定理証明器を用いたWide Mouth Frog Protocolの検証

Verification of Wide Mouth Frog Protocol Using Theorem Prover

安田 武史 † 高橋 和子 ††

Takeshi YASUDA Kazuko TAKAHASHI

† 関西学院大学理工学研究科

Graduate School of Science and Technology, Kwansei Gakuin University

†† 関西学院大学理工学部

School of Science and Technology, Kwansei Gakuin University

scbc1020@ksc.kwansei.ac.jp ktaka@kwansei.ac.jp

本研究では，定理証明器を用いて Wide Mouth Frog Protocol（以下 WMF）の検証を行った．定理証明器には Isabelle/HOL を使用し，Paulson による帰納的アプローチと Bella によるその拡張を用いて形式化を行い，安全性の性質として信頼性，規則性，単一性，信憑性，機密性の検証を試みた．WMF とは，共通鍵暗号方式を用いた，安全にセッション鍵を交換するのため暗号プロトコルで，その特徴は，タイムスタンプが十分に信頼できるものとして設計されている点である．そのため，WMF は時間感知型（time-sensitive）プロトコルとも呼ばれる．また，一般の暗号プロトコルはサーバがセッション鍵を作成するが，WMF はイニシエータがセッション鍵を作成するため，既存のアプローチをそのまま適用しただけでは安全性の検証は成功しない．そこで，単一性の検証では，タイムスタンプの性質をプロトコル全体を通してユニークであるように記述し，さらに機密性の検証では，必要なセッション鍵の取り扱いに関する前提条件を加えた．その結果，WMF の安全性の検証に成功した．

1 はじめに

ネットワークを介して安全に通信を行うためには，暗号プロトコルの使用が不可欠である．そのため，多種多様な暗号プロトコルが開発されている．一方で，それらが安全であるという保証を与えることは非常に困難であり，重要な研究課題となっている．以前はテストデータによるシミュレーションや経験的知識に頼り，安全性の検証を行っていた．しかし，これらの手法では，厳密な安全性を保証することができない．また，インターネットのような，複雑で大規模な公共ネットワークが登場したため，そこでの使用に耐えうる，より複雑で難解な暗号プロトコルが開発された．こうした事情により，これまでの検証手法では手に負えなくなったため，新たな検証の手法が必要となった．

そこで，暗号プロトコルの新たな検証の手法として，注目を集めているのが数理的技法である．数理的技法は，暗号プロトコルに数学的あるいは論理的な保証を与えることによって安全性を検証する手法である．つまり，暗号プロトコルの安全性について，以前よりも厳密な保証を与えることができる．

ただし，数理的技法を用いた暗号プロトコルの検証手法は，まだ十分に確立していない．そこで問題となるのが，暗号プロトコルのモデル化と安全性の定義，およびその定式化である．本研究の意図は，それらの方法を提案することである．

数理的技法を用いた，暗号プロトコルを検証するためのアプローチとしては，モデル検査（Model Checking）と定理証明（Theorem Proving）がある．

モデル検査は，検証したい暗号プロトコルを状態遷移の形で表現することでモデル化を行う．さらに，LTL（Linear Temporal Logic) や CTL（Computation Tree Logic）などの時相論理式を用いてそのプロトコルが満たすべき安全性の性質を定式化する．そして，すべての遷移を辿りながら，どの経路においてもその時相論理式の条件を満たしていることを検証する．もし，その条件を満たしていない経路を発見した場合は，反例となる経路を出力することもできる．ただし，可能性のある経路をすべて探索するため，検証に時間がかかるという欠点がある．代表的なモデル検査ツールとしては，SPIN[5]，FDR[4]，SMV[11] などがある．

モデル検査を使用した暗号プロトコルの検証は，

Lowe の仕事が有名である．Lowe はモデル検査ツール FDR とプロセス代数 CSP を使用して，公開鍵暗号方式の Needham Shroeder プロトコルの検証に成功した [8].

定理証明は，ある公理系とその推論規則を使用し，ある定式化した性質を満たしているかどうか証明することで検証を行う方法である．そのため，検証の速度はモデル検査と比べ非常に速い．ただし，目的の性質を証明する際には，あらかじめ，いくつかの性質を証明しておかなければならない場合がある．また，証明が成功しない場合の解析は，非常に困難である．代表的な定理証明ツールとしは，Isabelle[10], PVS[12], ACL2[6] などがある．

定理証明を使用した暗号プロトコルの検証は，Paulson による Isabelle/HOL[10] を用いた帰納的アプローチが有名であり，公開鍵暗号方式の Needham Shroeder プロトコルの検証に成功している [13]. HOL は，関数プログラミングにロジックを組み合わせた高階論理である．帰納的アプローチを用いた暗号プロトコルの検証は過去にも例があったが，すべてのモデル化を完全に帰納的な表現によって記述したのは彼が初めてであった．ただし，このアプローチは，ごく少数の古典的暗号プロトコルにしか適用できないものであった．それを Kerberos などの中級暗号プロトコルに適用できるよう拡張したのが Bella である．Bella はタイムスタンプ，スマートカード，受信イベントを拡張し，今まで困難とされてきた KerberosIV, Shoup-Rubin の検証に成功した [3].

本研究では，定理証明器 Isabelle/HOL を用い，Wide Mouth Frog Protocol（以下 WMF）の検証を行う．形式化では，Paulson の帰納的アプローチと Bella のその拡張に基づき，モデル化を行った．検証では，信頼性，規則性，単一性，信憑性，機密性の検証を行う．

通常の暗号プロトコルでは，サーバが鍵を生成するのに対して，WMF ではイニシエータと呼ばれる最初の通信者が鍵を作成する．そのため，暗号プロトコルとして脆弱であることが指摘されている [14]. 本研究の目的は，脆弱性が引き起こす具体的な問題点を指摘し，その解決をはかることである．

まず，タイムスタンプを使用するプロトコルである Kerberos の定式化 [3] にしたがって WMF の定式化を行って安全性の検証を試みた．失敗した検証結果を解析することにより，以下の 2 つの問題点が判明した．

WMF では，タイムスタンプが重要な役割を果たすが，記述方法によっては単一性が保てない．また，機密性を保証するためには，イニシエータがセッション鍵を作成する際の，その鍵の選び方にある．本論文では，この 2 つの問題点を発見した過程とその解決策を示す．

本論文の構成は以下の通りである．まず，2 章では暗号プロトコルをモデル化する方法について述べ WMF のモデル化を行う．3 章では暗号プロトコルの安全性を検証する方法について述べ WMF の検証を行う．4 章では検証で得られた結果について議論し，5 章で結論を述べる．

2 WideMouthFrog プロトコルのモデル化

2.1 モデル化の方法

WMF のモデル化は Paulson の帰納的アプローチに基づいて行った．そのモデル化の概観を述べる.（詳細については [3][10][13] を参照のこと）

プロトコルの実行単位は，プロトコルステップ（*protocolstep*）であり，ここではイベント（*event*）と呼ぶ．このモデル化では，プロトコルの動作をトレースによって監視する．トレース（*trace*）は，実行されたプロトコルステップを順にリスト形式で記録したもの（*event list*）である．トレースは，すべてのエージェントが参照できる．

暗号プロトコルでやり取りを行うメッセージは，以下のように帰納的に定義されている．

Agent ：暗号プロトコルを実行するエージェントである．エージェントの定義は，プロトコルを実行する主体であり,「ユーザ」「機械」，あるいは「プロセス」である [1]. 自然数（*nat*）への全単射として定義されている．

Number ：自然数で定義された,“ 偽造可能な ”任意の数字である．これは，主にタイムスタンプとして用いられる．タイムスタンプとはメッセージを作成した時刻であり，プロトコルステップが意図した順番で実行されたことを確認するために使用される．

Nonce ：自然数で定義された“ 偽造できない ”任意の数字である．これは主にナンスとして用いられる．ナンスとは十分にランダムな数字であり，その数字は世界に 1 つか存在しないと仮定する

ことができる．ナンスはメッセージを推測して偽造されることを防ぐために使用される．

Key ：自然数で定義された，メッセージを暗号化（復号化）する鍵である．

MPair ：メッセージとメッセージのペアもメッセージである．メッセージを組み合わせるときは，それらを {| と |} でくくる．

Crypt ：暗号化されたメッセージもメッセージである．暗号化されたメッセージは，それらを Crypt (shrK A){| と |} でくくる．ただし，これは共有鍵 A で暗号化した例である．

これらのメッセージは，自然数（*Number*，*Nonce*，*Key*）あるいは自然数への写像（*Agent*）として定義されており，その自然数も帰納的に定義されている．そのため，メッセージの種類は無限に存在する．

エージェントとしては，信頼できるエージェント *Server*，プロトコルを利用する *Friend*，プロトコルを攻撃する *Spy* の 3 種類が用意されている．これらの違いは，最初から持っている知識にある．*Server* は自身の秘密鍵とすべてのエージェントの公開鍵，共通鍵を知っている．*Friend Spy* は自身の共通鍵，秘密鍵とすべてのエージェントの公開鍵を知っている．もし，*Server Friend* の共通鍵，秘密鍵がネットワーク上を流れた場合，そのエージェントは危険であるとみなし *bad* という集合に入れられる．そして，そのエージェントの共通鍵，秘密鍵は *Spy* の知識に加えられる．

トレースを解析するための関数は以下のように定義されている．これらの関数はすべて *msg set* から *msg set* への写像である．

analz ：トレース（厳密には *msg set*）からメッセージを取り出すための関数である．ただし，暗号化されているメッセージは，その複合化鍵を知らない場合，中身を取り出すことができない．

synth ：トレース（厳密には *msg set*）からメッセージを偽造するための関数である．偽造に使用できるメッセージは，トレースに含まれているメッセージとナンス以外の任意のメッセージを組み合わせてできるメッセージである．

parts ：トレース（厳密には *msg set*）からメッセージを取り出すための関数である．analz は鍵がないと暗号化されたメッセージの中身まで取り出せなかったが，*parts* はたとえメッセージが暗号化されていても中身を取り出すことができる．

event は以下のように記述する．

Says A B message ：エージェント *A* からエージェント *B* へメッセージ *message* を送信するイベントである．

Notes A message ：エージェント *A* の知識に *message* を加えるイベントである．

これらの関数を用いて論理式を記述する．例えば，前提 P，Q から結論 R が演繹される（$P,\ Q \vdash R$）ことを，以下のように記述する．

$$[\![\ P\ ;\ Q\ ]\!] \Longrightarrow R$$

ここで，P，Q，R は，HOL で許される論理式である．前提は [| と |] でくくり，複数の式がある場合は ; で区切る．結論は，$\Longrightarrow$ の後に記述する

2.2 WMF のモデル化

下記に示すように，WMF のモデル化を行った．

Nil : $[] \in$ *wm-frog*

Fake : $[\![\ evsf \in wm\text{-}frog;\ X \in synth\ (analz\ (spies\ evsf))\ ]\!] \Longrightarrow Says\ Spy\ B\ X\ \#\ evsf \in wm\text{-}frog$

WM1 : $[\![\ evs1 \in wm\text{-}frog;\ Key\ KAB \notin used\ evs1;\ KAB \in symKeys;\ Nonce\ NA \notin used\ evs1\ ]\!] \Longrightarrow Says\ A\ Server\ \{|\ Agent\ A,\ Crypt\ (shrK\ A)\ \{|\ Nonce\ NA,\ Number\ (CT\ evs1),\ Agent\ B,\ Key\ KAB\ |\}\ |\}\ \#\ evs1\ \in\ wm\text{-}frog$

WM2 : $[\![\ evs2 \in wm\text{-}frog;\ Says\ A'\ Server\ \{|\ Agent\ A,\ Crypt\ (shrK\ A)\ \{|\ Nonce\ NA,\ Number\ TA,\ Agent\ B,\ Key\ KAB\ |\}\ |\} \in set\ evs2\ ;\ \neg Expired\ TA\ evs2;\ Nonce\ NB \notin used\ evs2\ ]\!] \Longrightarrow Says\ Server\ B\ (Crypt\ (shrK\ B)\ \{|\ Nonce\ NB,\ Number\ (CT\ evs2),\ Agent\ A,\ Key\ KAB\ |\})\ \#\ evs2\ \in\ wm\text{-}frog$

WM3 : $[\![\ evs3 \in wm\text{-}frog;\ Says\ S\ B\ (Crypt\ (shrK\ B)\ \{|\ Nonce\ NB,\ Number\ TB,\ Agent\ A,\ Key\ KAB\ |\}) \in set\ evs3\ ;\ \neg Expired\ TB\ evs3]\!] \Longrightarrow Notes\ B\ (Crypt\ (shrK\ B)\ (Key\ KAB))\ \#\ evs3\ \in\ wm\text{-}frog$

Oops : $[\![\ evso \in wm\text{-}frog;\ Says\ S\ B\ (Crypt\ (shrK\ B)\ \{|\ Nonce\ NB,\ Number\ TB,\ Agent\ A,\ Key\ KAB\ |\}) \in set\ evso;$

$$\begin{array}{l} \llbracket\ Expired\ TB\ evso\ \rrbracket \\ \Longrightarrow Notes\ Spy\ (Key\ KAB)\ \#\ evso \in wm\text{-}frog \end{array}$$

WMF は帰納的にモデル化されており，Nil がベースケース，$Fake$，$WM1$，$WM2$，$WM3$，$Oops$ がインダクションステップである．

タイムスタンプの値は，トレースの長さをとる．例えば，$Number\ (CTevs2)$ と記述した場合，CT はイベントリストの長さ（0 以上の整数値）の関数であり，$evs2$ はこれまでに発生したイベントリストである．タイムスタンプが期限切れである判定は $Expired$ 関数で判定する．$Expired$ の第 1 引数はタイムスタンプの時刻，第 2 引数は現在の時刻である．その差が 1 より大きければ期限切れと判定される．逆に，その差が 1 以下ならば期限内と判定される．

Nil はイベントが何も起こっていない状態，すなわちトレースが空リストである．

$WM1$ は WMF の最初のイベントである．任意のエージェント（A）から仲介サーバ（$Server$）へ，A と $Server$ 間で共有している鍵（$shrK\ A$）で暗号化した，タイムスタンプ（$Nonce\ NA\ Number\ (CT\ evs1)$），レスポンダの名前（$Agent\ \ B$），セッション鍵（$Key\ KAB$）を送信する．ここで，$A$ はプロトコルのイニシエータである．ただし，前提条件として鍵（Key）とタイムスタンプ（$Nonce$）は，過去のトレースに存在しないユニークな値が選ばれる．

$WM2$ は WMF の 2 番目のイベントである．仲介サーバ（$Server$）から任意のエージェント（B）へ，B と $Server$ 間で共有している鍵（$shrK\ \ B$）で暗号化した，タイムスタンプ（$Nonce\ NB\ Number\ (CTevs2)$），レスポンダの名前（$Agent\ B$），セッション鍵（$Key\ KAB$）を送信する．ただし，前提条件として，$WM1$ によって未知のエージェント A' からのメッセージを受信しており，タイムスタンプが期限内のときのみ，$WM2$ が実行される．

$WM3$ は WMF の補足イベントである．$WM2$ によって未知のエージェント（S）からレスポンダ（B）へ送信されたメッセージのタイムスタンプ（$Number\ TB$）が期限内であれば，レスポンダ（B）の知識にセッション鍵（$Key\ KAB$）が追加される．

$Oops$ は WMF の例外イベントである．もし $WM2$ から送信されたメッセージのタイムスタンプ（$Number\ TB$）が期限切れであった場合，そのメッセージは信頼できないので，セッション鍵（KAB）は Spy に知られているとする．

3 WideMouthFrog プロトコルの検証

3.1 安全性の保証

安全性（Security）とは，暗号プロトコルが正常に動作するために必要な，様々な保証の集合体である [2]．安全性を示すための保証は無数にあり，それらをすべて網羅することは不可能である．検証する保証の選択は，暗号プロトコルの目的や実行環境を考慮して決定しなければならない．本研究では，Bella による信頼性（Reliability），規則性（Regularity），単一性（Unicity）[3]，Lowe による信憑性（Authenticity）[9]，ISO Security Architecture Framework[ISO 7498-2:1989] による機密性（Confidenciality）の保証を選択し検証した．

3.2 信頼性（Reliability）

信頼性とは，モデル化された暗号プロトコルが，実際に動作するプロトコルに期待されるべき振る舞いをする保証である．たとえば，交換するための鍵がメッセージの暗号化鍵として使用されていないかなどである．この定式化を以下のように行い検証した．

$$\begin{array}{l} \llbracket\ Says\ A\ Server\ \{\!|Agent\ A,\ Crypt\ K'\ \{\!|Number\ Ts,\\ Agent\ B,\ Key\ KAB|\!\}\ |\!\}\ \in set\ evs;\ A \notin bad;\\ evs \in wm\text{-}frog\ \rrbracket \\ \Longrightarrow K' = shrK\ A \wedge KAB \notin range\ shrK \end{array}$$

$$\begin{array}{l} \llbracket\ Says\ Server\ B\ \ (Crypt\ K'\ \{\!|Nonce\ NB,\ Number\ Ts,\\ Agent\ A,\ Key\ KAB|\!\})\ \in set\ evs;\ B \notin bad;\\ evs \in wm\text{-}frog\ \rrbracket \\ \Longrightarrow K' = shrK\ B \wedge KAB \notin range\ shrK \end{array}$$

3.3 規則性（Regularity）

規則性とは，作成したモデルに対して，「ある特定のメッセージがネットワーク上に現れた場合，そのとき満たされるべき性質が成り立つ」ことの保証である．WMF の場合，暗号化鍵がネットワーク上を流れた場合，その鍵を所有するエージェントは bad の集合に入っているという性質である．この定式化を以下のように行い検証した．

$$\begin{array}{l} evs \in wm\text{-}frog \\ \Longrightarrow (Key\ (shrK\ B) \in parts\ (spies\ evs)) = (B \in bad) \end{array}$$

$$\begin{array}{l} evs \in wm\text{-}frog \\ \Longrightarrow (Key\ (shrK\ B) \in analz\ (spies\ evs)) = (B \in bad) \end{array}$$

3.4 信憑性（Authenticity）

信憑性とは，「メッセージの送信者は，受信者が意図した送信者である」という保証である．この定式

化を以下のように行い検証した．

⟦ *Says A′ Server* {| *Agent A*, *Crypt* (*shrK A*) {| *Nonce NA*, *Number TA*, *Agent B*, *Key KAB* |} |} ∈ *set evs*; *A′*∉*bad*; *evs* ∈ *wm-frog* ⟧ ⟹ *Says A Server* {| *Agent A*, *Crypt* (*shrK A*) {| *Nonce NA*, *Number TA*, *Agent B*, *Key KAB* |} |} ∈ *set evs*

⟦ *Says S B* (*Crypt* (*shrK B*) {| *Nonce NB*, *Number TB*, *Agent A*, *Key KAB*|}) ∈ *set evs*; *B* ∉ *bad*; *S*∉*bad*;*evs* ∈ *wm-frog* ⟧ ⟹ *Says Server B* (*Crypt* (*shrK B*) {| *Nonce NB*, *Number TB*, *Agent A*, *Key KAB*|}) ∈ *set evs*

3.5 単一性（Unicity）

単一性とは,「本来 1 つのみ存在すべきでないものは 2 つ存在しない」という保証である．WMF の場合，それはセッション鍵である．この定式化を以下のように行い検証した．

⟦ *Says A Server* {|*Agent A*, *Crypt* (*shrK A*) {|*Nonce NA*, *Number Ts*, *Agent B*, *Key KAB*|}|} ∈ *set evs*; *Says A′ Server* {|*Agent A′*, *Crypt* (*shrK A′*) {|*Nonce NA′*, *Number Ts′*, *Agent B′*, *Key KAB*|}|} ∈ *set evs*; *A* ∉*bad*; *A′*∉*bad*; *evs* ∈ *wm-frog* ⟧ ⟹ *A*=*A′* ∧ *NA*=*NA′* ∧ *Ts*=*Ts′* ∧ *B*=*B′*

⟦ *Says Server B* (*Crypt* (*shrK B*) {|*Nonce NB*, *Number TB*, *Agent A*, *Key KAB*|}) ∈ *set evs*; *Says Server B′* (*Crypt* (*shrK B′*) {|*Nonce NB′*, *Number TB′*, *Agent A′*, *Key KAB*|}) ∈ *set evs*; *B* ∉ *bad*; *B′* ∉ *bad*; *NB*=*NB′*; *TB*=*TB′*; *evs* ∈ *wm-frog* ⟧ ⟹ *B*=*B′* ∧ *A*=*A′*

3.6 機密性（Confidenciality）

機密性とは「機密であるべき情報が第三者に知られていない」という保証である．WMF の場合，機密であるべき情報は，共有するセッション鍵である．この定式化を以下のように行い検証した．

⟦ *Says A Server* {|*Agent A*, *Crypt* (*shrK A*) {|*Nonce NA*, *Number TA*, *Agent B*, *Key KAB*|}|} ∈ *set evs*; ¬*Expired TA evs*; *A*∉*bad*; *B* ∉ *bad*; *evs* ∈ *wm-frog* ⟧ ⟹ *Key KAB* ∉ *analz* (*spies evs*)

⟦ *Says Server B* (*Crypt* (*shrK B*) {|*Nonce NB*, *Number TB*, *Agent A*, *Key KAB*|}) ∈ *set evs*; ¬*Expired TB evs*; *A*∉*bad*; *B* ∉ *bad*; *evs* ∈ *wm-frog* ⟧ ⟹ *Key KAB* ∉ *analz* (*spies evs*)

4 WMF の検証の問題点とその解決策

検証を行った結果，2 つの問題点が判明した．

1 つ目は単一性の検証における，タイムスタンプの取り扱いである．WMF の安全性を保証するために，最も重要な役割を果たしているのがタイムスタンプである．Bella は Kerberos を定式化する際，タイムスタンプにトレースの長さを用いた．たとえば，最初に Nil が実行されタイムスタンプは 0，次に $WM1$ が実行されたときはタイムスタンプが 1，その次に $WM2$ が実行されたときはタイムスタンプが 2，... となる．この Bella のタイムスタンプは，メッセージの連続性のみを保証するものであった．

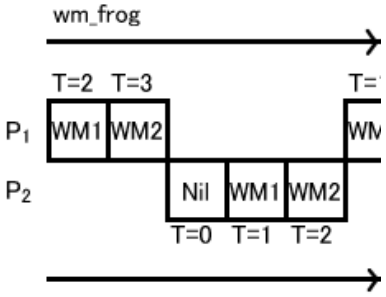

図 1: Bella のタイムスタンプ

ここで，図 1 のようにプロトコル P_1，P_2 を実行したとき，同じタイムスタンプ（T=2）を持った，異なるステップのメッセージが存在することがわかる．もし，この 2 つのメッセージの構成が異なるならば，両者を区別することができる．しかし，P_1 の $WM1$ と P_2 の $WM2$ を比較すると，暗号化された部分が，タイムスタンプ（$Number$），エージェントの名前（$Agent$），セッション鍵（Key）を共有鍵（$shrK$）で暗号化した構成になっており，同じ構成を持っている．

このため，両者のメッセージを区別することができず，$WM1$ のメッセージを $WM2$ のメッセージとして送信したり，逆に $WM2$ のメッセージを $WM1$ のメッセージとして送信することができる．したがって，異なるはずのメッセージが同一である可能性があるため，単一性を証明できなかった．

この原因は，WMF がタイムスタンプをグローバルクロックのようにユニークなものとして扱っているにもかかわらず，プロトコルごとにローカルな番号をつける定式化を行っているためである．本研究では，メッセージの連続性を保証するタイムスタンプ（$Number$）とメッセージのユニーク性を保証するナンス（$Nonce$）の複合メッセージを，WMF のタイムスタンプとして用いることで解決した．

WM1 の改良前のメッセージ {| *Agent A*, *Crypt* (*shrK A*){| *Number* (*CT evs1*), *Agent B*, *Key KAB* |} |}

WM2 の改良前のメッセージ
(*Crypt* (*shrK B*) {| *Number* (*CT evs2*), *Agent A*, *Key KAB* |})

2 つ目は機密性の検証におけるその性質の定式化である．Bella が Kerberos の検証で用いた定式化では，WMF の検証ができなかった．解析を行った結果，機密性を保証するためには，「$KAB \notin keysFor\ (analz(\ spies\ evs\ ))$」という前提条件が新たに必要となることが判明した．これは，作成したセッション鍵が，過去のトレースに含まれる暗号文の復号化鍵であってもならないという意味である．もし復号化鍵が流れたとするならばイニシエータが *Spy* であるということになる．言い換えれば，イニシエータが *Spy* でなければ，復号化鍵はネットワーク上を流れることはないので，この条件は妥当といえる．したがって，「$KAB \notin keysFor\ (analz(\ spies\ evs\ ))$」を前提条件として加えることで，機密性の検証は成功した．

これらの 2 つの問題点は，イニシエータがセッション鍵を作成することに起因するものである．なぜなら，単一性においては，もしセッション鍵を作成するイニシエータが *Spy* であった場合，過去に同じタイムスタンプをもったメッセージが存在するならば，それを代替して使用することができてしまうからである．また，機密性においては，イニシエータが復号化鍵と同じセッション鍵を作成した場合，本来通信経路を流れてはいけない復号化鍵が流れたことになり，問題が発生する．すなわち，それはイニシエータが *Spy* である可能性を示唆する．

5 まとめ

本研究では，定理証明器 Isabelle/HOL を使って WMF を検証した結果，このプロトコルの脆弱性が引き起こす具体的な問題点を指摘し，その解決策を示した．

今後の課題は，ゼロ知識証明やグループ署名などをモデル化し，より高度で実用的な暗号プロトコルを検証することである．また，Needham-Schroeder や Kerberos，WMF を用いて開発された実用レベルのプロトコルの検証を行うことである．

参考文献

[1] Martin Abadi and Roger Needam, Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering 22*, pages 6-15, 1995.

[2] Ross Anderson, Why Cryptosystems Fail. *In Proceedings of the 1st ACM Conference on Communications and Computer Security (CCS'93)*, 1993.

[3] Giampaolo Bella, Inductive Verification of Cryptographic Protocols. *Technical Report 493, Computer Laboratory, University of Cambridge*, 2000.

[4] Formal Systems (Europe) Ltd, Failures Divergence Refinement - User Manual and Tutorial. Version 1.3, 1993.

[5] Gerard J. Holzmann, The Spin Model Checker: Primer and Reference Manual. *Addison-Wesley Pub (Sd)*, ISBN: 0-3212-2862-6, 2003.

[6] Matt Kaufmann, Panagiotis Manolios, and J Strother Moore, Computer-Aided Reasoning: An Approach. Kluwer Academic Publishers, ISBN 0-7923-7744-3, 2000.

[7] Gavin Lowe, An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, volume 56, number 3, pages 131-133, 1995.

[8] Gavin Lowe, Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR. *In Tools and Algorithms for the Construction and Analysis of Systems*, Margaria and Steffen (eds.), volume 1055 of Lecture Notes in Computer Science, Springer Verlag, pages 147-166, 1996.

[9] Gavin Lowe, A Hierarchy of Authentication Specifications. *Proceedings of 10th IEEE Computer Security Foundations Workshop*, 1997,

[10] Tobias Nipkow and Lawrence C. Paulson and Markus Wenzel, Isabelle/HOL A Proof Assistant for Higher-Order Logic. *Springer*, ISBN 3-540-43376-7, 2002.

[11] W. Marrero, E.M. Clarke, and S. Jha. Model Checking for Security Protocols. *Technical Report CMU-SCS-97-139*, Carnegie Mellon University, May 1997.

[12] S. Owre, N. Shankar, J. M. Rushby, D. W. J. Stringer-Calvert, PVS Language Reference - Version 2.4, November 2001. *SRI International*, 2001.

[13] Lawrence C. Paulson, The Inductive Approach to Verifying Cryptographic Protocols. *J. Computer Security*, **6** pages 85-128, 1998.

[14] Bruce Schneier, 山形 浩生 監訳, 暗号技術大全. ソフトバンクパブリッシング株式会社, ISBN 4-7973-1911-9, page 65, 2003.