SPINに基づくセキュリティプロトコル検証システム

田中慎也† 佐藤文明†† 水野忠則††

本論文では，SPINによるセキュリティプロトコルの検証を行うシステムの提案と実装および評価について述べる．SPINに基づくセキュリティプロトコルの検証については，Jøsangがすでに考察を行っており，その実現性については問題があるという結論に達している．我々は，セキュリティプロトコルの運用にある制限を設定し，その範囲での動作について総当たりで検証するシステムの開発を行った．我々のシステムの特徴は，メッセージの暗号化，復号化などのセキュリティプロトコルに特有の処理を記述するためにPROMELAを拡張したSPROMELAを開発した点である．また，PROMELAから，SPROMELAへ変換し，さらに攻撃者プロセス，セキュリティ条件が破られたことを判定するモニタプロセスを自動的に生成するトランスレータを開発した．検証の際のシミュレーション条件設定やシミュレーション実行は，SPINの環境と統合され，GUIを使って操作できる環境となっている．この環境で，セキュリティホールがあることが分かっているOtway-Reesプロトコルの検証を行い，具体的なアタックの手順を導出することができた．

Security Protocol Verification System Based on SPIN

Shinya Tanaka,† Fumiaki Sato†† and Tadanori Mizuno††

This paper describes the system which verifies security protocols using SPIN. Jøsang has already considered verification of security protocols based on SPIN, and there is a problem about the implementability in conclusion. We restrict the operation of a security protocol within limits, and developed the system verifying the protocol. The feature of our system is extension to PROMELA (SPROMELA) for describing security protocols, such as encryption of a message, and decryption. And, we developed translater which converts SPROMELA to PROMELA, and which automatically generates the attacker process and the monitor process checking whether security conditions are broken. When we verify protocols, configuration and execution of the simulator are unified to the environment of SPIN using GUI. In this environment, we verified the Otway-Rees protocol that has wellknown security holes, and derived procedure of attack.

1. はじめに

インターネットを利用した電子商取引の普及にともない，通信のセキュリティを守ることによる不正防止の必要性が高まっている．通信のセキュリティを守るためには，個別のメッセージのセキュリティを守る暗号化技術は必須ではあるが，それだけでは十分ではない．メッセージのすり替えや，なりすましによる攻撃により，プロトコルの実行中にセキュリティが破られる可能性があるからである．そこで，このような攻撃に耐性のあるプロトコルの設計が必要とされているわけであるが，プロトコルのセキュリティホールを検証システムを利用せずに発見することは正確性を欠きやすく，多くの労力をともなう作業である．本論文では，悪意を持つ攻撃者の存在を前提として，セキュリティプロトコルの検証を行い，攻撃手順が存在するときには具体的な手順を導出するシステムをSPIN[1]を基に開発した．

従来のセキュリティプロトコルの検証法に関する研究としては，認証の論理によるプロトコルを解析するBAN論理[2]，これを拡張したSG論理[3]によるものがある．また，BAN論理による自動解析ツールの研究もある[4]．BAN論理に対しては，並行するセッションを利用した攻撃には対処できないとの批判[5)~8)]がある．メッセージの型チェックによるすり替えの可能性検出を提案しているSG論理もこの問題を解決はできていない．

一方，モデルチェックによる方法として，代数的なモデルに対する状態生成と解析による研究[9]，また我々

† 静岡大学大学院理工学研究科
Graduate School of Science and Technology, Shizuoka University
†† 静岡大学情報学部
Faculty of Information, Shizuoka University

の研究でも採用しているSPINによるセキュリティプロトコルの解析を検討した報告[10]などがある．モデルチェックによる方式では，並行セッションを利用した攻撃についての検証も可能であり，また検証の結果として実際の攻撃パターンが得られるなど非常に有用である．しかし，状態爆発問題という状態数が爆発的に増加し検証が困難になる問題が存在する．SPINによるセキュリティプロトコルの解析を検討した報告ではリプレイ攻撃などの様々な攻撃のパターンの検討や並行セッションを考慮した場合に必要となる状態数の考察を行っている．しかしながら，具体的な検証方式を提案して実際に検証を行ったわけではない．

本研究では，セキュリティプロトコルの運用にある制限を設定し，その範囲での動作について総当たりで検証するシステムの開発を行った．具体的には，並行に動作するセッション数を限定し，そのときに得られるすべてのメッセージを使ってあらゆる攻撃を試し，問題がないか検証するというシステムである．我々のシステムの特徴は，メッセージの暗号化，復号化などのセキュリティプロトコルに特有の処理を記述するためにPROMELAを拡張したSPROMELAを開発した点である．また，SPROMELAから，PROMELAへ変換し，さらに攻撃者プロセス，セキュリティ条件が破られたことを判定するモニタプロセスを自動的に生成するトランスレータを開発した．検証の際のシミュレーション条件設定やシミュレーション実行は，SPINの環境と統合され，GUIを使って操作できる環境となっている．この環境で，セキュリティホールがあることが分かっているOtway-Reesプロトコルの検証を行い，具体的な攻撃の手順を導出することができた．検証において具体的な攻撃の手順が示されるということは，その攻撃を回避するための対策を設計するために重要である．BAN論理などの論理型の検証方式と比較して，これは提案方式の大きな特長である．

以下，2章において本論文で使っている用語と前提条件について説明する．3章では，我々が開発したプロトコル検証システムについて述べる．4章では，Otway-Reesプロトコルに適用して，プロトコルを検証した結果について述べる．5章では，検証すべき状態数についての考察を行う．6章は本論文のまとめである．

2. 前提条件と用語

ここでは，本論文で使っている用語と前提条件について説明する．

2.1 用　　語

- セッション
  参加者間で行われる認証の単位．たとえば，ユーザAからユーザBへの認証のセッション，攻撃者からユーザAへのセッションというように用いる．
- ノンス
  必要に応じて生成される，過去に使用されたことのないデータ．ノンスにはタイムスタンプなどが使われ，メッセージやトランザクションの識別をするために使われる．
- メッセージ
  メッセージは構造を持っており，通常のデータ以外に，ノンス，暗号鍵，利用者識別情報，セッション情報などが必要に応じて暗号化されて各フィールドにセットされている．
- セキュリティ違反
  認証を受けるべき利用者と異なる利用者が認証されて，正常にプロトコルが終了したり，不正に利用者に鍵がわたってしまうなど，目的とする安全性が損なわれたときをセキュリティ違反とする．

2.2 前提条件

検証の対象となるプロトコルについての前提条件を以下に示す．

- 扱うプロトコルは1種類とする．
- プロトコルは2人の参加者の間での通信を定義する．
- プロトコルは途中で第三者に分岐しない．

セッションについての前提条件を以下に示す．

- セッションはプロトコルの一連の実行である．
- セッションは並行して複数実行できる．
- セッション間でメッセージの受け渡しはない．
- セッション中で使用されるメッセージは，セッションの開始時に与えられる．

参加者についての前提条件を以下に示す．

- 正規の参加者はプロトコルに従い不正をしない．
- プロトコルに必要な初期状態，想定されるメッセージの内容を知っている．
- セキュリティを守るため，可能な限り受け取ったメッセージが想定されている内容か否かをチェックする．

暗号についての前提条件を以下に示す．

- メッセージは暗号化鍵によって暗号化されたメッセージに変換され，復号化鍵によってその逆の変換がなされる．
- 暗号化されたメッセージは完全で，鍵なしでは解

読できない．
- 暗号化されたメッセージと同じものを偶然に作ることはできない．
- 暗号化されたメッセージの集合から，鍵を推定することはできない．

攻撃者とその攻撃についての前提条件を以下に示す．
- 対象とするセッション全体の開始時点では，自分の秘密鍵と，参加者の公開鍵以外の鍵は持っていない．
- 送受信されたメッセージはすべて傍受できる．
- 攻撃者はプロトコルを知っている．すなわち，各メッセージのフォーマットやプロトコルの状態，その遷移といった情報を知っている．
- 正規の参加者の送信するメッセージを，傍受，改変，または横取りすることができる．ただし，暗号化されたメッセージは鍵がなければ復号化や改変はできない．
- 正規の参加者になりすまして，偽造，あるいは過去に傍受したメッセージを送信することができる．知らない鍵で暗号化されたメッセージは，その内容を偽造することはできないが，暗号化されたメッセージそのものを使ってメッセージを作り送信することはできる．

3. SPINに基づくセキュリティプロトコル検証システム

3.1 SPIN

SPINは，分散システムの形式的な検証を行うためのソフトウェアパッケージであり，すでに広く配布されている．このソフトウェアは，ベル研究所の形式手法と検証研究グループによって1980年に開発されている．

SPINは，ソフトウェアの検証を効率的に実行することを目的としており，システム仕様を記述するためにハイレベル言語であるPROMELA（PROcess MEta LAnguage）を使用している．SPINは，オペレーティングシステム，通信プロトコル，交換システム，並列アルゴリズムなどの分散システム設計における論理的な誤りをトレースすることに利用されてきた．このツールによって，仕様の論理的な一貫性についての検査を行うことができる．たとえば，デッドロック，未定義の受信，フラグの不完全さ，レース状態，そして保証されないプロセス間の相対速度に関する仮定などについて検出することができる．

SPINは，またランダム，インタラクティブ，そしてガイド付きシミュレーションが可能で，徹底探索，および部分探索をサポートしている．したがって，このツールは問題のサイズにスムーズに対応し，かなり大規模な問題サイズも扱えるように設計されている．

3.2 セキュリティプロトコル

セキュリティプロトコルは，通常複雑なプロトコルではなく参加者も少ないため，メッセージの手順自体の検証は難しくない．しかし，そのプロトコルが本当に安全に通信できるのか，あるいは正しく利用者が認証されているのか，という意味の検証は容易ではない．

この検証に対しては，認証の論理を使ったBAN論理による論理検証が使われてきた．しかし，この検証方式では，安全でないことが分かった場合に具体的な攻撃手順までは分からない，また複数のセッションが並行に動作するときの検証ができないという欠点がある．一方，モデルチェックに基づくセキュリティプロトコルの検証も研究されている．しかし特に，SPINを使った検証に関する考察では，検証すべき状態数が非常に大きくなるため，検証できるプロトコルは限定されるという報告があった．

ここで，我々は同時に動作するセッション数を限定した運用を行う場合の検証を検討する．1人の参加者が，複数のセッションを張ることは，通常行われており，これを制限することは現実と乖離するシステムとなるが，そのセッション数を無限に可能とすることは現実の有限の資源では不可能である．つまり，並行動作できるセッション数を適切な数に制限することで，現実のシステム運用と同等の範囲で安全性を検証できるようにすることが可能となる．

3.3 PROMELAのセキュリティ拡張

PROMELAは本来分散システムの仕様を検証するために設計された言語であるため，セキュリティに関する情報を簡潔に表現するための構文は存在しない．そこで，我々は認証プロトコルを簡潔に表現するために必要ないくつかの構文を追加する．追加する構文は**表1**に示す．これらの構文は，実際はPROMELAの低レベルのデータ表現とそのアクセス関数によって実現されている．この構文により，暗号化などのデータ構造をユーザに隠蔽し，抽象的なシンボルを用いた検証をユーザに提供することができる．たとえば，SPROMELAでのメッセージの宣言は，PROMELAでのメッセージ本体と，暗号化のレベル（n重に暗号化されていることを許す），およびどの鍵で暗号化されているかの状態を表す変数群から構成される．また，SPROMELAでの暗号化処理は，PROMELAでは暗号化レベル変数のインクリメント，そしてそのレベルでの暗号化の鍵IDを変数にセットするなどの処理か

表1 SPROMELAの構文
Table 1 Syntax of SPROMELA.

追加した構文	構文の意味
$message_define(M, flag, ...)	プロトコルで使用するメッセージとそのメッセージを攻撃者が知っているかどうかの定義.
$session_define(...)	セッション名，メッセージ番号，送信者，受信者，攻撃者の挙動などセッションを定義する.
$security_check(...)	セキュリティが犯される条件を定義する.
$set(m, M, ...)	変数mの値をメッセージMにする.
$encrypt(m, K, ...)	変数mを鍵Kで暗号化する.
$decrypt(m, K, ...)	変数mを鍵Kで復号化する.
$nonce(m1, m2, ...)	m1とm2の値が等しいかどうかチェックする.
$send(A, B, N)	ユーザAからBにN番目のメッセージを送信する.
$receive(B, A, N)	ユーザBからAに送信されたN番目のメッセージを受信する.

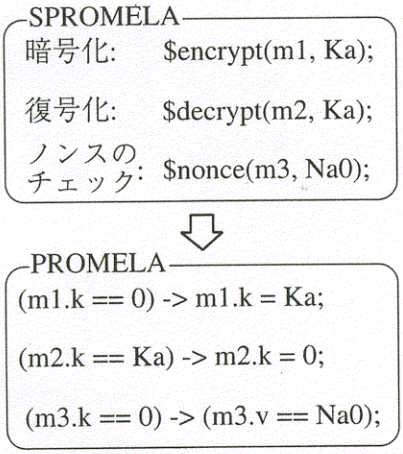


図1 SPROMELA記述とPROMELA記述の対応
Fig. 1 Mapping SPROMELA to PROMELA.

ら構成される．SPROMELA記述の例と変換されたPROMELA記述を**図1**に示す．

3.4 トランスレータ

SPROMELAは，SPROMELAトランスレータSPTranによってPROMELAに変換される．このトランスレータは，オブジェクト指向スクリプト言語Rubyを用いて実装した．トランスレータの持つ機能は，以下の3点である．

(1) SPROMELAからPROMELAに変換する．その際，SPROMELAに記述されている通信路を攻撃者を介した通信路に置き換える．
(2) 検証に必要な3つのプロセスを自動生成する．その3つのプロセスとは，最初にすべてのプロセスを立ち上げ，認証シーケンスを開始するinitプロセス，セキュリティ違反が発生していないかどうかを監視するモニタプロセス，さらに認証プロトコルに攻撃を仕掛け，認証プロトコルを破ろうとする攻撃者プロセスである．
(3) 攻撃者の挙動や送信するメッセージ，または通信路のキューのサイズなどの様々なパラメータを検証に与える．

3.5 通信路

SPROMELAで記述された通信路はトランスレータによって攻撃者を介した通信路に置き換えられる．これは以下の2つの理由で行われる．

- ネットワークを介した通信はすべて攻撃者が傍受，改変，横取りできることをシミュレートする．
- 攻撃者がセッションがどこまで進んでいるか把握する．これは，検証の停止のために必要である．

3.6 プロセス

トランスレータは3つのプロセスを自動生成する．以下にそれぞれ説明する．

initプロセスは，最初にすべてのプロセスを立ち上げ，認証シーケンスを開始する役割を持つ．このプロセスはSPINによる検証には不可欠である．

モニタプロセスは，$security_check構文によって指定されたセキュリティ違反の条件をつねに監視する．この監視には，PROMELAにおけるassert文が使われる．ここで指定する条件は次のようなものであり，この条件が両立した場合に検証を停止させる．

(1) 本来のプロトコルの動作では獲得しえないような鍵を攻撃者プロセスが獲得する．
(2) かつ，正しく鍵が配送できたと見なして，ユーザプロセスが認証シーケンスを終了する．

攻撃者プロセスは，**図2**に示すように1つの大きなdoループで構成されており，そのdoループ中はセッションの進行状況を表すローカル変数で制御されたブロックによって構成されている．検証において，ブロックの先頭の条件が成り立っているブロックがそれぞれランダムに選択される．また，セッションを示す変数がセッションの終わりを示していた場合，そのセッションはすでに終了しているため，攻撃者はそのセッションに干渉することができないと見なす．与えられたすべてのセッションが終了した場合にはdoループから抜け出るため，攻撃者プロセスは停止し，検証

```
proctype Attacker()
{
 /*ユーザA, B間のセッションを示す*/
 byte state_AB;
 do
 :: /*ユーザA→ユーザB メッセージ0 */
    (state_AB <= 0) -> state_AB = 1;

       /*メッセージの
        *  受信, 復号化, 記憶
        *  組み立て, 送信
        */

 :: /*ユーザA→ユーザB メッセージ1 */
    (state_AB <= 1) -> state_AB = 2;
       ⋮

 :: /* Give Up!! */
    (state_AB == 4) -> break;

 od
}
```

図2 攻撃者プロセスの構成
Fig. 2 Structure of attaacker process.

は終了する．

ブロックの内部には，主にメッセージを盗聴したり，メッセージを新たに生成し他プロセスに送信するなどの具体的な動作が記述される．この動作は，他プロセスから送信されたメッセージを盗聴する場合と攻撃者がメッセージを組み立て，他プロセスに送信する場合の2つに大きく分けることができる．

また，攻撃者は使用可能な鍵やメッセージを保持しておくローカル変数を持っている．使用可能なメッセージには暗号化されていない内容が分かっているメッセージと暗号化されていて内容が分からないメッセージの2種類ある．攻撃者自身の知りうる限りの鍵とは，攻撃者が初期状態で持っている鍵と，メッセージの盗聴によって入手した鍵を指す．同様に，攻撃者自身の知りうる限りのメッセージとは，攻撃者が初期状態で知っているメッセージと，メッセージの盗聴によって入手したメッセージを指す．

メッセージの盗聴において，盗んだメッセージを攻撃者自身の知りうる限りの鍵で復号化を試みる．復号化可能である場合には復号化した後のメッセージを，復号化が不可能である場合は暗号化されたメッセージそのものを，攻撃者プロセスのローカル変数に記憶する．これらのメッセージは今後の攻撃に使用される．

メッセージの送信において，攻撃者自身が知っているメッセージと，過去に盗聴し復号化が不可能なメッセージの両方を組み合わせ，さらに攻撃者が知っている鍵で暗号化し他のプロセスに送信することができる．メッセージは，可能な限りあらゆる組合せを網羅するように生成されるが，固定のメッセージを送信するように指定することもできる．

以上の動作の詳細は，SPROMELA の$session_define構文で定義することができる．$session_defineでは，通信路ごとに挙動を設定していくが，その通信路が攻撃者から見て受信に用いられるか送信に用いられるかによって設定できる内容が大きく異なる．受信の場合は，メッセージを受け取るかどうか，受け取ったメッセージのどの部分を記憶するのか，また，メッセージをどのように復号化するか，たとえば固定の鍵，または自分の知りうる限りの鍵で総当たりに復号化する，などを設定することができる．送信の場合は，メッセージを送信するかどうか，送信するメッセージの内容や，どのように暗号化するか，などを設定する．

3.7 セキュリティプロトコル検証システムの実行環境

本検証システムは，SPINの検証系と融合しており，SPROMELAで記述したセキュリティプロトコルを検証してセキュリティ違反を検出すると，その状態に至るまでのシーケンスをガイド付きシミュレーションによって表示することができる．このように，検出された手順を検討して，さらにメッセージ構造やプロトコルの手順をSPROMELAのレベルで変更し，再び検証することによってサイクリックにシステム設計を行うことができる．これは，SPINに付属のXSPINにセキュリティに関する表示の拡張などを行うことで実現している．

4. Otway-Reesプロトコルへの適用

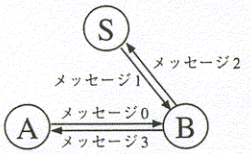
本検証方式をOtway-Reesプロトコル[2),11),12)]に適用し検証を行うことで評価する．Otway-Reesプロトコルには既知のセキュリティホールが存在し，攻撃をかけることが可能である．

4.1 Otway-Reesプロトコル

Otway-Reesプロトコルは，**図3**に示すように2人のユーザAとBと1つの認証サーバSによって行われるユーザ間の共有鍵を配送することを目的とした認証プロトコルである．

認証のシーケンスを簡単に説明する．まずAはBにメッセージを送信する．このメッセージにはAのノンスが含まれている．BはAから受け取ったメッセージにB自身のノンスなどを付け加えて認証サーバに転送する．サーバでは新しいセッション鍵Kabを生成し，A，B別々に分けてそれぞれKa，Kbで暗号化

(0) A → B : Na AB {Na}Ka {AB}Ka
(1) B → S : Na AB {Na}Ka {AB}Ka
Nb {Na}Kb {AB}Kb
(2) S → B : Na {Kab}Ka {Na}Ka {Kab}Kb {Nb}Kb
(3) B → A : Na {Kab}Ka {Na}Ka



A,B : ユーザ
S : サーバ
Na,Nb : ノンス
Ka,Kb : ユーザ・サーバ間の共有鍵
Kab : ユーザ間の共有鍵
AB : ユーザ間のセッションを示す情報
{...}Kx : ...を鍵Kxで暗号化

図3 Otway-Reesプロトコル
Fig. 3 Otway-Rees protocol.

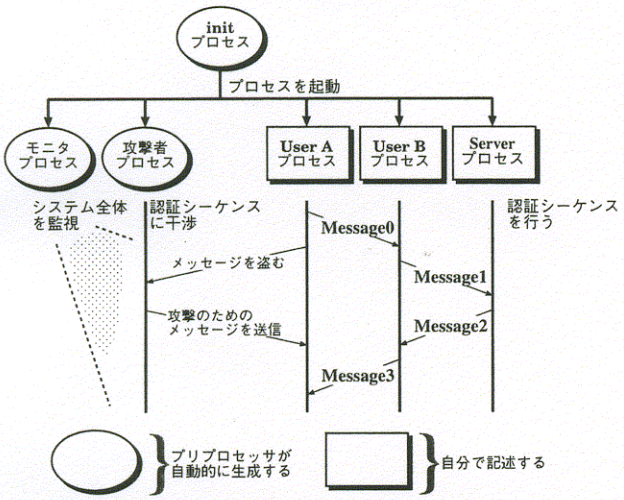


図4 プロセス間の関係
Fig. 4 Relation between processes.

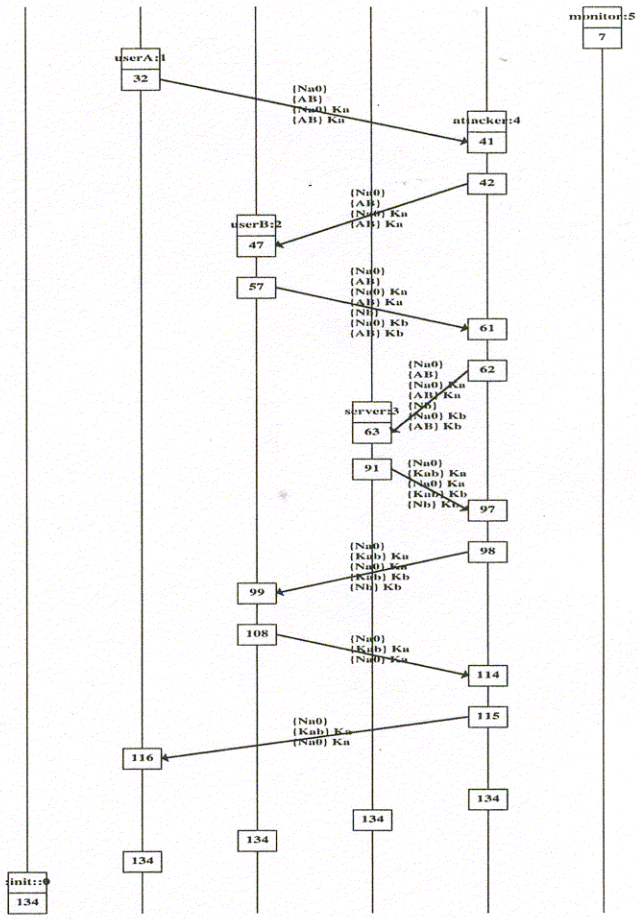


図5 ランダム・シミュレーションの結果
Fig. 5 Result of random simulation.

しBに送信する．次にBはサーバから返ってきたメッセージのB宛ての部分を復号化し，ノンスが以前送信したものと同じであることを確認する．そして残りのA宛ての部分をAに転送する．AはBから送られてきたメッセージを復号化しノンスを確認する．ここでノンスが正しいものだと確認できた場合には認証が成功したと見なし，KabをAB間のセッションで使用することができる．

本検証方式の検証モデルにおけるOtway-Reesプロトコルのプロセス間の関係を図4に示す．検証者は，トランスレータが自動生成するプロセス以外のOtway-Reesプロトコルに登場するプロセスをSPROMELAで記述すればよい．

4.2 攻撃者が攻撃しない動作の検証

まず，攻撃者プロセスが攻撃を行わないようにトランスレータに指示して，通常の状態のモデルを生成する．ランダム・シミュレーションを数回繰り返すとプロトコルが問題なく予想どおりに動作していることが確認できた．

変換後のPROMELAをSPINに読み込み，ランダム・シミュレーションを走らせた結果を図5に示す．縦軸は時間を表していて上から下に向かって時間が経過する．また四角の中の数字は探索の深さのステップ数である．なお，この図はXSPINのPostScriptダンプ機能を使って簡単に生成することができる．

次に，検証器を生成し検証を実行した．表明違反や到達不可能な文の検出など様々なフラグを付けて検証を実行したところ，特に問題となることは発見できなかった．したがって，セッションが並行に動作した場合においてもプロトコルは正常に動作することが確かめられた．

4.3 攻撃者が攻撃する場合の検証

攻撃者プロセスを生成するようにトランスレータに指示してPROMELAへの変換し，攻撃者が入った場合のプロトコルがうまく動作しているかどうか徹底探索を用いて検証する．本提案方式では，メッセージの組み合わせ方によって状態数が爆発的に増加し検証が困難になってしまう．そこで，試行錯誤しながら検証を行った結果，assert文の違反を発見することができた．発見したassert文の違反の場合の各プロセスの動作はファイルに保存され，後でガイド付きシミュレー

(0) A → C_B : Na0 AB {Na0}Ka {AB}Ka
(0)' C → A : Nc CA {Nc}Kc {CA}Kc
(1)' A → C_S : Nc CA {Nc}Kc {CA}Kc
Na1 {Nc}Ka {CA}Ka
(1)'' C_A → S : Nc CA {Nc}Kc {CA}Kc
Na0 {Nc}Ka {CA}Ka
(2)'' S → C_A : Nc {Kca}Kc {Nc}Kc {Kca}Ka {Na0}Ka
(3) C_B → A : Na0 {Kca}Ka {Na0}Ka

A: ユーザ
S: サーバ
Na0,Nc,Na1: ノンス
Cx: Xのふりをした攻撃者
Ka,Kc: ユーザ,攻撃者とサーバ間の共有鍵
Kca: ユーザ間の共有鍵
CA,AB: CとA, A とBのセッションを示す情報
{...}Kx: ...を鍵Kxで暗号化

図6 攻撃例
Fig. 6 Example of attack sequence.

ションでそのエラーに至るまでのシーケンスを再現することができる．発見した攻撃パターンを図6に示す．このパターンを見ると，ユーザAが過去に送信したメッセージを攻撃者が横取りし，そのメッセージを再利用したり，改竄し他プロセスに送信している様子が分かる．

5. 考　　察

本提案方式について，提案方式の特徴と状態数の2つの観点から考察する．

5.1 提案方式の特徴

本提案方式では，検証において発見したセキュリティホールが具体的な攻撃の手順として再現可能である．このような手順を示すことによって，どのようなセキュリティホールが存在しているのか理解することを助け，さらには，その攻撃を回避するための対策を設計するのに有効である．BAN論理などの論理型の検証方式と比較して，これは提案方式の大きな特長である．

さらに，提案方式では複数のセッションを扱うことが可能で，複数のセッション間でのメッセージの交換を利用した攻撃も発見可能である．これは，BAN論理に対する大きなメリットである．しかしながら，提案方式では，BAN論理のように認証の結果得られる信頼関係が正しいことを検証によって証明することはできない．

また，提案方式の重大な問題として状態爆発問題があげられる．これは，メッセージの組み合わせ方などの検証に与えるパラメータによって状態数が爆発的に増加し，計算機資源が不足することで検証が困難になってしまうことである．状態数については次節で詳しく述べる．

5.2 状　態　数

ここでは，我々が検証すべき状態数について考察する．我々の検証においては，攻撃者があらゆる組合せのメッセージを各プロセスに送信することから，そのメッセージ数が検証する状態数に最も大きな影響を持っている．したがって，そのメッセージ数について考察する．

並行に動作できるセッション数を k とする．また，対象プロトコルのステップ数を n とする．また，ステップ i におけるメッセージ種別を m_i とする．ステップ i における，1つのメッセージにおけるフィールドの組合せの数を c_i とすると，ステップ i におけるメッセージの組合せの数は，$M_i = \prod m_i c_i$ となる．全セッションにおけるメッセージの数は，$M = \prod_{i=1}^{n} M_i$ である．そして，最後に並行に動作する全セッションでのメッセージの送信種別は，$S = \prod kM$ となる．

SPINによる検証では，おおよそ 10^8 の状態数が徹底検証における限界の大きさであり，この程度のプロトコルが徹底検証可能なサイズということができる．このプロトコルを超えるサイズの検証を行うためには，並行動作できるセッション数を部分的，あるいは全セッションにわたって減らす，または安全性の高いメッセージやセッションについては検証対象から除くなどの，サイズを小さくする処理が必要と考えられる．

6. ま と め

本研究では，セキュリティプロトコルの運用にある制限を設定し，その範囲での動作について総当たりで検証するシステムの開発を行った．

我々のシステムの特徴は，メッセージの暗号化，復号化などのセキュリティプロトコルに特有の処理を記述するためにPROMELAを拡張したSPROMELAを開発した点である．また，SPROMELAから，PROMELAへ変換し，さらに攻撃者プロセス，セキュリティ条件が破られたことを判定するモニタプロセスを自動的に生成するトランスレータを開発した．検証の際のシミュレーション条件設定やシミュレーション実行は，SPINの環境と統合され，GUIを使って操作できる環境となっている．この環境で，セキュリティホールがあることが分かっているOtway-Reesプロトコルの検証を行い，具体的な攻撃の手順を導出することができた．

今後の課題として，提案方式ではメッセージやセッションの数によって状態数が爆発的に増加するという問題をかかえており，この問題の解決または緩和が重要である．さらに，今回は1つのプロトコルのみ検証評価したが，提案方式が任意のセキュリティプロトコルの検証に有効であることを示すためにも，様々なプロトコルの検証を行っていく必要がある．

参考文献

1) Holzmann, G.J.（著），水野忠則，東野輝夫，佐藤文明，太田　剛（共訳）：コンピュータプロトコルの設計法，カットシステム (1994).
2) Burrows, M., Abadi, M. and Needham, R.: logic of authentication, *ACM Trans. Comput. Syst.*, Vol.8, No.1, pp.18–36 (1990).
3) Gŕgens, S.: SG Logic – A Formal Analysis Technique for Authentication Protocols, *Secutiry Protocols 5th International Workshop*, LNCS, Vol.1361, pp.159–176, Springer (1997).
4) Kindred, D. and Wing, J.M.: Fast, Automatic Checking of Security Protocols, *Proc. 2nd USENIX Workshop on Electronic Commerce* (1996).
5) Bird, R., Copal, I., Herzburg, A., et al: Systematic Design of Two-Party Authentication Protocols, LNCS 576, *Advances in Cryptology CRYPTO '91*, pp.44–61, Springer (1991).
6) Syverson, P.: On Key Distribution Protocols for Repeated Authenticatiion, *SIGOPS Operating Systems Review*, Vol.27, No.4, pp.24–30 (1993).
7) Syversion, P.F. and van Oorschot, P.C.: On Unifying Some Cryptographic Protocol Logics, *IEEE Symposium on Research in Security and Privacy*, pp.14–28, IEEE (1994).
8) 根岸和義，米崎直樹：セキュリティプロトコルの一貫性およびその解決方策，情報処理学会研究報告，Vol.2000, No.30, pp.37–42 (2000).
9) Marrero, W., Clarke, E. and Jha, S.: Model Checking for Security Protocols, Research Report, CMU (1997).
10) Jøsang, A.: Security Protocol Verification using SPIN, *SPIN95, the 1st SPIN Workshop 95* (1995).
11) Otway, D. and Rees, O.: Efficient and Timely Mutual Authentication, *Operating Systems Review*, Vol.21, No.1, pp.8–10 (1987).
12) Paulson, L.C.: *The Indective Approach to Verifying Cryptographic Protocols*, Computer Laboratory, University of Cambridge (1998).

(平成12年 5 月19日受付)
(平成12年10月 6 日採録)

田中　慎也

昭和51年生．平成11年静岡大学工学部知能情報工学科卒業．同年静岡大学大学院理工学研究科計算機工学専攻入学，現在に至る．セキュリティ，形式的検証，通信ソフトウェアの研究に興味を持つ．

佐藤　文明（正会員）

昭和37年生．昭和61年東北大学大学院工学研究科電気及通信工学専攻博士前期課程修了．同年三菱電機（株）入社．通信ソフトウェアの研究開発に従事．平成7年1月より静岡大学工学部助教授，現在，静岡大学情報学部助教授．工学博士．通信ソフトウェア，形式言語，分散処理システムに関する研究に興味を持つ．電子情報通信学会，IEEE Computer Society 各会員．

水野　忠則（正会員）

昭和20年生．昭和43年名古屋工業大学経営工学科卒業．同年三菱電機（株）入社．平成5年静岡大学工学部情報知識工学科教授，現在，情報学部情報科学科教授．工学博士．情報ネットワーク，プロトコル工学，モバイルコンピューティングに関する研究に従事．著書としては「プロトコル言語」（カットシステム），「分散システム入門」（近代科学社）等がある．電子情報通信学会，IEEE，ACM 各会員．