2F-2

セキュリティプロトコルの代数モデルに基づく形式化*

金城直貴†, 二木厚吉†

北陸先端科学技術大学院大学 情報科学研究科

e-mail:{naok, kokichi}@jaist.ac.jp

1. はじめに

近年、インターネットの普及に伴い、多くの通信プロトコルが考案され、ますます大規模化、複雑化している。考案されたプロトコルには、暗号技術を利用したセキュリティプロトコルがあり、ネットワーク上での安全な商取引や選挙などへの適用が期待されている。

セキュリティプロトコルが本当に目的どおりに機能するのか、論理的矛盾や曖昧性はないのか、通信相手の認証、秘密通信というセキュリティ要件を満たすのか、その正当性が非常に重要となる。

しかし、セキュリティプロトコルの正当性を得ることは難しく、発表された数年後に欠陥が明らかにされるケースが存在する。一方、形式言語でセキュリティプロトコルを記述し、その検証によって欠陥を明らかにしたことが報告され、形式言語のセキュリティ検証への有効性が注目され、期待が高まっている。

そこで本論文では、セキュリティプロトコルの例としてNeedham-Schroeder Public-Key protocol を取り上げ、形式仕様言語 Maude によって行われた同プロトコルの形式化[4]をもとにした、代数仕様言語 CafeOBJ[1][2]による記述について報告する。

2. Needham-Schroeder Public-Key protocol

Needham-Schroeder Public-Key protocol(以下 NSPK protocol)は公開鍵暗号方式を利用したプロトコルであり、通信者の認証を目的とする。プロトコルは7ステップからなるが、公開鍵が周知であると仮定することで公開鍵配布に関する4ステップを省略し、3ステップに簡略化したプロトコルについて説明する。

Message 1. A → B : $\{Na, A\}_{Kb}$
Message 2. B → A : $\{Na, Nb\}_{Ka}$
Message 3. A → B : $\{Nb\}_{Kb}$

Message 1 は、A が nonce(=Na)を生成し、自身の ID を付加したメッセージ{Na, A}を、B の公開鍵 Kb で暗号化して B に送信することを表す。nonce とはセッションごとに新しくかつ、ランダムに生成される値である。

Message 2 は、B が Message 1 から A の nonce(=Na)を抜き取り、自身の nonce(=Nb)を付加したメッセージ{Na, Nb}を A の公開鍵 Ka で暗号化して A に送信することを表す。

Message 3 は A が Message 2 から B の nonce(=Nb)を抜き取り、メッセージ{Nb}を B の公開鍵 Kb で暗号化して B に送信することを表す。

公開鍵暗号方式の性質上、A の公開鍵で暗号化されたメッセージは、A の秘密鍵でのみ復号が可能であり、B も同様である。すなわち、Message 1、Message 3 を読むことが出来るのは B だけであり、Message 2 を読むことが出来るのは A だけである。NSPK protocol では、Message 2 を受け取り、メッセージ内に自身の nonce を確認した A は B を認証し、Message 3 を受け取り、メッセージ内に自身の nonce を確認した B は A を認証する。その結果、相互の認証が完了する。

しかし、NSPK protocol では、次の方法を用いることで、正しい認証が行われないことが知られている[3]。

Message 1. A → S : $\{Na, A\}_{Ks}$
Message 1'. S(A) → B : $\{Na, A\}_{Kb}$
Message 2. B → A : $\{Na, Nb\}_{Ka}$
Message 3'. A → S : $\{Nb\}_{Ks}$
Message 3. S(A) → B : $\{Nb\}_{Kb}$

A が S とセッションを開始した直後に、S は A から受け取ったメッセージをもとに B とセッションを開始する。B はメッセージ内にある ID(=A)を見て A に返信する。メッセージを受け取った A はメッセージ内に S に出した nonce(=Na)が含まれていることから、S に相手の nonce(=Nb)を送り返す。A からのメッセージを受け取った S はそのメッセージを B に送る。これで、S の B に対する A への成りすましが完了する。

3. NSPK protocol の形式化

3.1. Configuration

オブジェクトが生成されたり、メッセージが受け渡されたりすることを Configuration の記述で表現することができる。Configuration は状態を表し、オブジェクトとメッセージから構成される。さらに、Configuration

*Algebraic formalization of security protocol
†Naoki Kinjo, Kokichi Futatsugi
Japan Advanced Institute of Science and Technology
School of Information Science

上に次のような書き換え規則を定義することで、オブジェクトの生成、変更、消滅とメッセージの発信、消費が表現できる。

$M_1 \cdots M_n$ $\langle O_1 : C_1 \mid attrs_1 \rangle \cdots \langle O_m : C_m \mid attrs_m \rangle$
$\Rightarrow$ $\langle O_{i1} : C_{i1} \mid attrs_{i1} \rangle \cdots \langle O_{ik} : C_{ik} \mid attrs_{ik} \rangle$
$\langle Q_1 : D_1 \mid attrs_1' \rangle \cdots \langle Q_p : D_p \mid attrs_p' \rangle$
$M_1' \cdots M_q'$.

上記の規則は、メッセージ $M_1 \cdots M_n$ が消費され、オブジェクト $O_1 \cdots O_m$ はクラス $C_1 \cdots C_m$ や属性 $attrs_1$ が変更された上で $O_{i1} \cdots O_{ik}$ となって存続あるいは消滅し、新たなオブジェクト $Q_1 \cdots Q_p$ が生成され、新たなメッセージ $M_1' \cdots M_q'$ が発信されたことを意味する。

3.2. Agent

Agent をクラスとして宣言する。

```
class Agent {
  seckey : Sec-Key   — エージェントの秘密鍵
  sfield : Run       — 情報の保存
  dcom : FieldSet    — セッション要求
  cnt : Int          — nonce の ID
}
```

Agent は、秘密鍵(seckey)、セッションで得た情報の保存(sfield)、セッション要求(dcom)、生成した nonce に付加する ID(cnt)を属性として持つ。

3.3. Message

Message を次のように形式化する。

```
mod! MESSAGE {
  protecting(FIELD)
  [ Message ]
  op msg-init: -> Message
  op msg : Agent Agent Field -> Message
}
```

Message は空のメッセージである msg-init と送信者、受信者、メッセージ内容とで構成される msg で表現される。メッセージの内容は FIELD で表され、FIELD では、メッセージの要素と、要素に行われる操作(暗号化、復号化、組など)が定義されている。

3.4. シーケンス

プロトコルのシーケンスを、前述の Agent、Message で構成される Configuration 上の書き換え規則で表現する。一例として、セッション開始時の規則を挙げる。

```
<A : Agent | sfield = RI, dcom=fset+(B, FSET), cnt=CNT>
 => <A : Agent | sfield = run+(RI, run(n(A, CNT), B,
      mtfield)), dcom = FSET, cnt = (CNT + 1) >
    msg(A, B, ed(pkey(B), tuple(n(A, CNT), A))) .
```

B とのセッションの要求(dcom)を持つ A は、メッセージを生成(msg(A, B, ed(pkey(B), tuple(n(A, CNT), A))))して、セッションを開始する。その際、A の属性(sfield、dcom、cnt)が変更される。

セッションには、メッセージの盗聴、横取り、改竄が可能な、悪意を持つ Agent も存在する。これら悪意を持つ Agent を定義するには、全てのメッセージを受信し、その結果、自身の属性と発信するメッセージを変更する規則を記述する。一例として、Message 1 の盗聴に関する規則を挙げる。

```
msg(A, B, ed(pkey(B), tuple(n(A, CNT), A)))
<C : Agent | sfield = RI>
 => <C : Agent | sfield = run+(RI,  ed(pkey(B),
      tuple(n(A, CNT), A)))>
      msg(A, B, ed(pkey(B), tuple(n(A, CNT), A))) .
```

C は A から B へのメッセージを受信(C≠B)し、自身の既知の情報に B の公開鍵で暗号化されたメッセージを追加し、受信したメッセージをそのまま発信する。これで、C による A が B に宛てた Message 1 の盗聴が完了する。同様の方法で、受信メッセージを変更したメッセージを発信する規則を定義することでメッセージの改竄が表現され、メッセージを受信した後に、メッセージを発信しない規則を定義することで横取りが表現される。

4. まとめ

本稿では、通信者間の認証を目的とした NSPK protocol の書き換え論理による記述例を報告した。セキュリティプロトコルの正当性を証明することは重要であり、その検証に形式仕様の有効性が期待されている。

今後は、セキュリティプロトコルの振舞仕様による記述を試み、書き換え論理による推論規則とは異なる角度からの検証を行う予定である。

参考文献

[1] R. Diacoonescu, K. Futatsugi, "CaféOBJ Report", World Scientific, 1998.

[2] 二木 厚吉, "代数モデルの基礎", コンピュータソフトウェア Vol13, No1(1996), pp.4 - 22

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

[4] Denker. G, Meseguer. J, Talcott. C, "Formal Specification and Analysis of Active Networks and Communication Protocols: The Maude Experience", Internal Report, Computer Science Laboratory, SRI International, Menlo Park, CA, 1999.