コンピュータセキュリティ　14−10
(2001. 7. 25)

SOAPにおける否認不可性の実現について

羽田 知史

日本アイ・ビー・エム(株) 東京基礎研究所
〒242-8502 大和市下鶴間 1623-14
046-215-4241　　satoshih@jp.ibm.com

あらまし

SOAP(Simple Object Access Protocol)は、任意のXML文書を送受信するための汎用的なプロトコルとして注目されている。SOAPでは、送信したいXML文書を包むためのデータ構造（SOAPエンヴェロープ）が定義されている。SOAP Security Extensions: Digital Signature では、そのデータ構造にしたがって、否認不可性を実現ことを目的としてSOAPエンヴェロープに署名するための一般的な方法が定義されている。本稿では、その安全性をSOAPが規定するメッセージの通信モデルにしたがって、形式的に議論する。

キーワード　SOAP、否認不可性、ディジタル署名

On Non-repudiation in SOAP

Satoshi Hada

IBM Japan, Co.,Ltd.　Tokyo Research Laboratory
〒242-8502　1623-14, Shimotsuruma, Yamato-shi
046-215-4241　　satoshih@jp.ibm.com

Abstract

SOAP (Simple Object Access Protocol) is a general protocol for exchanging any XML documents. SOAP defines a data structure called the SOAP Envelope to carry XML documents to be sent. The SOAP Security Extensions: Digital Signature defines a general way of signing SOAP envelopes for the purpose of satisfying the non-repudiation requirement. In this paper, we analyze its security based on the message transmission model of SOAP.

key words　SOAP, Non-repudiation, digital signature

1 はじめに

SOAP(Simple Object Access Protocol)は、任意のXML文書を送受信するための汎用的なプロトコルとして注目されている[9]。SOAPの仕様は、SOAPエンヴェロープとSOAP符号化の二つのパートから構成されている。SOAPエンヴェロープは、送信したいXML文書を包むためのデータ構造であり、そのデータ構造はSOAPボディとSOAPヘッダから構成される。通常ボディにはアプリケーションデータが含まれるが、ヘッダにはアプリケーションのセマンティクスとは直接関係のないデータが含まれることが想定されている。一方、SOAP符号化は非XMLデータをXMLデータに符号化する規則を定義しており、主に、RPCなどのアプリケーションでの利用が想定されている。本稿では、SOAPエンヴェロープのデータ構造を用いて、SOAPエンヴェロープを署名することを考察するので、SOAP符号化は扱われない。図1にSOAPエンヴェロープの例をあげる(URIの値などは簡略化されている)。この例では、SOAPボディには、株式の買い注文の情報が含まれており、SOAPヘッダは空である。

SOAPでは以上のようにSOAPエンヴェロープにメッセージが包まれて送信される。XML署名[12]を用いて、SOAPエンヴェロープに署名するための一般的な方法として、SOAP Security Extensions: Digital Signatureが提案されている[10]。以下では、この方法をSOAP-DSIGと呼ぶことにする。そこでのアイデアは極めて単純であり、署名の対象(SOAPエンヴェロープ全体でも、そのフラグメントでもよい)をXML署名を用いて署名し、生成された署名データ(¡ds:Signature¿要素)をSOAPヘッダに¡SOAP-SEC:Signature¿要素として格納するというものである。図2にその例を示す。

SOAP-DSIGの主要な目的は、SOAPで送信するXMLデータの否認不可性(Non-repudiation)を実現することである。本稿の目的は、SOAP-DSIGを用いて否認不可性を実現するための十分条件は何かを、SOAPの通信モデルに従って、形式的に議論することである。ただし、本稿の議論は一般的なものであり、以下に述べるSOAPの通信モデルに従う限り、SOAP-DSIGに限定されるものではない。以下に、SOAPの通信モデルを整理しておく。

1. メッセージの一方向性:SOAPでのXMLデータの送信は、基本的に一方向である。リクエスト/レスポンス型の通信などは、その一方向のデータ送信を組み合わせで実現されることになる。したがって、本稿では、SOAPエンヴェロープを送信する際、SOAP-DSIGにより署名をヘッダに添付して送信する、という非対話型のプロトコルを前提とする。
2. トランスポート独立:SOAPでは、SOAPエンヴェロープを運ぶためのトランスポート層を指定していない。したがって、否認不可性の実現のためにHTTPやSSLなどの特定のトランスポート層の情報を用いることは想定しない。
3. 仲介者(Intermediaries)の存在:SOAPでは、送信者Aが受信者BにSOAPエンヴェロープを送信する際、AがBに直接送信するのではなく、複数の仲介者を介して、送信されてもよい。本稿では、送信者Aが自分の秘密鍵を用いて署名を生成および送信し、受信者Bが検証することを前提とする。つまり、仲介者は署名の生成および検証には一切関与しない。仲介者は通信路の一部とみなされる。悪意のある仲介者(データを改ざんしたりする)が存在する場合は、そのような仲介者の存在は、認証されていない通信路(unauthenticated channel)を仮定することによってカバーされる。実際、本稿では認証されていない通信路上で署名されたSOAPエンヴェロープを送信することを前提としている。

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://soap/envelope/">
 <SOAP-ENV:Header>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <order:買い注文 xmlns:order="http://stock/order">
   <order:銘柄>日本アイビーエム</order:銘柄>
   <order:銘柄コード>1234</order:銘柄コード>
   <order:買付数量>100</order:買付数量>
   <order:指値>10000</order:指値>
   <order:市場>ニューヨーク</order:市場>
  </order:買い注文>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図1: SOAPエンヴェロープの例

広く知られていることであるが、否認不可性は、図2のように単にディジタル署名を添付するだけでは実現することはできない。そのようなナイーブなやり方では、リプレー攻撃に弱いことが知られている。SOAP-DSIGの仕様においては、こういった攻撃を回避するためのデータ構造は規定されていない。ISOの国際標準[8]では、この種の攻撃を回避する

ためにどのように署名が生成されるべきかを標準化している[1]。特に、この標準は、Non-repudiation of origin token（以下、トークン）を定義しており、これは、基本的には、送信されるメッセージ m、いくつかの付加情報 z、そして、m と z に対する署名の値から構成される。つまり、あるメッセージの送信者は、トークンを生成して、受信者に送信することになる。付加情報はいくつかのデータ項目から構成される。具体的には、否認不可性に関するポリシーの識別子、Non-repudiation of Originを示す識別子、メッセージの送信者の識別子、メッセージの受信者の識別子、トークンが生成された日時、トークンが送信される日時、その他、メッセージ識別子や署名メカニズムなどに関するオプショナルデータが含まれる。本稿では、これらのデータ項目のうち、どれが否認不可性の実現のために必須であるべきか？そして、どれが省略可能か？という疑問を考察する。

本稿の主要な成果は、上述のトークンに基づくプロトコルの安全性の形式的な解析を与え、安全性の観点から上述の疑問に回答することである。従来メッセージ認証 (message authentication あるいは authenticated communication) に関しては、形式的な安全性の解析が行われてきた [1][4][3]。メッセージ認証は否認不可性よりも弱い概念である。通常、メッセージ認証は、認証付の鍵交換プロトコル (authenticated key-exchange protocol) により、セッション鍵を共有し、その鍵を用いて計算される MAC をメッセージに添付することにより実現される（SSL や TLS[11] が典型例である)。この種の組み合わせの形式的な安全性の解析は [1][5] で与えられている。メッセージ認証とは対照的に否認不可性の形式的な安全性の解析はこれまで行われていない。本稿で解析するプロトコルは本質的には、[1] で解析されている signature-based MT-authenticator と同じである。同様に、その安全性解析の内容も本質的には同じものである。本稿では、メッセージ認証ではなく、否認不可性の観点から、そのプロトコルの安全性の解析を行う。本稿での解析によると、一意なメッセージ識別子とメッセージの受信者の識別子がデータ項目として加えられるならば、否認不可性は満たされる。したがって、SOAP-DSIG に基づいて SOAP エンヴェロープを署名する際もこれらのデータ項目が署名の対象に含まれることが必要となる。本稿での解析は、トークンが送信される通信路は、認証なし (unauthenticated) で、かつ、公開 (public) であると仮定している。つまり、送信されるデータは盗聴可能であり、かつ、データの内容および送信者の識別子も改ざんされるかもしれない。

[1]ISO の国際標準 [7] では 4 種類の否認不可性を定義している。本稿で扱うのは、Non-repudiation of origin である。

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://soap/envelope/">
 <SOAP-ENV:Header>
  <SOAP-SEC:Signature
   xmlns:SOAP-SEC="http://soap/security/2000-12"
   SOAP-ENV:mustUnderstand="1">
   <ds:Signature xmlns:ds="http://xmldsig#">
    <ds:SignedInfo>
     <ds:CanonicalizationMethod
      Algorithm="http://CR-xml-c14n">
     </ds:CanonicalizationMethod>
     <ds:SignatureMethod
      Algorithm="http://xmldsig#dsa-sha1"/>
     <ds:Reference URI="#Body">
      <ds:Transforms>
       <ds:Transform Algorithm="http://CR-xml-c14n"/>
      </ds:Transforms>
      <ds:DigestMethod
Algorithm="http://xmldsig#sha1"/>
      <ds:DigestValue>j6lwx3...</ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>MC0Cs...</ds:SignatureValue>
    <ds:KeyInfo>
     <ds:KeyName>田中太郎</ds:KeyName>
    </ds:KeyInfo>
   </ds:Signature>
  </SOAP-SEC:Signature>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body
  xmlns:SOAP-SEC="http://soap/security"
  SOAP-SEC:id="Body">
  <order:買い注文 xmlns:order="http://stock/order">
   <order:銘柄>日本アイビーエム</order:銘柄>
   <order:銘柄コード>1234</order:銘柄コード>
   <order:買付数量>100</order:買付数量>
   <order:指値>10000</order:指値>
   <order:市場>ニューヨーク</order:市場>
  </order:買い注文>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 2: SOAP エンヴェロープの署名の例

以上では、一意なメッセージ識別子および受信者の識別子をデータ項目として加えることは否認不可性のために十分であることを説明した。メッセージ識別子はしばしばナンス (nonce) と呼ばれ、タイムスタンプ、カウンタなどとして実装される。しかしながら、そのようなナンスを管理することは、実際のシステムのアプリケーション開発者/管理者にとって面倒な場合もある。なぜなら、送信者と受信者の間で共通の状態を管理する必要があるからである。実際には、ナンスを省略したいかもしれない。前述

のように本稿の安全性解析では、署名が送信される通信路は、認証なし (unauthenticated) で、かつ、公開 (public) であると仮定している。しかし、認証つき (authenticated) で、かつ、安全 (secure) な通信路は [1][5] の方法で理論的にも実現可能であり、そして、SSL や IPSec などのように実用的な実装も今日のネットワーク環境で利用可能である。本稿が扱う2つ目の問題は、そのような通信路が利用可能であるとき、一意なメッセージ識別子や受信者の識別子といったデータ項目を省略することができるか？という問題である。残念ながら本稿の安全性解析によると、そのような通信路が利用されたとしても、二つのデータ項目は必須である。

本稿での安全性の定義および解析は Canetti のフレームワークに基づいている [3]。Canetti はこのフレームワークに基づいて、認証つき通信路、鍵交換プロトコル、暗号化、ディジタル署名、コミットメント、ゼロ知識、などの従来の暗号プリミティブおよびプロトコルんい対して新しい定義を与えている。[3] の主要な結果は、コンカレント構成定理 (concurrent composition theorem) であり、この定理は、Canetti のフレームワークで定義されたプロトコルの安全性は、そのプロトコルがあるアプリケーション内で複数同時に実行されたとしても満たされることを保証する。したがって、本稿で解析するプロトコルも、安全性を犠牲にすることなく、アプリケーション内で複数同時に実行可能である。

本稿では、Canetti のフレームワークについての説明は省略する。安全性の定義のしかたなどに関しての詳細は [3] を参照のこと。特にセッションおよびセッション識別子の概念は重要であるが説明は省略する。[3] では、通信路はすべて認証つきであると仮定されているが、本稿では、認証なしの通信路も利用する。これについては、[3, Sect. 4.4, Unauthenticated communication] を参照のこと。また、本稿では、Canetti のフレームワークを [1] のように公開鍵秘密鍵ペアの生成のための初期化処理により拡張したものを用いる。

2 否認不可性の定義

本章では、否認不可性の安全性を表現する理想的な機能 (ideal functionality) $\mathcal{F}_{\mathrm{NR}}$ を定義する。

各送信者/受信者ペア $(\tilde{P}_i, \tilde{P}_j)$ はナンスを生成するためのアルゴリズム $G_{i,j}$ をもっていると仮定する。それはセキュリティパラメータ k を入力として、一意性の保証された文字列を出力する。この生成されたナンスはメッセージ識別子として利用される。また、各 $G_{i,j}$ 毎に、対応する検証アルゴリズム $V_{i,j}$ が存在し、ナンスの一意性を検証することができる、と仮定する。つまり、$V_{i,j}$ はセキュリティパラメータ k とある文字列を入力として、その一意性を示す答え Unique あるいは Not-Unique を出力する。これらのアルゴリズムは異なるセッションにまたがって使用され、そのナンスの一意性は異なるセッションにまたがって保証されると仮定する。それらは、タイムスタンプやカウンタとして実装されるが、各ペア $(\tilde{P}_i, \tilde{P}_j)$ は異なるセッションにまたがって共通の状態を管理する必要がある。

$\mathcal{F}_{\mathrm{NR}}$ の完全なスペックは図 3 で示されている。$\mathcal{F}_{\mathrm{NR}}$ は [3, Section 7.1] で定義された認証つき通信路 $\mathcal{F}_{\mathrm{AUTH}}$ の拡張である。その違いは:

1. $\mathcal{F}_{\mathrm{AUTH}}$ では、一つのセッションでは、メッセージが一つだけ送信されると仮定されている。つまり、各メッセージは別々に、他のすべてのメッセージとは独立して認証される。言い換えると、$\mathcal{F}_{\mathrm{AUTH}}$ におけるセッション識別子 sid は $\mathcal{F}_{\mathrm{AUTH}}$ の異なる実行を区別するためだけでなく、異なるセッションにて同じ送信者によって複数回送信される同じメッセージを区別するためにも使われる。一方、$\mathcal{F}_{\mathrm{NR}}$ では、一つのセッションにて複数のメッセージが送信されてもよい。特に、同じメッセージが複数回同じセッションにおいて送信されてもよい。それらを区別するために、$\mathcal{F}_{\mathrm{NR}}$ では、受信者にフォワードする前に一意なメッセージ識別子を各メッセージに加える必要がある。このようにセッション識別子とメッセージ識別子を区別することは安全性解析を明確にできると思われる。
2. prove メッセージが新しく加えられており、これは否認不可性のために本質的なものである。このメッセージを $\mathcal{F}_{\mathrm{NR}}$ に送信することによって、あるメッセージの受信者 $(\tilde{P}_j)$ は誰か他のパーティ $(\tilde{P}_k)$ に、あるパーティ $(\tilde{P}_i)$ がそのメッセージを作成し送信したという事実を証明することができる。結果的に、いかなる悪意のある送信者も後にメッセージを送信した事実を否認することはできない。$\mathcal{F}_{\mathrm{AUTH}}$ はこの種の機能をサポートしていない。

さらにいくつかの $\mathcal{F}_{\mathrm{NR}}$ の重要なポイントを述べておく。$\mathcal{F}_{\mathrm{NR}}$ は送信者への send メッセージのレスポンスとしていかなるメッセージ送信の証拠も与えな

い。実際にレスポンスはない。つまり、$\mathcal{F}_{\mathrm{NR}}$ の実現はディジタル署名の使用を前提としていない。本稿はディジタル署名に基づいた否認不可性の実現を考えているが、ディジタル署名以外の別の手段により実現できるかもしれない。

受信者があるメッセージを受信したセッションは、受信者が後に受信した事実を証明するセッションとは別のセッションであってもよい。つまり、セッション sid であるメッセージ受信した受信者は別のセッション $sid2$ においてそのメッセージを受信した事実を証明してもよい。言い換えると, $\mathcal{F}_{\mathrm{NR}}$ はどのセッションでメッセージを受信したかについての否認不可性はサポートしていない。セッション識別子がデータベース DB に記録されないのはこれが理由である。

あるメッセージを受信すると、$\mathcal{F}_{\mathrm{NR}}$ は対応する通知メッセージ `send_request` あるいは `prove_request` をアドバーサリに送信する。これは $\mathcal{F}_{\mathrm{NR}}$ を実現するプロトコルの安全性が「アドバーサリがメッセージの内容や関係するパーティの識別子を知ることはできない」という仮定を要求してはいけないことを意味する。

3 否認不可性を実現するプロトコル

本章では、ディジタル署名に基づくプロトコル π_{NR} を示し、それが $\mathcal{F}_{\mathrm{NR}}$ を適応的に (adaptively) 買収 (corruption) を行うアドバーサリに対して安全に実現することを証明する。$\mathsf{ID}(P_i)$ により P_i の識別子を表すものとする。

π_{NR} は 2 種類のメッセージ通信から構成される。その完全なスペックは図 4 に示されている。一つのメッセージ通信は、あるパーティがあるメッセージを他のパーティに送信するために使われる。$(\mathtt{send}, sid, \mathsf{ID}(P_j), m)$ が入力されると、P_i はナンス $mid \leftarrow G_{i,j}(1^k)$ を生成し、$(m, mid, \mathsf{ID}(P_j))$ に署名し、m を署名とともに P_j に認証なしの通信路上で送信する。それを受信すると、P_j は署名とナンスの一意性を検証し、もし検証が OK ならばそれらをデータベースに記録する。すでに述べたように、$\mathcal{F}_{\mathrm{NR}}$ はどのセッションでメッセージが送信されたのかに関する否認不可性はサポートしないので、セッション識別子 sid は署名の対象ではない。もう一つのメッセージ通信はあるメッセージの受信者が別のパーティに、そのメッセージはその送信者によって送信されたという事実を証明するために用いられる。$(\mathtt{prove}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_k), m, mid)$ が入力されると、P_j は (m, mid) の署名をデータベースから取り出し、P_k に送信する。つまり、P_k にその署名を検証するように依頼する。このメッセージは認証つきの通信路で送信されるものとする。それは SSL のような通信路が用いられてもよいし、フロッピーディスクなどに保存され物理的に手渡されてもよい。

以下に、π_{NR} が $\mathcal{F}_{\mathrm{NR}}$ を適応的に買収を行うアドバーサリに対して安全に実現することを証明する。

定義 1 ([6]) 署名法 $Signature = (Gen, Sign, Ver)$ は次の条件を満たすならば選択メッセージ攻撃に対して偽造不可能 *(existentially unforgeable against chosen message attacks)* である。いかなる（非一様なモデルでの）多項式時間マシン G *(偽造者)* においても、G が公開鍵 PK およびセキュリティパラメータ 1^k を入力として、$Sign(SK, \cdot)$ を署名オラクルとして用いて、$Ver(PK, m, \sigma) = \mathsf{Valid}$ を満たし、かつ、m が署名オラクルに問い合わせられていないような (m, σ) を出力する確率は k に関して無視できる。ここで確率は Gen および $Sign$ の使う乱数上で計算される。

定理 1 $Signature = (Gen, Sign, Ver)$ を選択メッセージ攻撃に対して偽造不可能な署名法であるとする。その場合、π_{NR} は $\mathcal{F}_{\mathrm{NR}}$ を適応的に買収を行うアドバーサリに対して安全に実現する。

証明: 任意の現実世界でのアドバーサリ $\mathcal{A}$ に対して、それを模倣する (simulate) 理想的なプロセスでのアドバーサリ $\mathcal{S}$ が存在することを示す必要がある。詳細はスペースの都合上省略するが、任意の $\mathcal{A}$ に対する、そのような理想的なプロセスでのアドバーサリ $\mathcal{S}$ を図 5 に示す。

4 考察

[8] では、受信者の識別子とメッセージ識別子はオプションのデータ項目であるが、本稿での安全性解析によると、それらの両方は $\mathcal{F}_{\mathrm{NR}}$ を安全に実現する上で必須であることがわかる。SOAP-DSIG に基づいて、否認不可性を満たすシステムを構築する上でも、このことに注意する必要がある。以下では、このことをもう少し詳しく説明する。まず、メッセージ識別子は必須であることを説明する。署名 σ が $Sign(SK_i, (m, \mathsf{ID}(P_j)))$ として計算されるような π_{NR} の簡略化を考える。この簡略化によりあるメッセージの悪意のある受信者 P_j は別の（正直な）パーティ P_k に、P_j はそのメッセージ

を、たとえ一度しか受信していなくても、複数回受信したと偽って証明することが可能である。結果的に、π_{NR} はもはや $\mathcal{F}_{\mathrm{NR}}$ を実現するとは言えない。より具体的には、いかなる理想的なプロセスでのアドバーサリも以下のような現実世界のアドバーサリ $\mathcal{A}$ を模倣する (simulate) ことはできない：(1) $\mathcal{A}$ は $(\texttt{message\_transmission}, sid, m, mid, \sigma)$ を P_i から P_j へ配送する。(2) $\mathcal{A}$ は P_j を買収し、異なるメッセージ識別子 $mid2 \neq mid$ を用いて $(\texttt{verify}, sid, \mathsf{ID}(P_i), m, mid2, \sigma)$ を別の買収されていないパーティ P_k に送信させる。識別子 mid は署名の対象ではないので、P_k による検証は成功し、P_k は $(\texttt{proof}, sid, \mathsf{ID}(P_i), m, mid2, \mathsf{Yes})$ を出力する。

同様に、受信者の識別子は必須のデータ項目であることを説明する。署名 σ が $Sign(SK_i, (m, mid))$ として計算されるような π_{NR} の簡略化を考える。この簡略化により、悪意のあるパーティ P_m は、別の悪意のあるパーティ P_j と結託して、別の（正直な）パーティ P_k に、P_i から P_j に送信されたメッセージを自分 (P_m) が受信したと偽って証明することが可能である。より具体的には、いかなる理想的なプロセスでのアドバーサリも以下のような現実世界のアドバーサリ $\mathcal{A}$ を模倣することはできない：(1) $\mathcal{A}$ は $(\texttt{message\_transmission}, sid, m, mid, \sigma)$ を P_i から P_j に配送する。(2) $\mathcal{A}$ は P_j を買収し、配送された (sid, m, mid, σ) を取得する。(3) $\mathcal{A}$ は P_m を買収し、$(\texttt{verify}, sid, \mathsf{ID}(P_i), m, mid, \sigma)$ を別の買収されていないパーティ P_k に送信させる。本来の受信者の識別子 $\mathsf{ID}(P_j)$ は署名の対象ではないので、P_k による署名検証は成功し、P_k は $(\texttt{proof}, sid, \mathsf{ID}(P_i), m, mid, \mathsf{Yes})$ を出力する。

以上の議論は message_transmission メッセージが、認証つきで、かつ、安全な通信路上で送信される場合でも成り立つ。なぜなら、上述の二つの現実世界でのアドバーサリの振る舞いは、使用する通信路のモデルとは独立しているからである。このことは、そのような通信路が用いられたとしても、受信者の識別子およびメッセージ識別子は省略できないことを意味している。つまり、否認不可性の観点からは、π_{NR} を SSL のような認証つきの安全な通信路上で実行しても意味はないと言えるだろう。もちろん、SSL は受信者認証や機密性のためには有用ではある。第 1 章で述べたように、異なるセッションにまたがって一意性の保証されるメッセージ識別子を管理することは時にはやっかいである。特に、あるパーティが物理的に異なるコンピュータ上で動作しているときには、やっかいであろう。残念であるが、以上の議論は、メッセージ識別子を使用することは否認不可性の安全性確保のためには必須であることを意味している。

[1] での signature-based authenticator のように、安全性を犠牲にすることなく、メッセージ識別子を受信者によってランダムに選択されるチャレンジメッセージで置き換えることは可能である。しかし、その場合、プロトコルは対話的なものとなってしまい、SOAP のような一方向のメッセージ通信を基本とする場合には適用できない。

参考文献

[1] M. Bellare, R. Canetti, and H. Kraczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols, " Proceedings of 30th STOC, pp. 419-428. A full version (dated March 13, 1998) is available at author's home page.

[2] R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols, " Journal of Cryptology, Vol. 13, Num. 1, pp. 143-202, Winter 2000.

[3] R. Canetti, "A Unified Framework for Analyzing Security of Protocols, " *manuscript*, Dated December 22, 2000. Available at `http://eprint.iacr.org/2000/067/`

[4] R. Canetti, S. Halevi, and A. Herzberg, "How to Maintain Authenticated Communication in the Presence of Break-Ins, " Journal of Cryptology, Vol. 13, Num. 1, pp. 61-105, Winter 2000.

[5] R. Canetti and H. Kraczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels," Proceedings of Eurocrypt'01. A longer version (dated May 17, 2001) is available at `http://eprint.iacr.org/2001/040/`

[6] S. Goldwasser, S. Micali, and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks, " SIAM Journal on Computing, Vol. 17, No. 2, pp. 281-308, April 1988,

[7] ISO/IEC 13888-1, "Informatin Technology - Security techniques - Non-repudiation - Part 1: General, " First edition, 1997.

[8] ISO/IEC 13888-3, "Informatin Technology - Security techniques - Non-repudiation - Part 3: Mechanisms using aysmmetric techniques, " First edition, 1997.

[9] W3C Note, "Simple Object Access Protocol (SOAP) 1.1, " http://www.w3.org/TR/SOAP/, 08 May 2000.

[10] W3C Note, "SOAP Security Extensions: Digital Signature " http://www.w3.org/TR/SOAP-dsig/, 06 Feburuary 2001.

[11] IETF RFC 2246, "The TLS Protocol Version 1.0," http://www.ietf.org/rfc/rfc2246.txt, January 1999.

[12] W3C Candidate Recommendation, "XML-Signature Syntax and Processing," 31 October 2000.

Functionality $\mathcal{F}_{\mathrm{NR}}$

DB を送信されたメッセージを記録するためのデータベースとする。$\mathcal{F}_{\mathrm{NR}}$ は、ダミーパーティ $\tilde{P}_1, \tilde{P}_2, \cdots, \tilde{P}_n$ とアドバーサリ $\mathcal{S}$ とともに、以下のように実行される。ただしセキュリティパラメータは k である。 $(\mathtt{send}, sid, \mathsf{ID}(\tilde{P}_j), m)$ あるいは $(\mathtt{prove}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_k), m)$ のいずれかが入力されると、ダミーパーティは単にその入力を $\mathcal{F}_{NR}$ へフォワードする。

1. $\tilde{P}_i$ から $(\mathtt{send}, sid, \mathsf{ID}(\tilde{P}_j), m)$ を受信したとき、$\mathcal{F}_{\mathrm{NR}}$ は $(\mathtt{send\_request}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m)$ を $\mathcal{S}$ へ送信し、一意なメッセージ識別子 $mid \leftarrow G_{i,j}(1^k)$ を生成し、$(\tilde{P}_i, \tilde{P}_j, m, mid)$ を DB に追加し、そして、$(\mathtt{message}, sid, \mathsf{ID}(\tilde{P}_i), m, mid)$ を $\tilde{P}_j$ へ送信する。
2. $\tilde{P}_j$ から $(\mathtt{prove}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_k), m, mid)$ を受信したとき、$\mathcal{F}_{\mathrm{NR}}$ は $(\mathtt{prove\_request}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), \mathsf{ID}(\tilde{P}_k), m, mid)$ を $\mathcal{S}$ に送信し、識別子 mid をもつメッセージ m が以前 $\tilde{P}_j$ に送信されたことがあるかどうかをチェックする、すなわち、$(\tilde{P}_i, \tilde{P}_j, m, mid)$ が DB に記録されているかをチェックする。もし、記録されていれば、$\mathcal{F}_{\mathrm{NR}}$ は $result = \mathsf{Yes}$ をセットする、さもなくば、$result = \mathsf{No}$ とセットする。そして、$\mathcal{F}_{\mathrm{NR}}$ は $(\mathtt{proof}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m, mid, result)$ を $\tilde{P}_k$ へ送信する。

ダミーパーティは $(\mathtt{message}, sid, \mathsf{ID}(\tilde{P}_i), m, mid)$ あるいは $(\mathtt{proof}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m, mid, result)$ を $\mathcal{F}_{\mathrm{NR}}$ から受信したときはいつでも、それを出力する。

図 3: The Non-repudiation functionality $\mathcal{F}_{\mathrm{NR}}$

Protocol π_{NR}

$Signature = (Gen, Sign, Ver)$ を [6] で定義された署名法とする。初期化処理として、各パーティは公開鍵秘密鍵ペア $(PK_i, SK_i) \leftarrow Gen(1^k)$ を生成し、公開鍵を信頼できる手段により他のパーティに公開する。

π_{NR} はセキュリティパラメータ k を入力として、パーティ $P_1, \cdots, P_n$ およびアドバーサリ $\mathcal{A}$ とともに以下のように実行される。パーティ P_i の入出力の振る舞いは図 3 における対応するダミーパーティ $\tilde{P}_i$ のものと同じである。

1. 入力 $(\mathtt{send}, sid, \mathsf{ID}(P_j), m)$ が与えられると、P_i はメッセージ識別子 $mid \leftarrow G_{i,j}(1^k)$ を生成し、署名 $\sigma = Sign(SK_i, (m, mid, \mathsf{ID}(P_j)))$ を計算し、そして、$(\mathtt{message\_transmission}, sid, m, mid, \sigma)$ を P_j に送信する。この $\mathtt{message\_transmission}$ メッセージは認証なしの通信路上で送信される。
   $(\mathtt{message\_transmission}, sid, m, mid, \sigma)$ を受信すると、P_j は P_i の公開鍵 PK_i により署名 σ を検証し、mid の一意性を $V_{i,j}$ を用いてチェックする。すなわち、$Ver(PK_i, (m, mid, \mathsf{ID}(P_j)), \sigma) = \mathsf{Valid}$ かつ $V_{i,j}(k, mid) = \mathsf{Unique}$ であることをチェックする。もし両方が成り立てば、P_j は $(\mathsf{ID}(P_i), m, mid, \sigma)$ を自分のデータベースに保存し、$(\mathtt{message}, sid, \mathsf{ID}(P_i), m, mid)$ を出力する。さもなくば、P_j は何も出力しない。
2. P_j は入力として $(\mathtt{prove}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_k), m, mid)$ を与えられると、P_j と P_k は以下のことを実行する:
   (a) P_j は $(\mathsf{ID}(P_i), m, mid)$ に関連する署名 σ をデータベースから取り出す。もしそのような署名がなければ、P_j は $(\mathtt{verify}, sid, \mathsf{ID}(P_i), m, mid, \sigma = \mathtt{null})$ を P_k に送信する、さもなくば、$(\mathtt{verify}, sid, \mathsf{ID}(P_i), m, mid, \sigma)$ を P_k に送信する。この $\mathtt{verify}$ メッセージは認証つきの通信路上で送信される。
   (b) P_k は受信した署名 σ を検証し、もし署名が正当であるなら、すなわち、$\sigma \neq \mathtt{null}$ かつ $Ver(PK_i, (m, mid, \mathsf{ID}(P_j)), \sigma) = \mathsf{Valid}$ であるなら、P_k は $(\mathtt{proof}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_j), m, mid, \mathsf{Yes})$ を出力する、さもなくば、$(\mathtt{proof}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_j), m, mid, \mathsf{No})$ を出力する。

図 4: $\mathcal{F}_{\mathrm{NR}}$ を実現するプロトコル π_{NR}

Ideal-process Adversary $\mathcal{S}$

Adversary $\mathcal{S}$, interacting with the ideal functionality $\mathcal{F}_{\mathrm{NR}}$ and dummy parties $\tilde{P}_1, \tilde{P}_2, \cdots, \tilde{P}_n$, is activated in the following ways. Recall that all participants have security parameter k.

1. When $\mathcal{S}$ is activated for the first time (either by $\mathcal{Z}$ or by $\mathcal{F}_{\mathrm{NR}}$), it initializes a simulated interaction of the adversary $\mathcal{A}$ with real-life parties $(P_1, P_2, \cdots, P_n)$ running the protocol π_{NR}. $\mathcal{S}$ also generates a key pair $(PK_i, SK_i) \leftarrow Gen(1^k)$ for each simulated real-life party P_i.
2. When $\mathcal{S}$ is activated on an input value v (written by $\mathcal{Z}$), it runs, on the input v, the simulated $\mathcal{A}$ interacting with the simulated real-life parties $(P_1, P_2, \cdots, P_n)$. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs.
   (a) When $\mathcal{A}$ delivers a message $(\texttt{message\_transmission}, sid, m, mid, \sigma)$ from P_i to P_j, $\mathcal{S}$ runs the simulated P_j on the message. (1) If P_i is uncorrupted and the message is generated by P_i then there must be a message $(\texttt{send}, sid, \mathsf{ID}(\tilde{P}_j), m)$ for $\mathcal{F}_{\mathrm{NR}}$ on $\tilde{P}_i$'s outgoing communication tape. $\mathcal{S}$ delivers it to $\mathcal{F}_{\mathrm{NR}}$ and then delivers the response of $\mathcal{F}_{\mathrm{NR}}$ to $\tilde{P}_j$, that is, $\tilde{P}_j$ eventually receives $(\texttt{message}, sid, \mathsf{ID}(\tilde{P}_i), m, mid)$. (2) If the message is one forged by $\mathcal{A}$[a] then $\mathcal{S}$ runs the simulated P_j on the message. In addtion to that, $\mathcal{S}$ tries to simulate this forgery in the ideal process as follows: If both P_i and P_j are uncorrupted and P_j outputs $(\texttt{message}, sid, \mathsf{ID}(\tilde{P}_i), m, mid)$ then $\mathcal{S}$ outputs a special message `'I cannot continue'`. If P_i is uncorrupted, P_j is corrupted, and the message is valid for P_j[b] then $\mathcal{S}$ outputs `'I cannot continue'`. Otherwise, $\mathcal{S}$ delivers $(\texttt{send}, sid, \mathsf{ID}(\tilde{P}_j), m)$ from $\tilde{P}_i$ to $\mathcal{F}_{\mathrm{NR}}$.
   (b) When $\mathcal{A}$ delivers a message $(\texttt{verify}, sid, \mathsf{ID}(P_i), m, mid, \sigma)$ from P_j to P_k, $\mathcal{S}$ runs the simulated P_k on the message. (1) If P_j is generated by P_j then there must be a message $(\texttt{prove}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_k), m, mid)$ for $\mathcal{F}_{\mathrm{NR}}$ on $\tilde{P}_j$'s outgoing communication tape. $\mathcal{S}$ delivers it to $\mathcal{F}_{\mathrm{NR}}$ and then delivers the response of $\mathcal{F}_{\mathrm{NR}}$ to $\tilde{P}_k$, that is, $\tilde{P}_k$ eventually receives either $(\texttt{proof}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m, mid, \mathsf{Yes})$ or $(\texttt{proof}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m, mid, \mathsf{No})$. (2) If the message is generated by $\mathcal{A}$ then $\mathcal{S}$ runs the simulated P_k on the message. In addtion to that, $\mathcal{S}$ tries to simulate this forgery in the ideal process as follows (Recall that since the `verify` message is sent over an ideally authenticated communication channel, P_j is corrupted.): If the entry (P_i, P_j, m, mid) is not in the database DB of $\mathcal{F}_{\mathrm{NR}}$[c], P_k is uncorrupted, and P_k outputs $(\texttt{proof}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_j), m, mid, \mathsf{Yes})$ then $\mathcal{S}$ outputs `'I cannot continue'`. Otherwise, $\mathcal{S}$ delivers a message $(\texttt{prove}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_k), m, mid)$ from the corresponding corrupted dummy party $\tilde{P}_j$ to $\mathcal{F}_{\mathrm{NR}}$ and then delivers the response of $\mathcal{F}_{\mathrm{NR}}$ to $\tilde{P}_k$.
   (c) When $\mathcal{A}$ corrupts a party P_i, $\mathcal{S}$ also corrupts the corresponding dummy party $\tilde{P}_i$. From this point on, whenever $\mathcal{A}$ write a message on the output tape of P_i, $\mathcal{S}$ write the same message on the output tape of the corresponding dummy party $\tilde{P}_i$.
3. When $\mathcal{S}$ is activated on a message from $\mathcal{F}_{\mathrm{NR}}$, it proceeds as follows:
   (a) On receiving a message $(\texttt{send\_request}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m)$ from $\mathcal{F}_{\mathrm{NR}}$ (this occurs when $\tilde{P}_i$ is activated on a `send` message by $\mathcal{Z}$), $\mathcal{S}$ runs the simulated P_i on input $(\texttt{send}, sid, \mathsf{ID}(\tilde{P}_j), m)$. That is, P_i generates a message identifier $mid \leftarrow G_{i,j}(1^k)$, computes $\sigma = Sign(SK_i, (m, mid, \mathsf{ID}(P_j)))$, and sends (m, mid, σ) to P_j. Note that at this point, there is a message to $\mathcal{F}_{\mathrm{NR}}$ $(\texttt{send}, sid, \mathsf{ID}(\tilde{P}_j), m)$ on $\tilde{P}_i$'s outgoing communication tape.
   (b) On receiving a message $(\texttt{prove\_request}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), \mathsf{ID}(\tilde{P}_k), m, mid)$ from $\mathcal{F}_{\mathrm{NR}}$ (this occurs when $\tilde{P}_j$ is activated on a `prove` message by $\mathcal{Z}$), $\mathcal{S}$ runs the simulated P_j on input $(\texttt{prove}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_k), m, mid)$. Note that at this point, there is a message to $\mathcal{F}_{\mathrm{NR}}$ $(\texttt{prove}, sid, \mathsf{ID}(P_i), \mathsf{ID}(P_k), m, mid)$ on $\tilde{P}_i$'s outgoing communication tape.

[a] When P_i is corrupted, the message is always generated by $\mathcal{A}$.

[b] S can check this because P_j is corrupted.

[c] S can check this by sending $(\texttt{prove}, sid, \mathsf{ID}(\tilde{P}_i), \mathsf{ID}(\tilde{P}_j), m, mid)$ as a message generated by $\tilde{P}_j$. Recall that $\tilde{P}_j$ is corrupted.

図 5: The adversary for the ideal process with $\mathcal{A}$ and $\mathcal{F}_{\mathrm{NR}}$