SCIS 2013 The 30th Symposium on
Cryptography and Information Security
Kyoto, Japan, Jan. 22 - 25, 2013
The Institute of Electronics,
Information and Communication Engineers

計算論的に完全な記号的攻撃者と セキュリティ・プロトコルの計算論的に健全な検証

Computationally Complete Symbolic Adversary and Computationally Sound Verification of Security Protocols

バナ・ゲルゲイ* ウベール・コモン-ルンド† 櫻田 英樹‡
Gergei Bana Hubert Comon-Lundh Hideki Sakurada

あらまし セキュリティ・プロトコルの安全性を自動的に検証する方法として，記号検証と呼ばれる方法がある．記号検証では，メッセージをたとえば「$encrypt(N, K_{ab})$」などの記号列とし，攻撃者はたとえば「$encrypt(N, K_{ab})$ と K_{ab} から N を計算する」のようないくつかのルールのみを用いて攻撃を行うと仮定する．このようなルールは，攻撃者が明らかにできることを列挙して作られることが多い．このとき，記号検証で安全であるとされたプロトコルが計算論的にも安全であるという性質 (計算論的健全性) が成りたつことが望ましい．しかし多くの既存研究では，計算論的健全性のために非常に強い仮定が必要となるという問題がある．この問題を解決するため,「攻撃者が明らかにできること」をルールとして列挙するのではなく，暗号の安全性などの計算論的仮定から導かれる「攻撃者が破り得ないこと」を記号的なルールとして列挙し，これと矛盾しないあらゆるメッセージを送信することができる「計算論的に完全な」記号的攻撃者を仮定する記号検証を提案する．この方法の計算論的健全性と，Needham-Schroeder-Lowe プロトコルを検証するために十分なルールについて述べる．

キーワード プロトコル, 安全性, 自動検証

1 はじめに

セキュリティ・プロトコルの安全性を自動的に検証する方法として，記号検証と呼ばれる方法がある．記号検証では，メッセージをたとえば「$encrypt(N, K_{ab})$」などの記号列とし，攻撃者はたとえば「$encrypt(N, K_{ab})$ と K_{ab} から N を計算する」のようないくつかのルールのみを用いて攻撃を行うと仮定する．このようなルールは，攻撃者が明らかにできることを列挙して作られることが多い．この手法によるプロトコルの自動検証は多くの成功を収め，これを行うツールもいくつか存在する．

記号検証で安全であるとされたプロトコルが計算論的にも安全であるという性質 (計算論的健全性) が成りたつことが望ましい．しかし，Needham-Shroeder-Lowe のプロトコル [8] は記号検証で安全とされているが，メッセージのペアの構成が結合法則を満たすとき [9] や乱数をメッセージのペアと見なせるとき (第 6 節) に可能な計算論的な攻撃が存在する．さらに，これらの他に未知の攻撃が存在する可能性もある．記号検証の計算論的健全性については多くの研究 [1, 3, 2, 6] があるが，その多くは健全性のために暗号プリミティブとプロトコルの実装に非常に強い仮定を用いている．そのような仮定の例として，ビット列が一意に記号列にパースできる，暗号文から暗号化に用いられた公開鍵を知ることができる，鍵のサイクル（秘密鍵の対応する公開鍵による暗号化）がない，攻撃者は参加者をプロトコル実行中に買収しない，などがある．しかも，これらの仮定はプロトコルの安全性が成り立つためではなく，記号検証の計算論的健全性のために必要とされ，仮定が成り立たなくてもプロトコルは安全であるかもしれない．

本研究ではこの問題を解決するため，新しい記号検証の方法を提案する．本研究では,「攻撃者が明らかにできること」をルールとして列挙するのではなく，暗号の安全性などの計算論的仮定から導かれる「攻撃者が破り得ないこと」を記号的なルールとして列挙し，これと矛盾しないあらゆるメッセージを送信することができる「計算論的に完全な」記号的攻撃者を考える．

* Microsoft Research - INRIA Joint Centre, École Polytechnique, bâtiment Alan Turing, 1 rue Honorè d'Estienne d'Orves, 91120 Palaiseau, France, bana@math.upenn.edu
† Laboratoire Spécification et Vérification, École Normale Supérieure de Cachan, 61 avenue du Président Wilson, 94235 Cachan Cedex, France comon@lsv.ens-cachan.fr
‡ NTT コミュニケーション科学基礎研究所, 〒 243-0198 神奈川県厚木市森の里若宮 3-1 sakurada.hideki@lab.ntt.co.jp

従来法と本研究の違いは次のように説明できる．従来法の攻撃者は，計算論的なモデルにおいて明らかにできることのみが可能なため，一般に計算論的な攻撃者よりも弱い (図 1 左)．このため，健全性を成り立たせるため

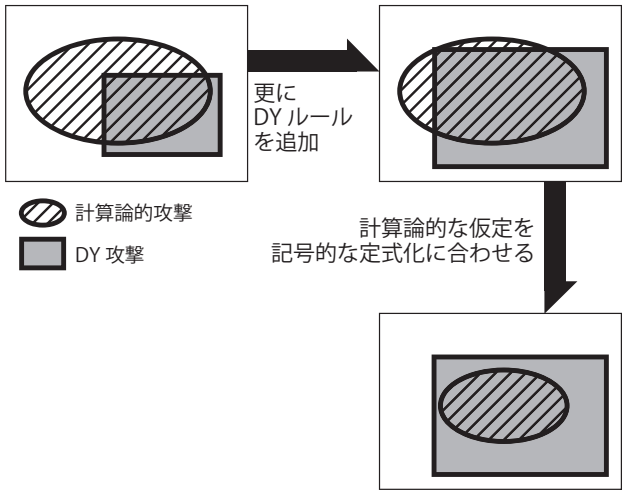

図 1: Dolev-Yao の方法の計算論的健全性

ルールを追加して攻撃者の能力を強める．しかし，ルールを追加しても十分強くならない場合 (図 1 右上) や，強くしすぎる場合があるため，計算論的な仮定を追加し，考慮すべき計算論的な攻撃を限定する (図 1 右下)．

一方，本研究では計算論的な攻撃よりもはるかに強い攻撃者を考え，それがプロトコルを検証するためには強すぎる場合は必要に応じて公理を追加し，攻撃者の能力を制限する (図 2)．このため，制限を加えるときにそれ

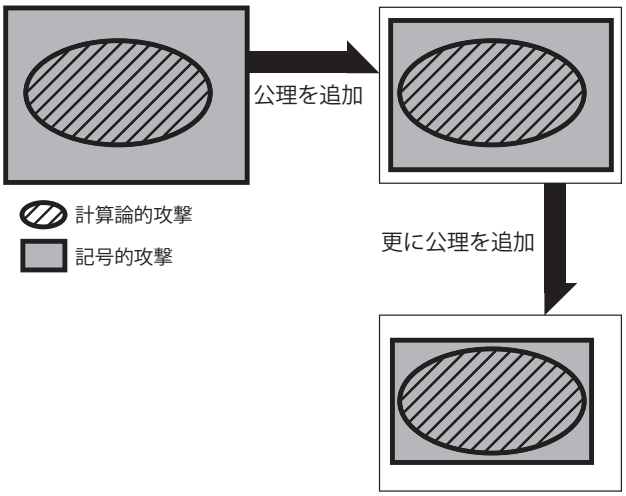

図 2: 我々の方法の計算論的健全性

が強すぎる制限でないことを証明しておけば，計算論的モデルを変更することなく計算論的健全性を得ることができる．このようにすれば記号的攻撃者は全ての計算論的攻撃者をカバーすることができる．つまり，記号的攻撃者が計算論的に完全であるといえる．

本研究は以上のような新しい記号検証の枠組みを提案するとともに，NSL プロトコルの安全性を証明するために十分な公理を示す．特に，暗号の IND-CCA 安全性に対応する秘匿性の公理および頑強性（non-malleability）の公理を示す．これら以外にほぼ自明な公理として，自己導出可能性，推測不能性 (no-telepathy)，導出可能性の遷移律，導出可能なものに関数を適用して得られるものもまた導出可能であることに対応する公理がある．さらに，以上の公理を用いて NSL プロトコルの安全性を用いて証明した．これにより，プロトコルが同時に有限個しか実行されないならば，公理と矛盾しない記号的な実行は安全性を満たすことがいえる．

なお，本研究の詳細は文献 [5] および文献 [4] されたい．

2 記号モデル

バナとコモン-ルンド [5] によるフレームワークを用いて記号モデルを定義する．このモデルは次ような考え方に基づく．

プロトコルの実行で交換されるメッセージは，項とよばれる記号列によって表わされる．たとえば項 $\{\langle N_1, A\rangle\}^{R_1}_{eK_B}$ はそれぞれノンス，参加者の名前，公開鍵，乱数を表わす記号 N_1，A，eK_B，R_1 および，ペアを表わす記号 $\langle \_, \_\rangle$ と暗号化を表わす記号 $\{\_\}^{\_}_{\_}$ からなる．従来のいわゆる Dolev-Yao の記号モデルでは攻撃者もまたこのような項を送信するが，ここではこれと異なり，攻撃者は「ハンドル」とよばれる記号 1 つだけからなる項を送信する．ハンドルが表す内容は論理式によって記述される．

Dolev-Yao の方法では，たとえば「$\{x\}_K$ および K から x を作れる (復号できる)」ことを表わすルール $\{x\}_K, K \vdash x$ などを考え，攻撃者はこのようなルールによって作れる項だけを送信できる．一方，我々の方法では,「攻撃者が項の集合 ϕ からハンドル h を作ることができる」ことを表す述語 $\phi \rhd h$ を考え，ϕ と h が満たすべき条件は公理と参加者のチェックの論理式によって記述される．公理は計算論的な実装における攻撃者の能力をもとに定義される．参加者のチェックの論理式はプロトコルのステップごとに決まる．たとえば参加者がハンドル h で表わされるメッセージをで受信し，それが以前に送信したノンス N_1 と一致することをチェックするとき，参加者が正しく次の動作に移るなら $h = N_1$ という性質が成り立つ．ハンドル h はこれらの条件に矛盾しない限りどのような性質を持っていてもよい．このため，公理が少ないほど強い攻撃者を考えることになり，公理が少なくても現実に可能な攻撃を網羅できる．

我々のモデルでは，プロトコルのセキュリティを表す論理式の否定，公理，そして正規参加者のチェックの論理式が互いに矛盾しないとき，プロトコルへの攻撃が成功すると定義する．

2.1 項とフレーム

参加者間で送受信されるメッセージは項によって表される．項は変数および定数に関数記号を繰り返し適用して得られる．定数には名前とハンドルがある．ここでいう名前は応用 π 計算の用語であり，通常の意味の名前とは異なる．名前は参加者名だけでなく，参加者が生成した乱数などのデータも含む．通常の意味の名前は「参加者名」あるいは「参加者の名前」と記述して区別する．ハンドルは攻撃者が送信したデータを表す．たとえば項 $\langle N_1, A\rangle$ は名前 N_1 および参加者の名前 A にペアを表わす 2 引数の関数記号 $\langle \_, \_\rangle$ を適用して得られる．さらに，項 $\{\langle N_1, A\rangle\}^{R_1}_{eK_B}$ は項 $\langle N_1, A\rangle$ および eK_B，および名前 R_1 に暗号化を表す 3 引数の関数記号 $\{\_\}^{\_}_{\_}$ を適用して得られる．ここで項 eK_B は名前 K_B に公開鍵を表わす関数記号を適用して得られる．本稿ではこの他，復号を表わす関数記号 $dec(\_, \_)$ を用いる．

フレームは名前の列 $\overline{n}$ と項の列 $t_1, \ldots, t_n$ からなり，$\nu\overline{n}.(t_1, \ldots, t_n)$ で表される．項の列 $t_1, \ldots, t_n$ は正規参加者が送信した項を表し，名前の列 $\overline{n}$ は正規参加者によってランダムに生成されたデータを表す．たとえばフレーム $\nu N.\langle N, A\rangle$ は，攻撃者が送ったメッセージ $\langle N, A\rangle$ を表し，このメッセージは正規参加者が生成したノンス N を用いて作られる．なお，$\nu\overline{n}$ を束縛子と呼ぶ．

2.2 論理式

攻撃者が送信するメッセージはそれまでに受信したメッセージから作ることができるものに限られる．正規参加者がこれを受信して次のステップに進むためにはプロトコルで決められた条件を満たさなければならない．このような条件を一階述語論理の論理式によって表す．特に述語記号として等号 $=$，導出可能性 $\_, \ldots, \_ \rhd \_$ および $\hat{\phi}, \_, \ldots, \_ \rhd \_$ および参加者の名前の述語 $\mathsf{AN}(\_)$ を用いる．これらの論理式は記号的実行についても計算論的実行についても用いられる．

論理式の計算論的な意味を第 3 節に詳しく説明するが，直観的には次のような意味である．$t_1, \ldots, t_n \rhd t_{n+1}$ は，項 $t_1, \ldots, t_n$ に対するビット列から項 t_{n+1} に対するビット列を確率的多項式時間アルゴリズムで計算できるという意味である．$\hat{\phi}, t_1, \ldots, t_n \rhd t_{n+1}$ は $t_1, \ldots, t_n$ に加えてそれまでに攻撃者が受けとったメッセージから t_{n+1} を計算できるという意味である．ただし，$\hat{\phi}, \_, \ldots, \_ \rhd \_$ はこれで 1 つの述語記号であり，$\hat{\phi}$ は変数でも具体的なフレームでもない．$\mathsf{AN}(t)$ は，項 t に対するビット列はある参加者の名前と等しいという意味である．

我々の方法と Dolev-Yao 方法では，記号的実行に対する述語の意味論が異なる．Dolev-Yao の方法では，述語 $t_1, \ldots, t_n \rhd t_{n+1}$ の記号的意味は項 $t_1, \ldots, t_n$ から t_{n+1} を Dolev-Yao のルールで計算できることである．しかしながら，我々はこのようなルールを定義しない．この述語の意味は一階述語論理のモデル $\mathcal{M}$ によって定義され，$\mathcal{M}$ は第 4 節の公理と第 2.3 節の正規参加者のチェックから得られる論理式を充足すればどのようなモデルでもよい．攻撃者が送信するメッセージもモデルによって決まる．したがって，それぞれのモデルは何らかの攻撃に対応する．さらに，$\hat{\phi}, t_1, \ldots, t_n \rhd t_{n+1}$ の意味は記号的実行の各状態に対して定義されている．ある状態での $\hat{\phi}, t_1, \ldots, t_n \rhd t_{n+1}$ の意味は，正規参加者がその状態までに送信したメッセージの列 $u_1, \ldots, u_n$ を $\hat{\phi}$ に代入して得られる $u_1, \ldots, u_n, t_1, \ldots, t_n \rhd t_{n+1}$ と同じである．述語 $\mathsf{AN}(t)$ の意味は，t がある参加者の名前に等しいという意味である．なお，参加者の名前の集合はプロトコルの実行前に決められる．論理和や限量子などの論理演算子の記号的意味は一階述語論理と同様である。

2.3 プロトコルの実行

記号的な実行および計算論的な実行は応用 π 計算の用語を用いて定義される．攻撃者はネットワークを完全にコントロールすることができる．記号モデルにおけるプロトコルの実行では，攻撃者がメッセージを送信することより参加者を含むネットワークの状態が変化する．ネットワークの状態を次の 4 つ組で表す．

- 制御状態 $q \in Q$ と名前 $n_1, ..., n_k$ の組 $q(n_1, ..., n_k)$
- ハンドルの列 $h_1, \ldots, h_n$
- 基底フレーム ϕ
- 論理式の集合 Θ

フレーム ϕ はこの状態までの実行で正規参加者が送信した項を表し，ハンドル $h_1, \ldots, h_n$ は攻撃者が送信したハンドルである．論理式の集合 Θ はハンドルの性質を表す論理式の集合である．正規参加者はハンドルで表わされるメッセージを受信するとき，その内容をチェックして次のステップに進む．このチェックの内容が集合 Θ に記録される．

例として，次の Needham-Schroeder-Lowe の公開鍵認証プロトコルを取り上げる．

1. $A \to B : \{N_1, A\}_{eK_B}$
2. $B \to A : \{N_1, N_2, B\}_{eK_A}$
3. $A \to B : \{N_2\}_{eK_B}$

このプロトコルの定式化のためにコンストラクタという関数記号の集合 $\mathcal{F}_c = \{\{\_\}^{\_}_{\_}, \langle \_, \_\rangle, e_{\_}, d_{\_}, K_{\_}\}$ およびデストラクタという関数記号の集合 $\mathcal{F}_d = \{dec(\_, \_), \pi_1(\_), \pi_2(\_)\}$ を固定する．次の等式も仮定する：(1) 暗号文の復号は平文と等しい，すなわち $dec(\{x\}^R_{eK}, dK) = x$．(2) ペアと投影に対する等式: $\pi_1(\langle x, y\rangle) = x$，$\pi_2(\langle x, y\rangle) = y$.

この事前の公開鍵の共有および最初の 2 つのメッセージの送受信の後の状態は次のフレームをもつ．

$$\phi_3 \equiv \nu_{K_A K_B N_1 R_1 N_2 R_2}\Big((A, B, eK_A, eK_B), \{\langle N_1, A\rangle\}^{R_1}_{eK_B}, \{\langle \pi_1\left(dec(h_2, dK_B)\right), \langle N_2, B\rangle\rangle\}^{R_2}_{eK_{\pi_2(dec(h_2, dK_B))}}\Big)$$

さらに，論理式の集合は次のようになる．

$$\Theta_3 = \{\phi_2 \rhd h_2, \mathsf{AN}(\pi_2\left(dec(h_2, dK_B)\right))\}$$

論理式 $\phi_2 \rhd h_2$ は，攻撃者がそれまで受けとった項の集合 ϕ_2 からハンドル h_2 を作ることができるという意味である．論理式 $\mathsf{AN}(\pi_2\left(dec(h_2, dK_B)\right))$ はハンドル h_2 を鍵 dK_B で復号でき，さらにそれがメッセージのペアになっており，その最後のパートは参加者の名前と等しいという意味である．ハンドル h がこの論理式を満たさなければ参加者 B は送信するメッセージを作れない．

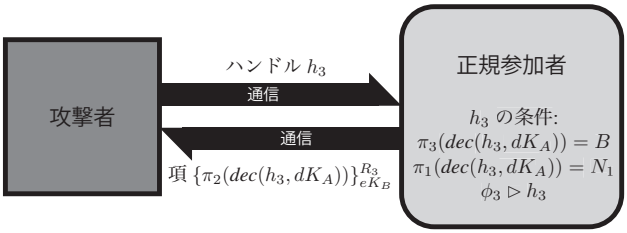

図 3: 記号的実行ラウンドの例

次の状態のフレーム ϕ_4 は ϕ_3 に加えて束縛子に R_3 が，項として $\{\pi_1\left(\pi_2\left(dec(h_3, dK_A)\right)\right)\}^{R_3}_{eK_B}$ が追加される．また，論理式の集合は $\Theta_4 = \Theta_3 \cup \{\phi_3 \rhd h_3,$ $\pi_1\left(dec(h_3, dK_h)\right) = N_1, \pi_2\left(\pi_2\left(dec(h_3, dK_A)\right)\right) = B\}$ となる．2 つの等式は暗号文を復号したその一部とそれぞれ N_1 および B との比較である．これは参加者 A によるチェックに対応する．

2.4 制約

これまでに述べた述語記号に加え，次のような制約を用いる．$\mathsf{Handle}(x)$ は x がハンドルであるという意味である．$\mathsf{RandGen}(x, \hat{\phi})$ は，x がフレーム $\hat{\phi}$ の束縛子に出現するという意味である．すなわち，x は x が正規参加者によって生成された乱数である．$x \sqsubseteq \hat{\phi}$ は x がフレームの項に出現するという意味である．すなわち，x は参加者が送ったメッセージの一部である．$x \sqsubseteq \vec{x}$ は x が $\vec{x}$ の一部であるという意味である．$dK \sqsubseteq_{\not d} \hat{\phi}$ は，dK が $\hat{\phi}$ の $dec(\_, dK)$ という形以外で出現するという意味である．$dK \sqsubseteq_{\not d} \vec{x}$ も同様である．

さらに，次のような略記を用いる．

- $x \sqsubseteq \hat{\phi}, \vec{x} \;\equiv\; x \sqsubseteq \hat{\phi} \vee x \sqsubseteq \vec{x}$
- $x \sqsubseteq_{\not d} \hat{\phi}, \vec{x} \;\equiv\; x \sqsubseteq_{\not d} \hat{\phi} \vee x \sqsubseteq_{\not d} \vec{x}$
- $\mathsf{fresh}(x; \hat{\phi}, \vec{x}) \;\equiv\; \mathsf{RandGen}(x, \hat{\phi}) \wedge x \not\sqsubseteq \hat{\phi}, \vec{x}$
- $\vec{x} \preccurlyeq \hat{\phi} \;\equiv\; h \sqsubseteq \vec{x} \wedge \mathsf{Handle}(h) \rightarrow \hat{\phi} \rhd h$

3 計算論的実行とその意味

計算論的実行もまた応用 π 計算の言葉で定義される．攻撃者はネットワークを完全にコントロールすることができるが，その計算能力は確率的多項式時間に限定される．記号的な実行との主な違いは，フレームに加えて参加者名やノンスやハンドルなどの記号に対応するビット列が記録される点である．さらに，記号的実行ではハンドル h の記号だけからは対応するメッセージの性質がわからないため，参加者がチェックする論理式を記録してこれを表現したが，計算論的実行ではこれは不要である．

記号的実行では論理式の真偽は実行の各状態において定義されるが，計算論的実行ではそうではない．なぜなら，以下に述べるように述語によっては計算論的実行の全体を確率分布とともに考える必要があるためである．項の計算論的な意味は，実行で用いるランダムネスの空間で定義される多項式時間アルゴリズムである．関数記号は多項式時間アルゴリズムとして解釈され，項 $f(t_1, \ldots, t_n)$ の解釈は項 $t_1, \ldots, t_n$ の解釈に関数記号 f の解釈を適用して得られる．述語記号の解釈は次のように確率的，計算量的に定義される．

等式 $t_1 = t_2$ が充足されるのは，項 t_1 および t_2 の解釈が無視できる確率を除いて等しいときである．より正確には，t_1 および t_2 の解釈を与えるアルゴリズムに，実行で用いたランダムネスを与えると無視できる確率を除いて同じ値が出力されるときである．

導出可能性 $t_1, \ldots, t_n \rhd t_{n+1}$ の解釈はより難しい．雑な言いかたをすると，ある確率的多項式時間アルゴリズム $\mathcal{A}$ が存在し，項 $t_1, \ldots, t_n$ の出力を与えられるときの出力が項 t_{n+1} の出力と一致する，というのがその解釈である．しかし，確率空間全体を 1 つのアルゴリズム $\mathcal{A}$ でカバーできるとは限らない．このため確率空間を無視できない大きさの部分集合に分割し，そのそれぞれで $t_1, \ldots, t_n$ の出力から t_{n+1} の出力を計算するアルゴリズムが存在するとき，導出可能性 $t_1, \ldots, t_n \rhd t_{n+1}$ が真であると定義する．

参加者名であることを表す述語 $\mathsf{AN}()$ の解釈は簡単で，ビット列が参加者の名前に等しければ真である．

一般に述語の解釈は，その引数の解釈であるアルゴリズムを，無視できる確率を除いて同じ値を出力する他のアルゴリズムで置き換えても変わらないと仮定する．

最後に，論理式の合成は通常の一階述語論理と同じようには定義されないことを強調したい．論理積 ($\wedge$) と全称限量子 ($\forall$) は通常と同じだが，それ以外は異なる．たとえば，論理和 $\theta_1 \vee \theta_2$ が真となるのは，確率空間を θ_1

が真になる空間と θ_2 が真になる空間に分けられるときである．存在限量子の意味も同様に，確率空間をさらに多くの空間に分けることで定義される．論理式 θ の否定 $\neg\theta$ が真となるのは，大きさが無視できないどの部分空間でも θ が真にならないときである．

これらの解釈は一階述語論理の通常の (Tarski 流の) 解釈と異なるが，一階述語論理の推論規則は健全である．実際，この解釈は Fitting による一階述語論理の (一階) 様相論理 S4 への埋め込みと深い関係にある [7].

以上のような解釈について以下のような計算論的健全性の定理 [5] が成り立つ．

定理 3.1 公理が計算論的に健全であると仮定する．任意のプロトコルとそのセキュリティについて，プロトコルが同時に有限個しか実行されないと仮定する．このとき，このプロトコルのセキュリティを無視できない確率で破るような計算論的攻撃者が存在すれば，このプロトコルのセキュリティを破るような記号的実行で公理に矛盾しないものが存在する．

いいかえれば，無視できない確率で実行可能な計算論的に成功する攻撃が存在すれば，記号的に成功する攻撃も存在するということである．

4 公理

本節では，計算論的に健全な公理について述べる．これらの公理によって NSL プロトコルだけでなく CCA2 安全な暗号を用いるその他のプロトコルの安全性が証明できる．さらに多くのプロトコルを検証するために，新しい公理を付け加えて拡張することもできる．このとき，もとの公理の健全性は保たれるので，この健全性を再度確認する必要はなく、新しい公理の計算論的な健全性を確認すれば十分である．

- **合同の公理．** 等価性"=" は合同である:
  - $x = x$ が成り立つ．また，述語記号の引数に表われる項は等価な項に置き換え可能である．ただし，制約の引数は置き換え可能とは限らない．

  前節において，述語の計算論的な意味論は「(述語の) 引数の解釈であるアルゴリズムを，無視できる確率を除いて同じ値を出力する他のアルゴリズムで置き換えても変わらない」と仮定したため，この公理は明らかに計算論的に健全である．

- **導出可能性の公理．** 確率的多項式時間 (PPT) 攻撃者を考えるとき，次の公理は健全である．最後の公理は，全ての関数記号が多項式時間計算可能関数を表すとき計算論的に健全である．
  - 自己導出可能性: $\hat{\phi}, \vec{x}, x \rhd x$
  - 導出可能性の強化: $\hat{\phi}, \vec{x} \rhd y \longrightarrow \hat{\phi}, \vec{x}, x \rhd y$
  - 可換性: $\vec{x}'$ が $\vec{x}$ を並べかえたものであるとき，$\hat{\phi}, \vec{x} \rhd y \longrightarrow \hat{\phi}, \vec{x}' \rhd y$
  - 導出可能性の推移律: $\hat{\phi}, \vec{x} \rhd \vec{y} \wedge \hat{\phi}, \vec{x}, \vec{y} \rhd \vec{z} \longrightarrow \hat{\phi}, \vec{x} \rhd \vec{z}$
  - 関数の導出可能性: $\hat{\phi}, \vec{x} \rhd f(\vec{x})$

- **関数記号に固有の等式．** これについては既に述べた:
  - $dec(\{x\}^R_{eK}, dK) = x$; $\pi_1(\langle x, y\rangle) = x$; $\pi_2(\langle x, y\rangle) = y$

- **乱数についての公理．** 以下の公理は制約 RandGen と $\sqsubseteq$ の関係を表す．推測不能性 (no telepathy) の公理は，送信されていない乱数は推測できないことを表す．この公理の計算論的健全性は，乱数が十分大きな空間から抽出されており無視できる確率でしか推測できないことからいえる．その次の公理の健全性は，乱数が独立であり，乱数 x を知っていてもそれが送信される前なら独立に生成された項 y について知るためには役に立たないことからいえる．ただし，$\{y\}^R_x$ のような x を使った項が送信された後では，x を知れば y を知ることができるかもしれない．条件 $\vec{x}, y \preccurlyeq \hat{\phi}$ は，$\vec{x}$ および y に出現するハンドルに攻撃者が後の実行で知る知識が含まれないようにするためにある．
  - 推測不能性 (No telepathy): $\mathsf{fresh}(x; \hat{\phi}) \longrightarrow \hat{\phi} \not\rhd x$
  - 新しい乱数は独立であり，他の乱数の情報をもたない: $\mathsf{fresh}(x; \hat{\phi}, \vec{x}, y) \wedge \vec{x}, y \preccurlyeq \hat{\phi} \wedge \hat{\phi}, \vec{x}, x \rhd y \longrightarrow \hat{\phi}, \vec{x} \rhd y$

- **IND-CCA2 暗号に特有の性質．** ここでは次の 2 つの公理を考える．両方とも，暗号が CCA2 安全であり乱数が無視できる確率でしか推測できない場合に健全である．最初の公理は秘匿性を表し，次の公理は頑強性 (non-malleability) を表す．両者は独立で，両者のうち一方から他の一方を導くことはできない．また，両者が充足される場合でも暗号が CCA2 安全であるとはいえない．計算論的健全性は次の定理で示される．

定理 4.1 (CCA2 暗号の秘匿性) 暗号スキームが *IND-CCA2* 安全であるとき，次の論理式は計算論

的に健全である．

$$\mathsf{RandGen}(K,\hat{\phi}) \wedge \hat{\phi} \rhd eK \wedge \mathsf{fresh}(R;\hat{\phi},\vec{x},x,y) \\ \wedge \vec{x},x,y \preccurlyeq \hat{\phi} \wedge \hat{\phi},\vec{x},\{x\}^R_{eK} \rhd y \\ \longrightarrow dK \sqsubseteq_{\not d} \hat{\phi},\vec{x},x \vee \hat{\phi},\vec{x} \rhd y$$

この公理の意味は，K が正しく生成され R が新しく生成された乱数であり y が $\{x\}^R_{eK}$ を使って導出できるとき，y は $\{x\}^R_{eK}$ を使わなくても導出できるというものである．式変形により $dK \not\sqsubseteq \hat{\phi},\vec{x},x$ を条件とすることができるが，これは dK が公開されていないという条件で，これにより鍵のサイクル (鍵をその鍵で暗号化) がないことがいえる．なお，より条件を弱めた公理で置き換えることも可能であるが，本稿では扱わない．さらに，条件 $\vec{x},x,y \preccurlyeq \hat{\phi}$ は $\vec{x},x,y$ が後の実行で得られる情報を含まないようにするためにある．なお，この公理は CPA 安全な暗号一般に成り立つように見えるかもしれないがそうではない，なぜならここでは正規参加者を復号オラクルとして使えるためである．この定理を証明するために，CCA2 ゲームの暗号オラクルが暗号文を作る前にのみ復号オラクルを使う必要がある．つまり，暗号は IND-CCA1 安全であれば十分である．

次に頑強性 (non-malleability) の公理を与える．制約 $\mathsf{MayCor}_{\mathrm{CCA2}}(u;\hat{\phi},\vec{x})$ は，項 u がフレームあるいは項の列 $\vec{x}$ に出現し，次の性質を満たすという意味である．すなわち，u は $\mathcal{F}_c \cup \mathcal{F}_d$ に含まれない関数記号あるいは復号関数記号の引数として暗号化によって保護されていない形で出現する．

定理 4.2 (CCA2 暗号の頑健性)　暗号スキームが *IND-CCA2* 安全であるなら，以下の公理は計算論的に健全である．

$$\exists u(\mathsf{MayCor}_{\mathrm{CCA2}}(u;\hat{\phi},\vec{x}) \wedge \hat{\phi},\vec{x},u \not\rhd N) \wedge \hat{\phi} \rhd eK \\ \wedge \mathsf{RandGen}(N,\hat{\phi}) \wedge \mathsf{RandGen}(K,\hat{\phi}) \wedge \vec{x} \preccurlyeq \hat{\phi} \\ \wedge N \sqsubseteq \hat{\phi},\vec{x} \wedge \hat{\phi},\vec{x} \rhd y \wedge \hat{\phi},\vec{x},dec(y,dK) \rhd N \\ \longrightarrow \exists K'(\mathsf{RandGen}(K',\hat{\phi}) \wedge dK' \sqsubseteq_{\not d} \hat{\phi},\vec{x}) \\ \vee \exists x R(y = \{x\}^R_{eK} \wedge \{x\}^R_{eK} \sqsubseteq \hat{\phi},\vec{x})$$

上の公理は次のような意味である．すなわち，N と K が正しく生成され，y が復号でき，その平文から N が導出でき，さらに N はその平文なしでは導出できないとき，正規参加者がある復号鍵 dK' を送信したか，あるいは y は暗号文として構成したかのいずれかである．この定理の証明では暗号化オラクルの呼び出しの前にも後にも復号オラクルを用いるため，CCA2 安全性が必要である．最初の制約は，暗号化およびペアとは無関係の関数記号が CCA2 暗号と干渉しないようにするために用いられている．原理的にはたとえば他の暗号化関数が我々の CCA2 暗号と同じ暗号文を生成することが可能である．さらに，CCA2 安全な暗号であっても $dec(N,dK)$ の結果が N とならない保証はない．NSL プロトコルの場合は，制約 $\mathsf{MayCor}_{\mathrm{CCA2}}$ では関数記号 $dec(\_,\_)$ の引数である項だけを考えればよい．

5　矛盾の証明の例

NSL プロトコルの証明の前に，公理の使いかたを例を用いて説明する．以下で用いる推論では，公理と一階述語論理の推論規則のほかに記号実行の性質を用いる．記号実行は一階述語論理で定義されていないので，以下の推論は純粋な一階述語論理とは異なる．

例 5.1　まずは自明な例を考える．第 2.3 節の NSL プロトコルの実行において，$\phi_2 \not\rhd A$ は公理に矛盾する．なぜなら，A は ϕ_2 に含まれており次のように導出できるためである．まず ϕ_2 は次のような性質を持つことに注意する．

$$\begin{aligned}\phi_2 &\equiv A,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B} \\ &\equiv \phi_0,A,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B}.\end{aligned} \tag{1}$$

ϕ_0 について自己導出可能性公理を用いて，$\phi_0,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B}, A \rhd A$ が得られる．さらに，可換性により $\phi_0,A,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B} \rhd A$ が得られる．したがってこの否定 $\phi_0,A,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B} \not\rhd A$ は公理に矛盾する．従って $\mathcal{M},\sigma \not\models A,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B} \not\rhd A$ であり，(1) により $\phi_2 \not\rhd A$ も公理に矛盾する．

例 5.2　同じく NSL プロトコルの例で，$\phi_2 \rhd N_1$ が公理に矛盾することも示せる．N_1 が一度だけ正しく暗号化されて送信されていることを用いる．次が成り立つため，$\phi_2 \equiv A,B,eK_A,eK_B,\{\langle N_1,A\rangle\}^{R_1}_{eK_B} \equiv \phi_1,\{\langle N_1,A\rangle\}^{R_1}_{eK_B}$, 次が公理に矛盾することを示せばよい．

$$\phi_1,\{\langle N_1,A\rangle\}^{R_1}_{eK_B} \rhd N_1. \tag{2}$$

秘匿性公理を用いるため，$\vec{x} = \langle\rangle, x = \langle N_1,A\rangle$ および $y = N_1$ とする．K_B は名前であるため正しく生成されており $\mathsf{RandGen}(K_B,\hat{\phi})$ が成り立つ．さらに $eK_B \sqsubseteq \phi_1$ が成り立つ．また $\mathsf{RandGen}(R_1,\hat{\phi}) \wedge R_1 \not\sqsubseteq \phi_1,\langle\rangle,\langle N_1,A\rangle,N_1$ であるから $\mathsf{fresh}(R_1;\phi_1,\vec{x},x,y)$ もいえる．$\vec{x} \preccurlyeq \phi_1$ であり，また x および y にはハンドルが現れないため $x \preccurlyeq \phi_1$ と $y \preccurlyeq \phi_1$ が成り立ち，式 (2) および $dK_B \not\sqsubseteq \phi_1,\vec{x},x$ が成り立つため，秘匿性公理を用いて次がいえる．$\phi_1 \rhd N_1$. ステップ 1 において，$\mathsf{RandGen}(N_1,\hat{\phi}) \wedge N_1 \not\sqsubseteq \phi_1$ より

$\mathsf{fresh}(N_1;\phi_1)$ が導かれ，$\mathsf{fresh}(N_1;\phi_1)\wedge\phi_1 \rhd N_1$ が成り立つ．これは推測不能性公理に矛盾する．

例 5.3 公理を用いて，攻撃者の知識がしだいに増えていくことも証明できる．すなわち，任意の m と x について $\phi_m \rhd x$ が公理と参加者のチェックの論理式から導かれるなら $\phi_{m+1} \rhd x$ もまた同様である．前者を仮定して後者を示す．t をプロトコルの第 $m+1$ ステップで送信された項であるとする．つまり，$\phi_{m+1} \equiv \phi_m, t$ であるとする．導出可能性の強化の公理を適用すると $\phi_m \rhd x$ から $\phi_m, t \rhd x$ がいえる．したがって次がなりたつ．$\phi_{m+1} \rhd x$ この例と例 5.1 を用いれば，任意の m について公理から $\phi_m \rhd A$ を導くことができる．

6 NSL プロトコルへの計算論的攻撃

本節では第 4 節の公理だけでは NSL プロトコルの安全性検証ができないことを示す．これは，正しく生成されたノンス N が $N = \langle n, Q\rangle$ のようにノンスのような文字列と悪意のある参加者の名前のペアとして使われる場合があるためである．

より正確には，論理式 $\mathsf{RandGen}(N,\hat{\phi}) \wedge \mathsf{AN}(\pi_2(N))$ が充足可能なときに，$N = \langle n, Q\rangle$ を満たす参加者の名前 Q とビット列 n が無視できない確率で存在する，ということが起りうる．実際，ペア $\langle x, y\rangle$ が x と y の単なる連結として実装されている場合は，参加者の名前がたとえば 8 ビットの固定長のとき，N が一様な確率で選ばれるならその最後の 8 ビットが任意の参加者名 Q に一致する確率は $1/2^8$ であり，無視できない．これを用いて，次のような攻撃が可能である (図 4)．(1) 攻撃者は名

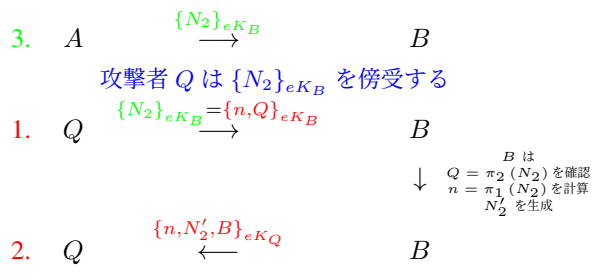

図 4: NSL プロトコルへの攻撃

前 Q 選ぶ．(2) 正規の参加者 A と B が実行したセッションの最後のメッセージ $\{N_2\}_B$ を攻撃者が受信する．(3) 攻撃者は参加者 Q のふりをして参加者 B と新しいセッションを開始し，$\{N_2\}_{eK_B}$ を送信する．(4) 参加者 B は $\{N_2\}_{eK_B}$ を $\{\langle N_1', Q\rangle\}_{eK_B}$ として扱う．(5) 参加者 B は新しいノンス N_2' を生成し，メッセージ $\{n, N_2', B\}_{eK_Q}$ を参加者 Q に送る．(6) 攻撃は参加者 Q として，メッセージ $\{n, N_2', B\}_{eK_Q}$ を復号し，ビット列 n を読み，$N_2 = \langle n, Q\rangle$ を計算でき，ノンス N_2 の秘匿性を破ることができる．

以上の攻撃により，等式 $N = \langle n, Q\rangle$ が無視できない確率で成り立つときはこのプロトコルは安全でないといえる．しかし，従来の記号検証はこのような攻撃を発見できない．したがって従来の記号検証で計算論的健全性が成り立つためには，異なったタイプの項に対応するビット列が混同されないという強い仮定を置き，たとえば等式 $N = \langle n, Q\rangle$ などが成り立たないようにする必要がある．

なおプロトコルの実装で参加者 B がビット列 n の長さを常にチェックするようにすれば，この攻撃ができないことは明らかである．このため，実装において論理式 $\mathsf{RandGen}(N,\hat{\phi}) \rightarrow \neg\mathsf{AN}(\pi_2(N))$ が成り立つことを確かめる必要がある．あるいは，ノンスであることを表す述語記号 $Nonce(\_)$ を使った論理式 $\mathsf{RandGen}(N,\hat{\phi}) \rightarrow \neg Nonce(\pi_2(N))$ を確かめてもよい．

ここで注意すべきなのは，この攻撃は通常のいわゆる型混同攻撃 (type-flaw attack) とは異なる．なぜなら，型混同攻撃でも通常はノンスをそれ分解できないとするためである．

7 NSL プロトコルの安全性証明

我々の手法がプロトコルの検証に使えることを示すため，この手法によってセッション数を任意に制限した場合の NSL プロトコルの安全性の証明を与える．プロトコルの記号実行において秘匿性 (secrecy) または認証 (authentication) が破られるとすると公理に矛盾することを示す．ここでは正規参加者 A および B はそれぞれイニシエータとレスポンダの役割のみを実行すると仮定する．ただし，A および B が攻撃者に買収された参加者ともセッションを行うことを許す．次のようにして証明を行う．

まず参加者 A および B が生成して互いに送ったノンスの秘匿性を示す．ノンス N について，初期状態では推測不能性 (no-telepathy) の公理より，秘匿性 $\phi_0 \not\rhd N$ が成り立つ．したがって帰納法によれば，実行の各ステップ m におけるノンス N の秘匿性 $\phi_m \not\rhd N$ を示すためには，各ステップ m で論理式 $\phi_m \not\rhd N$ および公理と参加者のチェックの論理式から論理式 $\phi_{m+1} \not\rhd N$ が導かれることを示せばよい，これは，論理式 $\phi_m \not\rhd N$，公理，参加者のチェックの論理式，および論理式 $\phi_{m+1} \rhd N$ が矛盾することを示せばよい．実際にはもっと複雑な帰納法の仮定を用いるが，本質的には以上のとおりである．

次に，ノンスの秘匿性と頑健性 (non-malleability) の公理を用いて認証とノンスの共有を示す．すなわち，参加者 B が A を相手としてセッションを完了したとき，参加者 A が B を相手として完了したある

セッションで，B によるセッションと同じノンスを用いたものが存在する．これを示すためには，B だけが

セッションを完了したかあるいは完了してもパラメータが一致しないと仮定し，公理および参加者のチェックとの矛盾を示せばよい．

以上の詳細は文献 [4] のオンライン入手可能なバージョンにある．したがって次の定理が得られる．

定理 7.1 (秘匿性，認証とノンスの共有) 任意の *NSL* プロトコルの記号実行を考える．この実行には任意の数の買収された参加者と，*2* つの正規参加者 A および B が参加しているとする．また，A および B はそれぞれイニシエータとレスポンダの役割のみを実行しているとする．ただし，A および B が実行するセッションの数はある定数以下であるとする．さらに，メッセージのペアについて等式 $\langle x, y, z\rangle \equiv \langle x, \langle y, z\rangle\rangle$ が成り立つとする．このき次が成り立つ．

- A または B によって生成されそれぞれ B または A に送られた任意のノンス N について，任意の $n \in \mathbb{N}$ および $\phi_n \not\vdash N$ が成り立つことが公理 $\mathsf{RandGen}(N, \hat{\phi}) \to \neg\mathsf{AN}(\pi_2(N))$ から導かれる．
- B が A を相手としてセッションを完了し，このセッションにおいてノンス N_1 およびノンス N_2 を用いたとき，A により実行されたあるセッションで，B を相手としており同じノンスを用いたものが存在する．

この定理では，参加者によるメッセージのパーシングについての条件は $\mathsf{RandGen}(N, \hat{\phi}) \to \neg\mathsf{AN}(\pi_2(N))$，すなわち,「ノンスをペアとみなして分解しても最後の要素が参加者名になることはない」という性質だけである．つまり，NSL プロトコルの安全性のためにはノンスやペアや暗号文が混同されないように常にタグをつけて識別したりする必要はない．一方，従来の記号検証では計算論的健全性を成り立たせるためにこのような性質を必要とする．

8 結論

本研究ではセキュリティ・プロトコルの記号検証の新しい方法を提案した．従来法のように記号的攻撃者に許される計算をルールとして列挙する方法では，ルールを追加しても計算論的攻撃者の全ての能力をカバーできるようにはならない．このため本研究では，記号的攻撃者に許されないことをルールとして列挙する．ここではこのようなルールを公理と呼ぶ．公理は計算論的に健全でなければならない．すなわち，公理はプロトコルの計算論的な実装に対して常に真でなければならない．このようにすれば，計算論的攻撃者ができることは全て記号的攻撃者にも可能となる．より厳密には，定理 3.1 で示した計算論的健全性が成り立つ．

我々の方法では次の 2 点が問題となる．(1) プロトコルの安全性を検証するために十分な公理を見つけられるか？(2) この方法を自動化できるか？前者については，これが可能であることを実際に NSL プロトコルについて示した．後者については現在研究中である．

参考文献

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[2] M. Backes, D. Hofheinz, and D. Unruh. CoSP: A general framework for computational soundness proofs. In *CCS'09*, 66–78, 2009.

[3] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *CCS'03*, 220–230. ACM, 2003.

[4] G. Bana, P. Adão, and H. Sakurada. Computationally Comlete Symbolic Attacker in Action. In *FSTTCS 2012*, LIPIcs, 2012. 掲載決定．ロングバージョンは `http://eprint.iacr.org/2012/316`.

[5] G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In *POST'12*, 189–208. Springer, 2012. ロングバージョンは `http://eprint.iacr.org/2012/019`.

[6] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *CCS'08*, 109–118. ACM, 2008.

[7] M. Fitting. An embedding of classical logic in S4. *The Journal of Symbolic Logic*, 35(4):529–534, 1970.

[8] G. Lowe. Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR. In *TACAS'96*, 147–166, 1996.

[9] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS'01*, 166–175. ACM, 2001.