SET 支払いプロトコルの秘匿性検証

櫻田英樹†

セキュリティ・プロトコルの安全性を証明によって検証する論理的検証は，プロトコル仕様記述の柔軟性と検証の完全性の点で優れている．特に Paulson による帰納的方法は，実際に使用されているプロトコルの検証の実績があり，証明スクリプトも公開されている．しかしこの方法では大きなプロトコル仕様記述が繁雑になりやすいという欠点があった．本論文ではプロトコル仕様を簡潔に記述する言語と，この言語で記述したプロトコル仕様を帰納的方法の論理式に変換する方法を提案する．これにより大きなプロトコルを検証する際にも帰納的方法の証明スクリプトの多くをそのまま利用でき，検証を効率的に行うことができる．本論文ではさらに，これを SET（Secure Electronic Transaction）の支払いプロトコルの検証に適用し，カード番号の秘匿性を検証することができた．その際，検証を現実的な時間内で可能にするために，検証を目指すセキュリティから参照されない部分を省略して簡略化されたプロトコル仕様を得た．

Verification of Secrecy of the SET Payment Protocol

Hideki Sakurada†

To verify security protocols by giving the proofs of security properties has advantages in flexibility of protocol specification and completeness of verification. In particular, Paulson's Inductive Method is used to verify several practical protocols and the proof modules are freely available. However, protocol specifications in this method tend to be complicated. This paper proposes a language in which one can express protocols concisely and also proposes a method to convert the protocol specification to logical formulae for Inductive Method. They make it possible to reuse the proof modules of Inductive Method in verification of large security protocols. This paper also applies them to the SET payment protocol and verifies the secrecy of payment card numbers in the protocol. To achieve this, this paper simplifies the protocol by omitting parameters that are not referred in security requirements we aim to verify.

1. はじめに

インターネットを用いた電子商取引や遠隔ログインのような通信では，暗号を用いた通信プロトコル仕様を用いてセキュリティを確保している．すでに広く使用されているプロトコルにセキュリティの欠陥が発見された場合，すべての端末においてプロトコルを使用するソフトウェアのバージョンアップが必要になり，大きな費用が発生する．このため，通信プロトコルの欠陥を運用前に発見するプロトコル検証の技術が重要である．

プロトコル検証技術のなかでも，論理式で記述したプロトコルを公理あるいは推論規則とし，同じく論理式で記述したセキュリティ要求を定理証明システムを用いて証明する方法[3),5),11)]は，プロトコルの並列実行の数や生成されるノンス（乱数またはタイムスタンプ）の数に一定の上限を設けて状態を網羅的に検査するモデル検査に比べ，プロトコル仕様やセキュリティを柔軟に与えられる点と上限のない完全な検証を行える点で優れている．なかでも Paulson による帰納的方法[11)]は KerberosIV や TLS の検証[1),12)]などの実績があり，証明スクリプトが定理証明システム Isabelle[10)]とともに配布されており誰でも利用できる．

本論文では帰納的方法を用いて SET（Secure Electronic Transaction）[15)~17)]の支払いプロトコルの秘匿性の検証を行う．配布されている証明スクリプトの多くをそのまま利用することにより効率的に検証を行う．このとき次の 2 点が問題となる．

(1) Paulson らはプロトコル仕様を論理式（公理あるいは推論規則）として直接記述しているが，SET のように大きなプロトコルの場合に記述が繁雑になり，記述そのものに誤りが生じる可

† 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation

能性がある．

(2) SET は仕様が 400 ページ以上もある巨大なプロトコルであるため，これをそのまま検証するのは現実的な時間内にはできない可能性が高い．

本論文ではこれらをそれぞれ次のように解決する．

(1) プロトコル仕様を簡潔に記述できる言語を定義するとともに，これを帰納的方法の論理式に変換する方法を定義する．これにより大きなプロトコルの仕様を簡潔に記述でき，さらに帰納的方法の既存の証明スクリプトを再利用して効率的に検証を行うことができる．

(2) 検証すべきセキュリティ要求に着目し，プロトコル仕様からセキュリティ要求とは関係ないと考えられる部分を省略する．これにより適度に簡略化されたプロトコル仕様を得ることができ，現実的な時間内で検証が可能になる．

本研究では Meadows ら[9] によって与えられた SET の支払いプロトコルのセキュリティ要求の検証をめざしているため，これを参照する．本研究で行う秘匿性の検証はこのセキュリティ要求を検証するために不可欠な部分である．

検証方法の点から本研究と関連が深い研究として，前述の Paulson による帰納的方法による検証がある．本研究のプロトコル仕様記述・変換は彼らの方法で問題となるプロトコル仕様記述の繁雑さの問題を解決するものである．また，SET の検証の研究としては，Bella らによる SET の支払いプロトコルの検証の研究がある[2]．彼らは帰納的方法を用いて支払い情報の秘匿性だけでなくデータの完全性や顧客と商店の認証についても検証を行っている．彼らはプロトコルを推論規則として直接記述しているのに対し，本論文ではプロトコル仕様記述をより簡潔に行うためにプロトコル仕様記述・変換を提案して用いている．このほかに SET の検証の研究として，SET の支払いプロトコルを CSP[7] を用いて記述し，秘匿性を含まないいくつかの性質をモデル検証器 FDR[6] を用いて検証した Lu らによる研究[8] がある．彼らが並列に動作する顧客の数を 2 人に限るなどの上限を設けて検証を行ったのに対し，本論文では並列実行の数などに上限を設けずに検証を行う．

本論文ではまず検証の対象とする SET プロトコルと，検証のバックエンドとして利用する Paulson の帰納的方法について概説する（2，3 章）．次にプロトコル仕様記述のための言語と論理式への変換法を提案する（4，5 章）．次に SET の支払いプロトコルを簡略化し，4 章で提案する言語で記述する（6 章）．そしてこれを 5 章の方法で変換し，SET の支払いプロトコルの秘匿性の検証を行う（7 章）．最後にまとめと今後の課題について述べる（8 章）．

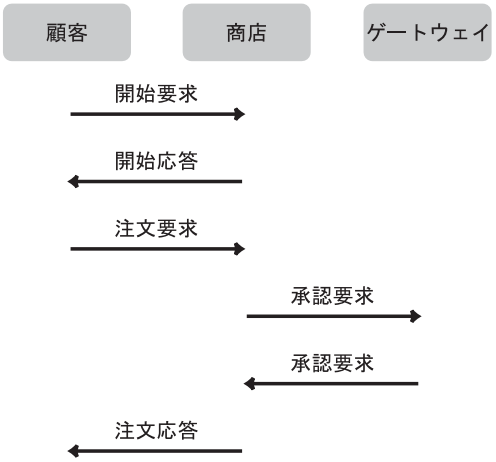

図 1 SET の支払いプロトコルにおけるメッセージの流れ
Fig. 1 Message flow in SET payment protocol.

2. SET の支払いプロトコル

SET はクレジットカード（payment card）を用いた電子決済のためのプロトコルで，Visa と MasterCard などが中心になって策定したものである．SET の目的は，

- 情報の秘匿性
- 支払いの信頼性
- 商店（merchant）と顧客（cardholder）の認証

を提供し，インターネット・ショッピングなどの新しい市場でのクレジットカードの使用を促し，クレジットカードがより広い範囲で受け入れられるようにすることである[15]．

SET は大きく分ければ，顧客と商店の鍵を認証局に登録する部分と，顧客・商店・カード会社の間で売買の情報を交換する支払いプロトコルの部分からなる．本研究ではこのうち実際の支払いを行う後者の部分を扱う．

SET の支払いプロトコルの特徴は，**図 1** のように顧客，商店，支払いゲートウェイの 3 者が参加する点である．支払いゲートウェイはカード会社（acquierer）に相当する参加者で，SET のネットワークと既存のクレジットカードのネットワークを結ぶ．顧客のクレジットカード番号（*PAN*）は注文要求のメッセージに含まれて商店に送信されるが，暗号化により商店には読まれない．このため商店によるカード番号を利用した詐欺を防止できると期待される．

しかし，SET の支払いプロトコルがセキュリティ要求を実現しているかは明らかではないため，検証を行う必要がある．SET の支払いプロトコルには図 1 に示した以外にも代金引き落としを取り消すメッセージなどがあるが本研究では売買の成立を確認するために必要な図 1 の 6 つのメッセージに絞って秘匿性の検証を行う．

3. Paulson の帰納的方法

プロトコルを定理証明の技術を用いて検証する Paulson の帰納的方法について説明する．

この方法では，プロトコルはトレースの集合を帰納的に定義する論理式（推論規則）の集まりとして記述される．トレースは Says $A\ B\ M$ という形のイベントのリストである．Says $A\ B\ M$ は参加者 A が参加者 B にメッセージ M を送信するイベントである．トレースの集合を定義する推論規則は一般に次の形をしている．

$$
\begin{array}{ll}
 & trace \in traceSet \\
\wedge & \text{Says } A\ B_1\ M_1 \in trace \\
 & \vdots \\
\wedge & \text{Says } A\ B_n\ M_n \in trace \\
\wedge & \text{Says } C_1\ A\ L_1 \in trace \\
 & \vdots \\
\wedge & \text{Says } C_m\ A\ L_m \in trace \\
\wedge & cond \\
\rightarrow & ([\text{Says } A\ B\ M] + trace) \in traceSet
\end{array}
\qquad (1)
$$

ただし，‘+’ はリストの連結を表し，‘∈’ は集合またはトレース（リスト）の要素であることを表す．‘∧’ および ‘→’ はそれぞれ論理積および含意を表す．記号の結合の強さは強いものから順に，‘+’，‘∈’，‘∧’，‘→’ であるとする．この推論規則は，「参加者 A が参加者 $B_1, \ldots, B_n$ にそれぞれメッセージ $M_1, \ldots, M_n$ を送信し，参加者 $C_1, \ldots, C_m$ が参加者 A にそれぞれメッセージ $L_1, \ldots, L_m$ を送信したならば，参加者 A は参加者 B にメッセージ M を送信する」という意味である．この推論規則により定義されるトレースの集合 *traceSet* は，参加者がプロトコルに従って行動したときに発生させるイベントのリストの集合である．*cond* にはメッセージの性質などが記述される．

例 1 Yahalom のプロトコル[4] におけるイニシエータを定義する推論規則[13] を**図 2** に示す．ここで $\mathsf{shrK}(A)$ は A と鍵サーバとの間であらかじめ共有されている鍵である．この 2 つの推論規則のうち最初の推論規則は，イニシエータである参加者 A が新しいノンス N_A を生成し，B に $\{A, N_A\}$ を送信するという意味である．2 番目の推論規則は，過去に参加者 A が参加者 B にメッセージ $\{A, N_A\}$ を送信し，さらに誰か (B') からメッセージ $\{\{B, K, N_A, N_B\}_{\mathsf{shrK}(A)}, X\}$ を受けとったとき，B に $\{X, \{N_B\}_K\}$ を送信するという意味である．used *trace* は *trace* の中で過去に用いられたパラメータの集合である．$N_A \notin$ used *trace* は式 (1) の *cond* に相当し，N_A が新しく生成したノンスであることを意味する．いま，

$$
\begin{array}{l}
trace \in traceSet \\
\wedge N_A \notin \text{used } trace \\
\rightarrow ([\text{Says } A\ B\ \{A, N_A\}] + trace) \in traceSet
\end{array}
$$

$$
\begin{array}{l}
trace \in traceSet \\
\wedge \text{Says } A\ B\ \{A, N_A\} \in trace \\
\wedge \text{Says } B'\ A\ \{\{B, K, N_A, N_B\}_{\mathsf{shrK}(A)}, X\} \\
\qquad \in trace \\
\rightarrow ([\text{Says } A\ B\ \{X, \{N_B\}_K\}] + trace) \\
\qquad \in traceSet
\end{array}
$$

図 2 Yahalom のプロトコルのイニシエータを定義する推論規則
Fig. 2 Inference rules that define initiator's action in Yahalom protocol.

$$[] \in traceSet$$

を仮定し，図 2 の最初の推論規則を適用すれば，

$$[\text{Says } A\ B\ \{A, N_A\}] \in traceSet$$

が得られる．さらにここではあげなかったレスポンダの推論規則を適用すれば，

$$
\begin{array}{l}
[\ \text{Says } B'\ A\ \{\{B, K, N_A, N_B\}_{\mathsf{shrK}(A)}, X\}, \\
\ \ \text{Says } A\ B\ \{A, N_A\}] \\
\ \ \in traceSet
\end{array}
$$

が得られる．このように推論規則を繰り返し適用して得られる集合が *traceSet* である．これは Yahalom のプロトコルを実行したときに参加者が発生させるイベントの列である．

帰納的方法では，攻撃者の行動も推論規則として記述することにより，攻撃者のいる環境でのプロトコル実行を考えることができる．攻撃者の能力に対する仮定は推論規則の定義の中で与えることができる．

プロトコルを帰納的方法の推論規則として与えておけば，これによって定義されるトレース集合 *traceSet* を用いてセキュリティ要求を記述できる．具体的には，「*traceSet* に含まれるすべてのトレース *trace* についてセキュリティ $P(trace)$ が成り立つ」というトレースに関する論理式として記述される．帰納的方法ではこれを証明することによりセキュリティを検証する．

セキュリティ要求の記述例は 7 章（図 12）で与える．

帰納的方法では，トレースの集合の定義とトレースの集合の性質がそれぞれプロトコル仕様とセキュリティ要求の記述に相当しており両者の関係が分かりやすい．しかしながら，この方法はトレースというプロトコル仕様そのものとは直接関係がない概念を利用するため，プロトコル仕様の記述が直感的でなくなり誤りをおかす可能性がある．特に SET に関していえば，プロトコル仕様書では参加者の動作はメッセージの受信，処理，作成および送信といった単位で記述されており，トレースと論理式を用いた記述との間にはギャップがある．さらに，式 (1) の論理式では，メッセージ M が過去の多くのイベントに依存している場合に前提部分の Says $A\ B_i\ M_i \in trace$ や Says $C_i\ A\ L_i \in trace$ という論理式の数が多くなり，記述が繁雑になるという問題もある．本研究では，プロトコル仕様を簡潔に記述できるプロトコル仕様記述言語を定義し，この言語で表現されたプロトコルを論理式へ変換することでこれらの問題を解決する．

4. プロトコル仕様記述言語

本章では，プロトコルを各参加者の役割に分けて記述でき，パターンマッチングを用いて簡潔な表現ができるプロトコル仕様記述言語を定義する．この言語ではプロトコルを参加者の動作を表すプロセスの集まりとして記述する．

プロセスの間で交換されるメッセージ M は次の文法で与えられる．

$$
\begin{array}{rcl}
M & ::= & A \\
& | & Num \\
& | & Nonce \\
& | & K \\
& | & \mathsf{H}(M) \\
& | & \{M_1, M_2\} \\
& | & \{M\}_K
\end{array}
$$

A は参加者の名前，Num はクレジットカードの番号のような繰り返し用いられる番号，$Nonce$ はタイムスタンプや乱数のようなノンス，K は暗号化あるいは復号化の鍵を表す．$\mathsf{H}(M)$ は M のハッシュ，$\{M_1, M_2\}$ は M_1 と M_2 の連結，$\{M\}_K$ は M の鍵 K による暗号化を表す．

変数とパターンマッチングを用いるためにメッセージの文法に変数 X を追加して項 T の文法を以下のように定義する．

$$
\begin{array}{rcl}
T & ::= & A \\
& | & Num \\
& | & Nonce \\
& | & K \\
& | & \mathsf{H}(T) \\
& | & \{T_1, T_2\} \\
& | & \{T\}_K \\
& | & X
\end{array}
$$

次に，参加者の動作を表すプロセス P の言語を定義する．

$$
\begin{array}{rcl}
P & ::= & \mathsf{End} \\
& | & \mathsf{Send}\ A\ T;\ P \\
& | & \mathsf{Recv}\ T;\ P \\
& | & \mathsf{New}\ X;\ P \\
& | & \mathsf{Let}\ X = T;\ P
\end{array}
$$

X は変数である．End は停止しているプロセスを表す．Send $A\ T;\ P$ は参加者 A にメッセージ T を送信した後で P を実行するプロセスを表す．Recv $T;\ P$ はメッセージ T を受信した後で P を実行するプロセスを表す．New $X;\ P$ は新しいノンスを生成してそれを変数 X に束縛した後で P を実行するプロセスを表す．Let $X = T;\ P$ は変数 X に T を束縛した後で P を実行するプロセスを表す．なお，セミコロン ‘;’ と End は誤解のないときは省略する．Recv $T;\ P$ ではパターンマッチングが行われる．たとえばプロセス Let $X_1 = M;$ Recv $\{X_1, \{X_2\}_K\};\ P$ は変数 X_1 に M を束縛した後に $\{M, \{M'\}_K\}$ という形のメッセージのみを受けとり，変数 X_2 に M' を束縛し，プロセス P を実行する．また，この場合 $\{M'\}_K$ から M' を取り出すためには対応する復号化のための鍵 K^{-1} を知っている必要がある．この言語では Recv $\{X\}_K;\ P$ のように記述した場合にはこのプロセスは暗黙に K^{-1} を知っていると仮定する．

この言語は非常に単純であるが，プロトコルを各参加者の動作を表すプロセスに分けて記述し，プロセスの内部で変数とパターンマッチングを用いることで記述を簡潔に行えるようになっている．たとえば，メッセージ $\{M, \{M'\}_K\}$ を受信するプロセスを Recv $\{X_1, \{X_2\}_K\};\ P$ と記述するか Recv $\{X_1, X_2\};\ P$ と記述するかによって，このプロセスが $\{M'\}_K$ を復号化して M' を読み出すか復号化せず変数 X_2 に束縛して暗号文のまま扱うかをそれぞれ書きわけることができる．

例 2 Yahalom のプロトコルにおけるイニシエータ

のプロセス *Init* を**図 3** に示す．これを図 2 と比較すると，トレースやトレースの集合などのプロトコルそのものとは直接関係のない変数がなく，簡潔である．また，図 2 のように送信を表すイベント Says $A\ B\ \{A, N_A\}$ が 2 カ所に出現するなどの冗長さもない．

5. プロトコル仕様の論理式への変換方法

4 章で定義した言語で記述したプロトコル仕様を，帰納的方法の推論規則へ変換する関数 convert を定義する．convert はプロトコル仕様を構成するプロセスを引数の 1 つとしてとり，推論規則の集合を返す．convert は実際には初期状態に関する推論規則のみを生成し，その他の変換は convert1 を用いて行う．convert および convert1 の定義を**図 4** に示す．

プロセスの定義により，プロセスは Send T，Recv T，New X，Let $X = T$ を要素とし，End によって終端される列と見なせる．特に Send T に注意すれば，e_{ij} を Recv T，New X，Let $X = T$ のいずれかとして，プロセスは一般に次の形をしていることが分かる．

$$\begin{array}{l} e_{11}; \cdots; e_{1n_1}; \mathsf{Send}\ B_1\ T_1; \\ \qquad \vdots \\ e_{m1}; \cdots; e_{mn_m}; \mathsf{Send}\ B_m\ T_m; \\ e_{(m+1)1}; \cdots; e_{(m+1)n_{m+1}}; \mathsf{End} \end{array} \tag{2}$$

convert1 はこの $e_{i1}; \cdots; e_{in_i}; \mathsf{Send}\ B_i\ T_i$ という部分を順に次の形の推論規則に変換する．

$$\begin{array}{l} trace \in traceSet\ \wedge\ last \in trace\ \wedge\ prem_i \\ \rightarrow ([(\text{Says}\ A\ B_i\ T_i, env)] + trace) \in traceSet \end{array}$$

ここで *traceSet* はトレースの集合を表す変数である．ただし，ここでは 3 章のトレースの定義を拡張して，トレースは (Says $C\ D\ T, env$) あるいは (Notes $C\ T, env$) という 2 つ組のリストであるとする．Notes $C\ T$ は参加者 C の状態をトレースに記録するためのもので，参加者 C のプロセスが初期状態でどのようなパラメータを与えられているかを記録するために主に用いられる．*env* は項のリストで，参加者 C がイベント Says $C\ D\ T$ を発生させたときに C のプロセスが保持している変数の値が格納されていると考えてよい．*env* に含まれているパラメータを T_i から参照できるため，T_i が過去の複数のイベントで生成あるいは受信されたパラメータを参照している場合でも推論規則の前提部分に複数の Says イベントを加えておく必要がない．これにより 3 章の式 (1) に表れる Says $A\ B_i\ M_i \in trace$ や Says $A\ B_i\ M_i \in trace$

$$\begin{array}{l} Init(A, B) = \\ \quad \mathsf{New}\ N_A; \\ \quad \mathsf{Send}\ B\ \{A, N_A\}; \\ \quad \mathsf{Recv}\ \{\{B, K, N_A, N_B\}_{\mathsf{shrK}(A)}, X\}; \\ \quad \mathsf{Send}\ B\ \{X, \{N_B\}_K\}; \\ \quad \mathsf{End} \end{array}$$

図 3 Yahalom のプロトコルのイニシエータを定義するプロセス
Fig. 3 A process that defines initiator's action in Yahalom protocol.

$$\begin{array}{rl} \text{convert}(P, [X_1, \ldots, X_n]) = & \{trace \in traceSet \\ & \rightarrow ([(\text{Notes}\ \{X_1, \ldots, X_n\}, [X_1, \cdots, X_n])] + trace) \\ & \quad \in traceSet\} \\ & \cup \text{convert1}(P, (\text{Notes}\ \{X_1, \ldots, X_n\}, [X_1, \cdots, X_n]), \\ & \quad [X_1, \cdots, X_n], \mathit{True}) \end{array}$$

$$\begin{array}{rl} \text{convert1}(\mathsf{End}, last, env, prems) = & \{\} \\ \text{convert1}(\mathsf{New}\ X; P, last, env, prems) = & \text{convert1}(P, last, env + [X], \\ & \quad prems \wedge X \notin \text{used}\ trace) \\ \text{convert1}(\mathsf{Let}\ X = T; P, last, env, prems) = & \text{convert1}(P, last, env + [X], \\ & \quad prems \wedge X = T) \\ \text{convert1}(\mathsf{Recv}\ M; P, last, env, prems) = & \text{convert1}(P, last, env + [X_1, ..., X_n], \\ & \quad prems \wedge (\text{Says}\ B'\ A\ M, env')\ \in trace) \\ \text{convert1}(\mathsf{Send}\ B\ M; P, last, env, prems) = & \{trace \in traceSet \wedge last \in trace \wedge prems \\ & \quad \rightarrow ([(\text{Says}\ A\ B\ M, env)] + trace) \in traceSet\} \\ & \cup \text{convert1}(P, (\text{Says}\ A\ B\ M, env), env, \{\}) \end{array}$$

図 4 プロトコル仕様の論理式への変換
Fig. 4 Conversion of protocol specification into logical formulae.

$$\begin{aligned}
&\mathrm{convert}(\mathsf{New}\ N_A;\cdots,[A,B]) = \{\mathit{trace}\in \mathit{traceSet}\rightarrow([(\mathrm{Notes}\ \{A,B\},[A,B])]+\mathit{trace})\in \mathit{traceSet}\}\cup R_1\\
&R_1 = \mathrm{convert1}(\mathsf{New}\ N_A;\cdots,(\mathrm{Notes}\ \{A,B\},[A,B]),[A,B],\mathit{True})\\
&\quad = \mathrm{convert1}(\mathsf{Send}\ B\ \{A,N_A\};\cdots,(\mathrm{Notes}\ \{A,B\},[A,B]),[A,B,N_A],\mathit{True}\wedge N_A\notin \mathrm{used}\ \mathit{trace})\\
&\quad = \{\mathit{trace}\in \mathit{traceSet}\wedge(\mathrm{Notes}\ \{A,B\},[A,B])\in \mathit{trace}\wedge(\mathit{True}\wedge N_A\notin \mathrm{used}\ \mathit{trace})\\
&\qquad \rightarrow([(\mathrm{Says}\ A\ B\ \{A,N_A\},[A,B,N_A])]+\mathit{trace})\in \mathit{traceSet}\}\cup R_2\\
&R_2 = \mathrm{convert1}(\mathsf{Recv}\ \{\{B,K,N_A,N_B\}_{\mathsf{shrK}(A)},X\};\cdots,(\mathrm{Says}\ A\ B\ \{A,N_A\},[A,B,N_A]),[A,B,N_A],\mathit{True})\\
&\quad = \mathrm{convert1}(\mathsf{Send}\ B\ \{X,\{N_B\}_K\};\mathsf{End},(\mathrm{Says}\ A\ B\ \{A,N_A\},[A,B,N_A]),[A,B,N_A,N_B,K,X],\\
&\qquad\qquad \mathit{True}\wedge(\mathrm{Says}\ S\ A\ \{\{B,K,N_A,N_B\}_{\mathsf{shrK}(A)},X\},\mathit{env}')\in \mathit{trace})\\
&\quad = \{\mathit{trace}\in \mathit{traceSet}\wedge(\mathrm{Says}\ A\ B\ \{A,N_A\},[A,B,N_A])\in \mathit{trace}\\
&\qquad \wedge(\mathit{True}\wedge(\mathrm{Says}\ S\ A\ \{\{B,K,N_A,N_B\}_{\mathsf{shrK}(A)},X\})\in \mathit{trace})\\
&\qquad \rightarrow([(\mathrm{Says}\ A\ B\ \{X,\{N_B\}_K,[A,B,N_A,N_B,K,X])]+\mathit{trace})\in \mathit{traceSet}\}\cup R_3\\
&R_3 = \mathrm{convert1}(\mathsf{End},(\mathrm{Says}\ A\ B\ \{X,\{N_B\}_K,[A,B,N_A,N_B,K,X]),[A,B,N_A,N_B,K,X],\mathit{True})\\
&\quad = \{\}
\end{aligned}$$

図 5 Yahalom のプロトコルにおけるイニシエータの論理式への変換
Fig. 5 Translation of initiator's process in Yahalom protocol into logical formulae.

の一部が省略できる．*last* は (Notes $A\ T, \mathit{env}'$) あるいは (Says $A\ B'\ T', \mathit{env}'$) で，参加者 A が直前に行った動作に相当する．prem_i は論理和 ($\wedge$) によって結合された論理式で，式 (1) の変数 *cond* に相当する．prem_i の中身は $e_{i1},\ldots,e_{in_i}$ によって決まる．e_{ij} が Recv T，Let $X=T$，New X のときそれぞれ Says $B\ A\ T\in \mathit{trace}$，$X=T$，$X\notin$ used *trace* が prem_i に追加される．

例 3 Yahalom のプロトコルにおけるイニシエータのプロセス（図 3）を変換した例を**図 5** に示す．プロセス *Init* の Send までの部分をそれぞれ式 (2) の形の推論規則に変換していることが分かる．また，推論規則の前提部分にノンスの生成 New とメッセージの受信 Recv に対応する論理式がそれぞれ追加されていることが分かる．この推論規則の集合は，Notes イベントと *env* の部分が冗長であることを無視すれば，図 2 の推論規則の集合と等価であることも容易に分かる．

6. SET の支払いプロトコルの簡略化

SET は仕様書が 400 ページ以上もある巨大なプロトコルであるため，これをそのまま検証することは膨大な時間と労力を要する．本研究では，検証すべきセキュリティ要求に着目してプロトコルの簡略化を行うことにより，この問題に対処する．具体的には，プロトコル仕様から検証すべきセキュリティ要求とは無関係な部分を省略し，より単純なプロトコルを作る．

本研究では Meadows らの与えたセキュリティ要求の検証をめざしているため，プロトコル仕様に含まれるパラメータのうち，このセキュリティ要求に出現しないものを省略する．Meadows らのセキュリティ要求で参照されるパラメータは以下のとおりである．

(1) 顧客が生成するトランザクション ID
(2) 商店が生成するトランザクション ID
(3) 顧客の名前
(4) 注文内容のハッシュ値
(5) 顧客の *PAN*
(6) 顧客の *PANSecret*
(7) 購入金額
(8) 商店の名前
(9) 引落とし金額
(10) 承認された購入金額
(11) *PReq*（購入要求）における電子署名の有無

このうち (9) および (10) は本研究で扱うメッセージに含まれないため省略する．また，本研究では署名つきの *PReq* メッセージのみを取り扱うため，(11) は省略する．SET では購入要求のメッセージに電子署名を添付するか否かは利用者が選択することができるが，本研究では顧客は購入要求のメッセージに必ず電子署名を添付し，これを受信する商店およびゲートウェイは必ず署名を検証すると仮定する．この仮定により電子署名を添付しない顧客についての安全性は検証できないが，(11) の省略が検証結果に影響を与えることはない．

たとえば *AuthReq*（承認要求）のメッセージの簡略化は**図 6** のように行われる．この図はメッセージの構造を木状に表したものである．省略されないパラメータには灰色の印をつけてある．得られたプロトコルを前章で述べる言語で記述したものが**図 7**，**図 8**，**図 9** である．それぞれ顧客，商店，ゲートウェイの動作で

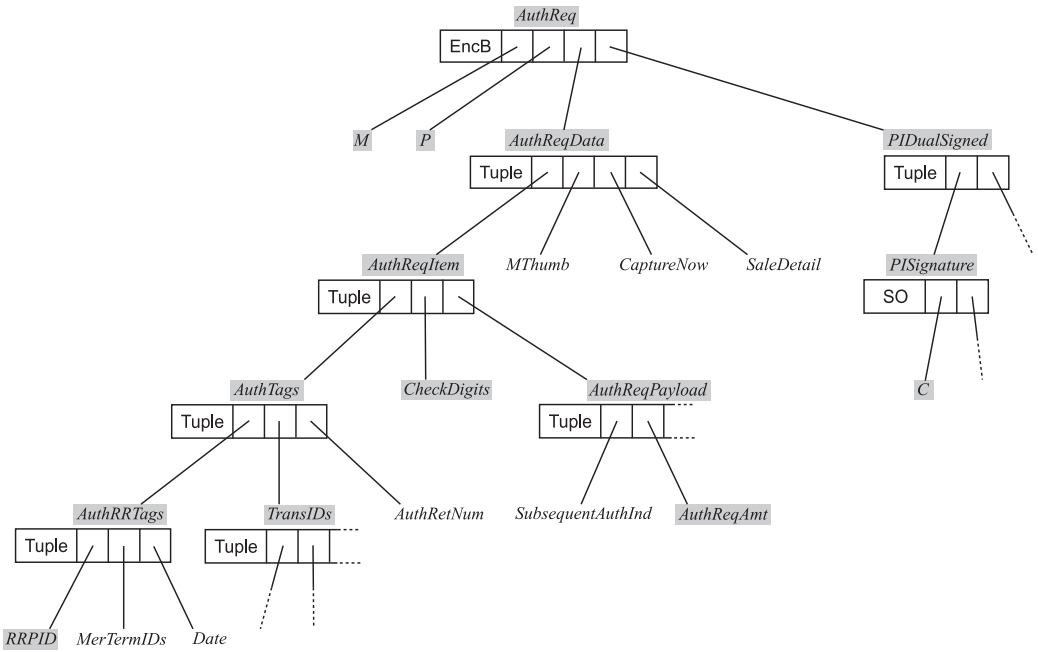

図 6 AuthReq（承認要求）メッセージの簡略化
Fig. 6 Simplification of AuthReq (Authentication Request) message.

$$
\begin{aligned}
&Customer(C,\ PAN,\ PANSecret,\ P,\ M,\ OD,\ ODSalt,\ PurchAmt) = \\
&\quad \mathsf{New}\ RRPID1,\ LID\_C,\ Chall\_C \\
&\quad ;;\ PInitReq \\
&\quad \mathsf{Send}\ M\ \{RRPID1,\ LID\_C,\ Chall\_C\} \\
&\quad ;;\ PInitRes \\
&\quad \mathsf{Recv}\ \{\{LID\_C,\ LID\_M,\ XID\},\ RRPID1,\ Chall\_C,\ Chall\_M\} \\
&\quad \mathsf{Let}\ TransIDs = \{LID\_C,\ LID\_M,\ XID\} \\
&\qquad PANData = \{PAN,\ PANSecret\} \\
&\qquad PIHead = \{TransIDs,\ \mathsf{H}(OD),\ PurchAmt\} \\
&\qquad OIData = \{TransIDs,\ RRPID2,\ Chall\_C,\ \mathsf{H}(OD),\ ODSalt\} \\
&\qquad PIData = \{PIHead,\ PANData\} \\
&\quad \mathsf{New}\ K\_PReq \\
&\quad ;;\ PReq \\
&\quad \mathsf{Send}\ M\ \{\ \{\ \mathsf{SO}(C,\ \{\mathsf{H}(PIData),\ \mathsf{H}(OIData)\}), \\
&\qquad\qquad \mathsf{EX}(P,\ \mathsf{L}(PIHead,\ OIData),\ PANData,\ K\_PReq)\}, \\
&\qquad \{OIData,\ \mathsf{H}(PIData)\}\} \\
&\quad ;;\ PRes \\
&\quad \mathsf{Recv}\ \mathsf{S}(M,\ \{TransIDs,\ RRPID2,\ Chall\_C\})
\end{aligned}
$$

図 7 顧客の動作
Fig. 7 Cusotmer's action.

ある．

ここではハッシュ H，暗号化 E などのメッセージ操作が用いられている．オリジナルの SET 仕様では，ハッシュアルゴリズムやビット列への符号化まで定義されているが，我々の言語ではメッセージは抽象的な集合の要素である．これらの操作は記述言語で**図 10** のように表現される．

$$
\begin{aligned}
&Merchant(M,\ OD,\ ODSalt,\ AuthReqAmt,\ C,\ P) = \\
&\quad ;;\ PInitReq \\
&\quad \mathsf{Recv}\ \{RRPID1,\ LID\_C,\ Chall\_C\} \\
&\quad \mathsf{New}\ LID\_M,\ XID,\ Chall\_M \\
&\quad \mathsf{Let}\ TransIDs = \{LID\_C,\ LID\_M,\ XID\} \\
&\quad ;;\ PInitRes \\
&\quad \mathsf{Send}\ C\ \mathsf{S}(M,\ \{TransIDs,\ RRPID1,\ Chall\_C,\ Chall\_M\} \\
&\quad \mathsf{Let}\ OIData = \{TransIDs,\ RRPID2,\ Chall\_C,\ \mathsf{H}(OD),\ ODSalt\} \\
&\quad ;;\ PReq \\
&\quad \mathsf{Recv}\ \{\ \{\mathsf{SO}(C,\ \{HPIData,\ \mathsf{H}(OIData)\}),\ PIBody\}, \\
&\qquad\quad \{OIData,\ HPIData\}\} \\
&\quad \mathsf{New}\ RRPID3,\ K1 \\
&\quad ;;\ AuthReq \\
&\quad \mathsf{Send}\ P\ \mathsf{EncB}(\ M,\ P,\ \{RRPID3,\ TransIDs,\ AuthReqAmt\}, \\
&\qquad\qquad \{\mathsf{SO}(C,\ \{HPIData,\ \mathsf{H}(OIData)\}),\ PIBody\},\ K1) \\
&\quad ;;\ AuthRes \\
&\quad \mathsf{Recv}\ \mathsf{Enc}(P,\ M,\ \{RRPID3,\ TransIDs,\ AuthReqAmt\},\ K2) \\
&\quad ;;\ PRes \\
&\quad \mathsf{Send}\ C\ \mathsf{S}(M,\ \{TransIDs,\ RRPID2,\ Chall\_C\})
\end{aligned}
$$

図 8 商店の動作
Fig. 8 Merchant's action.

$$
\begin{aligned}
&Gateway(P,\ C,\ M,\ PAN,\ PANSecret) = \\
&\quad ;;\ AuthReq \\
&\quad \mathsf{Recv}\ \mathsf{EncB}(M,\ P,\ \{RRPID3,\ TransIDs,\ AuthAmt\}, \\
&\qquad\quad \{\ \mathsf{SO}(C,\ \{\ \mathsf{H}(\{\{TransIDs,\ HOD,\ AuthAmt\},\ PANData\}), \\
&\qquad\qquad\quad HOIData\}), \\
&\qquad\qquad \mathsf{EX}(P,\ \{\{TransIDs,\ HOD,\ AuthAmt\},\ HOIData\}, \\
&\qquad\qquad\quad PANData,\ K\_PReq)\}, \\
&\qquad\quad K\_PReq) \\
&\quad \mathsf{Let}\ TransIDs = \{LID\_C,\ LID\_M,\ XID\}, \\
&\qquad PANData = \{PAN,\ PANSecret\} \\
&\quad \mathsf{New}\ K2 \\
&\quad ;;\ AuthRes \\
&\quad \mathsf{Send}\ M\ \mathsf{Enc}(P,\ M,\ \{RRPID3,\ TransIDs,\ AuthAmt\},\ K2)
\end{aligned}
$$

図 9 ゲートウェイの動作
Fig. 9 Gateway's action.

7. 秘匿性の検証

前章で得られた簡略化された SET の支払いプロトコル仕様を 5 章で提案した変換方法を用いて論理式に変換し，これを推論規則として秘匿性の論理式を証明した．証明は定理証明システム Isabelle 上で行った．たとえば図 7 の顧客の動作は**図 11** の 3 つの論理式に変換される．

ここで検証した秘匿性は，顧客のクレジット・カード番号（*PAN*）についてのものである．*PAN* の秘匿性の論理式を**図 12** に示す．この論理式は，攻撃者がネットワークを監視して得られるデータの集合 analz(spies *trace*) にカード番号 *PAN* が含まれるのは次の 3 つの場合のいずれかであるという意味である．

- 顧客自身が悪意を持っており（$C \in \mathrm{bad}$），攻撃者に直接 *PAN* を漏らしてしまう場合

$$
\begin{array}{lrcl}
\text{リンク：} & \mathsf{L}(M,N) & = & \{M, \mathsf{H}(N)\} \\
\text{署名：} & \mathsf{SO}(A,M) & = & \{\mathsf{H}(M)\}_{\mathsf{priK}(\mathsf{signer}(A))} \\
\text{暗号化：} & \mathsf{E}(A,M,K) & = & \{\{M\}_K, \{K\}_{\mathsf{pubK}(\mathsf{cryptor}A)} \\
\text{暗号化：} & \mathsf{EX}(A,M,Mp,K) & = & \{\{\mathsf{L}(M,Mp)\}_K, \{K,Mp\}_{\mathsf{pubK}(\mathsf{cryptor}A)}\} \\
\text{署名つきメッセージ：} & \mathsf{S}(A,M) & = & \{M, \mathsf{SO}(A,M)\} \\
\text{署名後暗号化：} & \mathsf{Enc}(S,R,T,K) & = & \mathsf{E}(R, \mathsf{S}(S,T), K) \\
\text{署名後暗号化：} & \mathsf{EncB}(S,R,T,B,K) & = & \{\mathsf{Enc}(S,R,\mathsf{L}(T,B),K), B\}
\end{array}
$$

図 10 メッセージ演算
Fig. 10 Message operations.

$$
\begin{array}{l}
\quad \mathit{trace} \in \mathit{traceSet} \\
\rightarrow (\text{Notes } C\ \{C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}\}, \\
\quad [C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}]) \in \mathit{trace}
\end{array}
$$

$$
\begin{array}{l}
\quad \mathit{trace} \in \mathit{traceSet} \\
\wedge\ (\text{Notes } C\ \{C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}\}, \\
\quad [C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}]) \in \mathit{trace} \\
\wedge\ \mathit{RRPID1}, \mathit{LID\_C}, \mathit{Chall\_C} \notin \text{used } \mathit{trace} \\
\rightarrow [(\text{Says } C\ M\ \{\mathit{RRPID1}, \mathit{LID\_C}, \mathit{Chall\_C}\}, \\
\quad [C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}, \\
\quad \mathit{RRPID1}, \mathit{LID\_C}, \mathit{Chall\_C}])] + \mathit{trace} \in \mathit{traceSet}
\end{array}
$$

$$
\begin{array}{l}
\quad \mathit{trace} \in \mathit{traceSet} \\
\wedge\ (\text{Says } C\ M\ \{\mathit{RRPID1}, \mathit{LID\_C}, \mathit{Chall\_C}\}, \\
\quad [C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}, \\
\quad \mathit{RRPID1}, \mathit{LID\_C}, \mathit{Chall\_C}]) \in \mathit{trace} \\
\wedge\ (\text{Says } M'\ C\ \{\{\mathit{LID\_C}, \mathit{LID\_M}, \mathit{XID}\}, \mathit{RRPID1}, \mathit{Chall\_C}, \mathit{Chall\_M}\}, \\
\quad \mathit{env}') \in \mathit{trace} \\
\wedge\ \mathit{TransIDs} = \{\mathit{LID\_C}, \mathit{LID\_M}, \mathit{XID}\} \\
\wedge\ \mathit{PANData} = \{\mathit{PAN}, \mathit{PANSecret}\} \\
\wedge\ \mathit{PIHead} = \{\mathit{TransIDs}, \mathsf{H}(\mathit{OD}), \mathit{PurchAmt}\} \\
\wedge\ \mathit{OIData} = \{\mathit{TransIDs}, \mathit{RRPID2}, \mathit{Chall\_C}, \mathsf{H}(\mathit{OD}), \mathit{ODSalt}\} \\
\wedge\ \mathit{PIData} = \{\mathit{PIHead}, \mathit{PANData}\} \\
\wedge\ \mathit{K\_PReq} \notin \text{used } \mathit{trace} \\
\rightarrow [(\text{Says } C\ M\ \{\{\mathsf{SO}(C, \{\mathsf{H}(\mathit{PIData}), \mathsf{H}(\mathit{OIData})\}), \\
\qquad\qquad \mathsf{EX}(P, \mathsf{L}(\mathit{PIHead}, \mathit{OIData}), \mathit{PANData}, \mathit{K\_PReq})\}, \\
\qquad\qquad \{\mathit{OIData}, \mathsf{H}(\mathit{PIData})\}\}, \\
\quad [C, \mathit{PAN}, \mathit{PANSecret}, P, M, \mathit{OD}, \mathit{ODSalt}, \mathit{PurchAmt}, \\
\quad \mathit{RRPID1}, \mathit{LID\_C}, \mathit{Chall\_C}, \\
\quad \mathit{LID\_M}, \mathit{XID}, \mathit{Chall\_M}, \\
\quad \mathit{TransIDs}, \mathit{PANData}, \mathit{PIHead}, \mathit{OIData}, \mathit{PIData}, \\
\quad \mathit{K\_PReq}])] + \mathit{trace} \in \mathit{traceSet}
\end{array}
$$

図 11 顧客の動作を定義する論理式
Fig. 11 Logical formulae that specify customer's action.

$$
\begin{aligned}
&trace \in traceSet \wedge PAN \in \mathrm{analz}(\mathrm{spies}\ trace) \rightarrow \\
&\quad \exists C \in \mathrm{bad}.\exists PANSecret\ env. \\
&\qquad (\mathrm{Notes}\ C\ \{PAN, PANSecret\}, env) \in trace \\
&\vee \exists \mathrm{cryptor}(P) \in \mathrm{bad}.\exists C\ M\ RRPID1\ LID\_C\ Chall\_C\ \cdots. \\
&\qquad (\mathrm{Says}\ C\ M\ \{RRPID1, LID\_C, Chall\_C\}, [\cdots]) \in trace \\
&\vee \exists P \in \mathrm{bad}.\exists C'\, PANSecret\ env. \\
&\qquad (\mathrm{Notes}\ P\ \{C', PAN, PANSecret\}, env) \in trace
\end{aligned}
$$

図 12 *PAN* の秘匿性の論理式
Fig. 12 Logical formula on secrecy of *PAN*.

- ゲートウェイの秘密鍵があらかじめ攻撃者に漏れており（$\mathrm{cryptor}(P) \in \mathrm{bad}$），顧客がそのゲートウェイを用いて注文をしようとする場合．
- ゲートウェイ自身が悪意を持っており（$P \in \mathrm{bad}$），攻撃者に直接秘密を漏らしてしまう場合．

これは，顧客かゲートウェイが悪意を持っていなければ商店が悪意を持っていても秘匿性が守られるということもできる．

秘匿性の証明自体は次のようなオーソドックスな順で行った．

(1) 参加者の秘密鍵の秘匿性の証明
(2) E，EX，Enc，EncB で用いられる対称鍵の秘匿性の証明
(3) *PAN* の秘匿性の証明

以上の証明のために，Isabelle とともに配布されている帰納的方法によるプロトコル検証の Isabelle スクリプト（合計約 1,600 行）を 5 章の *env* に相当する若干の変更を加えて再利用することができた．これに加え，SET の支払いプロトコル固有のスクリプト（合計約 800 行）を新たに追加し，検証を行った．これらの変更・追加したスクリプトは文献 14) として公開されている．このスクリプトに記述されている証明が正しいことを Isabelle99-1 でチェックするために，CPU が PentiumIII-S 1.26 GHz，メモリを 4 G バイト搭載した計算機を用いて約 3 分 23 秒を要した．なお，OS は RedHat Linux 7.2 で，使用した ML の処理系は Poly/ML 3.3X である．

8. まとめと今後の課題

本論文ではプロトコル仕様を簡潔に記述できる言語を定義し，この言語で記述されたプロトコル仕様を帰納的方法で用いられる論理式に変換する方法を定義した．これにより大きなプロトコルを簡潔に表現でき，また，セキュリティの検証には帰納的方法の証明スクリプトの多くを再利用できるため検証を効率良く進めることができる．さらに，この言語と変換方式を SET の支払いプロトコルに適用し，カード番号（*PAN*）の秘匿性を検証することができた．その際，現実的な時間内での検証を可能にするため，検証すべき性質とは関係ないと考えられる部分をプロトコル仕様から省略し，適度に簡略化されたプロトコルを得ることができた．

本研究には次のような課題が残されている．

- 検証，すなわちセキュリティの論理式の証明は，プロトコル仕様を変換した論理式を用いて行うため，検証を行う者は変換後の論理式を理解する必要が生じてしまう．検証をより見通し良く行うために，プロトコル記述言語のレベルでもセキュリティに関する推論を可能とし，帰納的方法のレベルでの推論との対応関係をつけるという方法が考えられる．
- SET の支払いプロトコルの簡略化を，検証対象のセキュリティ要求からは参照されない部分を省略することにより行った．しかし，この簡略化が本当にセキュリティに影響を与えないことは明らかではない．オリジナルのプロトコルの安全性を厳密に示すためには，セキュリティを保存する簡略化の理論が必要となる．

謝辞 日頃よりご指導いただいている NTT コミュニケーション科学基礎研究所の塚田恭章氏，白柳潔グループリーダ，平原達也人間情報研究部長に深く感謝いたします．本論文に対して有益なコメントを下さった査読者の皆様に深く感謝いたします．

参 考 文 献

1) Bella, G. and Paulson, L.C.: Kerberos Version IV: inductive analysis of the secrecy goals, *4th European Symposium on Research in Computer Security: ESORICS98* (1998).
2) Bella, G., Paulson, L.C. and Massacci, F.: The verification of an industrial payment protocol: the SET purchase phase, *Proc. 9th ACM Conference on Computer and communications se-*

curity, pp.12–20 (2002).
3) Bolignano, D.: Towards the Formal Verification of Electronic Commerce Protocols, *10th IEEE Computer Security Foundations Workshop*, pp.133–146 (1997).
4) Burrows, M., Abadi, M. and Needham, R.: A Logic of Authentication, *Trans. Computer Systems*, Vol.8, No.1, pp.18–36 (1990).
5) Dutertre, B. and Schneider, S.: Using a PVS Embedding of CSP to Verify Authentication Protocols, *TPHOLs*, Gunter, E.L. and Felty, A.(Eds.), Lecture Notes in Computer Science, Vol.1275, pp.121–136, Springer-Verlag (1997).
6) Formal Systems Ltd: FDR2 User Manual (1998).
7) Hoare, C.A.R.: *Communicating Sequential Processes*, Prentice Hall (1985).
8) Lu, S. and Smolka, S.: Model Checking SET Secure Electronic Transaction Protocol, *Proc. 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (*MASCOTS'99*), pp.358–365 (1999).
9) Meadows, C. and Syverson, P.: A Formal Specification of Requirements for Payment Transactions in the SET Protocol, *Financial Cryptography '98*, LNCS 1465, Springer-Verlag (1998).
10) Paulson, L.C.: *Isabelle: A Generic Theorem Prover*, LNCS 828, Springer-Verlag (1994).
11) Paulson, L.C.: The Inductive Approach to Verifying Cryptographic Protocols, *Journal of Computer Security*, Vol.6, No.1, pp.85–128 (1998).
12) Paulson, L.C.: Inductive analysis of the Internet protocol TLS, *ACM Trans. Computer and System Security*, Vol.2, No.3, pp.332–351 (1999).
13) Paulson, L.C.: Relations Between Secrets: Two Formal Analyses of the Yahalom Protocol, *Journal of Computer Security*, Vol.9, No.3, pp.197–216 (2001).
14) Sakurada, H.: Isabelle Proof Scripts for the Secrecy in the SET Payment Protocol (2003), Available from http://www.brl.ntt.co.jp/people/sakurada/set-secrecy/.
15) SET Secure Electronic Transaction LLC: SET Secure Electronic Transaction Book 1: Business Description (1997).
16) SET Secure Electronic Transaction LLC: SET Secure Electronic Transaction Book 2: Programmer's Guide (1997).
17) SET Secure Electronic Transaction LLC: SET Secure Electronic Transaction Book 3: Formal Protocol Definition (1997).

(平成 14 年 12 月 3 日受付)
(平成 15 年 6 月 3 日採録)

櫻田 英樹（正会員）

1974 年生．1999 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年日本電信電話（株）入社．現在 NTT コミュニケーション科学基礎研究所に所属．セキュリティプロトコルの検証技術に興味を持つ．