EAP-AKA の Scyhter による評価結果

国立研究開発法人 情報通信研究機構

1. 基本情報

- ✧ 名前

  Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement

- ✧ 機能

  移動体通信（3G）における認証モジュール IM を用いた相互認証・鍵交換プロトコル。暗号として IM に組み込まれた AKA アルゴリズムを利用する。

- ✧ 関連する標準

  RFC4187 (http://www.ietf.org/rfc/rfc4187.txt)

2. Scyther の文法による記述

2.1. プロトコル仕様

```
usertype String;
const Success: String;
/* key generation function */
hashfunction f1;
hashfunction f2;
hashfunction f5;
/*
  Instead of declaration that XOR is invertible,
  a set of functions (+, -) is used.
 */
const Add: Function;
const Sub: Function;
inversekeys(Add, Sub);
protocol EAP-AKA(A, P)
{
  role A
  {
    //variables
    fresh RAND: Nonce;
    fresh SQN: Nonce;

    //sequence
```

```
    send_!0(A, A, SQN); // with packet 0, Weakagree is not satisfied.
    send_1(A, P,  A); //with packets 1-2, Nisynch is not satisfied.
    recv_2(P, A,  P);
    send_3(A, P,  RAND, Add(SQN, f5(k(A,P), RAND)), f1(k(A,P), SQN,
RAND));
    recv_4(P, A,  f2(k(A,P), RAND));
    send_5(A, P,  Success);
  }
   role P
  {
     //variables
    var RAND: Nonce;
    var SQN: Nonce;

     //sequence
    recv_1(A, P,  A);
    send_2(P, A,  P);
    recv_3(A, P,  RAND, Add(SQN, f5(k(A,P), RAND)), f1(k(A,P), SQN,
RAND));
    claim_p0(P, Running, A, RAND, SQN);
    send_4(P, A,  f2(k(A,P), RAND));
    recv_5(A, P,  Success);
  }
}
```

2.2. 攻撃者モデル

Scyther はデフォルトで Dolev-Yao モデルを想定しており、特に記載すべき項目はない。

2.3. セキュリティ要件

```
// ロール A のセキュリティプロパティ
claim_a0(A, Running, P, RAND, SQN);
claim_a1(A, Secret, k(A,P));
claim_a2(A, Weakagree);
claim_a3(A, Niagree);
claim_a4(A, Nisynch);
claim_a5(A, Commit, P, RAND, SQN);
// ロール P のセキュリティプロパティ
claim_p1(P, Secret, k(A,P));
claim_p2(P, Weakagree);
claim_p3(P, Niagree);
claim_p4(P, Nisynch);
claim_p5(P, Commit, A, RAND, SQN);
```

3. Scyther による評価結果

3.1. 出力

Scyther で評価可能なセキュリティプロパティについての、bounded な評価結果は以下のとおりである。non-injective agreement が満たされない例が示されているが、次の節で現実の攻撃環境としては意味のない例であることを説明する。

```
claim id [EAP-AKA,a1], Secret(k(A,P))       : No attacks.
claim id [EAP-AKA,a2], Weakagree    : No attacks.
claim id [EAP-AKA,a3], Niagree      : No attacks.
claim id [EAP-AKA,a4], Nisynch      : At least 1 attack.
claim id [EAP-AKA,a5], Commit(P,RAND,SQN)   : No attacks.
claim id [EAP-AKA,p1], Secret(k(A,P))       : No attacks.
claim id [EAP-AKA,p2], Weakagree    : No attacks.
claim id [EAP-AKA,p3], Niagree      : At least 2 attacks.
claim id [EAP-AKA,p4], Nisynch      : At least 2 attacks.
claim id [EAP-AKA,p5], Commit(A,RAND,SQN)   : No attacks.
```

Scyther による評価では、EAP-AKA プロトコルはロール A について non-injective agreement を、 ロール P について weak agreement を満たすが、それ以上の性質を満たさないとしている。これは、主に EAP が認証プロセスを開始する以前、もしくは終わった後に送信されるメッセージ（EAP-Success など）の存在が原因であり、本質的な問題ではない。実際、いずれのロールも、EAP-AKA において本質的に重要な RAND に関する non-injective agreement を満たしている。また、メッセージ 1,2,5 をコメントアウトして評価することで、EAP-AKA がすべてのロールについて、Scyther で評価可能なすべての性質を満たすことを確認できる。

3.2. 攻撃の解説

次ページの図.2 では、EAP-AKA プロトコルがロール P について non-injective agreement を満たさない例を表している。これは現実的な意味で攻撃とは言えない例である。ここで注目すべきは、メッセージ 1,2 のやり取りが繋がっていないこと、そしてメッセージ 5 を攻撃者が送信している点である。これらのメッセージは誰でも生成できるため、たとえばメッセージ“Success”を共有ができていない、という評価結果になる。

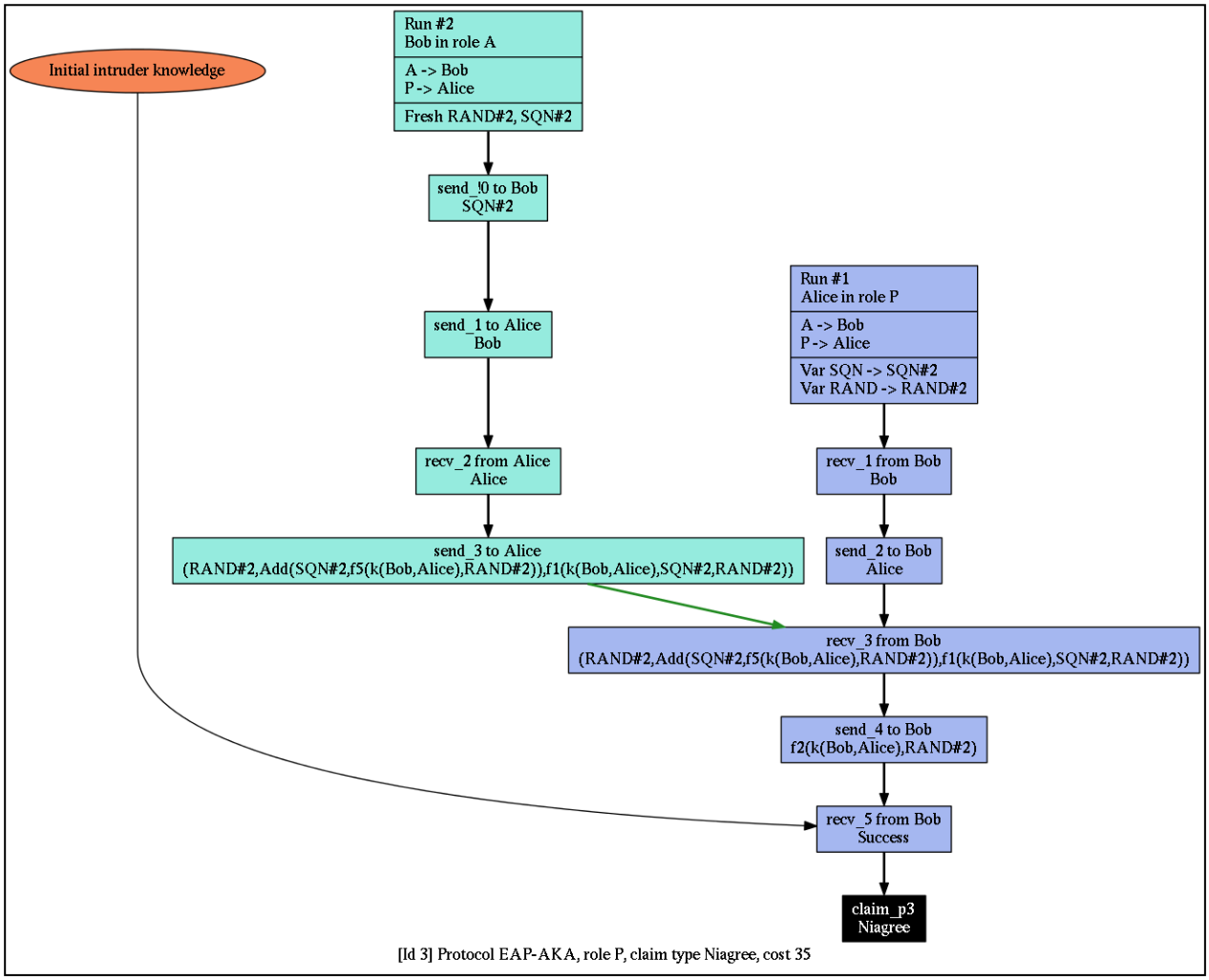

図.1　攻撃に関する解説

4. 形式化

4.1. 方針

IETF の通信プロトコルでは、ペイロードの階層ごとに、ペイロードタイプ、ペイロード長などが記載されている。しかし、これらの情報は冗長であり、また、Scyther では長さなどの情報をチェックすることはできない。そこで、メッセージのペイロードを特定するための、最低限のメッセージタイプ情報は残し、それ以外の認証に関係ないと思われる情報はモデルから削除した。

EAP プロトコルでは、Identifier フィールドを用いて Request と Response を 1 対 1 対応させている。しかし、Scyther ではメッセージのタイプチェックで不整合があるとエラーを返す。すなわち、ペイロードが異なるメッセージは自動的に判別する。EAP-AKA ではペイロ

ードが同じになるようなメッセージがないため、Identifier フィールドは省略した。

EAP-AKA プロトコルでは、MAC の計算に用いる関数を HMAC-SHA1-128 などと特定しているが、Scyther ではアルゴリズム個別の特徴を捉えることができないので関数名を省略し、また、入力についても、鍵とそれ以外の情報を区別していない。

ピアを特定する情報として、固有名や NAI などがあるが、差異はないので、モデル上はすべてロール名で統一している。

EAP-AKA では、AUTN 属性値の詳細が記述されておらず、また、データ長が 128 ビットであるため、AKA の仕様をそのまま当てはめることはできない。今回のモデル化では、AUTN=SQN XOR AK と仮定した。

4.2. 妥当性

抽象化は行っているが、情報が大きく損なわれたり、特殊なデータ変換を用いたりはしていないので、妥当な形式化であると考える。

4.3. 検証ツールとの相性

プロトコル仕様、攻撃者モデルを記述するにあたって、特に制限はなかった。

セキュリティ要件を記述するにあたって、Scyther では、鍵長など数値化された情報を記述することができない。同様に辞書攻撃について評価することはできない。これは暗号プロトコルではなく、暗号プリミティブの安全性として評価されるべき項目である。データの完全性、秘匿性は本評価の対象である認証プロトコルの範囲外であるため、評価を行っていない。

相互認証については、暗号プロトコルが満たすべき性質が記載されていない。しかし、リプレイ攻撃に対する耐性を謳っていること、鍵共有を目的としていることから、injective agreement が達成されるべきセキュリティプロパティであるとみなした。

Scyther では、injective な性質について評価ができないため、non-injective agreement よりも強い性質である non-synchronization について評価を行った。また、セッション鍵は RAND と事前共有鍵 k(A,P)から生成されるため、RAND の共有が成功していれば、セッション鍵は秘密裡に共有できたと考える。これについては、特定データについて non-injective agreement が成立していることを確認する記述を行っている。

4.4. 検証ツール適用時の性能

検証時間は 0.3 秒だった。実行環境は以下のとおり。

- CPU：AMD Phenom X4 9750B (2.4GHz)
- メモリ：1.7GB

5. 備考

本文書は、**総務省「暗号・認証技術等を用いた安全な通信環境推進事業に関する実証実験の請負　成果報告書」**からの引用である。