コンピュータセキュリティ 17－5
（２００２．５．２３）

Spi計算の型付けによる公開鍵暗号方式を用いたプロトコルのメッセージ認証の検証

畑山 研† 萩原 茂樹‡ 米崎 直樹‡

†（株）日立製作所 公共システム事業部

‡ 東京工業大学 大学院 情報理工学研究科 計算工学専攻

これまでに、Gordon らにより、Spi 計算に型付けを行うことにより、対称鍵暗号を用いるセキュリティプロトコルのメッセージ認証を検証する手法が提案されている。しかし、公開鍵暗号系を用いるプロトコルについては明らかではなかった。本稿ではそのようなプロトコルのメッセージ認証を検証する手法について報告する。公開鍵を用いるプロトコルでは、認証者が、相手を認証するために、最初に意図した相手のみに読めるように乱数を暗号化して送信し、その乱数を返信してもらう。プロトコルがこの構造をしていることを確認するために、Spi 計算に公開鍵・秘密鍵の型と新たな乱数の型を加え、これに対応する推論規則を構成した。さらに、この推論規則が Spi 計算の意味論に対して健全であることを示した。

Type-based Verification of Authenticity in Protocols with Public Key Encryption using Spi-Calculus

Ken Hatayama† Shigeki Hagihara‡ Naoki Yonezaki‡

† Government & Public Corporation Information Systems Division, Hitachi Ltd.

‡ Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology.

M.Abadí and A.D.Gordon invented the “Spi-Calculus”[1] and constructed a verification method of secrecy [2] and authenticity in protocols using symmetric-keys [3] by typing. However, the method of verifying authenticity in protocols using public-keys and private-keys cannot be done by a simple extension of the method of symmetric-keys. The purpose of this paper is to construct the verification method of authenticity in such protocols. In order to achieve this, we introduced types and constructed typing rules. And we proved that the typing system we defined is sound and justified.

1 はじめに

一見欠陥のないプロトコルに対して、攻撃が可能であるという発見が近年になってなされている。そのため、形式的な手法を用いることにより、プロトコルに情報が第三者に漏洩しない性質 (秘密性) やネットワークを通して受け取ったメッセージが意図している相手から確実に届いたことを確認できる性質 (メッセージ認証) があることを厳密に検証する方法を確立することが重要である。

M.Abadíと A.D.Gordon は、並行計算を扱うための π 計算を暗号を扱えるように拡張した Spi 計算 [1] を提案し、その後、メッセージやプロセスに安全であるかどうかを表す付加情報である型をつけることによって、静的にこれらの性質を検証する手法を考案した [2, 3]。しかし、公開鍵・秘密鍵を扱うプロトコルについては次の２つの点で対称鍵と性質が異なるために同様の検証方法を構築することは明らかではない。一つは秘密鍵で署名されたメッセージは攻撃者 (イントルーダ) が復号して確認できてしまうという点、もう一つは公開鍵を用いるプロトコルではナンス (乱数) の流れが対称鍵と異なるという点である。

そこで本稿では、[3] の型付け規則を拡張することにより、公開鍵・秘密鍵を扱うプロトコルにおいて、メッセージ認証を検証する手法を構築する。まず基本メッセージの型として公開鍵と秘密鍵の型を追加し、それに伴って、公開鍵暗号で用いられるナンスの型をはじめ、いくつかの型を追加する。次に、それらの型を持つメッセージを合成して得られるメッセージの型を計算する、項の型付け規則を定義する。続いて、メッセージ認証を満たすためのプロトコルの構造を考察し、プロセスの型付け規則にそれを反映させる。そして、その振る舞いに制限を加えることによってイントルーダのプロセスを定義する。そして、これらの定義が正当であることを示す証明の指針を述べる。

同じ目的で、Gordon らも [4] で検証方法を発表している。そこでは、我々が行うプロトコルの解析以外の解析も行っている。そのアプローチや検証できる性質の違いについても考察する。

2 Spi 計算

2.1 構文と意味

本節では型つき Spi 計算の構文規則とその形式的な意味について述べる。Spi 計算の構文は、メッセージやデータを表す項とその処理を表すプロセスの２つに分かれる。

定義 1 (項) Spi 計算の項は以下のように定義される。

$$M ::= n|(M_1, M_2)|()|\{M_1\}_{M_2}|\mathsf{Inl}(M)|\mathsf{Inr}(M)$$

n は名前を表し、原子的な項である。これは、変数と定数の区別をしない。(M_1, M_2) は２つの項の組を表したものである。また $()$ は空の組を表す。これらを用いてメッセージの並びを表すことができる。$\{M_1\}_{M_2}$ は項 M_1 を項 M_2 で暗号化したものを表す。$\mathsf{Inl}(M)$ と $\mathsf{Inr}(M)$ はそれぞれタグがついた項であり、後述する case イベントにおいて場合分けを行うときに使用するものである。

定義 2 (プロセス) プロセスを以下のように定義する。

$P ::=$	プロセス
out $M_1\ M_2; P$	出力
inp $M\ x:T; P$	入力
$P_1 \mid P_2$	合成
repeat P	繰り返し
new$(n:T); P$	生成
new$(n_1, T, n_2); P$	鍵生成
split $M = (n_1:T, n_2:U); P$	組分け
case $M =$ Inl$(n_1:T); P$ Inr$(n_2:U); Q$	場合分け
decrypt $M_1\ (M_2)$ to $n:T; P$	復号
match $M_1 = (M_2, y:T); P$	等式判定
STOP	停止
begin $M; P$	開始ラベル
end $M; P$	終了ラベル
cast M_1 is $M_2:T; P$	型変換
check $M_1 = M_2; P$	ナンス確認

ここで、開始ラベル、終了ラベル begin M, end M はメッセージ認証を定式化する際に利用される特別のイベントである。また、cast, check は検証の際に用いる特別なイベントである。これらの詳細は後述する。

次に、形式的意味を定義する。意味は簡約操作によって発生するイベント列 (トレース) として定義される。まず、構造等価関係を定義する。なお、fn は free name、gn は generated name の略である。

定義 3 (構造等価関係)

$P \equiv P$
$P \mid \mathsf{STOP} \equiv P$
$P \mid Q \equiv Q \mid P$
$P \equiv Q \Rightarrow Q \equiv P$
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$
repeat $P \equiv P \mid$repeat P
$P \equiv Q \Rightarrow$ new$(x:T); P \equiv$ new$(x:T); Q$
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$
$x \notin fn(P) \Rightarrow P \mid$new$(x); Q \equiv$ new$(x)(P \mid Q)$
$x \neq y, x \notin fn(U), y \notin fn(T) \Rightarrow$
new$(x:T)$; new$(y:U); P \equiv$ new$(y:U)$; new$(x:T); P$

次に、ラベル付き遷移規則 $P \xrightarrow{\alpha} P'$ を定義する。これは、P から P' に簡約されるとき、イベント α が発生するということを意味する。

定義 4 (ラベル付き遷移規則)

out $M\ N; P \mid$inp $M\ x:T; Q \xrightarrow{\tau} P \mid Q[N/x]$
split $(M, N) = (x:T, y:U); P \xrightarrow{\tau} P[M/x, N/y]$
decrypt $\{M\}_N\ (N^{-1})$ to $x:T; P \xrightarrow{\tau} P[M/x]$
check $M = M; P \xrightarrow{\text{check } M} P$
case Inl$(M) =$ Inl$(x:T); P$ Inr$(y:U); Q \xrightarrow{\tau} P[M/x]$
case Inr$(M) =$ Inl$(x:T); P$ Inr$(y:U); Q \xrightarrow{\tau} Q[M/y]$
match $(M, N) = (M, x:T); P \xrightarrow{\tau} P[N/x]$
cast M is $x:T; P \xrightarrow{\text{cast } M} P[M/x]$
new$(n:T); P \xrightarrow{\text{new } n:T} P$
new$(n_1, T, n_2); P \xrightarrow{\tau} P$
begin $L; P \xrightarrow{\text{begin } L} P$
end $L; P \xrightarrow{\text{end } L} P$
$gn(\alpha) \cap fn(Q) = \phi \Rightarrow P \xrightarrow{\alpha} P' \Rightarrow P \mid Q \xrightarrow{\alpha} P' \mid Q$
$P \equiv Q, Q \xrightarrow{\alpha} Q', Q' \equiv P' \Rightarrow P \xrightarrow{\alpha} P'$
$x \notin fn(\alpha) \Rightarrow P \xrightarrow{\alpha} P' \Rightarrow$
new$(x:T); P \xrightarrow{\alpha}$ new$(x:T); P'$

2.2 メッセージ認証の定式化

プロトコルのメッセージ認証が成り立つか否かを検証するためのプロトコルの仕様は、[3] に従い 2 つの対応するイベント begin M, end M を用いて表す。begin M は、通信を行う前に挿入し、end M は通信でメッセージを受け取り、ナンスのチェック等の適切な処理を行った後に挿入する。これらのイベントをはさんだ通信に対して、イントルーダによる攻撃が成功するかどうかを検証することになる。M は前章で定義した項であり、この中にお互いに共有したいデータを含ませることで、メッセージ認証を定式化する。

検証の手法について述べる。Spi 計算を簡約していく中で、end M が出現したときに、その前に同じラベルを持つ begin M が出現しているかどうかを調べる。あるプロセスがこれを満たしている場合、そのプロセスは safe であると呼ぶ。

これを用いると、任意のイントルーダのプロセス I に対してプロセス $P \mid I$ が safe となるならば、プロセス P にモデル化されたプロトコルのメッセージ認証が保証されることになる。これを、図を用いて説明する。図 1 は、正常な通信を行っている様子を表す。ラベル付きのイベントは、通信の前後に挿入する。図 2 は、イントルーダによって攻撃を受けたが、その後受取人の処理によってそれが検出でき、受取人のプロセスが end イベントを実行する前にデッドロックに陥ったことを表している。この場合は、begin $M1$ に対して end $M1$ がないことになるが、safe である。これは、イントルーダによる攻撃を受取人が検出していることに対応する。最後に図 3 は、イントルーダによってリプレイ攻撃を受けた場合を表している。このとき、1 つの begin $M1$ イベントの実行に対して 2 つの end $M1$ イベントが実行されており、2 回目に実行された end $M1$ イベントに対応する begin $M1$ イベントが存在しないため safe ではない。このような実行系列が存在するプロトコルは、安全なプロトコルとはいえない。イントルーダのプロセスは次節で定義する。

3 検証手法

本検証手法では、Spi 計算で記述されたプロトコルに対して、安全であるかどうかを表す付加情報である型を計算し、それが、空のマルチ集合 [] になったとき、そのプロトコルは安全であると判定する。本章では、項に与えられる型と、プロセスに与えられる型を定義し、次にそれらの型を計算する規則を定義する。そして、イントルーダのプロセスを定義し、最後に、これらの定義が正当であることを示す証明の指針を述べる。

3.1 項の型

項に割り当てられる型は、イントルーダによって盗聴が可能か、どのように使用されうるかによって大きく 3 種類に分けられる。まずイントルーダに盗聴され、あらゆる用途に使用されるおそれのある項の型を Un(untrusted) 型と呼ぶ。この型を持つ項のみがネットワークに流すことができる。次に、盗聴は可能である

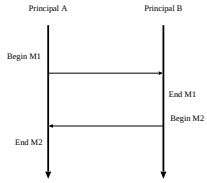

図 1: 正常な通信

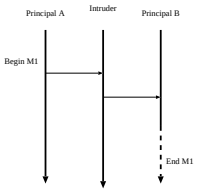

図 2: 攻撃の検出

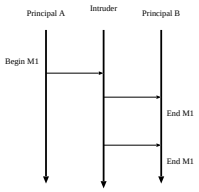

図 3: リプレイ攻撃を受けた場合

が使用目的が限定されているものとして被認証者が暗号化するナンスの型と公開鍵の型がある。その他、盗聴出来ないデータの型として、チャンネル、認証者が暗号化するナンス (チャレンジ)、対称鍵の型・秘密鍵の型がある。また、項の組に割り当てられる型と、タグ付きの項の型についても定義する。

定義 5 (項の型)

$$T ::= \mathsf{Un} | (x : T_1, T_2) | () | \mathsf{ch}(T) | \mathsf{symkey}(T) | \mathsf{prikey}(T) | \mathsf{pubkey}(T) | \mathsf{nonce}\ \ es | \mathsf{challenge}\ es | T_1 + T_2 | \mathsf{sig} | \mathsf{mistake}$$

$(x : T_1, T_2)$ は組の項に割り当てられる型で、x は T_2 の中で束縛されている。() は空の組に割り当てられる型である。$\mathsf{ch}(T), \mathsf{symkey}(T), \mathsf{prikey}(T), \mathsf{pubkey}(T)$ はそれぞれチャンネルの型、対称鍵、秘密鍵、公開鍵の型である。チャンネルの型は、型 T を持つ項を送信するということを意味する。チャンネルにこの型が割り当てられた場合、そのチャンネルを用いた通信はイントルーダに盗聴されない通信であることを意味する。これは、プロセスの内部処理を行う場合等に用いられる。インターネット等、公開の通信路には Un 型が割り当てられる。

鍵の中に出現する型 T は、その鍵で暗号化する際には T 型の項を暗号化でき、また、その鍵で復号すると T 型の項になるという意味である。T 型の項を暗号化する際に用いる公開鍵や秘密鍵と、それらに対応する秘密鍵や公開鍵では、その T が常に一致している必要がある。つまり、$\mathsf{new}(n_1, \mathsf{prikey}\ (T), n_2); P$ というプロセスが実行されると、n_2 は $\mathsf{pubkey}(T)$ という型にならなければならない。これは、後にプロセスの型付け規則として定義する。

また、ナンスの型とチャレンジ・レスポンスの型に含まれる es はプロセスの型であるエフェクトを表す。これについては後述する。

$T_1 + T_2$ は T_1 あるいは T_2 のいづれかの型を持つことを表し、例えば、M が $T_1 + T_2$ 型を持つとき、$\mathsf{Inl}(M)$ は型 T_1 を、$\mathsf{Inr}(M)$ は型 T_2 をそれぞれ持つ。sig は秘密にすべき情報を秘密鍵で署名した場合に、署名された項につく型である。これを公開鍵もしくは対称鍵で暗号化すれば、Un 型となる。mistake は、ナンスまたは公開鍵を公開鍵で暗号化した場合に、その項につく型である。これらはイントルーダが知ることができるデータであり、これを公開鍵で暗号化したものはイントルーダも生成可能である。従って、これらは対称鍵で暗号化するか、秘密鍵で署名しなければネットワークに流すことができない。

3.2 プロセスの型

プロセスの持つ型をエフェクトと呼ぶ。エフェクトは 3 つのアトミックエフェクトのマルチ集合として定義される。以下にエフェクトを定義し、直感的な意味を述べる。

定義 6 (エフェクト) エフェクト es は以下の 3 つのアトミックエフェクトのマルチ集合である。

$$\mathsf{end}\ M \qquad \mathsf{check}\ \ M \qquad \mathsf{checkend}\ M$$

エフェクト $\mathsf{end}\ M$ は、あるプロセスにおいて $\mathsf{begin}\ M$ イベントに対応していない $\mathsf{end}\ M$ イベントが存在する場合につく型である。例えば、以下のプロセスのエフェクトは、項 M が Un 型を持つときに $\mathsf{end}\ M$ となる。

$$\mathsf{out}\ M\ ; \mathsf{end}\ M$$

次に $\mathsf{check}\ \ M$ は、ナンスをチェックしたプロセスが持つエフェクトである。このとき、ナンスは前節で述べたようにエフェクトが情報として付加されており、これらをチェックした場合にはその付加されたエフェクトがキャンセルされる。つまり、以下のプロセスのエフェクトは $\mathsf{check}\ \ M$ となる。ここで、N の型は $\mathsf{nonce}\ \ [\mathsf{end}\ \ L]$、$M$ と L の型は Un であるとする。

$$\mathsf{check}\ N = M; \mathsf{end}\ \ L$$

最後に $\mathsf{checkend}\ M$ は $\mathsf{check}\ \ M$ とほぼ同様の働きがある。check エフェクトはレスポンスをチェックしたときにつくものであるのに対して、$\mathsf{checkend}\ M$ エフェクトはチャレンジをチェックした場合につくエフェクトである。

3.3 項の型付け規則

まず、名前がどのような型をもっているかを表す環境 E を定義する。この環境のもとで、項とプロセスの型付けがなされていく。

定義 7 (環境)

$$E ::= \phi | E, x : \tau$$

また環境の定義域 (domain) を次のように定義する。

$$dom(x_1 : \tau_1, \cdots, x_n : \tau_n) = \{x_1, \cdots, x_n\}$$

次に判定式を定義する。判定式は、環境が正しく定義されているかを判定するもの、型が正しく定義されているかを判定するもの、エフェクトが正しく定義されているかを判定するもの、ある項がある型を持つかを判定するもの、あるプロセスがあるエフェクトを持つかを判定するものと 5 種類がある。

定義 8 (判定式)

$E \vdash \diamond$	good environment
$E \vdash T$	good type
$E \vdash x : T$	good term of type T
$E \vdash es$	good effect
$E \vdash P : es$	good process effect es

続いて、これらの判定式を導出するための規則を定義する。

定義 9 (環境における規則) 環境における規則を次のように定義する。

$$\frac{}{\phi \vdash \diamond} \qquad \frac{E \vdash T}{E, x : T \vdash \diamond}$$

定義 10 (エフェクトにおける規則) エフェクト (プロセスの型) における規則を次のように定義する。

$$\frac{E \vdash \diamond}{E \vdash []} \qquad \frac{E \vdash es \quad E \vdash L : T}{E \vdash es + [\mathsf{end}\ L]}$$

$$\frac{E \vdash es \quad E \vdash N : \mathsf{Un}}{E \vdash es + [\mathsf{check}\ N]} \qquad \frac{E \vdash es \quad E \vdash M : \mathsf{challenge}\ es'}{E \vdash es + [\mathsf{checkend} M]}$$

定義 11 (項の型における規則) 項の型における規則を次のように定義する。

$$\frac{E \vdash \diamond}{E \vdash \mathsf{Un}} \qquad \frac{E \vdash \diamond}{E \vdash ()}$$

$$\frac{E \vdash T \quad E, x : T \vdash U}{E \vdash (x : T, U)} \qquad \frac{E \vdash T}{E \vdash \mathsf{ch}(T)} \qquad \frac{E \vdash \diamond}{E \vdash \mathsf{sig}}$$

$$\frac{E \vdash T}{E \vdash \mathsf{symkey}(T)} \qquad \frac{E \vdash T}{E \vdash \mathsf{prikey}(T)}$$

$$\frac{E \vdash T}{E \vdash \mathsf{pubkey}(T)} \qquad \frac{E \vdash T \quad E \vdash U}{E \vdash T + U}$$

$$\frac{E \vdash es}{E \vdash \mathsf{nonce}\ \ es} \qquad \frac{E \vdash es}{E \vdash \mathsf{challenge}\ es}$$

ここで、型の拡張を行う。前節で説明した、イントルーダが盗聴可能な項に pub という型を割り当てる。これは $\mathsf{Un}, \mathsf{nonce}\ \ es, \mathsf{pubkey}(T, U), \mathsf{mistake}$ の 4 つの型の複合として定義される。また、このうち Un 型を除いたものを steal 型と呼ぶことにする。これらは、イントルーダが盗聴可能であるが使用用途が決まっているために、プロトコルの正規の参加者の署名か対称鍵での暗号化がなければ、ネットワークに流すことができない型である。

定義 12 (拡張型) 拡張型 pub と steal は型から導出され、その導出を演算子 $\triangleright$ で表し、その導出規則を以下のように定義する。

$$\mathsf{Un} \triangleright \mathsf{pub} \qquad \mathsf{pubkey}(T) \triangleright \mathsf{steal}$$
$$\mathsf{nonce}\ \ es \triangleright \mathsf{steal} \qquad \mathsf{mistake} \triangleright \mathsf{steal}$$

$$\frac{T \triangleright \mathsf{steal} \quad U \triangleright \mathsf{steal}}{(x : T, U) \triangleright \mathsf{steal}} \qquad \frac{T \triangleright \mathsf{steal}}{T \triangleright \mathsf{pub}} \qquad \frac{T \triangleright \mathsf{pub} \quad U \triangleright \mathsf{pub}}{(x : T, U) \triangleright \mathsf{pub}}$$

以上の準備のもとで、項の型付け規則を定義する。

定義 13 (項の型付け規則)

$$\frac{E', x : T, E'' \vdash \diamond}{E, x : T, E'' \vdash x : T}$$

$$\frac{E \vdash M : T \quad E \vdash N : U[M/x]}{E \vdash (M, N) : (x : T, U)}$$

$$\frac{E \vdash M : T \quad E \vdash N : \mathsf{prikey}(T) \quad T \triangleright \mathsf{pub}}{E \vdash \{M\}_N : \mathsf{Un}}$$

$$\frac{E \vdash M : T \quad E \vdash N : \mathsf{prikey}(T) \quad T \not\triangleright \mathsf{pub}}{E \vdash \{M\}_N : \mathsf{sig}}$$

$$\frac{E \vdash M : T \quad E \vdash N : \mathsf{pubkey}(T) \quad T \not\triangleright \mathsf{steal}}{E \vdash \{M\}_N : \mathsf{Un}}$$

$$\frac{E \vdash M : T \quad E \vdash N : \mathsf{pubkey}(T) \quad T \triangleright \mathsf{steal}}{E \vdash \{M\}_N : \mathsf{mistake}}$$

$$\frac{E \vdash M : T \quad E \vdash N : \mathsf{symkey}(T)}{E \vdash \{M\}_N : \mathsf{Un}} \qquad \frac{E \vdash \diamond}{E \vdash () : ()}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un}}{E \vdash \{M\}_N : \mathsf{Un}} \qquad \frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un}}{E \vdash (M, N) : \mathsf{Un}}$$

$$\frac{E \vdash M : T \quad E \vdash U}{E \vdash \mathsf{Inl}(M) : T + U} \qquad \frac{E \vdash T \quad E \vdash N : U}{E \vdash \mathsf{Inr}(N) : T + U}$$

$$\frac{E \vdash M : \mathsf{Un}}{E \vdash \mathsf{Inl}(M) : \mathsf{Un}} \qquad \frac{E \vdash M : \mathsf{Un}}{E \vdash \mathsf{Inr}(M) : \mathsf{Un}}$$

3.4 プロセスの型付け規則

プロセスの型付け規則は、大きく 2 つの場合に基づいている。1 つはレスポンスを暗号化する場合であり、もう 1 つはチャレンジを暗号化する場合である。レスポンスを暗号化する場合は、認証者がナンスを Un 型として生成し、それを被認証者に送る。被認証者は受け取ったナンスの型を cast イベントによって nonce es 型に変更し、暗号化して送る。認証者は受け取ったナンスが自分がこのセッションで生成したものかどうかを確認する。この流れを表したものが図 4 である。このことを満たすように型付け規則を定義することで、メッセージ認証が保証されるプロトコルにおいては、begin イベントと end イベントの対応がとれてプロセス全体が safe となる。

次に、チャレンジを暗号化する場合を述べる。この場合には認証者が challenge es 型としてナンスを生成し、暗号化して被認証者に送る。被認証者はこれを Un 型に変換して認証者に送信する。認証者はこれを生成したものと一致しているかを確認する。これを図 5 に図示する。この場合もナンスを用いた場合と同様に、メッセージ認証が保証されるプロトコルについては safe が成り立つ。以上を踏まえた上で型付け規則を定義する。

定義 14 (生成可能型) 以下の型を生成可能型という。

$$\mathsf{Un}, \mathsf{ch}(T), \mathsf{symkey}(T), \mathsf{challenge}\ es$$

生成可能型とは、$\mathsf{new}(x, T)$ イベントで項を生成する際に、その項が持つことができる型である。ナンスの型や、組の型等は持つことができない。また、秘密鍵は $\mathsf{new}(x_1, T, x_2)$ イベントで生成するので、ここには含まず、公開鍵は自由変数として扱うので生成は認めない。

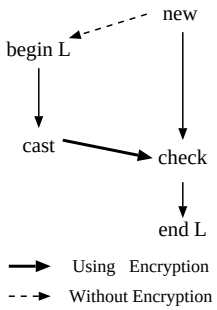

図 4: レスポンスの暗号化の場合

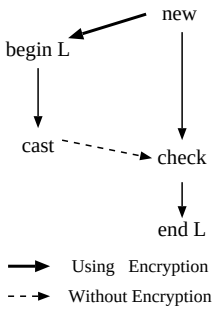

図 5: チャレンジの暗号化の場合

定義 15 (プロセスの型付け規則) プロセスの型付け規則は以下のように定義される。$x, y \notin fn(es)$ とする。
(基本規則)

$$\frac{E \vdash \diamond}{E \vdash \mathsf{STOP} : []}$$

$$\frac{E \vdash P : es}{E \vdash \mathsf{begin}\ \ L; P : es - [\mathsf{end}\ \ L]}$$

$$\frac{E \vdash P : es}{E \vdash \mathsf{end}\ \ L; P : es + [\mathsf{end}\ \ L]}$$

$$\frac{E \vdash P : es_P \quad E \vdash Q : es_Q}{E \vdash P|Q : es_P + es_Q} \qquad \frac{E \vdash P : []}{E \vdash \mathsf{repeat}\ \ P : []}$$

$$\frac{E, x : T \vdash P : es \quad T \text{ は challenge 以外の生成可能型}}{E \vdash \mathsf{new}(x : T); P : es - [\mathsf{check}\ x]}$$

$$\frac{E \vdash x_2 : \mathsf{pubkey}(T) \quad E, x_1 : \mathsf{prikey}(T) \vdash P : es}{E \vdash \mathsf{new}(x_1, \mathsf{prikey}(T), x_2); P : es}$$

$$\frac{E, x : \mathsf{challenge}\ es' \vdash P : es}{E \vdash \mathsf{new}(x : \mathsf{challenge}\ es'); P : es - [\mathsf{checkend}\ x(es')]}$$

(Un 型以外の項をもつプロセスの規則)

$$\frac{E \vdash M_1 : \mathsf{ch}(T) \quad E \vdash M_2 : T \quad E \vdash P : es}{E \vdash \mathsf{out}\ M_1\ M_2; P : es}$$

$$\frac{E \vdash M : \mathsf{ch}(T) \quad E, x : T \vdash P : es}{E \vdash \mathsf{inp}\ M\ x : T; P : es}$$

$$\frac{E \vdash M : (x : T, U) \quad E, x : T, y : U \vdash P : es}{E \vdash \mathsf{split}\ M = (x : T, y : U); P : es}$$

$$\frac{E \vdash M : (x : T, U) \quad E \vdash N : T \quad E, y : U[N/x] \vdash P : es}{E \vdash \mathsf{match}\ M = (N, y : U[N/x]); P : es}$$

$$\frac{E \vdash M : T + U \quad E, x : T \vdash P : es \quad E : y : U \vdash Q : fs}{E \vdash \mathsf{case}\ M = \mathsf{Inl}(x : T); P\ \mathsf{Inr}(y : U); Q : es \cup fs}$$

$$\frac{E \vdash M : \mathsf{Un} \quad (E, x : \mathsf{nonce}\ es') \vdash P : es}{E \vdash \mathsf{cast}\ M\ \mathsf{is}\ x : \mathsf{nonce}\ es'; P : es + es'}$$

$$\frac{E \vdash M : \mathsf{challenge}\ es' \quad E, x : \mathsf{Un} \vdash P : es}{E \vdash M : \mathsf{cast}\ M\ \mathsf{is}\ x : \mathsf{Un}; P : es + es'}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{nonce} \quad es' \quad E \vdash P : es}{E \vdash \mathsf{check}\ M = N; P : es - es' + [\mathsf{check}\ M]}$$

$$\frac{E \vdash M : \mathsf{challenge}\ es' \quad E \vdash N : \mathsf{Un} \quad E \vdash P : es}{E \vdash \mathsf{check}\ M = N; P : es - es' + [\mathsf{checkend}\ M(es')]}$$

$$\frac{\begin{array}{c}E \vdash M : (\mathsf{Un}\ \text{または}\ \mathsf{sig}\text{または}\ \mathsf{mistake}) \\ E \vdash N : \mathsf{symkey}\ (T) \quad E, x : T \vdash P : es\end{array}}{E \vdash \mathsf{decrypt}\ M\ (N)\ \mathsf{to}\ x : T; P : es}$$

$$\frac{\begin{array}{c}E \vdash M : (\mathsf{Un}\ \text{または}\ \mathsf{mistake}) \\ E \vdash N : \mathsf{prikey}\ (T) \quad E, x : T \vdash P : es\end{array}}{E \vdash \mathsf{decrypt}\ M\ (N)\ \mathsf{to}\ x : T; P : es}$$

(イントルーダに関わる型)

$$\frac{E \vdash M_1 : \mathsf{Un} \quad E \vdash M_2 : un \quad E \vdash P : es}{E \vdash \mathsf{out}\ M_1\ M_2; P : es}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E, x : \mathsf{Un} \vdash P : es}{E \vdash \mathsf{inp}\ M\ x : \mathsf{Un}; P : es}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E, x : \mathsf{Un}, y : \mathsf{Un} \vdash P : es}{E \vdash \mathsf{split}\ M = (x : \mathsf{Un}, y : \mathsf{Un}); P : es}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un} \quad E, y : \mathsf{Un} \vdash P : es}{E \vdash \mathsf{match}\ M = (N, y : \mathsf{Un}); P : es}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E, x : \mathsf{Un} \vdash P : es \quad E, y : \mathsf{Un} \vdash Q : fs}{E \vdash \mathsf{case}\ M = \mathsf{Inl}(x : \mathsf{Un}); P\ \mathsf{Inr}(y : \mathsf{Un}); Q : es \cup fs}$$

$$\frac{\begin{array}{c}E \vdash M : (\mathsf{Un}\ \text{または}\ \mathsf{sig}) \\ E \vdash N : \mathsf{pubkey}\ (T) \quad E, x : T \vdash P : es\end{array}}{E \vdash \mathsf{decrypt}\ M\ (N)\ \mathsf{to}\ x : T; P : es}$$

$$\frac{E \vdash M : \mathsf{Un} \quad E \vdash N : \mathsf{Un} \quad E, x : \mathsf{Un} \vdash P : es}{E \vdash \mathsf{decrypt}\ M\ (N)\ \mathsf{to}\ x : \mathsf{Un}; P : es}$$

3.5 イントルーダ

イントルーダは任意の処理ができるということが大前提である。しかしながら、イントルーダにとって意味のない処理もいくつかある。例えば、組の項でないものを split イベントで処理をしたり、ナンスのチェックを行う等、イントルーダ自身のプロセスが停止する恐れのある処理は、しないように制限してもイントルーダの表現力は変化しない。また、拡張型 pub を持つすべての項はイントルーダが持ちうる項から生成可能であるが、公開の通信路から送信することができるデータは Un 型のデータのみとする。これは、正規の参加者が公開の通信路から受信する唯一の項の型であるためである。

また、begin , end イベントと cast, check イベントは、メッセージ認証の検証を行うための仕様を表したイベントであるため、イントルーダのプロセス中には存在してはならない。以上より、イントルーダを以下のように定義する。

定義 16 (イントルーダ) イントルーダのプロセスは、以下を満たす任意のプロセスである。

- begin , end , cast, check イベントを持たない。
- split $M = (x : T, y : U)$ は M が Un 型か $(x : T, U)$ という型のいづれかの場合に実行できる。
- 公開鍵の型を持つ項は、それを用いた暗号化と復号のみに使用することができる。
- 送信できる項は Un 型のみである。(ナンス型の項を公開鍵で暗号化して送信することはできない。)
- 受信や生成等のその他のプロセスにおいて出現する型は Un 型のみである。

3.6 健全性定理

本節では、以上で定義された規則が健全であることの証明の指針を示す。まず、任意のイントルーダのプロセスが [](空のマルチセット) を持つという補題を示す。

補題 1 (イントルーダの型付け可能性) イントルーダのプロセスを I とする。また、$E \triangleq x_1 : T_1, \cdots, x_n : T_n$ とし、各 T_i が Un か pubkey(T) か nonce es のいずれかを持つとき、$E \vdash I : []$ となる。

証明 イントルーダの持つ項の構造に関する帰納法とプロセスの構造に関する帰納法により示すことができる。次に以下の定理が成立する。

定理 1 (Safety)

$$E \vdash P : [] \text{ ならば } P \text{ は safe}$$

上記の定理を証明する際に、重要となるのが次に示す主部簡約定理である。

補題 2 (主部簡約定理) $E \vdash P : es$ のとき、次が成り立つ。

- $P \xrightarrow{\tau} P' \Rightarrow E \vdash P' : es$
- $P \xrightarrow{\mathsf{begin}\ L} P' \Rightarrow E \vdash P' : es + [\mathsf{end}\ L]$
- $P \xrightarrow{\mathsf{end}\ L} P' \Rightarrow E \vdash P' : es - [\mathsf{end}\ L], \mathsf{end}\ M \in es$
- $P \xrightarrow{\mathsf{new}\ x:T} P' \Rightarrow$
  (a) $E, x : T \vdash P' : es + [\mathsf{check}\ \ x], T = \mathsf{Un}$
  (b) $E, x : T \vdash P' : es + [\mathsf{checkend}\ x(es')], T = \mathsf{challenge}\ es'$
  (c) $E, x : T \vdash P' : es, T$ が生成可能型
- $P \xrightarrow{\mathsf{cast}\ x:T} P' \Rightarrow$
  (a) $E, +x : \mathsf{nonce}\ \ es' \vdash P' : es - es'(es \geq es')$
  (b) $E \vdash x : \mathsf{challenge}\ es'$かつ $E, +x : \mathsf{Un} \vdash P' : es - es'(es \geq es')$
- $P \xrightarrow{\mathsf{check}\ x} P' \Rightarrow$
  (a) $E \vdash x : \mathsf{nonce}\ es', E \vdash P' : es + es' - [\mathsf{check}\ \ x], \mathsf{check}\ \ x \in es$
  (b) $E \vdash x : \mathsf{challenge}\ es', E \vdash P' : es + es' - [\mathsf{checkend}\ x(es')], \mathsf{checkend}\ x(es') \in es$
  (c) $E \vdash x : \mathsf{Un}, E \vdash P' : es$

本補題は各々の場合に分けて示すことで証明される。定理 1 により、以下のことが成立する。

定理 2 (健全性) $E = x_1 : T_1 \cdots x_n : T_n$ とする。ある $T_1 \cdots T_n$(ただし、Un, pubkey(T), nonce es のいづれか) に対して $E \vdash P : []$ となるとき、任意のイントルーダ I に対して $P|I$ は safe である。

証明 $dom(E') = \{T_{n+1} \cdots T_{n+m}\}$(ただし、Un, pubkey($T$), nonce es のいづれか) であるとする。また、イントルーダのプロセス I の環境を E, E' とするときに定理 1 より $E, E' \vdash I : []$ となる。さらに、$E \vdash P : []$ ならば $E, E' \vdash P : []$ となり、$E, E' \vdash P|I : []$ となる。従って成立する。

4 Gordon らによる検証手法 [4] との違い

本章では、[4] の検証手法との、アプローチや検証できる性質の違いについて述べる。[4] では、多重の署名や暗号化を行うプロトコルを扱えなかったが、本稿の検証手法では、そのようなプロトコルを扱うことができる。認証と秘密性を両方とも満たすために、ITU X.509 等、実際にそのようなプロトコルは存在する。

[4] では、イントルーダが公開鍵やナンス等を取り扱えるようにするために、部分型関係を導入し、イントルーダが取り扱えるデータの型を Un に統一している。これにより、イントルーダの振る舞いに本質的な制限がなく、様々な攻撃に対する安全性を検証することが可能である。それに対し、本稿では部分型関係を導入していない。本稿が想定しているイントルーダはメッセージのすり替え等は行うことができるが、公開鍵を用いたイントルーダの自発的なメッセージの作成に制限があり、本稿の検証手法ではそのような攻撃 (例えば、Needham-Schroeder 公開鍵プロトコルに対する Lowe の攻撃 [5]) の検出に制限が存在してしまう。なぜなら、イントルーダは公開鍵を使えるにも関わらず、それを適用するデータ型が、その鍵の用途のデータ型に制限されてしまうためである。しかし、部分型関係の規則がないため、検証する際適用できる規則の数が少なく、検証にかかる時間が短いと予想できる。

5 まとめ

本稿では、公開鍵暗号方式を用いたプロトコルのメッセージメッセージ認証の検証手法を提案し、その手法の正当性の証明の指針を示した。

4 章で述べたイントルーダの攻撃の制約を外すためには、[4] のアプローチで見られたような部分型関係を導入する方法がある。我々も部分型関係の導入を検討していたが、現段階では、我々は部分型関係を含む推論規則の健全性の証明ができていない。これを解決することが今後の課題となる。

参考文献

[1] Martin. Abadí and Andrew. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, Vol. 148, No. 1, January 1999.

[2] Martin. Abadí. Secrecy by typing in security protocols. *Journal of the ACM*, Vol. 46, No. 5, pp. 749–786, September 1999.

[3] Andrew D. Gordon and Alan Jeffrey. Authenticity by typing for security protocols. *IEEE Computer Security Foundations Workshop*, pp. 145–159, May 2001.

[4] Andrew D. Gordon and A Jeffrey. Types and effects for asymmetric cryptographic protocols. In *15th IEEE Computer Security Foundations Workshop (CSFW 2002)*, To appear.

[5] G. Lowe. Breaking and fixing the Needham-Schroeder Public-key Protocol using FDR. *Lecture Notes in Computer Science*, Vol. 1055, pp. 147–166, 1996.