〔論　説〕

知識グラフを利用した暗号プロトコル解析

大矢野　　潤

1　はじめに

近年、ソフトウエアの安全性を運用の前段階で形式的方法により検証することでリスクを軽減しようとするアプローチ、すなわちソフトウェアの形式的検証 (Formal Verification) が盛んに研究されている。本論文では、特に電子政府や電子商取引の基礎となる暗号プロトコル検証のための基礎技術を提案する。

公開されたネットワーク上で安全なシステムを構築するためには暗号の利用は不可欠である。暗号の強度は、暗号文の解読に必要な計算量により与えられる。しかし、暗号の強度が十分であっても暗号鍵の受け渡しなどのプロトロルが脆弱であれば、盗聴された暗号鍵により暗号自体を無効にされてしまう。すなわち、安全なシステムの構築には強い暗号と同時に耐攻撃性に優れた暗号プロトコルが必要である。

暗号プロトコルの厳密な耐攻撃性検証を行うためには、攻撃者の能力を明確にする必要があるため、本研究では YANPI 法 (Yielding Arrow Nonce Protocol Inspection Method) を用いる。YANPI 法では筆者が以前提案した同名の方法 [12] を改良した発話可能世界により、SPI 計算 [1] の操作的意味を記述する。SPI 計算は暗号プロトコルの研究分野において標準的に用いられている枠組みであり、本方法を用いることで多数の先行研究の理解を深めることができる。

論文の構成は以下のようになっている。第２節で一般的な暗号プロトコル検証について議論し、記述力を本研究の目的に制限した SPI 計算、SSPI (Simple SPI) 計算を定義する。第３節では YANPI 法、特に発話グラフと発話可能世界を定義し、第４節で実際の使用例を紹介する。最後に、結論と今後の研究について述べる。

2　暗号プロトコル検証

電子政府、電子商取引の仕組みを構築するためには、システムの安全性を脅かす脅威、すなわち盗聴やなりすまし、改ざんの可能性を排除しなければ

ならない。暗号技術はそのような攻撃に対抗するために必要不可欠な技術である。暗号系は必然的にそれ自身が攻撃の対象となるため、耐攻撃性が高く、かつ、利用しやすい暗号系の開発が重要となる。

ここでいう暗号系とは利用している暗号自体、および暗号を利用するための規約、すなわち暗号プロトコルからなる。暗号自体の強度に関する議論は筆者の能力を超えるため、本研究では暗号プロトコルの強度に的をしぼって議論する。

暗号プロトコルの検証には計算理論的アプローチと数理論理的アプローチがある。計算論的アプローチでは、主として確率的多項式時間チューリングマシンの枠組みを用い、暗号自体の脆弱性も考慮にいれた暗号プロトコルの解析を行う。一方、数理論理的アプローチでは、使われている暗号そのものは十分に強いという理想化を行ったうえで、通信路に流れるメッセージを記号化し、形式技法を応用した暗号プロトコルの検証を行う。前述の通り、本研究では数理論理的アプローチをとる。

代表的なプロトコルの記述方法としてプロトコルナレーション (Protocol Narration) がある。プロトコルナレーションとは暗号プロトコルの挙動を暗号メッセージの交換列としてインフォーマルに記述したものであり、その読みやすさから広く用いられている。反面、詳細な挙動を省略しているため、交換されたメッセージの意味付けなどは解釈者に依存してしまう。この曖昧性を排除するため、プロトコルナレーションから形式的な表現形式に一旦変換し、検証をおこなうアプローチ [3, 11] もある。

プロトコルの形式記述として SPI 計算を用いる場合、その正当性の根拠として双模倣性を用いることがある。例えば、秘密を漏洩しないことが確認されたプロセスと検証の対象となるプロセスが同じ振る舞いをする事を示すことができれば、検証の対象が情報を漏洩する事はないと結論づける事ができる。これに対し、時相論理など用いてシステムの性質を記述し、検査対象が与えられた論理式のモデルになっているかを調べる方法もある。この場合、時間に関する性質に関する可読性は当然向上するが、利用する論理によってその表現力、判定に要する計算量、決定可能性などが大きく変動するため、検証する性質によって適切な論理を選択しなければならない。

本研究では、プロトコルナレーションによるプロトコル記述を SSPI 計算の表現式に翻訳する。そして、プロトコル実行の意味として、参加プロセスの発話可能なメッセージをノードとして持つグラフを定義する。このグラフを調べる事でプロトコルがもつ性質の解析を試みる。

2.1 SPI 計算

プロセスの挙動を研究するための枠組みとしては Robin Milner による CCS (Calculus of Communicating Systems) [8] や C. A. R. Hoare の CSP (Communi-

cating Sequential Processes) [4] などが代表的である。CCS は双模倣性 (bisimulation) によりデッドロックやライブロックなどのシステムの持つ性質を証明する。さらに CCS を、通信路が変化するモバイルプロセスに対応するために拡張した枠組みとして Pi 計算 (Pi Calculus) [9] がある。CCS や Pi 計算は簡潔で豊かな表現力を持つが、語彙自体は非常に基本的であるため、応用システムなど複雑なプロセスの記述に用いると見通しが悪くなる。この点を解消するために、M. Abadi, A. D. Gordon 等による Pi 計算の暗号システム記述のための拡張、すなわち SPI 計算 [1] が提案され、現在、SPI 計算を利用した暗号システムの研究が盛んに行われている。

しかし、従来の SPI 計算の枠組みでは、“相手が正しい通信者であると信じる” と言ったような信念情報を陽に扱うことができない。筆者は、この点を克服するために SPI 計算の意味論に、発話可能性を表すグラフを追加する。さらに SPI 計算の ν の使用を制限する。通常、SPI 計算では秘密の通信を表現するために、ν という新しい値を生成する構文上のプリミティブを使用する。例えば、$(\nu c)(\bar{c}\langle M\rangle|c(x))$ という SPI 計算の式は、“新しく通信路 c を生成し、その通信路を経由して一方のプロセスがメッセージ M を送り、他方のプロセスが同名のチャネルを通じてメッセージを受け取る。このとき通信路は通信を始める直前に生成し、その名前は二つのプロセス間でのみ共有されるため、第三者のプロセスはこの通信を傍受できない” と解釈する。しかし、実際はインターネット上のすべてのメッセージは通信主体間に位置する通信機器により傍受可能である。暗号等で通信を保護する目的は、暗号化されたデータは傍受されても、その意味を解読する事ができないとする方が自然であろう。すなわち、メッセージ M を暗号鍵 K で暗号化したメッセージ $\{M\}_K$ が送信された場合、第三者はこの暗号化されたメッセージを受け取る事はできる。メッセージの解読可能性は複合鍵 K^{-1} を持っているかどうかにより決まるのであり、通信路自体の新規性によるものではない。

このことから、SSPI では SPI 計算で使用される通信路を制限するという目的では ν は使用せず、単に「新しいメッセージを作り出す」という意味でのみ使用する。 SPI 計算との混同を避けるため、新しいメッセージの創出を表すために **fresh** を使用する。

3 Yielding Arrow Nonce Protocol Inspection

簡単に言うと、Yielding Arrow Nonce Protocol Inspection Method (以下、YANPI 法) とは、知識生成グラフを伴った SPI 計算である。

定義 3.1 (知識生成グラフ) 知識生成グラフ *(*以下、単に知識グラフ*)* とは、メッセージをノード、メッセージの生成アロー *(Yielding Arrow)* をエッジとして持つラベル付きグラフ $W = (\mathcal{N}, \mathcal{E})$ である。$N, M \in \mathcal{N},\ N \xrightarrow{f} M \in \mathcal{E}$ である

とき、M は N から f を使用して生成されることを表す。なお、グラフ W のノードの集合を $W_{\mathcal{N}}$、エッジの集合を $W_{\mathcal{E}}$ と記述する。

プロセスがメッセージを創出する場合、ランダムに生成したメッセージを除けば、プロセスは今までに獲得した知識から生成できないメッセージはそもそも発話できない。したがって、メッセージを発話できない場合には、そのメッセージの発話を伴う状態遷移は起こらない。YANPI 法では、発話されたメッセージがどのように構成されたのかの根拠を明確にするため、プロセスの入出力規則に知識グラフを書き換える規則を付加する。このグラフの書き換えは機械的に行うことができるため、適当な支援環境の元での解析作業は効率的に行われる。

3.1 Simple SPI 計算

ここでは、簡略化した SPI 計算 (Simple SPI 計算) を定義する。なお、本論文ではオリジナルの SPI 計算からの主な変更点のみ記すことにする。SPI 計算の詳しい説明は [1, 8] を参照されたい。

3.1.1 SSPI:シンタックス

定義 3.2 (関数) 関数適用 $f(x)$ を $x;f$ 、関数合成 $g(f(x))$ を $x;f;g$ と記す。通常の関数合成とは記述の順序が異なる点に注意する。

定義 3.3 (メッセージ) 名前と変数の集合を **Name**、**Var** とし、それぞれの領域を動く変数を $n, c, P \ldots \in$ **Name**, $x, y, z \in$ **Var** とする。なお、名前にはチャネルやプロセスの識別子も含まれるものとする。

このとき、メッセージの集合 **Msg** を次のように定義する。

$$\begin{aligned} M, N, K, L \in \textbf{Msg} \quad ::= \quad & n \mid c \mid P \mid x \\ \mid \quad & (M, N) \\ \mid \quad & \{M\}_K \end{aligned}$$

ここで、(M, N) によりメッセージ M, N のタプル、$\{M\}_K$ は鍵 K でメッセージ M を暗号化したものを表している。

メッセージ M, N から新しいメッセージ (M, N) を作り出す操作に対してその逆の操作を $(M, N);\pi_1 = M$、$(M, N);\pi_2 = N$ とする。同様に、$\{M\}_K$ はメッセージ M を鍵 K によって暗号化したものを表す。つまり、ここでは鍵 K と鍵 K を用いて暗号化する関数 encode(K) を同一視している。したがって $\{M\}_K$ は $K(M)$ つまり、encode(K)(M) と同義である。

通常、 SPI 計算では暗号化の種類として対称鍵、非対称鍵、ハッシュを区別するが、本論文ではそれらを区別しない。対称鍵 K は $K;K = \mathbf{Id}$ (ただし、$\forall x.\mathbf{Id}(x) = x$) と、非対称鍵は $\forall K \exists K^{-1}$ s.t. $K;K^{-1} = K^{-1};K = \mathbf{Id}$、ハッシュ関数は $\forall K \nexists K^{-1}$ s.t. $K;K^{-1} = \mathbf{Id} \vee K^{-1};K = \mathbf{Id}$ など、暗号鍵に対する複合鍵の存在で区別する。メッセージ M の鍵 K を用いた暗号文 $\{M\}_K$ を鍵 K^{-1} で複合する操作は $\{\{M\}_K\}_{K^{-1}} = \{M\}_{K;K^{-1}} = \{M\}_{\mathbf{Id}} = \mathbf{Id}(M) = M$ となる。

定義 3.4 (メッセージ空間) メッセージ空間 **Msg** は最小値 $\bot$、最大値 $\top$ を持つ半順序空間とする。すなわち、

$$\forall M \in \mathbf{Msg}.\bot \leq M \leq \top$$

である。また、$M \neq M' \in \mathbf{Msg}$ かつ $M, M' \neq \bot, \top$ の場合は M と M' の間に順序はつかないものとする。

Msg で、*2* つの要素の最大値を与える演算 $\sqcup$ を次のように定義する。

$$M \sqcup M' = \begin{cases} M & \textbf{if } M' \leq M \\ M' & \textbf{if } M \leq M' \\ \top & \textbf{otherwise} \end{cases}$$

特に、

$$\forall M \in \mathbf{Msg}.\bot \sqcup M = M \sqcup \bot = M$$

である。

定義 3.5 (プロセス) プロセスの集合 **Proc** とその上を動く変数 $P, Q, \ldots \in \mathbf{Proc}$ を次のように定義する。

$$\begin{array}{rcl} P, Q \in \mathbf{Proc} & ::= & \textbf{stop} \\ & | & \textbf{out}\ c\ M; P \\ & | & \textbf{inp}\ c\ x; P \\ & | & P|Q \\ & | & P + Q \\ & | & [M\ \textbf{is}\ N]P \\ & | & !P \\ & | & \textbf{fresh}\ n; P \\ & | & \textbf{let}\ x = M\ \textbf{in}\ P \\ & | & \textbf{case}\ M\ \textbf{of}\ \{x\}_K\ \textbf{in}\ P \\ & | & \textbf{case}\ M\ \textbf{of}\ (x, y)\ \textbf{in}\ P \end{array}$$

これらのプリミィティブの意味は表 1 の通りである。

構文	意味
stop	停止するプロセス
out $c\,M;P$	チャネル c からメッセージ M を出力して P になるプロセス。
inp $c\,x;P$	チャネル c からメッセージを受け取り、P の局所変数 x にそのメッセージを格納したプロセス。
$P\|Q$	プロセス P と Q を合成したプロセス。
$P+Q$	プロセス P もしくは Q のどちらかを選択。
$[M$ **is** $N]P$	M が N と等価である場合に限りプロセス P になる。その他のケースは **stop** と同じ。
$!P$	プロセス P の複製。
fresh $n;P$	P 中の n に新しい値を割当てたプロセス。
let $x=M$ **in** P	P 中の変数 x に M を割り当てたプロセス、すなわち $\{M/x\}P$ 。
case M **of** $\{x\}_K$ **in** P	メッセージ M が $\{x\}_K$ にパターンマッチした場合は P 中の x を $\{M\}_{K^{-1}}$ で置き換えたプロセス、すなわち $\{\{M\}_{K^{-1}}/x\}P$ 。
case M **of** (x,y) **in** P	メッセージ M が (x,y) にパターンマッチした場合、P 中の x,y をそれぞれ対応する値で置き換えたプロセス、すなわち $\{M;\pi_1/x, M;\pi_2/y\}P$ 。

表 1: SSPI:プロセスの意味

3.1.2 SSPI:セマンティクス

ここでは直前に定義した SSPI 計算に対する操作的意味を定義する。通常、SPI 計算の意味はラベル付き遷移システム (LTL) によって与えられる。LTL に関する意味は文献 [1, 8] に譲り、ここでは、入出力を伴う遷移によって変化する知識グラフをその意味に付け加える。

定義 3.6 (プロセス ID) プロセスはそれぞれ識別子 *(PID)* を持つものとし、状態遷移を通じて変化しないものとする。

定義 3.7 (局所変数、大域変数) 局所変数とは他のプロセスから直接参照される事がない変数である。そうでない変数を大域変数と呼ぶ。*PID* P の局所変数 x を x^P と記述する。

定義 3.8 (チャネル) チャネルは大域変数であり、任意のプロセスからアクセス可能である。ここでは、任意のプロセスの局所変数としても解釈可能な変数として定義する。例えばチャネル c がプロセス P と Q により共有される場合、$c^P = c^Q$ という条件のついた双方の局所変数として定義する。

定義 3.9 (入出力) *PID* P の入力プロセス **inp** $c\,x$ は入力チャネル c を通じて受け取ったメッセージを分解したものを現在の知識に局所変数 x^P に追加する。逆に **out** $c\,M$ は現状の知識から新しいメッセージ M を構成しチャネル c から出力する。M のデストラクタを $f, g, \ldots$、Q の局所変数を $N, x, y, \ldots$ とすると入出力に伴う知識の変化は次のようになる。

$$\textbf{out}\ c\,M; P \mid\ \textbf{inp}\ c\,N; Q \to P \mid \{M; f/x, M; g/y, \ldots\}Q$$

3.2 知識グラフ付きセマンティクス

前述の通り、知識グラフとはプロセスの発話可能性の根拠となる。プロセスの実行は知識の獲得にともないグラフの書き換えを起こすため、知識グラフの書き換えをプロセス実行の操作的意味と考えることができる。通常、SPI 計算はプロセスがそれまでに獲得した知識を陽に指定する仕組みを持たないため、ここでは PID P のプロセスに知識世界 W を付け加え、$P : W$ と記述する。また実行のタイミングを明示する場合には、時間 **Time** によるインデックスがついてるものとする。なお、本論文では **Time** を単なる非負整数集合とする。また、特にプロセス P の世界とプロセス Q の世界とを陽に区別したい場合にはそれぞれ W^P, W^Q のように、あるタイミング $T \in$ **Time** での世界と記す場合には W^T のように記す。特に、プロセス P の内部の時間 P_T を指定する場合には W^{P_T} のようにする。

以下、SSPI 計算のそれぞれのプリミティブの意味を定義する。

3.2.1 置換 (Substitution)

定義 3.10 (指示ノード) 知識グラフ W で $x := y$ となるノード y を x が指示するノードと呼び、$W(x) = y$ と表現する。

定義 3.11 (置換) $W[x := M]$ は x が M を *(*辺 $:=$ により*)* 指し示すということ以外は W と同じグラフであるとする。具体的にはグラフ W から $x := N$ の形の辺を取り除いた後 $x := M$ を加える事で得られる。この操作は通常の置換と同じような効果を持つ。

$$W[x := M](y) \quad = \quad \begin{array}{l} M \text{ if } y = x \\ W(y) \text{ otherwise} \end{array}$$

直感的には、$x := y$ により、メッセージ x の発話が メッセージ y によって裏付けられるということを表している。

注意 1 $:=$ の両辺に同一の変数が出てくる場合には注意が必要である。例えば $x := x + 1$ という表現は $x^{P_{T+1}} := x^{P_T} + 1$ のように $:=$ の左右で時間が異なるとする。

知識グラフを用いて置換を伴うプリミティブの意味を次のように定義する。

$$\textbf{let}\ x = M\ \textbf{in}\ P : W \to \{M/x\}P : W[x^P := M]$$

つまり、P 中の（構文上の）変数 x を M で置き換えるのと同時に、知識グラフ W 中の P の局所変数 x^P の値は M を指示する。

case 文の意味も同様に定義できる。

$$\begin{aligned}
\textbf{case}\ M\ \textbf{of}\ \{x\}_K\ \textbf{in}\ P : W &\to \{\{M\}_{K^{-1}}/x\}P : W[x^P := \{M\}_{K^{-1}}] \\
\textbf{case}\ M\ \textbf{of}\ (x, y)\ \textbf{in}\ P : W &\to \{M;\pi_1/x, M;\pi_2/y\}P : \\
&\qquad W[x^P := M;\pi_1, y^P := M;\pi_2]
\end{aligned}$$

定義 3.12 (判定、選択) 変数の値によって処理を変更するためには、判定、選択を用いて例えば $[M\ \textbf{is}\ N]P + [M\ \textbf{is}\ L]Q$ のように記述する。

本論文において判定が必要になるのは “ある変数がある値と等しい場合には計算を継続し、そうでない場合には停止する” という場合のみである。これは **case** 文を用いて次のように計算する事ができる。

$$\textbf{case}\ M\ \textbf{of}\ N\ \textbf{in}\ P : W \to \{M/N\}P : W[N^{P_{T+1}} := N^{P_T} \sqcup M]$$

$N = \bot$ (未定義) であれば M が新しい N となる。従って、$N = \bot$ が明らかな場合には、単に $W[N := M]$ と記述する。さらに、$\bot \lneq N \wedge N \neq M$ すなわち M と N の値が異なる場合には $N \neq M = \top$ (矛盾) となり、その時点で計算が停止する。本論文ではこの方法を限定された判定文として使用する。

定義 3.13 (dom、codom) $:=$ を二項関係と解釈する事により、知識グラフ W の領域 $dom(W)$、値域 $codom(W)$ を次のように定義する。

$$\begin{aligned}
\mathrm{dom}(W) &\triangleq \{x | \exists y \text{ s.t. } (x, y) \in :=\} \\
\mathrm{codom}(W) &\triangleq \{y | \exists x \text{ s.t. } (x, y) \in :=\}
\end{aligned}$$

3.2.2 入出力

入出力のセマンティクスを定義するために、現在の知識グラフ W の次の時間のグラフ W' の定義をする。

定義 3.14 (知識グラフの時間) $pwid, cwid, nwid \in \textbf{Time}$ とはそれぞれ直前、現在、次の時間を意味する。当然、$pwid < cwid < nwid$ である。また、時間で修飾しない知識グラフ W 、およびその変数 x はグラフの現在の時間が省略されているもの、すなわち $W(x)$ は $W^{cwid}(x^{cwid})$ の略記である。

定義 3.15 (次の知識グラフ) 現在の知識グラフ W に対して直後の知識グラフ W' を次のように定義する。

$$\begin{aligned} x' &\triangleq x^{nwid} \\ W' &\triangleq W \cup \{x^{nwid} := x | x \in \mathit{dom}\,(W)\} \end{aligned}$$

また、補助的な記法として、直前の変数を表す ‘x を次のように定義する。

$$‘x \triangleq x^{pwid}$$

$W(x)$ の x は x^{cwid} を、$W'(x)$ の x は x^{nwid} を参照している事に注意する。

以上を利用して、入出力のセマンティクスを定義する。

定義 3.16 (入出力)

$$\begin{aligned} (\ \textbf{inp}\ \ c\,x; P) : W &\rightarrow (\{c/x\}P) : W'[x^P := c^P] \\ (\ \textbf{out}\ \ c\,M; Q) : W &\rightarrow Q : W'[c^Q := M^Q] \end{aligned}$$

3.2.3 フレッシュ(fresh)

fresh n とは新たに生成されたメッセージ n を表現する。構文 **fresh** に対応するメッセージ生成の関数を o とする。必要に応じてメッセージの唯一性を保証するために発話グラフの ID (=時間)、PID 、生成されたメッセージの用途などを引数とすることができる。

定義 3.17 (fresh) **fresh** $n; P$ とは、P に存在する変数 n に新しいユニークな値を割り当てる。

$$(\ \textbf{fresh}\ n; P) : W \rightarrow P : W[n := o(\mathit{args})]$$

3.2.4 合成

プロセスの合成は次のように定義される。

$$\begin{aligned} P : W^P | Q : W^Q &\triangleq (P|Q) : W^{P|Q} \\ &\quad \text{where } W^{P|Q} \triangleq W^P \cup W^Q \end{aligned}$$

知識グラフの和は通常のグラフの和として解釈する。この合成されたプロセスが通信をした場合には、お互いのプロセスの時間が一致したと解釈するため、入出力が行なわれた直後のそれぞれの世界の時間は一致する。

3.3 プロセスの通信の例

“あるプロセス (PID P) が新たなメッセージ M を生成し、鍵 K を使用して暗号化したメッセージを PID Q のプロセスに送信したのち停止する。プロセス Q は受け取ったメッセージを鍵 K^{-1} を使って複合し、Q の変数 x に格納した後に停止する。” という状況を SSPI で記述すると次のようになる。

$$
\begin{aligned}
P &\triangleq \textbf{fresh}\ M;\ \textbf{out}\ \ c\,\{M\}_K;\ \textbf{stop} \\
Q &\triangleq \textbf{inp}\ \ c\,N;\ \textbf{case}\ N\ \textbf{of}\ \{x\}_K\ \textbf{in}\ \ \textbf{stop}
\end{aligned}
$$

これを通信が始まる前の世界 W とともに遷移させてみる。ただし W の時刻は cwid $= 0$ とする。また、可読性のため、知識世界の時間は矛盾のない範囲で省略している。

$$
\begin{aligned}
(P|Q) : W \quad &\triangleq \quad (\ \textbf{fresh}\ M; \ldots |Q) : W \\
&\rightarrow \quad (\ \textbf{out}\ \ c\,\{M\}_K;\ \textbf{stop}\ |Q) : W[M := o(0)] \\
&= \quad (\ \textbf{out}\ \ c\,\{M\}_K;\ \textbf{stop}\ |\ \textbf{inp}\ \ c\,N; \ldots) : W[M := o(0)] \\
&\rightarrow \quad (\ \textbf{stop}\ |\ \textbf{case}\ c\ \textbf{of}\ \{x\}_K\ \textbf{in}\ \ \textbf{stop}\ ) : \\
&\qquad (W[M := o(0)])'[c^P := \{M\}_K, N^Q := c^Q] \\
&\rightarrow \quad (\ \textbf{stop}\ |\ \textbf{stop}\ ) : \\
&\qquad (W[M := o(0)])'[c^P := \{M\}_K, N^Q := c^Q, x^Q := \{N^Q\}_{K^{-1}}]
\end{aligned}
$$

ここでプロセスが停止した際の世界は $(W[M := o(0)])'[c^P := \{M\}_K, N^Q := c^Q, x^Q := \{N^Q\}_{K^{-1}}]$ となり、この世界で Q の局所変数 x^Q を解析すると次の事がわかる。

$$
\begin{aligned}
x^Q &:= \{N^Q\}_{K^{-1}} \\
&:= \{c^Q\}_{K^{-1}} \quad \because\ N^Q := c^Q \\
&:= \{\{M\}_K\}_{K^{-1}} \quad \because\ c^Q = c^P := \{M\}_K \\
&= \{M\}_{K;K^{-1}} \\
&= \{M\}_{\mathbf{Id}} \\
&= M \\
&:= o(0)
\end{aligned}
$$

$X^Q := \ldots := o(0)$ であること、すなわち、Q は P が発行したフレッシュなメッセージ $o(0)$ を受け取って終了していることがわかる。

3.4 知識世界

いよいよ知識世界を定義する。

定義 3.18 (知識世界) 知識世界とは知識グラフ W を生成元とする推移的閉包であり、$\mathcal{W}$ と書く。

本論文では具体的に推移的閉包 $\mathcal{W}$ を求めるのではなく、後述する発話可能性を用いて間接的に使用する。これは知識グラフ W が有限であっても、推移的閉包 $\mathcal{W}$ は一般に無限のサイズを持つため、あらかじめ $\mathcal{W}$ を計算する事は機械的な検証にそぐわないためである。

注意 2 知識世界と知識グラフを混同しないように注意したい。知識グラフは実際に獲得した知識を表すグラフであり、知識世界とは得られた知識から生成することが可能なすべてのメッセージからなるノードと、その生成関係を表すエッジからなるグラフである。

3.4.1 基本的な推移

まず、知識世界で使用する基本的な推移関係について定義する。

定義 3.19 (基本推移規則)

$$\frac{x := y \quad y := z}{x := z}\ (:=\text{Trans}) \qquad \frac{x \xrightarrow{f} y \quad y \xrightarrow{g} z}{x \xrightarrow{f;g} z}\ (\text{FuncComp})$$

$$\frac{f := g \quad x \xrightarrow{f} y \quad x \xrightarrow{g} z}{y := z}\ (:=\text{Func1}) \qquad \frac{x := z \quad x \xrightarrow{f} y \quad z \xrightarrow{f} w}{y := w}\ (:=\text{Func2})$$

同様に基本的な等号関係を定義する。

定義 3.20 (等号関係)

$$(M_1, \ldots, M_i, \ldots); \pi_i = M_i\ (\text{Prod})$$

$$K; K^{-1} = K^{-1}; K = \mathbf{Id}\ (\text{Inv}) \qquad M; \mathbf{Id} = \{M\}_{\mathbf{Id}} = M\ (\text{Id})$$

3.4.2 発話可能性

もう一度 $\{M\}_K$ について考えてみよう。これは $\{M\}_K$ というメッセージは鍵 K の暗号を複合するための鍵 K^{-1} を知っている、すなわち発話できるときに限りメッセージ M を発話できるということを意味している。

定義 3.21 (発話可能) プロセス P がメッセージ M を発話可能であるということを $P \Rightarrow M$ と記述する。特に、P が世界 $\mathcal{W}$ においてメッセージ M を発話可能であるということを明示する場合は $P \overset{\mathcal{W}}{\Rightarrow} M$ と記述する。

“P がメッセージ M と鍵 K の両方を発話できるとき、メッセージ $\{M\}_K$ を発話できる” という事は次のように記述できる。

$$\frac{P \Rightarrow M \quad P \Rightarrow K}{P \Rightarrow \{M\}_K}$$

発話可能性の基本的アイデアは [2] 中の Entailment relation に基づいている。しかし、Entailment relation は単にメッセージから新たなメッセージを作り出すルールを規定しているのに対し、本研究での発話可能性はどの世界で、どのプロセスが発話できるのかを陽に指定する。この事により、あるプロセスには解読可能なメッセージが別のプロセスには解読不可能であるというような状況を明示的に記述することができる。

P, Q によりプロセスを $\mathcal{W}$ で発話世界を表すものとする。

定義 3.22 (発話可能性：基本ルール)

$$\frac{}{P \Rightarrow \pi_{i\in\{1,2,\ldots\}}}\ (\pi) \qquad \frac{}{P \Rightarrow x^P}\ \text{(Local)} \qquad \frac{}{\forall P, Q \in \mathbf{Proc}.P \Rightarrow Q}\ \text{(PID)}$$

$$\frac{P \Rightarrow M}{P \overset{\mathcal{W}}{\Rightarrow} M}\ \text{(Comm)} \qquad \frac{\text{wid} < \text{wid}' \quad P \overset{\mathcal{W}^{\text{wid}} M}{\Longrightarrow}}{P \overset{\mathcal{W}^{\text{wid}'} M}{\Longrightarrow}}\ \text{(Mono)} \qquad \frac{P \overset{\mathcal{W}}{\Rightarrow} x \quad x := y}{P \Rightarrow y}\ (:=)$$

$$\frac{P \overset{\mathcal{W}}{\Rightarrow} M \quad P \overset{\mathcal{W}}{\Rightarrow} (x \xrightarrow{f} y)}{P \overset{\mathcal{W}}{\Rightarrow} M; f}\ \text{(Comp1)} \qquad \frac{P \overset{\mathcal{W}}{\Rightarrow} (x \xrightarrow{f} y) \quad P \overset{\mathcal{W}}{\Rightarrow} (y \xrightarrow{f} z)}{P \overset{\mathcal{W}}{\Rightarrow} (x \xrightarrow{f;g} z)}\ \text{(Comp2)}$$

上の基本ルールを利用して、以下のルールを導出する事ができる。

$$\frac{P \overset{\mathcal{W}}{\Rightarrow} M \quad P \overset{\mathcal{W}}{\Rightarrow} K}{P \overset{\mathcal{W}}{\Rightarrow} \{M\}_K}\ \text{(Enc)} \qquad \frac{P \overset{\mathcal{W}}{\Rightarrow} M \quad P \overset{\mathcal{W}}{\Rightarrow} N}{P \overset{\mathcal{W}}{\Rightarrow} (M, N)}\ \text{(Tupple)}$$

$$\frac{P \overset{\mathcal{W}}{\Rightarrow} (M, N)}{P \overset{\mathcal{W}}{\Rightarrow} M \wedge P \overset{\mathcal{W}}{\Rightarrow} N}\ \text{(Proj)} \qquad \frac{P \overset{\mathcal{W}}{\Rightarrow} \{M\}_K \quad P \overset{\mathcal{W}}{\Rightarrow} K^{-1}}{P \overset{\mathcal{W}}{\Rightarrow} M}\ \text{(Dec)}$$

これらの規則はよく用いられるため、略記として自由に用いる。例として Proj 規則 、および Dec 規則の導出過程を示す。

$$\frac{\dfrac{P \overset{\mathcal{W}}{\Rightarrow} (M, N) \quad \dfrac{\dfrac{}{P \Rightarrow \pi_1}\ (\pi)}{P \overset{\mathcal{W}}{\Rightarrow} \pi_1}\ \text{(Comm)}}{P \overset{\mathcal{W}}{\Rightarrow} (M, N); \pi_1}\ \text{(Comp1)}}{P \overset{\mathcal{W}}{\Rightarrow} M}\ \text{(Prod)} \qquad \frac{\dfrac{\dfrac{P \overset{\mathcal{W}}{\Rightarrow} \{M\}_K \quad P \overset{\mathcal{W}}{\Rightarrow} K^{-1}}{P \overset{\mathcal{W}}{\Rightarrow} \{M\}_{K;K-1}}\ \text{(Enc)}}{P \overset{\mathcal{W}}{\Rightarrow} \{M\}_{\mathbf{Id}}}\ \text{(Inv)}}{P \overset{\mathcal{W}}{\Rightarrow} M}\ \text{(Id)}$$

最後に、ノードの値について定義する。

定義 3.23 (ノードの値) ある世界 $\mathcal{W}$ において、$x := y \wedge \nexists z \in \mathcal{W}_{\mathcal{N}}$ *s.t.* $y{:=}z$ であるとき、ノード y を世界 $\mathcal{W}$ における x の値と呼び、$\mathcal{W}(x) = y$ と記述する。

4 プロトコルナレーションへの応用

ここでは、前節で作った知識世界を Needham Schroeder 公開鍵プロトコルへの Lowe の攻撃のプロトコルナレーションに適用する。

4.1 Needham Schroeder 公開鍵プロトコル

1978 年、 Roger M. Needham と Michael D. Schroeder により、セッションキーなど共通の秘密 (Nonce) の安全な受け渡しを目的とした公開鍵プロトコル (Needham Schroeder 公開鍵プロトコル：以下、NS プロトコル) [10] が提案された。

NS プロトコルは次のような手順で鍵交換を行う。

1. $A \rightarrow B{:}\{A, N_A\}_{K_B}$
2. $B \rightarrow A{:}\{N_A, N_B\}_{K_A}$
3. $A \rightarrow B{:}\{N_B\}_{K_B}$

ここで、$A \rightarrow B{:}\{A, N_A\}_{K_B}$ とは、“A が B に対してメッセージ $\{A, N_A\}_{K_B}$ を発話する” ことを意味する。このとき、次のことがわかる。

- 仮定 1：メッセージ $\{A, N_A\}$ を K_B で暗号化して得られるメッセージ $\{A, N_A\}_{K_B}$ は、${K_B}^{-1}$ でしか解読できない。
- 仮定 2：${K_B}^{-1}$ は B しか知らない。
- 仮定 3：メッセージを作った主体 A と B は秘密を漏らさない。
- 結論 1：仮定 1、2、3 より、メッセージ $\{A, N_A\}$ を解読できる主体は、自分と B だけであると「メッセージを作った主体 A が」信じることができる。
- 仮定 4：$\{N_A, N_B\}_{K_A}$ を作れる主体は $\{N_A, N_B\}$ を知っている主体だけである。
- 結論 2：結論 1、仮定 4 より $\{N_A, N_B\}_{K_A}$ を送り返した主体は (A でないとすると) B である。
- 仮定 5：メッセージ $\{N_A, N_B\}$ を K_A で暗号化して得られるメッセージ $\{N_A, N_B\}_{K_A}$ は ${K_A}^{-1}$ でしか解読できない。

- 結論 3：$\{N_A, N_B\}_{K_A}$ から $\{N_A, N_B\}$ を取り出せる主体は A (自分) だけである。

以上より、“「もし、B が正直であれば、$\{N_A, N_B\}_{K_A}$ を送り返した主体は B であり、B のノンスは N_B である。また、通信の間に秘密は漏れていない」と A が信じる” ことは妥当である。

4.2 Lowe の攻撃

1978 年に提唱され、17 年間安全であると考えられていた NS プロトコルが Gavin Lowe によりなりすまし攻撃の存在と、Lowe 自身による修正バージョン (Needham Schroeder Lowe Protocol: 以下、NSL プロトコル) が提案された [6, 7]。

Lowe の攻撃とは次のようなものである。ただし、C を侵略者、$C(A)$ で主体 A になりすました C を意味する。

1. $A \to C : \{A, N_A\}_{K_C}$
   A が C に話しかけてしまう。

2. $C(A) \to B : \{A, N_A\}_{K_B}$
   C は A から得た情報を利用して A になりすまし、B に発話する。

3. $B \to C(A) : \{N_A, N_B\}_{K_A}$
   B は A から話しかけられたものと信じ、A との間の秘密 N_B を含んだメッセージを C に返す。

4. $C \to A : \{N_A, N_B\}_{K_A}$
   C は (このメッセージを読むことができないため) そのまま A に渡す。

5. $A \to C : \{N_B\}_{K_C}$
   A はこのメッセージは C から来たものと信じ、確認としてそのメッセージ内に入っていた B の秘密を C に知らせてしまう。

6. $C(A) \to B : \{N_B\}_{K_B}$
   C は A のふりをして B に B の秘密 N_B を理解したことを伝える。

この結果、B は A と会話が成立しているものと思い込んでしまい、今後 A にしか伝えてはならない会話をノンス N_B を用いて開始してしまう。C はノンス N_B を手に入れているため、このセッションすべてのメッセージを傍受、改ざんすることができてしまう。これが Lowe の攻撃である。

この攻撃を防ぐために Lowe 自身が改良版した NSL プロトコルとは次のようなものである。

1. $A \to B : \{A, N_A\}_{K_B}$

2. $B \to A : \{B, N_A, N_B\}_{K_A}$

3. $A \to B : \{N_B\}_{K_B}$

NS プロトコルとの違いは、2 行目のメッセージを発話した主体は B であるということを示す項目を追加し、A に通信相手に関する錯誤を気づかせる部分のみである。このように、暗号プロトコルの検証は非常にデリケートである。

なお、入力と現在の知識との錯誤を気づかせるという処理はナレーション中で明示的に出現せず、YANPI 法では暗黙のうちに **case** 文の処理に含めている。すなわち、入力 c と直前の知識 N との錯誤を調べるために、

$$N := \text{`}N \sqcup c$$

という記述を追加する。すなわち $\bot \lneq \text{`}N, c$ かつ $\text{`}N \neq c$ の場合は $N = \top$ となり、異変に気がついたプロセスはそれ以降の計算を打ち切る。

4.3 Needham-Schroeder プロトコルナレーションの SSPI 表現

ここでは、NS プロトコルナレーションを SSPI に変換し、攻撃者のプロセスと組み合わせて実行したときに Lowe の攻撃がどのように成功するのかを解析する。

以下、簡単に NS プロトコルの記述に必要な前提を列挙する。

主体: プロトコルに参加する主体 (プロセス) は A, B, C とする。

役割: プロトコルに参加する発話者 I (イニシエータ)、応答者 R (レスポンダ)、そして攻撃者 P (ペネトレータ) がある。ここでは、プロセス A が役割 I を、B が R、C が P を演じる。

鍵: $\mathbf{Pb}(A)$ は A の公開鍵を、$\mathbf{Pb}^{-1}(A)$ で A の秘密鍵を表す。また、すべての公開鍵で暗号化したメッセージは対応する秘密鍵で、逆に秘密鍵で暗号化したメッセージは公開鍵で複合できる。

$$\forall P \in \mathbf{Proc}, \mathbf{Pb}(P); \mathbf{Pb}^{-1}(P) = \mathbf{Pb}^{-1}(P); \mathbf{Pb}(P) = \mathbf{Id}$$

また、すべての参加者はすべての参加者の公開鍵を知っている、

$$\forall P, Q \in \mathbf{Proc}, P \Rightarrow \mathbf{Pb}(Q)$$

すべての参加者は自分の秘密鍵を知っている、

$$\forall P \in \mathbf{Proc}, P \Rightarrow \mathbf{Pb}^{-1}(P)$$

などが成り立つと仮定する。

信念: それぞれの主体、例えば A がある時間 $A^T \in$ **Time** で信じている事柄 N を N^{A_T} とする。もしくは、単に A が信じている N を N^A と表す。信じている事柄には表 2 のようなものがある。

信念	意味
Init	イニシエータ。例えば $\text{Init}^{A_0} := A$ とは “A は初期状態 0 でイニシエータが自分 (A) であると信じている” ということを表している。
NI	イニシエータのノンス。
Resp	レスポンダ。
NR	レスポンダのノンス。

表 2: 信念項目

4.3.1 イニシエータ A の知識グラフ

まず、イニシエータ A の SSPI 表現は次のようになる。

$$
\begin{aligned}
A0 &\triangleq \textbf{fresh}\ \text{NI};\ A0' \\
A0' &\triangleq \textbf{out}\ \ c\ \{\text{Init}, \text{NI}\}_{\mathbf{Pb}(\text{Resp})};\ A1 \\
A1 &\triangleq \textbf{inp}\ \ c\ z;\ A1' \\
A1' &\triangleq \textbf{case}\ z\ \textbf{of}\ \{\text{NI}, \text{NR}\}_{\mathbf{Pb}(\text{Init})};\ A2 \\
A2 &\triangleq \textbf{out}\ \ c\ \{\text{NR}\}_{\mathbf{Pb}(\text{Resp})};\ A3 \\
A3 &\triangleq \textbf{stop}
\end{aligned}
$$

つぎに、参加者 A の時間 T を A_T と表し、時間 A_T の A の知識グラフを W^{A_T} と記述することにする。特に参加者の初期状態、すなわち時間 0 の知識グラフを W^{A_0} とする。

$$
\begin{aligned}
W^{A_0} \quad = \quad & \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}^{-1}(A) := {K_A}^{-1}, \mathbf{Pb}(C) := K_C, \\
& \text{Init}^{A_0} := A, \text{Resp}^{A_0} := C, \text{NI}^{A_0} := \bot, \text{NR}^{A_0} := \bot\}
\end{aligned}
$$

Lowe の攻撃を簡単に説明するため、ここでは、 A は自分を発話者、応答者を C と仮定している。

初期状態 W^{A_0} の元、 SSPI 表現を実行すると次のように知識が変化する。

$$
\begin{aligned}
A0 : W^{A_0} &= (\ \textbf{fresh}\ \mathrm{NI}; A0') : W^{A_0} \\
&\to A0' : W^{A_0}[\mathrm{NI} := N_A] \text{ where } N_A = o(A_0, \mathrm{NI}) \\
&= (\ \textbf{out}\ \ c\ \{\mathrm{Init}, \mathrm{NI}\}_{\mathbf{Pb}(\mathrm{Resp})}; A1) : W^{A_0}[\mathrm{NI} := N_A] \\
&\to A1 : W^{A_1}[c := \mathrm{m0}] \\
&\quad \text{where } \mathrm{m0} = \{\mathrm{Init}, \mathrm{NI}\}_{\mathbf{Pb}(\mathrm{Resp})}, W^{A_1} = (W^{A_0}[\mathrm{NI} := N_A])' \\
&= (\ \textbf{inp}\ \ c\ z; A1') : W^{A_1}[c := \mathrm{m0}] \\
&\to A1' : W^{A_2}[z := c] \quad \text{where } W^{A_2} = (W^{A_1}[c := \mathrm{m0}])' \\
&= (\ \textbf{case}\ z\ \textbf{of}\ \{\mathrm{NI}, \mathrm{NR}\}_{\mathbf{Pb}(\mathrm{Init})}; A2) : W^{A_2}[z := c] \\
&\to (\{\{z\}_{\mathbf{Pb}^{-1}(\mathrm{Init})}; \pi_1/\mathrm{NI}, \{z\}_{\mathbf{Pb}^{-1}(\mathrm{Init})}; \pi_2/\mathrm{NR}\}A2) : W^{A_2}[z := c] \\
&\to A2 : W^{A_2}[z := c, \\
&\quad \mathrm{NI} := \text{`}\mathrm{NI} \sqcup \{z\}_{\mathbf{Pb}^{-1}(\mathrm{Init})}; \pi_1, \mathrm{NR} := \text{`}\mathrm{NR} \sqcup \{z\}_{\mathbf{Pb}^{-1}(\mathrm{Init})}; \pi_2] \\
&= (\ \textbf{out}\ \ c\ \{\mathrm{NR}\}_{\mathbf{Pb}(\mathrm{Resp})};\ \textbf{stop}\ ) : W^{A_2}[\ldots] \\
&\to \ \textbf{stop}\ : W^{A_3}[c := \mathrm{m2}] \text{ where } \mathrm{m2} = \{\mathrm{NR}\}_{\mathbf{Pb}(\mathrm{Resp})}, W^{A_3} = (W^{A_2}[\ldots])'
\end{aligned}
$$

この実行によりプロセス A の終了時 A_3 の世界 (単に W^A とする) は次のようになる。

$$
\begin{aligned}
W^A \ = \ & \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}^{-1}(A) := {K_A}^{-1}, \mathbf{Pb}(C) := K_C, \\
& \mathrm{Init}^{A_0} := A, \mathrm{Resp}^{A_0} := C, \mathrm{NI}^{A_0} := N_A, \mathrm{NR}^{A_0} := \bot, \\
& {c}^{A_1} := \mathrm{m0}, \\
& \mathrm{Init}^{A_1} := \mathrm{Init}^{A_0}, \mathrm{Resp}^{A_1} := \mathrm{Resp}^{A_0}, \mathrm{NI}^{A_1} := \mathrm{NI}^{A_0}, \mathrm{NR}^{A_1} := \mathrm{NR}^{A_0}, \\
& z^{A_2} := {c}^{A_2}, \\
& \mathrm{Init}^{A_2} := \mathrm{Init}^{A_1}, \mathrm{Resp}^{A_2} := \mathrm{Resp}^{A_1}, \\
& \mathrm{NI}^{A_2} := \mathrm{NI}^{A_1} \sqcup \{z^{A_2}\}_{\mathbf{Pb}^{-1}(\mathrm{Init})}; \pi_1, \mathrm{NR}^{A_2} := \mathrm{NR}^{A_1} \sqcup \{z^{A_2}\}_{\mathbf{Pb}^{-1}(\mathrm{Init})}; \pi_2, \\
& \mathrm{Init}^{A_3} := \mathrm{Init}^{A_2}, \mathrm{Resp}^{A_3} := \mathrm{Resp}^{A_2}, \mathrm{NI}^{A_3} := \mathrm{NI}^{A_2}, \mathrm{NR}^{A_3} := \mathrm{NR}^{A_2}, \\
& {c}^{A_3} := \mathrm{m2}\} \\
\text{where} \quad & N_A = o(A_0, \mathrm{NI}), \mathrm{m0} = \{\mathrm{Init}^{A_1}, \mathrm{NI}^{A_1}\}_{\mathbf{Pb}(\mathrm{Resp}^{A_1})}, \mathrm{m2} = \{\mathrm{NR}^{A_3}\}_{\mathbf{Pb}(\mathrm{Resp}^{A_3})}
\end{aligned}
$$

最後に変数とプロセス実行の意味を定義する。

定義 4.1 (変数の意味) 変数 x の知識世界 $\mathcal{W}$ での解釈を次のように定める。

$$\|x\|^{\mathcal{W}} = a \text{ where } a \in \{y | x := y\} \wedge \nexists b \in \mathcal{W}_{\mathcal{N}}.a := b$$

これを $\mathcal{W}$ での x の解釈とする。

定義 4.2 (知識世界の意味) 知識世界 $\mathcal{W}$ のすべての変数の意味を集めたものをその世界の意味 $\|\mathcal{W}\|$ とする。すなわち次のようになる。

$$\|\mathcal{W}\|(x) = \|x\|^{\mathcal{W}}$$

プロセス A が実行した後の世界の意味、すなわちプロセス A の実行の意味は次のようになる。

$$\begin{aligned}
\|\mathcal{W}^A\| \quad = \quad & \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}^{-1}(A) := {K_A}^{-1}, \mathbf{Pb}(C) := K_C, \\
& \mathrm{Init}^{A_0} := A, \mathrm{Resp}^{A_0} := C, \mathrm{NI}^{A_0} := N_A, \mathrm{NR}^{A_0} := \bot, \\
& \mathrm{Init}^{A_1} := A, \mathrm{Resp}^{A_1} := C, \mathrm{NI}^{A_1} := N_A, \mathrm{NR}^{A_1} := \bot, \\
& c^{A_1} := \{A, N_A\}_{K_C}, \\
& z^{A_2} := c^{A_2}, \\
& \mathrm{Init}^{A_2} := A, \mathrm{Resp}^{A_2} := C, \\
& \mathrm{NI}^{A_2} := N_A \sqcup \{c^{A_2}\}_{{K_A}^{-1}}; \pi_1, \mathrm{NR}^{A_2} := \{c^{A_2}\}_{{K_A}^{-1}}; \pi_2, \\
& \mathrm{Init}^{A_3} := A, \mathrm{Resp}^{A_3} := C, \\
& \mathrm{NI}^{A_3} := N_A \sqcup \{c^{A_2}\}_{{K_A}^{-1}}; \pi_1, \mathrm{NR}^{A_3} := \{c^{A_2}\}_{{K_A}^{-1}}; \pi_2, \\
& c^{A_3} := \{\{c^{A_2}\}_{{K_A}^{-1}}; \pi_2\}_{K_C}\}
\end{aligned}$$

特にプロセスの実行が終わったときの変数の値に注目してみる。

$$\begin{aligned}
\|\mathcal{W}^{A_3}\|(\{\mathrm{Init}, \mathrm{Resp}, \mathrm{NI}, \mathrm{NR}\}) \quad = \quad & \{\mathrm{Init} := A, \mathrm{Resp} := C, \\
& \mathrm{NI} := N_A, \mathrm{NR} := \{c^{A_2}\}_{{K_A}^{-1}}; \pi_2\}
\end{aligned}$$

プロセス A はイニシエータを A、レスポンダを C、イニシエータの秘密を N_A、そしてレスポンダのノンスを $\{c^{A_2}\}_{{K_A}^{-1}}; \pi_2$ すなわち、入力されたメッセージを自分の秘密鍵でデコードしたものの 2 番目の項目であると信じている事がわかる。

4.3.2 レスポンダ B の知識グラフ

同様にレスポンダの役を演じる B の知識グラフを見てみる。

まず、B の SSPI 表現は次のようになる。

$$\begin{aligned}
B0 \quad & \triangleq \quad \textbf{inp}\ \ c\ z; B0' \\
B0' \quad & \triangleq \quad \textbf{case}\ z\ \textbf{of}\ \{\mathrm{Init}, \mathrm{NI}\}_{\mathbf{Pb}(\mathrm{Resp})}; B1 \\
B1 \quad & \triangleq \quad \textbf{fresh}\ \mathrm{NR}; B1' \\
B1' \quad & \triangleq \quad \textbf{out}\ \ c\ \{\mathrm{NI}, \mathrm{NR}\}_{\mathbf{Pb}(\mathrm{Init})}; B2 \\
B2 \quad & \triangleq \quad \textbf{inp}\ \ c\ z; B2' \\
B2' \quad & \triangleq \quad \textbf{case}\ z\ \textbf{of}\ \{\mathrm{NR}\}_{\mathbf{Pb}(\mathrm{Resp})}; B3 \\
B3 \quad & \triangleq \quad \textbf{stop}
\end{aligned}$$

知識グラフの初期状態 W^{B_0} を次のように定める。

$$\begin{aligned} W^{B_0} \quad = \quad & \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}(B) := K_B, \mathbf{Pb}^{-1}(B) := {K_B}^{-1}, \\ & \mathrm{Init}^{B_0} := \bot, \mathrm{Resp}^{B_0} := B, \mathrm{NI}^{B_0} := \bot, \mathrm{NR}^{B_0} := \bot\} \end{aligned}$$

プロセス B が終了した時点の世界の解釈は次のようになる。

$$\begin{aligned} \|\mathcal{W}^B\| \quad = \quad & \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}^{-1}(B) := {K_B}^{-1}, \mathbf{Pb}(B) := K_B, \\ & \mathrm{Init}^{B_0} := \bot, \mathrm{Resp}^{B_0} := B, \mathrm{NI}^{B_0} := \bot, \mathrm{NR}^{B_0} := \bot, \\ & {z}^{B_1} := {c}^{B_1}, \\ & \mathrm{Init}^{B_1} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_1, \mathrm{Resp}^{B_1} := B, \\ & \mathrm{NI}^{B_1} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_2, \mathrm{NR}^{B_1} := N_B, \\ & \mathrm{Init}^{B_2} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_1, \mathrm{Resp}^{B_2} := B, \\ & \mathrm{NI}^{B_2} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_2, \mathrm{NR}^{B_2} := N_B, \\ & {c}^{B_2} := \mathrm{m1}, \\ & {z}^{B_3} := {c}^{B_3}, \\ & \mathrm{Init}^{B_3} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_1, \mathrm{Resp}^{B_3} := B, \\ & \mathrm{NI}^{B_3} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_2, \mathrm{NR}^{B_3} := N_B \sqcup \{{c}^{B_3}\}_{{K_B}^{-1}}\} \\ \text{where} \quad & N_B = o(B_1, \mathrm{NR}), \mathrm{m1} = \{\{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_2, N_B\}_{\mathbf{Pb}(\{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_1)} \end{aligned}$$

同様にプロセス B の実行が終わったときの変数の値に注目してみる。

$$\begin{aligned} \|\mathcal{W}^{B_3}\|(\{\mathrm{Init}, \mathrm{Resp}, \mathrm{NI}, \mathrm{NR}\}) \quad = \quad & \{\mathrm{Init} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_1, \mathrm{Resp} := B \\ & \mathrm{NI} := \{{c}^{B_1}\}_{{K_B}^{-1}}; \pi_2, \mathrm{NR} := N_B \sqcup \{{c}^{B_3}\}_{{K_B}^{-1}}\} \end{aligned}$$

すなわち、プロセス B はイニシエータを最初の入力を自分の秘密鍵で複合したものの第 1 要素 、レスポンダを B、イニシエータのノンスを最初の入力を自分の秘密鍵で複合したものの第 2 要素、そしてレスポンダのノンスが $N_B \sqcup \{{c}^{B_3}\}_{K_B^{-1}}$ すなわち、自分で生成したノンス N_B と次の入力を自分の秘密鍵で複合したものが一致していればプロトコルが正常終了したと判断する。

4.3.3 ペネトレータ C の知識グラフ

イニシエータプロセス、レスポンダプロセスは **fresh** で生成するメッセージ以外はあらかじめ決められた動作を行う。これに対してペネトレータプロセスは、それまでに獲得した知識から発話可能なメッセージを自由に生成することができる。ここでは、ペネトレータプロセスを C とし、次のように定義する。

$$C \triangleq (\ \mathbf{inp}\ \ {c}^{A}; C) + (\ \mathbf{inp}\ \ {c}^{B}; C) + (\ \mathbf{out}\ \ {c}^{A}\ M; C) + (\ \mathbf{out}\ \ {c}^{B}\ M; C)$$

C はチャネル c^A を使ってプロセス A と、チャネル c^B を使ってプロセス B と通信をする。受け取った入力の解釈は定まっていないため、**inp** $c\,x$ の x は省略されている。また **out** $c\,M$ の M のコンストラクタも未定のままである。

Lowe の攻撃は次のような mu 計算の式 [5] で表される。

$$A0|C|B0 \models \mu Z.((A3 \wedge B3) \wedge (\text{Init}^A \neq \text{Init}^B \vee \text{Resp}^A \neq \text{Resp}^B)) \vee \langle - \rangle Z$$

直感的には “プロセス A, B の双方とも最終状態 $A3, B3$ に到達し、その時点でプロセス A, B がそれぞれ信じているイニシエータ、もしくはレスポンダが食い違う遷移が存在する” と解釈する。この論理式のモデルとなる遷移列とそれに伴う発話の列が Lowe の攻撃であった。

このプロセス C と $A0$ (PID A) および $B0$ (PID B) を合成したプロセスが以下のような遷移をとり、最終的に A, B ともに正常な最終状態 $A3, B3$ に遷移するとしよう。

$$\begin{aligned}
A0|C|B0 &\rightarrow (\ \textbf{out}\ c^A\, m0; A1|\ \textbf{inp}\ c^A\, x; C)|B0 \\
&\rightarrow A1|(\ \textbf{out}\ c^B\, M0; C|\ \textbf{inp}\ c^B x; B1) \\
&\rightarrow A1|C|B1 \\
&\rightarrow A1|(\ \textbf{inp}\ c^B\, x; C|\ \textbf{out}\ c^B\, m1; B1) \\
&\rightarrow A1|C|B2 \\
&\rightarrow (\ \textbf{inp}\ c^A\, x; A2|\ \textbf{out}\ c^A\, M1; C)|B2 \\
&\rightarrow A2|C|B2 \\
&\rightarrow (\ \textbf{out}\ c^A\, m2; A3|\ \textbf{inp}\ c^A\, x; C)|B2 \\
&\rightarrow A3|C|B2 \\
&\rightarrow A3|(\ \textbf{out}\ c^B\, M2; C|\ \textbf{inp}\ c^B\, x; B2) \\
&\rightarrow A3|C|B3
\end{aligned}$$

このプロセスとその時間の遷移を表すと表 3 のようになる。ここで、$A^{A_0}_{A_1} \xrightarrow{m0} C^{C_0}_{C_1}$ は “プロセス A は時刻 A_0 までの知識を使い、時刻 A_1 にメッセージ $m0$ を発話する。プロセス C は時刻 C_1 にそのメッセージを受け取る” という事を表している。

この時、入出力の同期による制限により、$A_i < A_j, B_i < B_j, C_i < C_j$ if $i < j$、および $A_1 = C_1, C_2 = B_1, B_2 = C_3, C_4 = A_2, A_3 = C_5, C_6 = B_3$ が成立している。これに伴い、$c^{A_1} = c^{C_1}, c^{C_2} = c^{B_1}, c^{B_2} = c^{C_3}, c^{C_4} = c^{A_2}, c^{A_3} = c^{C_5}, c^{C_6} = c^{B_3}$ が成り立つ。

A, B と同様に C の知識グラフの初期状態 W^{C_0} を次のように定める。

$$W^{C_0} = \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}(B) := K_B, \mathbf{Pb}(C) := K_C, \mathbf{Pb}^{-1}(C) := {K_C}^{-1}\}$$

$$
\begin{array}{lcll}
A_{A_1}^{A_0} & \xrightarrow{m0} & C_{C_1}^{C_0} & \\
 & & \downarrow & \\
 & & C_{C_2}^{C_1} & \xrightarrow{M0} \quad B_{B_1}^{B_0} \\
 & & & \downarrow \\
 & & C_{C_3}^{C_2} & \xleftarrow{m1} \quad B_{B_2}^{B_1} \\
 & & \downarrow & \\
A_{A_2}^{A_1} & \xleftarrow{M1} & C_{C_4}^{C_3} & \\
\downarrow & & & \\
A_{A_3}^{A_2} & \xrightarrow{m2} & C_{C_5}^{C_4} & \\
 & & \downarrow & \\
 & & C_{C_6}^{C_5} & \xrightarrow{M2} \quad B_{B_3}^{B_2}
\end{array}
$$

表 3: C による Lowe の攻撃手順

プロセス C が終了した時点の世界の解釈は非常に単純である。

$$
\begin{aligned}
\|\mathcal{W}^C\| \quad = \quad & \{\mathbf{Pb}(A) := K_A, \mathbf{Pb}(B) := K_B, \mathbf{Pb}(C) := K_C, \mathbf{Pb}^{-1}(C) := {K_C}^{-1}, \\
& c^{C_1} := \mathrm{m0}, c^{C_2} := \mathrm{M0}, c^{C_3} := \mathrm{m1}, c^{C_4} := \mathrm{M1}, c^{C_5} := \mathrm{m2}, c^{C_6} := \mathrm{M2}\}
\end{aligned}
$$

Lowe の攻撃によると、この遷移列内の $m0, m1, m2, M0, M1, M2$ は表 4 のようなものであった。

$$
\begin{array}{lcllcl}
\mathrm{m0} & = & \{A, N_A\}_{K_C} & \mathrm{M0} & = & \{A, N_A\}_{K_B} \\
\mathrm{m1} & = & \{N_A, N_B\}_{K_A} & \mathrm{M1} & = & \{N_A, N_B\}_{K_A} \\
\mathrm{m2} & = & \{N_B\}_{K_C} & \mathrm{M2} & = & \{N_B\}_{K_B}
\end{array}
$$

表 4: Lowe の攻撃中のメッセージ

一方、知識世界 $\mathcal{W}^A, \mathcal{W}^B$ の元で、 m0、m1、m2 はそれぞれつぎの値をとった。

$$
\begin{aligned}
\|\mathcal{W}^A\|(\mathrm{m0}) \quad &= \quad \{A, N_A\}_{K_C} \\
\|\mathcal{W}^B\|(\mathrm{m1}) \quad &= \quad \{\{c^{B_1}\}_{{K_B}^{-1}}; \pi_2, N_B\}_{\mathbf{Pb}(\{c^{B_1}\}_{{K_B}^{-1}}; \pi_1)} \\
\|\mathcal{W}^A\|(\mathrm{m2}) \quad &= \quad \{\{c^{A_2}\}_{{K_A}^{-1}}; \pi_2\}_{\mathbf{Pb}(C)}
\end{aligned}
$$

すなわち、C が C_1, C_3, C_5 の発話世界でそれぞれ $M0, M1, M2$ 発話可能であり、その状況において $\|\mathcal{W}^B\|(\mathrm{m1}) = \{N_A, N_B\}_{K_A}$、$\|\mathcal{W}^A\|(\mathrm{m2}) = \{N_B\}_{K_C}$ が示せれば Lowe の攻撃が成立した事になる。

見通しを良くするため、以下の証明では (Common)、(Mono) 規則による導出は省略してある。

証明 4.1 ($M0$ の発話可能性)

$$\dfrac{\dfrac{\dfrac{C \overset{\mathcal{W}^{C_1}}{\Longrightarrow} c^{C_1=A_1} \quad c^{A_1} := \{A, N_A\}_{K_C} (= m0)}{C \overset{\mathcal{W}^{C_1}}{\Longrightarrow} \{A, N_A\}_{K_C}} \quad C \overset{\mathcal{W}^{C_0}}{\Longrightarrow} {K_C}^{-1}}{C \overset{\mathcal{W}^{C_1}}{\Longrightarrow} (A, N_A)} \quad C \overset{\mathcal{W}^{C_0}}{\Longrightarrow} K_B}{C \overset{\mathcal{W}^{C_1}}{\Longrightarrow} \{A, N_A\}_{K_B} (= M0)}$$

証明 4.2 ($M1$ 発話可能性)

$$\dfrac{C \overset{\mathcal{W}^{C_3}}{\Longrightarrow} c^{C_3=B_2} \quad c^{B_2} := \{N_A, N_B\}_{K_A} (= m1)}{C \overset{\mathcal{W}^{C_3}}{\Longrightarrow} \{N_A, N_B\}_{K_A} (= M1)}$$

証明 4.3 ($M2$ 発話可能性)

$$\dfrac{\dfrac{\dfrac{C \overset{\mathcal{W}^{C_5}}{\Longrightarrow} c^{C_5=A_3} \quad c^{A_3} := \{N_B\}_{K_C} (= m2)}{C \overset{\mathcal{W}^{C_5}}{\Longrightarrow} \{N_B\}_{K_C}} \quad C \overset{\mathcal{W}^{C_0}}{\Longrightarrow} K_C^{-1}}{C \overset{\mathcal{W}^{C_5}}{\Longrightarrow} N_B} \quad C \overset{\mathcal{W}^{C_0}}{\Longrightarrow} K_B}{C \overset{\mathcal{W}^{C_5}}{\Longrightarrow} \{N_B\}_{K_B} (= M2)}$$

証明 4.4 ($\|\mathcal{W}^B\|(\mathrm{m1}) = \{N_A, N_B\}_{K_A}$)

$$\begin{aligned}
\|\mathcal{W}^B\|(\mathrm{m1}) &= \{\{c^{B_1}\}_{{K_B}^{-1}}; \pi_2, N_B\}_{\mathbf{Pb}(\{c^{B_1}\}_{{K_B}^{-1}};\pi_1)} \\
&= \{\{\mathrm{M0}\}_{{K_B}^{-1}}; \pi_2, N_B\}_{\mathbf{Pb}(\{\mathrm{M0}\}_{{K_B}^{-1}};\pi_1)} \quad \because c^{B_1} = c^{C_2} := \mathrm{M0} \\
&= \{\{\{A, N_A\}_{K_B}\}_{{K_B}^{-1}}; \pi_2, N_B\}_{\mathbf{Pb}(\{\{A,N_A\}_{K_B}\}_{{K_B}^{-1}};\pi_1)} \\
&= \{(A, N_A); \pi_2, N_B\}_{\mathbf{Pb}((A,N_A);\pi_1)} \\
&= \{N_A, N_B\}_{\mathbf{Pb}(A)} \\
&= \{N_A, N_B\}_{K_A}
\end{aligned}$$

証明 4.5 ($\|\mathcal{W}^A\|(\mathrm{m2}) = \{N_B\}_{K_C}$)

$$\begin{aligned}
\|\mathcal{W}^A\|(\mathrm{m2}) &= \{\{c^{A_2}\}_{{K_A}^{-1}}; \pi_2\}_{\mathbf{Pb}(C)} \\
&= \{\{\mathrm{M1}\}_{{K_A}^{-1}}; \pi_2\}_{\mathbf{Pb}(C)} \quad \because c^{A_2} = c^{C_4} := \mathrm{M1} \\
&= \{\{\{N_A, N_B\}_{K_A}\}_{{K_A}^{-1}}; \pi_2\}_{\mathbf{Pb}(C)} \\
&= \{(N_A, N_B); \pi_2\}_{\mathbf{Pb}(C)} \\
&= \{N_B\}_{\mathbf{Pb}(C)} \\
&= \{N_B\}_{K_C}
\end{aligned}$$

以上より、C は $m1, m2$ の制約を満たしながら、$M0, M1, M2$ すべてのメッセージを発話可能である事が示された。

5 まとめ

本研究では暗号プロトコル解析のための枠組みとして、簡略化した SPI 計算、およびその意味として発話可能グラフを生成元とする発話世界を定義した。プロトコルの実行は発話可能グラフの書き換えであり、それぞれのプロセスは発話可能グラフから生成された発話世界のノードをメッセージとして発話可能である。これにより、事前にネットワークに流れた知識を利用して、プロトコルの意図していない振る舞いを誘導するペネトレータを形式化することができる。例として、NS プロトコルに対する Lowe の攻撃が実現可能であることを本方法を用いて示し、その有効性を確認した。

今回は、 SSPI の生成、およびその実行過程にともなう発話グラフの書き換えを手動で行った。それぞれのルールは単純で理解しやすいものであるが、人手による方法では誤りが混入しやすい。このため、今後は本方式の自動化を行っていく。

また、今回のプロトコル参加者は有限で、攻撃パターンはよく知られているものを用いた。実際のネットワークは事実上有限ではあるが膨大な数の参加者が存在し、多種のプロトコルが同時に使用されている。これらの一般的な状況下でも暗号プロトコルの安全性を簡潔に示す枠組みが望まれる。今後は本来の目標である、より一般的な状況下での暗号プロトコル検証の自動化に向けて努力していく。

謝辞

この論文は、平成 19 年度在外研究の成果をまとめたものである。快適な研究環境を提供して下さった Dr. Masanobu Shinozuka、Dr. Maria Feng 先生をはじめ University of California Irvine の関係者の皆様にお礼を申し上げます。

参考文献

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.

[3] S. Briais and U. Nestmann. A formal semantics for protocol narrations. In *In Proceedings of TGC05, LNCS 3705*, pages 163–181. Springer, 2005.

[4] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[5] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 23, 1983.

[6] G. Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[7] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using fdr. In *Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166, London, UK, 1996. Springer-Verlag.

[8] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[9] R. Milner. Communicating and mobile systems: the pi-calculus, cambridge univ. press, 1999.

[10] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[11] E. SUMII, H. TATSUZAWA, and A. YONEZAWA. Translating security protocols from informal notation into spi calculus. **情報処理学会論文誌. プログラミング**, 45(12):1–10, 20041115.

[12] 大矢野潤. セキュリティプロトコルの簡易検証 i 〜 具体意味論 〜. **千葉商大論叢**, 44(4):27–52, 2007.

〔抄　録〕

本論文では，特に電子政府や電子商取引の基礎となる暗号プロトコル検証のための基礎技術としてYANPI法（Yielding Arrow Nonce Protocol Inspection Method），すなわち簡略化されたSPI計算の操作的意味として発話可能グラフを用いる方法を提案する。

公開されたネットワーク上で安全なシステムを構築する上で暗号の利用は不可欠である。暗号自体の強度は，暗号文の解読に必要な計算量により与えられるが，暗号の強度が十分であっても暗号鍵の受け渡しなどのプロトロルが脆弱であれば，盗聴された暗号鍵により暗号自体を無効にされてしまう。すなわち，安全なシステムの構築には強い暗号と同時に耐攻撃性に優れた暗号プロトコルが必要である。暗号プロトコルの厳密な耐攻撃性検証のためには，攻撃者攻撃者の能力を明確にしなければならないが，この定式化が困難であることがプロトコル検証を難しくしている原因の一つである。本論文では，通信主体の能力を表現するために発話可能グラフを定義した。そして，本方法をNeedham Schroeder公開鍵プロトコルへのLoweの攻撃のプロトコルナレーションに適用し，その攻撃が成立することの証明が簡潔に与えられることを示した。

SPI計算は暗号プロトコルの研究分野において標準的に用いられている枠組みであり，本方法を用いることで多数の先行研究の理解を深めることができると考えている。