（The Platform for Private Preferences）[3] がある．P3P では各サイト（企業）が定義しているプライバシポリシとユーザプリファレンスを比較することで，自動的な情報の提供可否判断を可能にする．しかし，これらは企業によってプライバシポリシが遵守されていることを保証しない．

企業内での情報の取り扱い方を制御するための仕組みとして EPAL（Enterprise Privacy Authorization Language）[4] がある．しかし，EPAL は企業内従業員による不正な情報利用を防止するものであって，企業内での情報の取り扱い方を利用者に保証するものではない．

利用者間の信頼に基づいて情報へのアクセス権やサー

* 鳥取大学大学院工学研究科情報エレクトロニクス専攻，鳥取県鳥取市湖山町南 4-101, Department of Information and Electronics, Graduate School of Engineering Tottori University, 4-101 Koyama Minami Tottori 680-8550, Japan

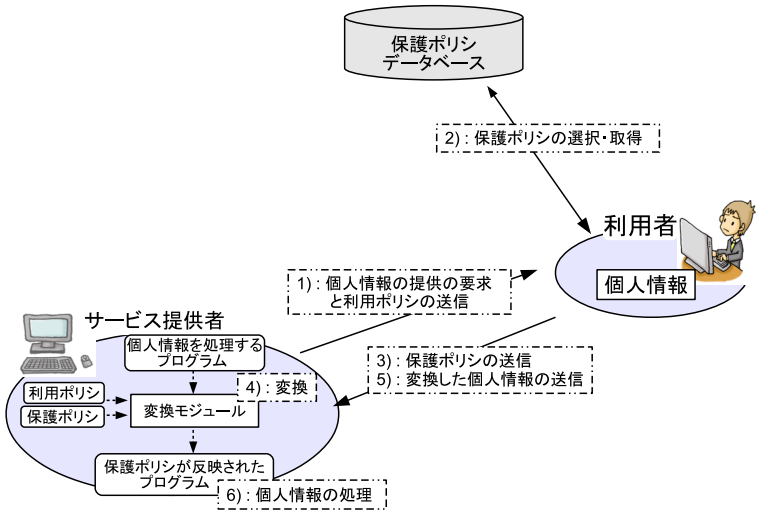

図 1: 提案モデル

ビスの利用権を付与するための様々な研究 [5, 6] が行なわれている．これらの研究では相手が持つ属性によってその相手が信頼可能かを決定する．しかし，信頼関係は両者が置かれた状況や環境に依存するものであり，一般的な方法を定義することが難しく，アプリケーションに依存したものとなる．また，利用者はサービス提供者を信頼することで，情報が不正に利用されないことを信じているだけであり，サービス提供者は容易に個人情報を不正に利用することができる．

利用者の匿名性が確保できる情報の組み合わせだけを相手に開示することでプライバシを確保する研究として [7] や [8] がある．これらの研究では，その情報単体だけで利用者の特定に繋がらない情報であれば，情報の組み合わせを制御することで利用者のプライバシを守ることができる．しかし，クレジットカード番号やパスワードなど，その情報単体で利用者の特定に繋がる情報を守ることはできない．

個人情報を外部に出さないことで利用者の個人情報を守ろうとする研究として EMAPP[9] がある．EMAPP では利用者が個人情報を管理する空間（Encapsulated Space）を持ち，サービス提供者が利用者の個人情報を確認するためのモバイルエージェントを持つ．利用者はモバイルエージェントを Encapsulated Space 内に受け入れ，利用者の情報確認結果だけをサービス提供者に送信させることで利用者の情報を守る．しかし，利用者はモバイルエージェントが送信する結果を利用者の情報を漏らさないものであると確認する必要があり，またモバイルエージェントが Encapsulated Space で正しく動作したといったことをサービス提供者に対して保証する必要がある．

更に，これらの研究では，利用者の情報操作に対してサービス提供者が主導権を握っており，利用者はサービス提供者に情報を提供するか，提供しないかの選択権しかない．そのため，我々は利用者が個人情報の利用方法を制御できる仕組みを提案する．

3　提案モデル

提案モデルはサービスを利用する利用者とサービスを提供するサービス提供者からなる．利用者は安心できる処理方法をサービス提供者に伝える．サービス提供者はその方法に従ってプログラムを変換する．その後，サービス提供者は変換したプログラムで個人情報を処理する．これにより，利用者は自身が選択した方法により，個人情報を保護することができる．

プログラムの変換はプログラム変換モジュールで実現する．プログラム変換モジュールでプログラムを変換するためにはプログラムを変換するルールが必要となる．しかし，一般的な利用者はプログラムに関する知識がなく，プログラムを変換するためのルールを定義することが難しい．そこで，プログラムの変換ルールとルールにより情報をどのように保護するのかを示す説明文から構成される保護ポリシを定義し，複数の保護ポリシを管理する保護ポリシデータベースを準備する．利用者は保護ポリシデータベースにアクセスし，保護ポリシ中の説明文を読むことで，自分が安心できる情報の処理方法が定義されている保護ポリシを選択することができる．

また，保護ポリシを選択するためには，どの保護ポリシがサービス提供者が持つプログラムに適用できるのか知る必要がある．しかし，サービス提供者が持つプログ

ラムによって情報の処理方法が異なる．そこで，利用ポリシを定義する．利用ポリシはサービス提供者が利用者に要求する情報ごとに準備され，サービス提供者がプログラム中でどの情報をどのように処理するかが定義される．これにより，利用者はサービス提供者が持つプログラムが提供した情報に対しどのような処理を行うか知ることができる．図 1 に提案モデルの概要を示す．

1. 利用者はサービス提供者にサービスを要求する．この時，サービス提供者はサービスを利用するための条件として利用者に対して個人情報の提供を要求する．しかし，利用者はサービス提供者を信頼できず，個人情報を与えることに不安に感じる．そこで，利用者はサービス提供者による個人情報の処理方法を確認するために利用ポリシを要求する．サービス提供者はその個人情報に対する利用ポリシを利用者に送信する．利用者は利用ポリシにより，サービス提供者がどのように個人情報を処理しようとしているか知ることができる．

2. 利用者は保護ポリシデータベースにアクセスし，利用ポリシに定義された処理方法に合致する保護ポリシの一覧を得る．利用者はその中から個人情報の処理方法が定義された保護ポリシを取得する．

3. 利用者は指定した保護ポリシをサービス提供者に送信する．

4. サービス提供者は保護ポリシ，利用ポリシにより，個人情報処理プログラムを変換モジュールで保護ポリシが反映されたプログラムに変換する．

5. 利用者は保護ポリシに従い保護したい個人情報を変換し，変換した個人情報をサービス提供者に送信する．

6. サービス提供者は，変換された個人情報を保護ポリシが反映されたプログラムで処理する．

これにより，利用者は自身が選択した処理方法でサービス提供者に個人情報を処理してもらうことが可能となる．第 4 章で保護ポリシ，第 5 章で利用ポリシ，第 6 章でプログラムの変換方法について述べる．

4 保護ポリシ

保護ポリシは利用者がサービス提供者に自身が安心できる個人情報の処理方法を伝えるために利用する．保護ポリシの構成を図 2 に示す．

保護ポリシは XML 形式で定義される．保護ポリシでは利用者に対する説明文を<EXPLANATION>，保護対象の情報と操作を<INFORMATION>と<OPERATION>，プログラム変換ルールを<CONVERT-RULE>で定義する．

```
<EXPLANATION>
   自然言語で書かれた説明文
</EXPLANATION>
<INFORMATION> information </INFORMATION>
<OPERATION> operation </OPERATION>
<CONVERT-RULE>
   rules
</CONVERT-RULE>
```

図 2: 保護ポリシ

<EXPLANATION>は自然言語で記述され．利用者は<EXPLANATION>を読むことで，自分が安心できる処理を実現する保護ポリシを選択することができる．

保護ポリシが保護対象とする情報とそれに対する操作はそれぞれ<INFORMATION>と<OPERATION>で定義する．これらは利用者が保護ポリシデータベース内からプログラムに適用可能な保護ポリシを検索するときに利用する．これにより，利用者は自身が守りたい情報が定義された保護ポリシでかつサービス提供者が持つプログラムに適用可能な保護ポリシの一覧を取得することができる．

<CONVERT-RULE>は複数のルールからなり，利用者に対する説明文を実現するためのプログラム変換方法を定義する．例えば，利用者がサービス提供者のことを信頼できず，個人情報を保護したいとする．しかし，個人情報を提供しなければ，サービスを利用することはできない．このため，個人情報 (データ) を保護するために個人を特定できないようにデータを変換する必要がある．そこで，このようなデータを変換するためのルールを<DATA-CONVERT-RULE>として<CONVERT-RULE>内に定義する．また，データが変換された場合，変換前のデータに行っていた操作が適用できなくなる可能性がある．そこで，変換前のデータに行っていた操作を変換後のデータに対する操作に変換するためのルールが必要となり，それを<OPERATION-CONVERT-RULE>として定義する．また，変換前のデータが利用者からサービス提供者に送信されると，個人情報を保護することはできない．そのため，変換前のデータの通信を阻止すると共に，変換後のデータを送受信するためのルールが必要となり，それを<COMMUNICATION-RULE>として定義する．

これらのルールを用いることで，個人情報を変換し，変換した個人情報を利用できるようにプログラムを変換する．

5 利用ポリシ

図 3 に利用ポリシを示す．利用ポリシは xml 形式で定義される．利用ポリシでは，定義対象の情報を<INFORMATION>，データを格納する変数を<VARIABLE>，処理方法を<OPERATION>で定義する．

```
<INFORMATION>information<INFORMATION>
<VARIABLE>
   <NAME>name</NAME>
   <TYPE>type</TYPE>
</VARIABLE>
<OPERATION>
   <FORMAT>
      <METHOD>(*<PARAMETER>)
   </FORMAT>
   <METHOD>
      <NAME>name</NAME>
      <RETURN>type</RETURN>
   </METHOD>
   *<PARAMETER>
      <DATA>information</DATA>
      <NAME>name</NAME>
      <TYPE>type</TYPE>
   </PARAMETER>
</OPERATION>
```

図 3: 利用ポリシ

<INFORMATION>は利用者が保護ポリシデータベースから保護ポリシを絞り込む時に利用する．保護ポリシの<INFORMATION>と，利用ポリシの<INFORMATION>を比較することで，絞り込むことができる．

<VARIABLE>は<NAME>と<TYPE>から構成され，利用者の個人情報（データ）を格納される変数を定義する．<NAME>は変数名を定義する．<TYPE>はその変数の型を定義する．これにより，変換対象のプログラムが利用者からの個人情報をどの変数に格納しているか知ることができる．

<OPERATION>は<FORMAT>，<METHOD>，<PARAMETER>から構成され，利用者の個人情報（データ）をどのような操作で利用するかを定義する．<FORMAT>はメソッドへの引数の与え方を定義する．<METHOD>は操作名を表す<NAME>とその操作の結果として返される戻り値を表す<RETURN>から構成される．<PARAMETER>では引数の情報を，<DATA>，<NAME>，<TYPE>により定義する．<DATA>は変数に格納される情報（データ）を示し，<NAME>は変数名，<TYPE>は変数の型名を示す．これにより，プログラム中で利用者の個人情報に対してどのような操作が適用されるか知ることができる．

6 プログラムの変換

利用者が選択した保護ポリシをサービス提供者の持つプログラムに適用するためにプログラムの変換を行う．保護ポリシには利用者が守りたい情報とプログラム変換ルールが定義される．一方，利用ポリシには変換するプログラムの情報が定義されている．プログラムの変換は変換対象のプログラムと保護ポリシ，利用ポリシをプログラム変換モジュールに与えることで行う．プログラム変換モジュールは Java プログラムを変換の対象とする．また，保護ポリシと利用ポリシは XML 形式で記述されている．図 4 にプログラムの変換例を示す．

図 4 は利用者からプレインテキストのパスワードを受信して認証するプログラムをハッシュ化したパスワードを認証するものに変換する例である．利用ポリシはパスワードについて定義されており，パスワードが受信された後に格納される変数とパスワードに対して行われる処理が定義される．一方，保護ポリシは，パスワードの通信を禁止し，代わりにハッシュ変換したパスワードで認証するようにプログラムを変換するためのものである．

プログラム変換ルールはデータ変換ルール（<DATA-CONVET-RULE>）と操作変換ルール（<OPERATION-CONVERT-RULE>），送受信制御ルール（<COMMUNICATION-RULE>）から構成される．<DATA-CONVET-RULE>にはパスワードをハッシュ化するルール，

hash_password_sp ← hash(password_sp, nonce)

が定義されている．これは hash 関数に password_sp と nonce を与えることで hash_password_sp を生成することを示す．また，<OPERATION-CONVERT-RULE>には

authentication(hash_password, hash_password_sp)
← authentication(password, password_sp)

が定義されており，authentication という操作に password と password_sp の情報を利用するところを hash_password と hash_password_sp の情報を利用するように変換することを示す．<COMMNUICATION-RULE>には

password : denied

hash_password, nonce : allowed

が定義され，前者は password の通信を禁止することを示し，後者は hash_password と nonce の通信を許可することを示す．

プログラム変換モジュールによるプログラム変換の流れを図 5 に示す．プログラム変換モジュールはまず保護ポリシと利用ポリシ，変換対象のプログラムを読み込む．次に保護ポリシの情報と利用ポリシの情報を結びつける．これにより，保護ポリシが保護対象とする情報がプログラム中でどのような変数に格納されて処理されるか知ることができる．次にプログラムを解析し，変数やメソッド等に分解する．その後，プログラム変換ルールを読み込み，各ルールに従ってプログラムを変換する．これにより，利用者はサービス提供者の持つプログラムに保護ポリシで定義された処理を適用する．

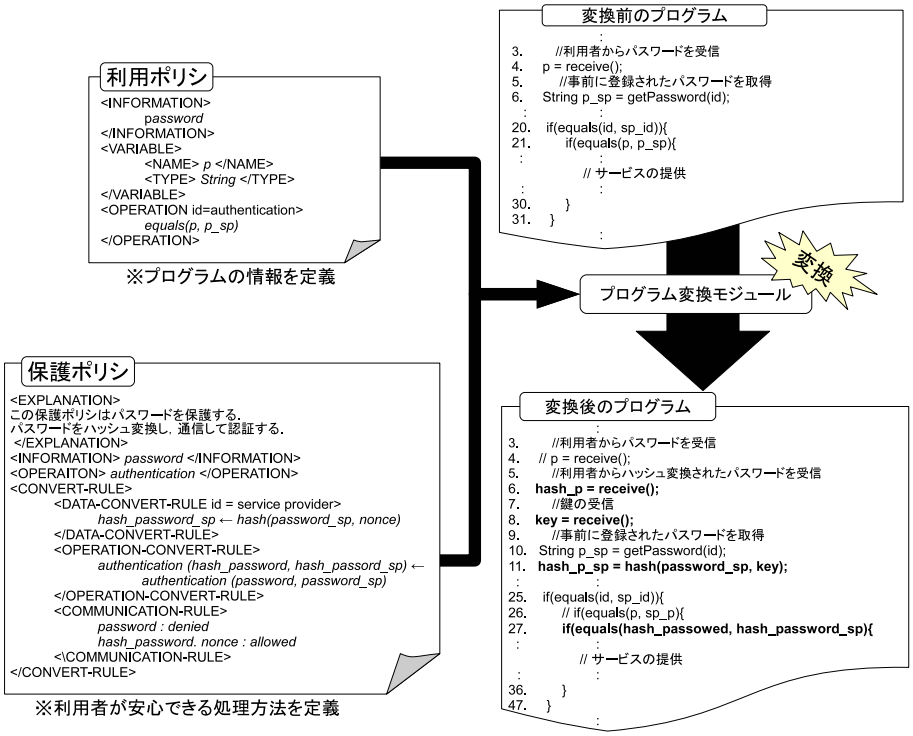

図 4: プログラムの変換例

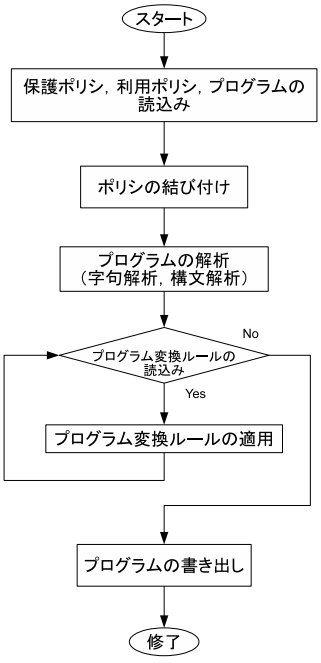

図 5: プログラム変換モジュールの流れ

6.1 読み込み

保護ポリシと利用ポリシ，個人情報処理プログラムを読み込む．その後，ポリシにどのような情報が定義されているか知るために，XML 形式で記述されたポリシを解析する．

6.2 ポリシの結びつけ

保護ポリシが変換対象とする情報がプログラム中のどこで利用されているか，また，変換対象の操作がプログラム中でどのような形式で処理されているかわからない．そこで，これらの情報を利用ポリシと保護ポリシを結びつけることで得る．

保護ポリシの<INFORMATION>には変換対象の情報が定義されている．また，利用ポリシの<VARIABLE>には利用ポリシの<INFORMATION>に定義された情報が格納される変数が定義されている．このため，保護ポリシと利用ポリシの<INFORMATION>を結びつけることで，保護ポリシの<INFORMATION>に定義された情報が<VARIABLE>の変数で利用されていることがわかる．図 4 の例では *password* が *String* 型の変数 *p* で利用されることがわかる．

同様に，<OPERATION>に対しても結びつけを行うことで，変換対象の操作がプログラム中のどの処理で実現されているか知ることができる．図 4 の例では，保護ポリシの<OPERATION>に定義された操作 *authentication* がプログラム中の *equals* 関数で実現されていることがわかる．

結果，情報が格納される変数やその情報に対する処理がわかり，プログラム変換ルールを適用する箇所を特定することが可能となる．

6.3 プログラムの解析

プログラム中から変数名を検索したくても，ソースコード内のどの文字列が変数名を示すか特定しなければならない．プログラム中ではメソッドを跨ぐことにより，

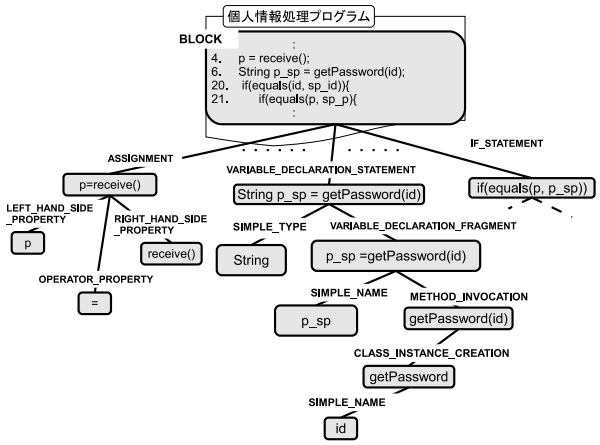

図 6: ASTParser により構造化されたプログラムの例

格納するデータは同じでも変数名が変わる場合が存在する．そこで，プログラムの解析を行う．プログラムの解析には ASTParser[10] を用いる．ASTParser は Java プログラムに対し字句解析と構文解析を行い，抽象構文木を生成する．解析により構造化されたプログラムの例を図 6 に示す．

例えば図 4 のプログラムの 4 行目 `p = receive()` は ASSIGNMENT と解析され，この行が代入式であることがわかる．同様に，6 行目は VARIABLE_DECLARATION_STATEMENT と解析され，この行が変数の宣言文であることがわかる．また，21 行目 `if(equals(p, sp)){` は IF_STATEMENT と解析され，この行は if 文であることがわかる．このため，変数 p が出現する箇所を特定する場合には，ASSIGNMENT と VARIABLE_DECLARATION_STATEMENT を探すことですべて抽出することができる．更に，式の左辺に p がある行を抽出することで，p へ代入する行を取得することができる．

6.4 プログラム変換ルールの適用

プログラム変換ルールを適用することで，保護ポリシに定義された処理をサービス提供者が持つプログラムに実現する．プログラム変換ルールは複数のルールから構成されるため，このことを実現するためには，まずこれらのルールの適用順序を決定する必要がある．プログラム変換ルールを適用する流れを図 7 に示す．

図 4 の例ではプログラム変換ルールは<DATA-CONVERT-RULE>と<OPERATION-CONVERT-RULE>，<COMMUNICATION-RULE>からなる．<DATA-CONVERT-RULE>はサービス提供者が認証時に受信したパスワードと比較する予め登録されたパスワードをハッシュ変換するためのルールであり，*hash_password_p* を生成するために *password_sp* と *nonce* を必要とする．しかし，*password_sp* と *nonce* がどの変数に対応するか定義されていない．そこで先に<OPERATION-CONVERT-RULE>の適用を検討する．ここ

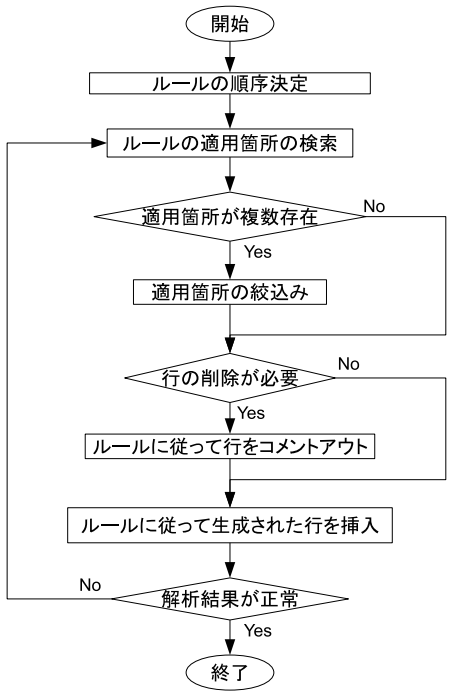

図 7: プログラム変換ルールの適用の流れ

で，利用ポリシには *password* が *p*，*authentication* が *equals* と定義されている．このため，<OPERATION-CONVERT-RULE>の変換前の操作

```
equals(p, password_sp)
```

を特定することで，*password_sp* が格納されている変数を特定することができる．しかし，本ルールを適用するためには，変換後の操作で必要となる *hash_password* がわからず，変換することができない．そこで，先に<COMMUNICATION-RULE>を適用する．これにより，ハッシュ化されたパスワードが格納される変数がわかるため，<OPERATION-CONVERT-RULE の適用が可能となり，更に<DATA-CONVERT-RULE>の適用も可能となる．

次に各プログラム変換ルールを適用する箇所を特定する．このことを実現するために ASTParser を利用してプログラムを解析する．

ASTParser により，生成された抽象構文木の各ノードは特定の識別子とが付けられ，その識別子は ASTParser のライブラリと結びつけられる．図 4 の変換前のプログラムでは図 6 のように解析される．

例えば，<DATA-CONVERT-RULE>は変換対象の情報が変数の値が確認した後であり，かつ，特定の操作が行われる前に適用する必要がある．そのため，Asaignment クラスなどのライブラリを利用することで，代入箇所を特定し，また，保護ポリシの<OPERATION>に定義された特定の操作を検索する．これにより，各ルールごとに適用可能な箇所を抽出する．

最後にそれぞれのプログラム変換ルールを適用し，プログラムを変換する．プログラム変換ルールの適用は，変換前の処理が必要なくなる箇所があればその箇所をコメントアウトし，変換後に必要となる処理を追加することで行われる．以下，図 4 の例を用いて，それぞれのプログラム変換ルールによるプログラムの変換方法を具体的に示す．

送受信ルール： <COMMUNICATION-RULE>は *password* の通信を禁止し，*hash_pasword* と *nonce* の通信を許可すると定義されている．このルールを適用するために，まず，禁止される *password* の通信箇所を特定する．保護ポリシと利用ポリシの結びつけにより，*password* は変数 *p* に格納されることがわかる．そのため，変数 *p* に情報が格納される行を特定する．図 4 のプログラムの例では 4 行目が取得できる．次に *password* の通信を禁止と定義されているため，4 行目をコメントアウトする．また，*hash_pasword* と *nonce* は通信許可とあり，利用者から受信するデータであるため，それらを受信するためのコード（変換後の 6 行目と 8 行目）を挿入する．これにより，*password* の通信を禁止すると共に，*hash_password* と *nonce* を受信するコードを生成することができる．

操作変換ルール： <OPERATION-CONVERT-RULE>には au-thentication という操作には password と password_sp の情報を利用するところを hash_password と hash_password_sp の情報を利用するように変換すると定義されている．また，保護ポリシと利用ポリシの結び付けにより，*authentication* は関数 *equals* で実現されていること，*password* は変数 *p* に格納されていること，*password_sp* は変数 *p_sp* に格納されることがわかる．そこで，*equals(p, sp_p)* の操作を行っている行を特定する．図 4 の例では 21 行目が取得できる．そこで，21 行目のこの操作を *equals(hash_p, hash_p_sp)* に変換する．これにより，操作を変換することができる．

データ変換ルール： <DATA-CONVERT-RULE>は *hash_p_sp* を *password_sp* と *nonce* から *hash* 関数により生成するためのルールである．ここで，すでに適用された<OPERATION-CONVERT-RULE>により，*password_sp* は変数 *p_sp* に格納されていることがわかる．また，*hash_p_sp* を生成するためには，*password_sp* と *nonce* が必要となる．そこで，まず，変数 *p_sp* と *nonce* の両方が揃う箇所を特定する．図 4 の例では 6 行目が両方揃う箇所と特定できる[1]．このため，6 行目以降に *hash_p_sp* を生成するコードを挿入すればよいとわかる．また，*hash_p_sp* を利用するコードを生成する操作変換ルールが 21 行目に適用されている．このため，7〜20 行目のいずれかに *hash_p_sp* を生成するコードを挿入する必要がある．図 4 の例では 7 行目（変換後の 11 行目）に挿入されている．これにより，データ変換ルールを適用し，変数 *p_sp* をハッシュ化することができる．

すべてのルールの適用が確認できたらプログラムの変換を終了する．

6.5 書き出し

変換できたプログラムを java 形式で書き出す．その後，コンパイルしたプログラムをサービス提供者が実行することで，利用者は自身が選択した処理方法を反映したプログラムで個人情報を処理してもらうことができる．

7 評価

プログラム変換モジュールにより，どれくらいのプログラムの変換が可能かを検証した．検証では Web 上で公開されている 20 個のプログラムを利用した．また，7 個の保護ポリシを準備した．保護ポリシとしては，特定の操作・通信を禁止するポリシや通信を暗号化するポリシ，認証方法を変更するポリシなどを準備した．プログラムによって適用する意味のない保護ポリシが存在し，その組み合わせを除いた 89 通りの変換を検証した．89 通りの変換を行った結果を表 1 に示す．

表 1: 検証結果

変換成功	実行可能	意図した動作	76 通り
		意図しない動作	2 通り
	実行不可能	-	0 通り
変換失敗	-	-	11 通り

89 通りの変換の内，76 通りの変換については，意図した動作を実現するプログラムに変換できた．しかし，意図した動作と異なるプログラムへの変換が 2 通り，変換を失敗したものが 10 通り存在した．

変換が失敗したものは，グローバル変数を利用しているプログラムや利用者からの情報を条件分岐の条件として利用しているものであり，これらのことを考慮していなかったために発生していた．このため，これらのことを考慮した実装にすることで解決できるものと思われる．

また，意図しない動作に変換されたものについては，利用者の個人情報を格納していた変数が，他のデータを格納するために再利用されており，この結果，データの

[1] 変換後のコードでは *nonce* の受信は 8 行目となっているが，これは変換前の 4 行目に挿入された結果である．

変換が間違ったデータに適用されたため発生していた．これは変数への代入文に注目し，その変数が利用者の個人情報を保持しているプログラム中の範囲を特定することで解決できるものと思われる．

8 おわりに

本研究では，提案モデルを実現するために保護ポリシ，利用ポリシ，プログラム変換ルールを定義し，それらによるプログラムの変換方法を提案した．本提案方法をプログラム変換モジュールとして実装し，評価した結果，変換を失敗するプログラムと意図しない処理に変換されたプログラムが存在した．今後の課題として，これらを変換できるようにプログラム変換モジュールを改良することが挙げられる．

謝辞

本研究は科学研究費補助金（23700098）の支援を受けて行なった．

参考文献

[1] NPO 日本ネットワークセキュリティ協会，2009 年情報セキュリティインシデントに関する調査報告，
http://www.jnsa.org/result/incident/2009.html

[2] フィッシング対策協議会，月次報告書，
http://www.antiphishing.jp/report/monthly/

[3] P3P project, http://www.w3.org/P3P.

[4] The EPAL 1.1, http://www.zurich.ibm.com/security/enterpriseprivacy/epal/.

[5] G. Theodorakopoulos, J. Baras, ‘‘Trust Evaluation in Ad-Hoc Networks,’’ WiSe04, pp. 1-10, 2004.

[6] D. Xiu, Z. Liu, ‘‘A Dynamic Trust Model for Pervasive Computing Environments,’’ FTDCS 2004, pp. 80-85, 2004.

[7] 今田美幸，高杉耕一，太田昌克，小柳 恵一，‘‘ユビキタスネットワーキング環境におけるプライバシー保護手法 : LooM,’’ 信学論 (B), Vol.J88-B, No.3, pp. 563-573, 2005.

[8] 宮本崇弘，竹内亨，奥田剛，春本要，有吉勇介，下條真司 ‘‘プライバシとサービス品質のトレードオフを考慮した個人情報制御機構の提案,’’ DEWS 2005, 6A-o1, 2005.

[9] S. Yamada, E. Kamioka, ‘‘Access Control for Security and Privacy in Ubiquitous Computing Environments,’’ IEICE Trans. on Comm., Vol.E88-B, No.3, pp. 846-856, 2005.

[10] Eclipse の ASTParser を試す,
http://www.ibm.com/developerworks/jp/opensource/library/os-ast/