セキュリティプロトコルの簡易検証 I
～具体意味論～

大矢野　　潤

1　はじめに

近年の情報技術の進歩は，我々の生活に急激な変化をもたらしている。自宅やオフィスから，わずか数秒で世界中に配信される電子メール，世界中の情報を瞬時に検索を可能にする WWW システム，ワンクリックで海外の品物を購入できるオンラインショッピングは，人と人とのコミュニケーションのみならず，巨大企業のビジネスモデルをも変革し続けている。これらのシステムを支えるのは，高速なデータの複製と伝達を基礎とした通信と豊富な機能を持つソフトウェア群である。

「よい」ソフトウェアの満たすべき条件として「現実世界で望まれる性質を忠実に仮想世界で再現していること」を挙げることはきわめて自然であろう。例えば，オンラインショップで利用される貨幣や電子投票における投票用紙は偽造されにくいという性質を持つべきであり，電子決済は電源異常などの不測の事態が起った場合には，迅速性をある程度犠牲にしてでも，正確に処理されなければならない。しかし，「偽造されにくい」という性質は，前出の「高速なデータの複製と伝達」と相反する概念であり，「不測の事態にも正確に」という性質を満たすために必要な，すべてのシステムの不具合を検出する万能計算機は，ゲーデルの不完全性定理とチューリングマシンの停止性問題によりその存在が否定される。つまり，我々の生活を根底で支える情報技術が「よい」性質をもつことは原理的に困難であり，経験から生まれる試行錯誤を通じて「よい」システムの近似解を探求することしかできない。このため，我々の生活は未経験の不測の事態によりその足下から崩壊する危険性を秘めている。

公開されたネットワーク上で「堅牢な」性質を表現するための仕組みとして，暗号とそれを応用したシステムが存在する。暗号自体の強度は，暗号文の解読に必要な計算量の議論により数学的に与えられる。これに対して，暗号の強度は十分であると理想化した上での，鍵受け渡しプロトコルの強度は証明論的に与えることができる。これらの計算論と証明論という異なる分野からのアプローチは，歴史的に別々のコミュニティにより議論されてきたが，近年，これらの隔たりを埋めるための試みが提案されている［1］。

通信プロトコルの正当性への証明論的アプローチには，一般的な枠組みに加えて通信，暗号，知識，信念に関する公理を組み込んだ，BAN logic［3］をはじめとした専用の証明系が提案されている。しかし，BAN logic は未だ意味論が確定せず，健全性や完全性に関する議論が不十分である。特に，プロトコルの設計過程では完全性が成立することが重要であるが，証明系の完全性の証明は一般に健全性の証明よりも困難である。

筆者は，通信プロトコルの証明体系に対して健全性，および完全性の議論を展開するために YANPI 法（Yeild-Arrow Nonce Protocol Inspection Method）を提案する。この方法により，通信における偽造，成り済まし，改ざんから防御に特化した簡便なプロトコ

ル検証を目標とし，さらに利用する具体意味論を拡張した抽象意味論による通信プロトコルの簡便な検証を行う。本論文では，代表的なプロトコル検証への証明論的アプローチと本研究の位置づけ，YANPI 法の具体意味論について述べ，抽象意味論とモデル検査への応用については次の論文で述べることとする。

2 情報化社会と暗号の必要性

堅牢な暗号システムは安全な電子政府，電子商取引などのサービスを実現するために欠かせない技術である。コンピュータネットワーク上で電子投票を実現する場合には，そのシステム上での通信主体が確かに選挙人としての資格を有すること，さらに，その主体が投じた投票用紙自体が正当なものであることを検証できなければならない。同時に，投票用紙は，盗み見（盗聴），改ざん，破棄などの妨害から守られなければならない。現実世界では，偽造や複製が容易にできるようなものは貨幣としての価値を持たず，電子商取引において単なるテキストデータを電子決済に利用することはできない。このように電子政府，電子商取引は通常よりも安全なシステムとして実現されなければならない。

一方，電子メディア上ではデータの伝達，検索，複製を容易に行うことができる。電子メールや WWW（World Wide Web）などのネットワーク上のサービスは，この性質を利用して，世界中の通信主体間での情報交換を可能にしている。インターネットはネットワークが相互に結合したネットワークの集合体であり，情報の伝達は一般に複数の第三者を経由して行われるため，交換するデータは不特定多数の通信主体によって読み取られることを前提としなければならない。このような公開が前提の通信メディア上で，非公開の情報を含み，改ざん，偽造，複製を許さない通信を可能にするために，古くから暗号技術が利用されてきた。

誤解を恐れずに言うと，暗号とは情報を含むデータを変換し，そこから元の情報の抽出を難しくする反面，正当な通信相手には変換されたデータから返還前の情報をある程度容易に復元する方法を提供するものである。ここで，変換前の情報を容易に解釈できるデータを平文（plaintext，cleartext），変換操作を暗号化（encryption）されたデータを暗号文（ciphertext），暗号文から平文に戻す操作を複合化（decryption）と呼ぶことにする。さらに，正当な通信相手が複合を容易にするために，その暗号文を解読するための鍵（key）を同時に用意する。特に，平文から暗号文を得るための鍵を暗号鍵（encryption key），暗号文から平文を得るための鍵を複合鍵（decryption key）と呼ぶことにする。

例えば，オンラインショッピングにおいて，顧客とショップの間では注文伝票，支払いに使用するクレジットカード番号などの秘匿すべき「情報」は互いに合意した鍵で暗号化した上で通信を行う。このとき，何らかの方法で通信を傍受した第三者はその「データ」を読むことはできるが，そこから意味のある「情報」を抽出することは困難であり，意味のある情報通信は事実上顧客とショップの二者間でのみ行われているとすることができる。

暗号強度とは，解読のための鍵を知らされていない主体が暗号文を解読するためにどの程度の努力（計算量）が必要になるのかということに依存する。現代暗号では，通常，暗号文から複合文を得るために想定される計算量は，複合鍵がない場合に非決定的多項式時間（Nondeterministic Polymonial Time，単に NP 時間），鍵を持った正当な通信主体が複合するのに必要な時間は多項式時間（Polynomial Time，単に P 時間）を仮定する。

暗号の歴史は古く，古典暗号としては，シーザー暗号，ポリュビオス暗号，ヴィジュネル暗号，スキュタレー暗号などが知られている。また，現代の情報通信に耐えうるだけの強度，利用方法に合った現代暗号としてはAES，DESなどの共通鍵暗号，RSA，楕円曲線暗号などの公開鍵暗号が知られている。現代暗号の特徴としては，2^{128}程度の巨大な数の素因数分解，楕円曲線上の離散対数問題など，現代数学における問題の解発見の困難さを，暗号解読の安全性の根拠にしていることがあげられる。

暗号文は，それを解読するための鍵を持たない主体には事実上解読不可能であるが，鍵を持っている主体は容易に復号できること，つまり，暗号文を使って通信する場合には，あらかじめ鍵を合意しておくことが必要である。この時，暗号鍵を通信に先立ってどのようにして合意すればよいのかという，いわゆる鍵配布問題が存在する。複合鍵をオンラインで配布し，鍵自体が第三者の手に落ちた場合には，その鍵に対応した暗号鍵で暗号化した通信は事実上平文で通信しているのと同じである。つまり，鍵配布プロトコルが脆弱な場合には，どのような強い暗号を用いたとしてもシステム全体が脆弱になる。

現代の情報システムの主たる目的は軍事や諜報活動ではなく，電子商取引など世界中の人々の日常的な利用である。さらに，通信をサポートするソフトウェアやハードウェアも任意のメーカーが供給できる必要があるため，そこで使用される暗号システム，鍵配布プロトコルなどはその仕組みを公開することが原則である。このため，セキュリティシステムの安全性は，仕組みの秘匿に因らず，純粋に暗号システムと鍵配布プロトコルの強さに根拠をおかなければならない。本論文では，このうち，鍵配布プロトコルなどのセキュリティプロトコルについて論じ，暗号系内部の計算量的な議論には立ち入らない。

3 セキュリティプロトコル

現代暗号には，大きく共通鍵によるものと公開鍵によるものがあることはすでに述べた。これらを組み合わせて効率よく安全に通信をするための仕組みをセキュリティプロトコルと呼ぶことにする。セキュリティプロトコル自体について語る前に，まず，本論文で使用する用語の定義を与えよう。

定義3.1 通信を開始する主体のことをイニシエータ（*initiator*），通信相手をレスポンダ（*responder*），通信を傍受し攻撃を加えるものをペネトレータ（*penetrator*）とよび，それぞれA，B，Pなどと記述する。

定義3.2 鍵（*key*）とは，暗号アルゴリズムの手順を制御するためのデータであり，Kと記述する。データmを鍵Kで暗号化して得られたデータを$\{m\}_K$と記述する。

定義3.3 共通鍵暗号（*Secret Key Cryptosystem*）では，暗号化と復号化に同一の鍵（共通鍵）を用いる。つまり，N_Aを主体Aが生成した共通鍵としたとき，$N_A = N_A^{-1}$となる。いいかえると，$\{\{m\}_{N_A}\}_{N_A} = m$：メッセージ$m$を鍵$N_A$で暗号化したものは同一の鍵$N_A$で復号化することができる。

定義3.4 公開鍵暗号（*Public Key Cryptosystem*）では，暗号化と復号化に別々の鍵を用い，一方の鍵を公開鍵として公開し，もう一方の鍵を秘密鍵とする。K_A，K_A^{-1} をそれぞれ A の公開鍵と秘密鍵としたとき，$\{\{m\}_{K_A}\}_{K_A^{-1}} = \{\{m\}_{K_A^{-1}}\}_{K_A} = m$ となる。いいかえると，K_A で暗号化したメッセージ $\{m\}_{K_A}$ は，A の秘密鍵 K_A^{-1} でのみ複合可能であり，また逆も成立する。

共通鍵暗号による暗号化は公開鍵によるものよりも比較的高速に行われる。反面，暗号としての強度が十分でないため，同一の鍵を長期間使用し続けることはできない。また，共通の鍵を異なる通信相手に使用することができないため，多数の鍵とその鍵配布問題が生じる。

これに対して，公開鍵暗号は，公開する鍵は一般に公開し，それを復号する秘密鍵のみを秘密にしておけばよく，一般に鍵の管理は容易である。さらに，秘密鍵で暗号化することは，その鍵を保持している主体しかできないため，秘密鍵の暗号文をその主体の証明書として用いることができる。これらの理由から，現代の暗号に対するニーズには公開鍵暗号が適していると言えるが，公開鍵暗号は暗号化/複合化にかかる時間がかかりすぎ，実用に耐えうる性能を引き出すことが難しいことも知られている。

これらの問題を克服するために，セッションごとに共通鍵を生成し，そのセッションに先立ち，公開鍵暗号を用いて共通鍵を交換する方法が用いられる。共通鍵の交換に必要なデータは通常のメッセージに比べごく小さいデータであるため，セッション全体では公開鍵を用いるオーバヘッドは無視することができる。しかし，このとき，共有鍵の受け渡しに失敗して鍵が第三者の手に渡った場合には，そのセッション中のすべてのメッセージが解読されてしまう。

3.1 Needham Schroeder Public-Key Protocol

1978年，Roger M. Needham と Michael D. Schroeder により，安全なセッションキーなど共通の秘密（Nonce）の受け渡しを目的とした公開鍵プロトコル（Needham Schroeder Public-Key Protocol：以下，NS プロトコル）[11] 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 Bであると Aが信じることができる。

● 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。

以上のことより，「もし，Bが正直（Honest）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が信じることができる。」と結論づけることは妥当であるように見える。

3.2 Lowe's attack

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により有効ななりすまし攻撃（Lowe's attack）の存在が指摘され，その後Lowe自身により安全なプロトコル（Needham Schroeder Lowe Protocol：以下，NSLプロトコル）が提案された［8，9］。

Loweの攻撃とは次のようなものである。ただし，Iは侵略者（intruder）とし，$I(A)$とは，主体 Aのになりすました Iを意味する。

1．$A \rightarrow I : \{A, N_A\}_{K_I}$
Aが Iに話しかけてしまう。

2．$I(A) \rightarrow B : \{A, N_A\}_{K_B}$
I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．$B \rightarrow I(A) : \{N_A, N_B\}_{K_A}$
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 Iに返す。

4．$I \rightarrow A : \{N_A, N_B\}_{K_A}$

I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．$A \rightarrow I : \{N_B\}_{K_I}$
AはこのメッセージはI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Iに知らせてしまう。

6．$I(A) \rightarrow B : \{N_B\}_{K_B}$
I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。
I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．$A \rightarrow B : \{A,\ N_A\}_{K_B}$

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

3．$A \rightarrow B : \{N_B\}_{K_B}$

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，改竄者Iの存在を知らせる部分のみである。このことは，セキュリティに関するプロトコルの検証は非常にデリケートであること，またその作業は誤りを含みやすく，人手で行うことが困難であることを物語っている。実際にLoweはモデル検査機FDRにより機械的にNSLプロトコルの正当性を示している。
オリジナルのNSプロトコルでは，通信している相手を両方の主体が錯誤していない場合には情報の漏洩はない。このことは，プロトコルに出現する主体の状況，動作，そして何をもってプロトコルが安全であるのかということを正確に定義しなければ安全性の議論はできないことを物語っている。本論文では，通信の秘匿性に次の対応性を正しさの定義として加える［14］。

定義3.5 対応性（*correspondence property*）：通信プロトコルは主体Bに対して次のことを保証する。主体Bがレスポンダーとしてそのプロトコルにおける役割を完了し，そこで認識したデータ列$\vec{x}$，特に通信のイニシエータがAであるという情報を得たとする。このとき，主体Aがイニシエータの役割をするユニークなプロトコル実行系列があり，そこで認識されるデータ列は同じ値$\vec{x}$で，そこに現れるレスポンダーはBとなる。

上の定義は，主体 B がレスポンダーとしてプロトコルの完了に成功した場合には，双方が認識するイニシエータ，レスポンダー，合意したデータに食い違いがないことを意味している。

Lowe が指摘したのは，NS プロトコルの対応性の不備であった。

4　セキュリティプロトコル検証法

セキュリティプロトコルの不備は，そこで使われている暗号システム自体を無力化し，暗号システムの強靭さをその安全性の根拠におく，電子政府や電子商取引などの情報社会基盤を崩壊させてしまう。17年間安全であると信じられてきた NS プロトコルの弱点は，Lowe の攻撃により誰の目にも明らかな方法で露呈した。この後，より安全なセキュリティプロトコルの確立にむけ，さまざまなプロトコル検証方法が提案されている。以下，代表的なものについて簡単に説明する。

4.1　モデル検査

まず，Lowe が NS プロトコルの攻撃と修正にモデル検査機（Model Checker）FDR を用いたことは既に述べた。モデル検査（Model Checking）とは，特定の有限状態機械が論理的な性質を充足するかどうかを体系的に，かつ自動的に判定する技術である［5］。M を状態システム，属性を表す論理式 Φ が与えられた時，M が Φ のモデルになっているかどうか，つまり $M \models \Phi$ の充足性をモデル検査機を使って，（半）自動的に判定する。モデル検査は近年急速に発達した技術で，ハードウェアや，ネットワークプロトコルの検証などに盛んに応用されている。さらにそれを支援するために SPIN，SMV，Mocha などさまざまなモデル検査機が提供されている。

モデル検査はシステムの動的な性質を検証するために，そこで用いられる論理体系は時制を含んだ論理式を表現できることが要求される。代表的なものとしては，CTL (Computation Tree Logic)，LTL (Linear Time Logic)，様相 mu 計算などがあげられる。

モデル検査はモデル空間を網羅的に調査するため，検査対象のモデルは原則として有限である。反面，ネットワーク上には可算無限個の主体，そしてそれが生成する攻略パターンも無限に存在することができる。このため，抽象化技法などを用いて，検証したい性質を残したまま無限状態システムを有限状態システムに変換する必要がある。また，あるシステムが有限状態数 $|P|$ を持つ $n < \omega$ 個のプロセスから生成される場合でも，システム全体の状態数 $|M|$ は $|M| = |P|^n$ となり，モデルの状態数が爆発する。さらに，検証したい時間の性質が複雑になる場合には，その検証にかかる時間が論理式の複雑さ（様相 mu 計算の場合には不動点交代数 d）の乗数 $|M|^d$，モデル検査全体にかかる計算量はモデルの状態数に対して PSPACE 完全となり，検査の実行は事実上不可能である。これらの理由から，プロトコル検証にモデル検査を応用するために，階層的にモデルを抽象化し検証する性質に応じて順次詳細化していく方法［6］や，検証する性質自体を制限する方法を用い，その上で効率の良い状態数探査のアルゴリズム［4，2］を適応していく方法が提案されている。

筆者は，セキュリティプロトコルの検証には，安全性（Safety Property）で十分であ

ると考えている。あるクラスの安全性の検査はモデル空間での到達可能性（reachability）の検証で十分であり，到達可能性の検証の計算量はモデルの大きさ対して線形時間で検証可能である。また，CTL の特別なクラスである ACTL に論理式を限定できる場合には，次の推論が可能であることも知られている。

$$M \preceq M',\ M' \models \Phi \Rightarrow M \models \Phi$$

これは，抽象領域M'がある ACTL の論理式Φのモデルになっている場合には，その具体領域$M \preceq M'$もΦのモデルとなっていると主張している。抽象領域の検査を具体領域の検査に代えることができれば，具体領域を構築しない分だけ，検査時に構築するモデル空間を小さくすることが期待される。

4.2 ストランド空間解析

ストランド空間の定義は必要最小限にとどめる。詳しくは［13］を参照すること。

それぞれの主体が通信プロトコルを実行していく上で得られる主体の状態変化列の集合を状態遷移システムという。状態遷移システム上で可能な状態遷移列を解析し，好ましくない状態に遷移しないことを証明することが安全性の検証の基本方針である。

定義4.1 プロトコルの実行時に交換されるデータ（メッセージ）の集合を **A**，符号付きメッセージの列を $(\pm\mathbf{A})^*$と記述する。

直感的には，メッセージ$+M$はメッセージMを発すること，$-M$メッセージMを受け取ることと理解してよい。さらに，$\langle -K,\ -h,\ +\{h\}_K \rangle \in (\pm\mathbf{A})^*$はメッセージ列の例であり，このメッセージはあらかじめ受け取った鍵Kと情報hを暗号化したもの$\{h\}_K$を出力すると解釈する。

定義4.2 A 上のストランド空間（*Strand Space*）とは，トレース写像 tr：$\Sigma \to (\pm\mathbf{A})^*$を伴う集合$\Sigma$である。

つまり，ある空間上の状態$s \in \Sigma$を指定すると，その状態で交換されるメッセージ列が指定できる。次にΣの具体的な与え方について定義する。

通信に参加する複数の主体が，メッセージの授受に関して協調してプロトコルで規定された動作を行うことを考える。例えば，メッセージhを発話する主体の動作$+h$と，それを受領する主体の動作$-h$が協調したものを$\bullet \xrightarrow{h} \bullet$，メッセージの授受により各主体の状態が変化することを$\bullet \Rightarrow \bullet$のように 2 種類のエッジをもつグラフとして記述すると見通しがよくなる。以下，このメッセージの受け渡しについて整合的（メッセージを発する主体と受け取る主体が記述されている）グラフをバンドル（bundle）と呼ぶ。例えば，前述の NS プロトコルの期待した動作と，それに対する Lowe の攻撃はそれぞれ図 1，2 のように表すことができる。

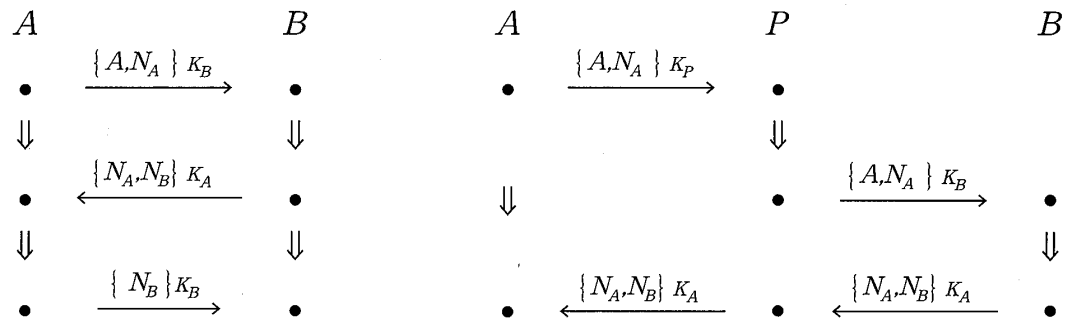


図1：Needham-Schroederプロトコル

図2：Loweのアタック

このストランドによる主体間でのメッセージの授受から，メッセージの生成に関する順序とそこから生成されるイデアルを定義し，ストランド空間内でメッセージ傍受，改ざんのパターンなどが存在するかどうかを解析するアプローチ［13］や，メッセージの授受のパターンを公理としてとらえ，定理証明系内で攻撃パターンの存在証明を，定理証明機を用いて示すものもある［12]。以下，これらの方法で共通して用いられる項（メッセージ）の生成に関する順序と，そこで定義されるイデアルについて定義しておく。ここでの定義，命題，補題は［13］によった。

定義4.3 メッセージ（項）の間に，部分項に関する二項関係（*subterm relation*）$\sqsubseteq$ を次のように定義する。

1．$a \sqsubseteq a$

2．$a \sqsubseteq g$ ならば $a \sqsubseteq \{g\}_K$

3．$a \sqsubseteq g$ か $a \sqsubseteq h$ ならば $a \sqsubseteq gh$

このようにして定義した二項関係は明らかに半順序になる。

定義4.4 $\mathrm{K} \in \mathbf{K}$ のとき，$\mathbf{A}$ の K イデアル（*ideal*）とは，$\forall h \in \mathbf{I}$，$g \in \mathbf{A}$，$K \in \mathrm{K}$ のとき次の条件を満たす $\mathbf{A}$ の部分集合 $\mathbf{I}$ のことをいう。

1．hg，$gh \in \mathbf{I}$

2．$\{h\}_K \in \mathbf{I}$

定義4.5 $\mathbf{I}_{K[h]}$ を h を含む最小の K イデアルとする。

定義4.6 $\mathbf{I}_{K[S]}$ を S を含む最小の K イデアル（$S \subseteq \mathbf{A}$）とする。

命題4.1 $S \subseteq \mathbf{A}$ ならば $\mathbf{I}_K[S] = \cup_{x \in S} \mathbf{I}_K[x]$

補題4.1 $S_0 = S$, $S_{i+1} = \{\{g\}_K \mid g \in \mathbf{I}_0[S_i],\ K \in \mathbf{K}\}$ とする。このとき，$\mathbf{I}_K[S] = \cup_i \mathbf{I}_K[S_i]$

攻撃者のとれるメッセージパターンは，攻撃者の持つ知識の集合から構成されるイデアルであると数学的に特徴づけることができる。

4.3 制約解消系

主体がとりうる行動パターンを一種の制約充足としてとらえるアプローチ［10］も有力である。プロトコルがその安全性を保証するために侵略者に対して行う防御パターンを制約，侵略者はその制約を充足し，かつ目的のデータの改ざんや傍受を行うことを制約充足と考える。

主体 A, B の動作はそれぞれ次のようなものであった。ただし $A\leftarrow$ は A に対する入力，$A\rightarrow$ は A の出力を表す

・A の動作

1．$A\leftarrow$: “talk to X”
　X に話しかけようと思う。
2．$A\rightarrow$: $\{N_A, A\}_{K_X}$
　X に話しかける。
3．$A\leftarrow$: $\{N_A, Z\}_{K_A}$
　A が伝えたノンスとともに，相手のノンス Z を受け取る。
4．$A\rightarrow$: $\{Z\}_{K_X}$
　X にノンス Z を返す。

・B の動作

1．$B\rightarrow$: “talk to B”
　B が何者からか話しかけられる。
2．$B\leftarrow$: $\{W, Y\}_{K_B}$
　B が W からのノンス Y を受け取る。
3．$B\rightarrow$: $\{Y, N_B\}_{K_W}$
　W が伝えたノンス Y とともに，自分のノンス N_B を返す。
4．$B\leftarrow$: $\{N_B\}_{K_B}$
　B が自分のノンス N_B をうけとる。

ここで，W, X, Y, Z は変数である。この時，「Lowe の攻撃に対応する W, X, Y, Z

の具体化はあるか」という問題に帰着する。

1．$B \rightarrow E$：“talk to B”
2．$A \leftarrow E$：“talk to X”
3．$A \rightarrow E$：$\{N_A, A\}_{K_X}$
4．$B \leftarrow E$：$\{A, N_A\}_{K_B}$
5．$B \rightarrow E$：$\{N_A, N_B\}_{K_A}$
6．$A \leftarrow E$：$\{N_A, N_B\}_{K_A}$
7．$A \rightarrow E$：$\{N_B\}_{K_X}$
8．$\leftarrow E$：N_B

このとき，攻撃者はそれまでに得た知識（$\mathcal{J}$）から攻撃のメッセージを作り出さなければならないため，次の制約が存在する。

1．$A \leftarrow E$：“talk to X” の Xは攻撃者の初期の知識$\mathcal{J}_0$から作り出せる。
2．$B \leftarrow E$：$\{A, N_A\}_{K_B}$の$\{A, N_A\}_{K_B}$は$\mathcal{J}_0$，$\{N_A, A\}_{K_X}$から作り出せる。
3．$A \leftarrow E$：$\{N_A, N_B\}_{K_A}$の$\{N_A, N_B\}_{K_A}$は$\mathcal{J}_0$，$\{N_A, A\}_{K_X}$，$\{N_A, N_B\}_{K_A}$から作り出せる。
4．$\leftarrow E$：N_BのN_Bは$\mathcal{J}_0$，$\{N_A, A\}_{K_X}$，$\{N_A, N_B\}_{K_A}$，$\{N_B\}_{K_X}$から作り出せる。

これらの制約を充足したパターン存在する時，具体的な攻撃パターンの存在が示される。

4.4 代表的アプローチの長所と短所

いままで紹介して来たプロトコル検証に関する代表的なアプローチについて，その長所と短所を考察する。

まず，モデル検査の長所としては，プロトコル設計者はモデル検査機を使用したトライアンドエラーを通じてプロトコルの解析と，改良を進めていくことができる。この作業はプログラマがコンパイラを利用する状況に類似している。面倒な字句解析，構文解析，型検査はコンパイラが自動的に行い，プログラマは純粋にコード生成に集中することができる。モデル検査においても，部分モデルから全体モデルの合成，モデル探査，抽象状態生成や，論理式の示す数理論理的意味はツールが取り扱うために，ツール使用者はエラーが出ないようにモデル空間を操作することに集中すればよい。短所としては，モデルは一般に無限状態であり，また有限状態であってもモデル合成時には容易に状態数爆発が起る。状態数爆発を防ぎ，モデルをモデル検査機の扱える大きさに変換することは一般的に容易ではなく，この作業中にエラーが混入しやすい。

次に，プロトコルを実行する主体の状態空間を数学的な対象としてとらえる場合を考える。ここでは，ストランド空間を前順序空間としてとらえ，定義4.3で定義したイデアルの性質に基づく公理を与え，安全性の証明に定理証明機を利用する場合を考えてみる。この手法の長所は，通常の数学的手法を応用するため，整数集合など自然に無限状態を記述する道具を利用することができる。反面，帰納法や証明論を検証のベースとしており，プロトコル開発者はこれらの数学的知識に通じている必要がある。定理証明機により，証明

の作業が軽減されることが期待できるが，定理証明機は定理証明を行うための支援ツールであり，開発者が証明論に通じている必要があることには変わりがない。このことは，制約解消系を用いたアプローチでも同様である。

無限のモデル空間を有限に変換する操作の正当性や，探査空間に関する定理を証明に証明系を用い，得られた空間を網羅的に探査する部分をモデル検査の技術を使う，いわゆるハイブリッド方式が現実的であるとする考え方もある。ツールの活用により，実際の検証作業が軽減されることが期待されるが，やはりツールの使用者はモデル検査機と定理証明機に関する複数の検査機に関する知識と経験があることが前提となる。

4.5 セキュリティプロトコル解析の隘路

Lowe の攻撃，そのストランド空間表現，制約解消系における正解代入など，プロトコルの挙動や，その不備は非常にわかりやすい形で与えられるが，プロトコル自体を解析することは容易ではない。解析を困難にしている理由としては次の点が考えられる。

1．通信に使われるメッセージは有限長のデータ列であり，正当でない手続きにより偶然生成されたり，辞書攻撃など可能性を総当たりにする方法で偶然発見される可能性がある。
2．通信に参加する主体の数，攻撃者の生成する攻撃パターンは（可算）無限の可能性がある。

以下それぞれに関して考察する。

4.5.1 メッセージの有限性

主体間で交換されるメッセージデータ，特に暗号鍵の長さは有限であり，理論的には，総当たり攻撃や辞書攻撃などデータが解読できるまで鍵を変えていく方法を用いることにより，有限時間内に鍵の値を解読することができる。この意味で，暗号システムは必ず有限時間内に破ることができる。ここで，暗号とは情報の解読を「困難にする」技術であることを思い出そう。つまり，絶対に解読ができないようにするということではなく，常識的な手続きで解読ができなければよい。例えば，2^{128}個の状態からなる空間の網羅的探査は常識的な時間内に終わらないため，却下するという立場をとる。

また，メッセージは生成者の意図した方法でのみ作ることができ，他の方法で偶然生成されることがないとする。この偶然性を排除するために，プロトコル内で使用するメッセージの生成規則に条件を付ける場合がある。例えば，[13] では次のような条件を用いている。

定義4.7 メッセージの生成に関する制限で，次のものを自由性の仮定（*Freeness Assumptions*）と呼ぶ。

・m, $m' \in \mathbf{A}$ と K, $K' \in \mathbf{K}$ について，$\{m\}_K = \{m'\}_K \Rightarrow m = m' \wedge K = K'$ が成り立つ。

・m_0，m_0'，m_1，$m_1' \in \mathbf{A}$ と K，$K' \in \mathbf{K}$ について，つぎの条件が成り立つ。

1．$m_0 m_1 = m_0' m_1' \Rightarrow m_0 = m_0' \wedge m_1 = m_1'$

2．$m_0 m_1 \neq \{m'\}_{K'}$

3．$m_0 m_1 \notin \mathbf{K} \cup \mathbf{T}$

Freeness Assumptions では，あるメッセージは複数の手段によって生成されることがないこと，そのためには，暗号文から鍵やもとの平文を推測できる可能性が無視できるほど小さいことを前提にしており，Abadi-Rogaway による type-0 暗号を仮定しているとしてよい［1］。

4.5.2 モデルの無限性

通信には無限個の主体が参加する可能性がある。このため，解析の対象となるモデル空間は無限の大きさのものを考えなければならない。

しかし，モデル空間解析で重要な点は通信主体の数ではなく，侵略者の発話パターンの個数であることがわかる。極端な場合，単体の侵略者が無限の主体を偽って通信しても他の通信主体にはそれが単体なのか複数の仲間が連携しているのかを区別することはできない。

同時に，侵略者が任意のパターンを発話可能であるわけでもない。侵略者はそれまでに得た知識を組み合わせて攻撃メッセージを生成する。

本論文では，攻撃者が発話可能なメッセージ（項）はメッセージに含まれる情報を表す部分項に分解され，攻撃者の発話パターンは，攻撃者の正当に知り得る知識を S_0 とした上で，補題4.1で定義されるイデアルで与えられるものと仮定する。このとき，イデアルの要素は部分項関係に関して整礎（well-founded）となるため，具体的な攻撃パターンは有限回の操作によって既知のデータから構成することができる。

5 Yield-Arrow Nonce Protocol Inspection

本節では，セキュリティプロトコルの検証法として YANPI 法（Yeild-Arrow Nonce Protocol Inspection Method）を提案する。

前節において，セキュリティプロトコルの解析を困難にしている原因は，任意のメッセージは任意の主体が偶然発話可能であるということを排除できないという問題と，プロトコルに参加する可能性のある主体数，攻撃者の発話パターンが無限性であるため，検査するモデル空間が無限になってしまうということを指摘した。本方法では，総当たり方法などのにより偶然意味が解析される可能性を排除しない。十分な強度を持ち，生成されるメッセージ空間が広い暗号システムであれば，攻撃者による総当たり的な攻撃やある種の統計処理は現実的でないと仮定する。プロトコル検証者は，交換されるメッセージから記述されている情報が解析される可能性が指摘された場合には，それが攻撃者の適当な解析により得られ結果であるのか，総当たり方法により偶然得られたものなのかが区別できればよい。

主体が情報を知ることができるということは，主体からある情報に直接アクセスできるか，もしくはアクセスして得たデータに何らかの操作を施して，必要な情報を取り出すこ

とが可能な場合である。この，情報の生成関係を表すグラフを，以下単に生成グラフ（Yeild Arrow Graph，以下 YAG）ということとする。

直感的には，YAG はある時点のシステムの状態であり，プロトコル実行列は YAG の遷移列により与えられる。正確には，YAG はノードとしてメッセージを，エッジとしてメッセージの生成関係を示すアロー（Yeild-Arrow）を持つ有向グラフである。

5.1 バンドルトレース

NS プロトコルのストランド空間解析で用いた図 1，2 を思い出そう。それぞれの主体の内部状態は省略され，それぞれの主体の状態はただの点（●）として記述されているため，詳細な解析を行うための情報が不足している。ここでは，内部状態を省略せずに表現しよう。

主体 A が「信じている」イニシエータは A，イニシエータのノンスを N_A，レスポンダーを B，そして，レスポンダーのノンスをノンス全体の集合 N，つまりすべてのノンスが値の候補となるということを A:[Init:A, NI:N_A, Resp:B, NR:N] と表現することにする。同様に，B の状態を B:[Init:P, NI:N, Resp:B, NR:N_B] と記す。

この状態の記述法を使用すると NS プロトコルの状態遷移列は表 1，Lowe の攻撃のトレースは表 2 のようになる。そして，この状態遷移列を改めてバンドルトレース（bundle trace）と呼ぶことにする。

表 1 において，ストランド空間内に A と B の主体しか存在せず，空間全体を単に ● と表すとすると空間自体の初期値はつぎのように書けることに注意する。

● : [A:[Init:A, NI:N_A, Resp:B, NR:N], B:[Init:P, NI:N, Resp:B, NR:N_B]]

次では，この表現からグラフ YAG を構成する。

A:[Init:A, NI:N_A, Resp:B, NR:N]	$A \xrightarrow{\{A,\ N_A\}_{K_B}} B$	B:[Init:P, NI:N, Resp:B, NR:N_B]
A:[Init:A, NI:N_A, Resp:B, NR:N]	$A \xleftarrow{\{N_A,\ N_B\}_{K_A}} B$	B:[Init:A, NI:N_A, Resp:B, NR:N_B]
A:[Init:A, NI:N_A, Resp:B, NR:N_B]	$A \xrightarrow{\{N_B\}_{K_B}} B$	B:[Init:A, NI:N_A, Resp:B, NR:N_B]
A:[Init:A, NI:N_A, Resp:B, NR:N_B]		B:[Init:A, NI:N_A, Resp:B, NR:N_B]

表 1 ：NS プロトコルのバンドルトレース

A:[Init:A, NI:N_A, Resp:P, NR:N]	$A \xrightarrow{\{A,\ N_A\}_{K_P}} B$	B:[Init:P, NI:N, Resp:B, NR:N_B]
A:[Init:A, NI:N_A, Resp:P, NR:N]	$A \xrightarrow{\{A,\ N_B\}_{K_B}} B$	B:[Init:P, NI:N, Resp:B, NR:N_B]
A:[Init:A, NI:N_A, Resp:B, NR:N_B]	$A \xleftarrow{\{N_A,\ N_B\}_{K_A}} B$	B:[Init:A, NI:N_A, Resp:B, NR:N_B]
A:[Init:A, NI:N_A, Resp:P, NR:N_B]	$A \xrightarrow{\{N_B\}_{K_P}} B$	B:[Init:A, NI:N_A, Resp:B, NR:N_B]

表2：Loweのアタックのトレース

5.2 バンドルグラフから生成グラフへ

バンドルトレースは単なるバンドルを解析するよりも多くの情報を含み，プロトコルの実行の意味が明確になった。しかし，プロトコルの実行はある主体の知識を加工したものを他の主体に送り，結果として他の主体の知識を増加させる行為であるにも関わらず，主体の知識を調べてもその知識がどのようにもたらされたのかが判別不可能である。単にサイコロを振って得られた値と，性格に知識を解析して得られた値とを区別することさえできない。

ここでは，状態に現れる属性値の因果関係を表すために，属性の生成関係をアローとして持つグラフを導入する。

まず，直感的なアイデアを説明するために，代表的な手続き型プログラミング言語で次の文を実行することを考えてみよう。

y:=10; x:=y; y:=20; x=？

この手続きの評価は，変数x，yともに整数型であるとすると，「yの値は10。次に，xにyの値（10）を代入するので，xの値は10になる。」と解釈するのか一般的であろう。ここでは，一旦，型の不整合を無視し，変数x，yを次のように解釈する。「10という値を持つ変数のポインタで10という名前を持つものがある。このポインタをyという名前のポインタ変数に代入する。つまりyという名前を呼ぶと，これは10という名前のポインタをさし，そのポインタを評価すると10という値が返る。その上でxという名前のポインタ変数にyというポインタ変数へのポインタを代入する。」と解釈する。前者であれば，最後のy:=20までを実行した場合でも，最終的なxの値は単に10となるが，後者であれば，xはyのポインタをさしており，yのポインタは20をさすポインタをさしているため，xは間接的に20を指し示している。前者では，xとyの値が一致していたとしても，その根拠はその状態を調べただけではわからないことに注意しよう。後者のでは，xとyの値が一致している理由は「xはyの知識の格納場所を知っているから」という因果関係を保存している。

このアイデアを使ってYAGを再定義しよう。

定義5.1 *YAG* (*Yeild-Arrow Graph*) とはノードの集合 N, アローの集合 $\mathcal{E}$ からなるグラフ $\mathcal{G} = (N, \mathcal{E})$ である。

定義5.2 $\mathcal{G}$ のノードの集合 Nは基本ノードとポインタノードからなる。基本ノードは値として整数値やメッセージなどの具体的な値を持つことができる。ポインタノードは，値として他のノードの名前を持つ。ポインタノード x がノード y の名前を参照しているということを $x \triangleright y$ と記述する。

定義5.3 ノードの評価とは，基本ノードの場合には，そのノードの持つ具体的な値を，ポインタノードは，その指し示しているノードの値を自身の評価値とする。ノード x の評価値を $\|x\|$ と記述する。

例えば，基本ノード ℓ_{10} の値が10の場合には，$\|\ell_{10}\| = 10$，$x \triangleright y$ の場合には，$\|x\|$ は $\|y\|$ と等しい。

定義5.4 ポインタノード x の内容が未定義の場合 $x = \top$ とする。また，評価値が不明であることを□で表す。

定義5.5 $\mathcal{G}$ はルートノード．をただ一つ持つ。便宜上，$. = \top$，$\|.\| = \square$ と定める。

ノードの評価に関するルールを表3のように与える。

$$\frac{x \triangleright y}{\|x\| = \|y\|}$$

$$\|\top\| = \square$$

$$\|x.\ell\| = \square \text{ where } \ell \notin L_P$$

表3：ノードの評価規則

YAGにおいてノードはメッセージを，アロー（エッジ）はあるメッセージから他のメッセージへの生成関係を表すものとする。例えば，$M \xrightarrow{f} M'$ は，「メッセージ M' は M に関数 f を適応して得られる」と読む。この時，アローは関数的であるとする。つまり，

$$\forall x, y, y' \in N,\ f \in A\, x \xrightarrow{f} y \wedge x \xrightarrow{f} y' \Longrightarrow y = y'$$

が成り立つ。

定義5.6 主体 P, Q, R はそれぞれラベル L_P, L_Q, L_R を持つ。すべての可能なラベルを単に $\mathcal{L}$ と書く。

定義5.7 アロー A は次の性質を満たすラベルの集合である。$. \in A$, $\forall a \in A$, $\ell \in \mathcal{L} \Longrightarrow$

a. $l \in A$

定義5.8 ポインタノードが参照しているノードを探査するためにラベルのリダクションルールを表4のように定める。

$$\frac{x \triangleright y \quad y \triangleright z}{x \triangleright z}$$

$$\frac{x \triangleright y}{x.z \triangleright y.z}$$

$$\frac{x \triangleright y}{z.x \triangleright z.y}$$

$$x.\varepsilon \triangleright x$$

表4：ラベルのリダクションルール

ラベル，アローは関数に対応することはすでに述べた。このアローの種類を定めることにより，解析するプロトコルの性質が定まることに注意する。言い換えると，ノード，アローの種類を規定することは，述語論理における関数記号を定めているようなものである。

ここでは基本的なものをいくつか例示しよう。

基本ノード，アロー 基本ノードはルートノード．に基本ラベルを適用したものである。基本ラベルは，定数関数と考えればよい。

基本的な足し算引き算を定義しよう。例えば，定数 $n \in N$ を返す定数関数 l_n，n に1を足して $n+1$ を計算する1引数関数 succ，n から1をひいて $n-1$ を計算する1引数関数 succ をそれぞれ考える。この時，次のように定める。

$$\forall n \in N \| .l_n \| = n$$

$\forall n \in N$，l_n，$l_{n+1} \in A$ について，

$$.l_n.\ \mathrm{succ} \triangleright .l_{n+1}$$
$$.l_n.\ \mathrm{dec} \triangleright .l_{n-1}$$

のように定める。

例えば公開鍵暗号の場合には $K_P.K_P^{-1} = K_P^{-1}.K_P = \epsilon$，共有鍵暗号の場合には $K.K = \epsilon$ を定義すればよい。

以下ではYAGの利用について例を挙げてみよう。

5.2.1 基本ノード

前の例にもある，自然数に関する基本ノードは図3のようになる。

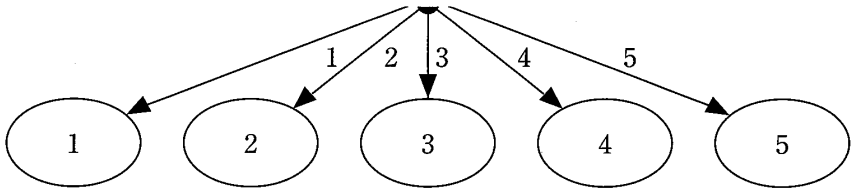


図 3 ：basetype

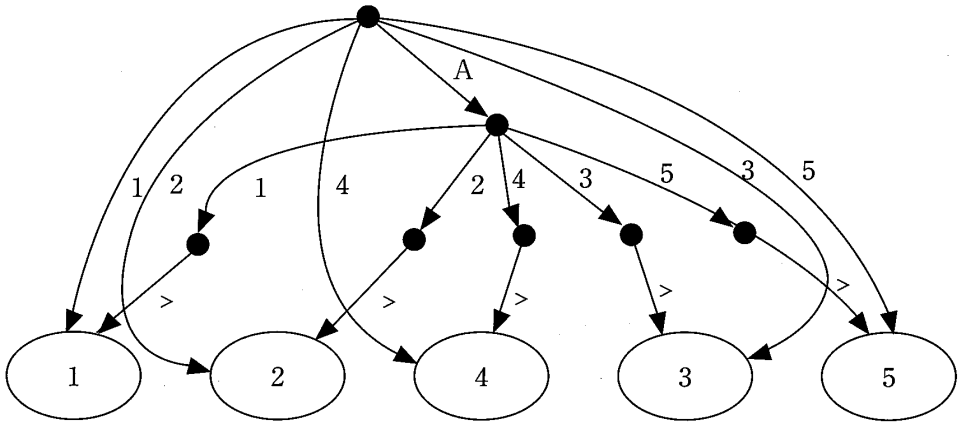
基本タイプのノードには各主体から基本ラベル（定数関数）をつかってアクセスできる。

例えば，A が基本タイプノード 5 にアクセスするとする。この時，基本ラベル 5 を使って，$.A.5$ は次のような解釈をする。

$$.A.5 \triangleright .5$$

この仕組みを利用して，図5. 2. 1のように，A は基本ラベル 5 を使用して，. 5 のノードにアクセスする。

$$||.A.5\ || = ||.A.5\ \triangleright .5|| = ||.5|| = 5$$



$\| .A.5 \| = \| .A.5 \triangleright .5 \| = \| .5 \| = 5$

図 4 ：basetype 1

5.3 メッセージから得られる知識

ここでは次の

$$.A.\text{sec} \quad \triangleright \quad .M$$
$$.B.\ \text{guess} \quad \triangleright \quad .A.\text{sec}$$

と，

$$.A.\,\mathrm{sec} \quad \triangleright \quad .M$$
$$.B.\,\mathrm{guess} \quad \triangleright \quad .M$$

のプロトコル実行列があるとする。
上の実行列で生成されるグラフは次のリダクション

$$.B.\,\mathrm{guess} \triangleright .A.\,\mathrm{sec} \triangleright .M$$

を生成し，BはAの秘密を理解していることを解析することができる。これに対して下の例では，Bがたまたま，Aと同じメッセージを guess していあるが，それがAの秘密であるという確証はない。

$$.B.\,\mathrm{guess} \triangleright .M$$

このように，Penetrator PがAもしくはBの秘密を知り得るかどうかは，$.P$から到達可能なノード，Reach（$.P$）に$.A.\,\mathrm{sec}$が含まれるかどうか，つまり

$$.A.\,\mathrm{sec} \in \mathrm{Reach}\,(.P)$$

で判定できる。

5.4 プロセスの挙動

通信主体とは，単に「（1）入力を得て，（2）自分の持っている知識と組み合わせ，（3）出力する」ものであると考える。また，ここでの通信主体は，通常の（Honest）な主体Aと悪意を持ったペネトレータPの二種類を考え，それぞれ次のように定義しよう。

・入力
- $.P$の出力ノードを$.A$の入力ノードに$\triangleright$を用いて接続する
  $.A.\,\mathrm{Local} \triangleright .P.\,\mathrm{Out}_i$
- 現在の状態で自分（A）が到達可能なノード Reach（$.A$）を計算する
- 通信プロトコルの仕様に従い，到達可能なノードから，新たなメッセージを作成する

・出力
- $.A$の新たなメッセージを$.P$の入力ノードに$\triangleright$を用いて接続する
  $.P.\,\mathrm{In}_{j+1} \triangleright .A.\,\mathrm{Local}$

一方，ペネトレータの挙動は次のように定義される。

・入力

－到達可能なノードからイデアルを作成し，そのイデアルに属するメッセージを一つ選び新たなメッセージとする

・出力

－$.P$の新たなメッセージを$.A$の入力ノードに　を用いて接続する

$.A.\,\mathrm{Local} \triangleright .P.\,\mathrm{Out}_j$

このように，一般の主体とペネトレータプロセスの違いは，一般の主体メッセージの授受のパターンが決まっているのに対し，ペネトレータ入力，出力のタイミング，偽造可能なメッセージをその時生成できるイデアルの範囲で，任意に生成することができる。

5.5 非決定性

主体 A が送る具体的なメッセージはプロトコル設計の段階で規定（決定）することはできない。例えば，そのプロトコルが実行されるときにどのノンスが選択されるかは非決定的である。これを決定的な選択にした場合には，常にそのプロトコルでは同じノンスが選択されるため，プロトコルの仕様を知っているペネトレータは A のメッセージを解析すらすることなく A の通信内容を知ることができてしまう。

非決定性が存在する場合には $\triangleright$ は関数的でなくなるため，便宜上 $\triangleright$ は $\triangleright_1$，$\triangleright_2$ のインデックスを削除したものとして考えるものとする。

つぎのシナリオを考えてみよう。主体 A は送出するメッセージとして M_2 を選択する。それをコミュニケーションチャネル $P.\,\mathrm{com}$ に接続する。ペネトレータは同じく，適当な別のメッセージ M_3 をコミュニケーションチャネル $P.\,\mathrm{com}$ に dup する。このどちらを B が受信すべきメッセージとして選択するか，もしくはさせられるかは非決定的である。

$$.A.\,\mathrm{msg} \quad \triangleright \quad .M_2 \tag{1}$$
$$.P.\,\mathrm{com} \quad \triangleright_1 \quad .A.\,\mathrm{msg} \tag{2}$$
$$.P.\,\mathrm{com} \quad \triangleright_2 \quad .M_3 \tag{3}$$
$$.B.\,\mathrm{rcv} \quad \triangleright \quad .P.\,\mathrm{com} \tag{4}$$

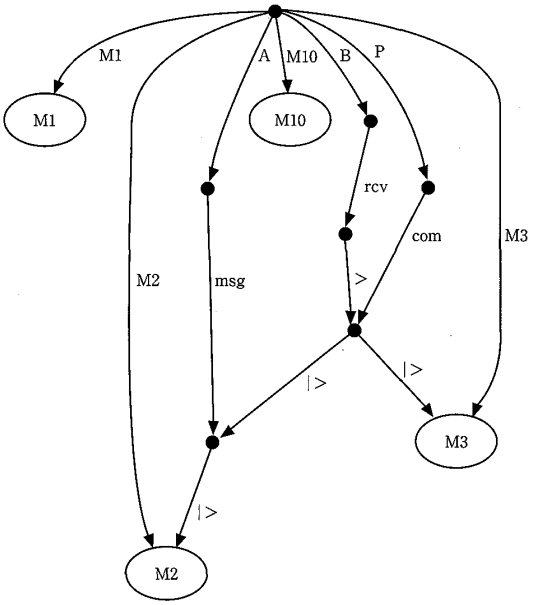
このとき，B が受信するメッセージ $.B.\,\mathrm{rcv}$ は次の可能性がある。

$$.B.\,\mathrm{rcv} \triangleright .P.\,\mathrm{com} \triangleright \begin{cases} .A.\,\mathrm{msg} \triangleright .M_2 \\ .M_3 \end{cases}$$

つまり，

$$.B.\,\mathrm{rcv} = .M_3 \neq .M_2 = .A.\,\mathrm{msg}$$

となり，A が送ったものと，B が受け取ったものが別のものとなる可能性がある（図5.5参照)。ペネトレータはこの錯誤を攻撃として利用する。



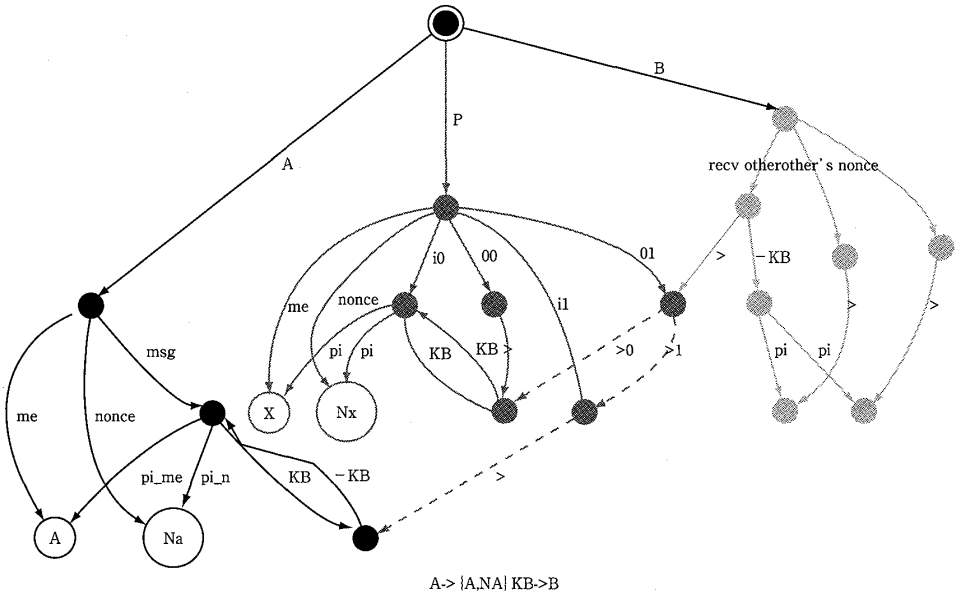
5.6 なりすまし可能性の解析例

ここでは，YANPI 法の具体モデルの目標であった，なりすましの攻撃の検証可能性を，NS プロトコルの一部分を解析することで例示しよう。

以下のメッセージの送信についてのアタックを考える。

$$A \xrightarrow{\{A,\ N_A\}_{K_B}} B$$

このメッセージに対するアタックは次のようなグラフが考えられる。



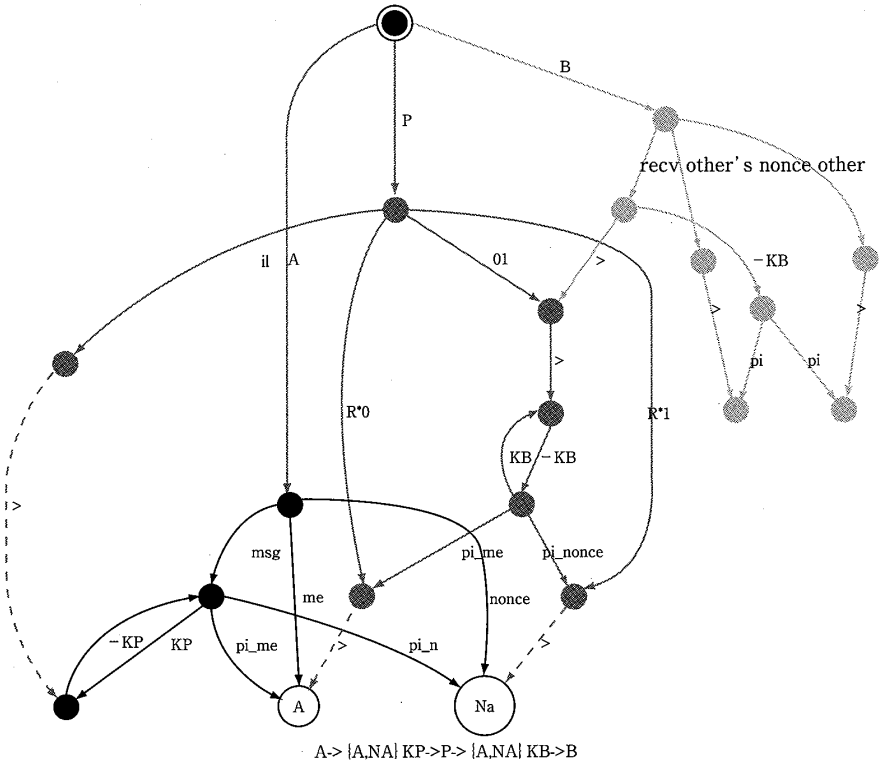
これは，次のようなグラフの書き換えをもたらす。

$$(.B.\,\text{other}, .B.\,\text{others}'\ \text{nonce}) \triangleright\ .B.\,(\text{other}, \text{others}'\ \text{nonce})$$
$$\triangleright .\underline{B.R.}K_B^{-1}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .\underline{P.O_1}.\ K_B^{-1}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.i_0.\ \underline{K_B.\ K_B^{-1}}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.i_0.\underline{\epsilon}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.\underline{i_0}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.\,(\text{me},\ \text{nonce})$$
$$\triangleright (.X, .N_X)$$

次に，以下のメッセージの送信についてのアタックを考える。

$$A \xrightarrow{\{A,\ N_A\}_{K_P}} P \xrightarrow{\{A,\ N_A\}_{K_B}} B$$

このメッセージに対するアタックは次のようなグラフが考えられる。



これは，次のようなグラフパスの書き換えをもたらす。

$$(.B.\,\text{other}, .B.\,\text{others' nonce}) \triangleright\ .B.\,(\text{other}, \text{others' nonce})$$
$$\triangleright .\underline{B.R.}K_B^{-1}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .\underline{P.O_1}.K_B^{-1}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.\text{msg}.\ \underline{K_B.\ K_B^{-1}}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.\text{msg}.\ \underline{\in}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.\text{msg}(\pi_{me,}\ \pi_{\text{nonce}})$$
$$\triangleright .P.(R_0^*,\ R_1^*)$$
$$\triangleright .A.(\text{me},\ \text{nonce})$$
$$\triangleright (.A, .N_A)$$

このとき，PがAのノンスを解析できる理由は，Pが到達可能ノードを解析して，Aの作ったメッセージの内容に次のようにアクセスできることによる。

$$.P.R_0^* \equiv .Pi_1 \triangleright \cdot \xrightarrow{K_P^{-1}} \cdot \xrightarrow{\pi_{\text{me}}} .A.\,\text{me} = A$$

$$.P.R_1^* \equiv .Pi_1 \triangleright \cdot \xrightarrow{K_P^{-1}} \cdot \xrightarrow{\pi_{\text{nonce}}} .A.\,\text{nonce} = N_A$$

以上の例から，YANPI法を利用して具体的なプロトコル解析が可能なことがわかった。

6　評価・結論

本論文では，プロトコル安全検証のための簡易検証のための数学的枠組みを示し，その具体意味論を構築した。この数学的枠組みはモデル検査機を作る上で必要な十分の仕様に関する情報を含んでいる。ここでいうモデル検査機はすべてのプロトコルのパターンに対して次のことを行う

1．実行グラフ書き換え
2．到達可能ノードの生成，ペネトレータイデアルの作成
3．エラーノードへの到達可能性検証

この実行グラフ作成，詳細化，到達可能性検証は状態グラフのサイズに対して線形の時間で検査でき，また，状態グラフは十分に小さい。しかし，具体意味論にしたがって直接モデル検査機を作成することはできない。それは，例の中でおいても，ペネトレータが生成するイデアルのなかから適当なパターンを人手で選択しており，また，イデアル自体は未だ無限空間のままである。

本研究ではこのことに対応するために，無限の具体空間を有限化する，抽象意味論を導入する。抽象意味論の構築とモデル検査のアルゴリズムについては次作の論文で紹介する。

謝辞

本研究は，平成17年度千葉商科大学研究助成を受け遂行したものである。本研究に理解を示していただいた関係各位に感謝の意を表します。

参考文献

[1] 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.

[2] A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. Marrero. An im-proved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178 (1－2): 237 - 255, 1997.

[3] M. Burrows, M. Abadi, and R. Needham. A logic of authentication, from proceedings of the royal society, volume 426, number 1871, 1989. In *William Stallings, Practical Cryptography for Data Internetworks, IEEE Computer Society Press, 1996*. Digital Equipment Corporation, 1996.

[4] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In K. G. Larsen and A. Skou, editors, *Proceedings of Computer Aided Verification (CAV' 91)*, volume 575, pages 48－58, Berlin, Germany, 1992. Springer.

[5] J. Edmund M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, 1999.

[6] T. A. Henzinger, S. Qadeer, S. K. Rajamani, and S. Tasiran. An assume-guarantee rule for checking simulation. In *Formal Methods in Computer-Aided Design*, pages 421－432, 1998.

[7] http://www.graphviz.org/. Graphviz-graph visualization software.

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

[9] G. Lowe. Breaking and fixing the Needham-Schroeder public-key proto-col 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.

[10] J. Millen and V. Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation, 2003.

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

[12] K. Takahashi, Y. Toda, and M. Hagiya. Nonce analysis and strand space model.

[13] J. Thayer, J. Herzog, and J. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 1999.

[14] T. Y. C. Woo and S. S. Lam. Verifying authentication protocols: Methodology

and example. In *1993 International Conference on Network Protocols*, pages 36－45, 19－22 1993.

—Abstract—

A Lightweight Verification Method
—Concrete Semantics—

To represent secure systems over open networks, there exists cryptosystems and its application. Robustness of a cryptosystem is given by computational approaches and formal ones. The computational approach relies on issues of complexity and probability, on the other hand, the formal method omit discussions about complexity of systems using an idealized cryptography, which is called "Computational hardness assumptions". Even if we omit computational discussion, it is still hard to prove the safety properties of network protocols. To cope with undecidability and complexity of analyzing formal systems, the author introduces "Yield-Arrow Nonce Protocol Inspection Method (YANPI method)". Using this method, we expect to get lightweight and intuitive verification method. The YANPI Method's model is established on a concrete semantics using directed graphs, however, its analyzing algorithms use abstracted directed graphs and its refinement.

In this paper, we introduce the formal concrete model of the YANPI method. In the next paper, we are planning to present YANPI method's abstraction and refinement algorithms.