EAP-SIM 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 Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)

- ✧ 機能

  FDD-TDMA 方式で実現されている第二世代携帯電話規格 (GSM) における SIM を用いた相互認証・鍵交換プロトコル。

- ✧ 関連する標準

  RFC4186 (https://www.ietf.org/rfc/rfc4186.txt)

2. Scyther の文法による記述

2.1. プロトコル仕様

```
usertype String;
hashfunction MAC;
hashfunction GSM-KeyGen;
hashfunction SHA1;
hashfunction PRF;
const EAP-Request, EAP-Responce, EAP-Success, EAP-Failure;
const Identity, Notification, Nak: String;
const SIM-Start, SIM-Challenge: String;
const Left, Right: String;
const SIM-TEK1, SIM-TEK2, SIM-MSK, SIM-EMSK: String;
protocol EAP-SIM(P, A)
{
        role P
        {
                fresh NONCE-MT: Nonce;
```

```
                    var SessionID: Nonce;
                    var RAND: Nonce;
                    recv_1(A, P, (EAP-Request, Identity, SessionID));
                    send_2(P, A, (EAP-Responce, Identity, SessionID,
P));
                    recv_3(A, P, (EAP-Request, SessionID, SIM-Start));
                    send_4(P, A, (EAP-Responce, SessionID, SIM-Start,
NONCE-MT));
                    recv_5(A, P,
                              (EAP-Request, SessionID, SIM-Challenge,
RAND,
                              MAC(EAP-Request, SessionID, SIM-Challenge,
RAND,
                              NONCE-MT,
                              PRF(SHA1(P, GSM-KeyGen(RAND, k(P, A),
Right), NONCE-MT),
                              SIM-TEK1))));
                    send_6(P, A,
                              (EAP-Request, SessionID, SIM-Challenge,
                              MAC(EAP-Request, SessionID, SIM-Challenge,
                              GSM-KeyGen(RAND, k(P, A), Left),
                              PRF(SHA1(P, GSM-KeyGen(RAND, k(P, A),
Right), NONCE-MT),
                              SIM-TEK1))));
                    recv_7(A, P, (EAP-Success, SessionID));
          }
          role A
          {
                    fresh SessionID: Nonce;
                    fresh RAND: Nonce;
                    var NONCE-MT: Nonce;
                    send_1(A, P, (EAP-Request, Identity, SessionID));
```

```
                    recv_2(P, A, (EAP-Responce, Identity, SessionID,
P));
                    send_3(A, P, (EAP-Request, SessionID, SIM-Start));
                    recv_4(P, A, (EAP-Responce, SessionID, SIM-Start,
NONCE-MT));
                    send_5(A, P,
                              (EAP-Request, SessionID, SIM-Challenge,
RAND,
                              MAC(EAP-Request, SessionID, SIM-Challenge,
RAND,
                              NONCE-MT,
                              PRF(SHA1(P, GSM-KeyGen(RAND, k(P, A),
Right), NONCE-MT),
                              SIM-TEK1))));
                    recv_6(P, A,
                              (EAP-Request, SessionID, SIM-Challenge,
                              MAC(EAP-Request, SessionID, SIM-Challenge,
                              GSM-KeyGen(RAND, k(P, A), Left),
                              PRF(SHA1(P, GSM-KeyGen(RAND, k(P, A),
Right), NONCE-MT),
                              SIM-TEK1))));
                    send_7(A, P, (EAP-Success, SessionID));
          }
}
```

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 P のセキュリティプロパティ
claim(P, Running, A, NONCE-MT, RAND);
claim_p1(P, Secret,
          SHA1(P, GSM-KeyGen(RAND, k(P, A), Right), NONCE-MT));
claim_p2(P, Alive);
```

```
claim_p3(P, Commit, A, NONCE-MT, RAND);
claim_p4(P, Weakagree);
// ロールAのセキュリティプロパティ
claim(A, Running, P, NONCE-MT, RAND);
claim_a1(A, Secret,
         SHA1(P, GSM-KeyGen(RAND, k(P, A), Right), NONCE-MT));
claim_a2(A, Alive);
claim_a3(A, Commit, P, NONCE-MT, RAND);
claim_a4(A, Weakagree);
```

3. Scytherによる評価結果

3.1. 出力

```
claim              EAP-SIM,P                                    SKR_P1
SHA1(P,GSM-KeyGen(RAND,k(P,A),Right),NONCE-MT,VerList,hash(VerList,C
hoose))     Ok     [proof of correctness]
claim    EAP-SIM,P        Alive_P2         -        Ok       [proof of
correctness]
claim    EAP-SIM,P        Commit_P3        (A,NONCE-MT,RAND)        Ok
[proof of correctness]
claim    EAP-SIM,P        Weakagree_P4     -        Ok       [proof of
correctness]
claim    EAP-SIM,P       Niagree_P5       -       Fail     [at least 2
attacks]
claim    EAP-SIM,P       Nisynch_P6       -       Fail     [at least 2
attacks]
claim              EAP-SIM,A                                    SKR_A1
SHA1(P,GSM-KeyGen(RAND,k(P,A),Right),NONCE-MT,VerList,hash(VerList,C
hoose))     Ok     [proof of correctness]
claim    EAP-SIM,A        Alive_A2         -        Ok       [proof of
correctness]
claim    EAP-SIM,A        Commit_A3        (P,NONCE-MT,RAND)        Ok
```

```
[proof of correctness]
claim    EAP-SIM,A        Weakagree_A4      -        Ok        [proof of
correctness]
claim    EAP-SIM,A        Niagree_A5        -        Ok        [proof of
correctness]
claim    EAP-SIM,A        Nisynch_A6        -        Fail      [at least 2
attacks]
```

Scyther による評価では、EAP-SIM 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-SIM の認証プロセスにおいて本質的に重要な 2 つのナンス NONCE-MT と RAND に関する non-injective agreement を満たしている。また、メッセージ 1,2,7 をコメントアウトして評価することで、EAP-SIM 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

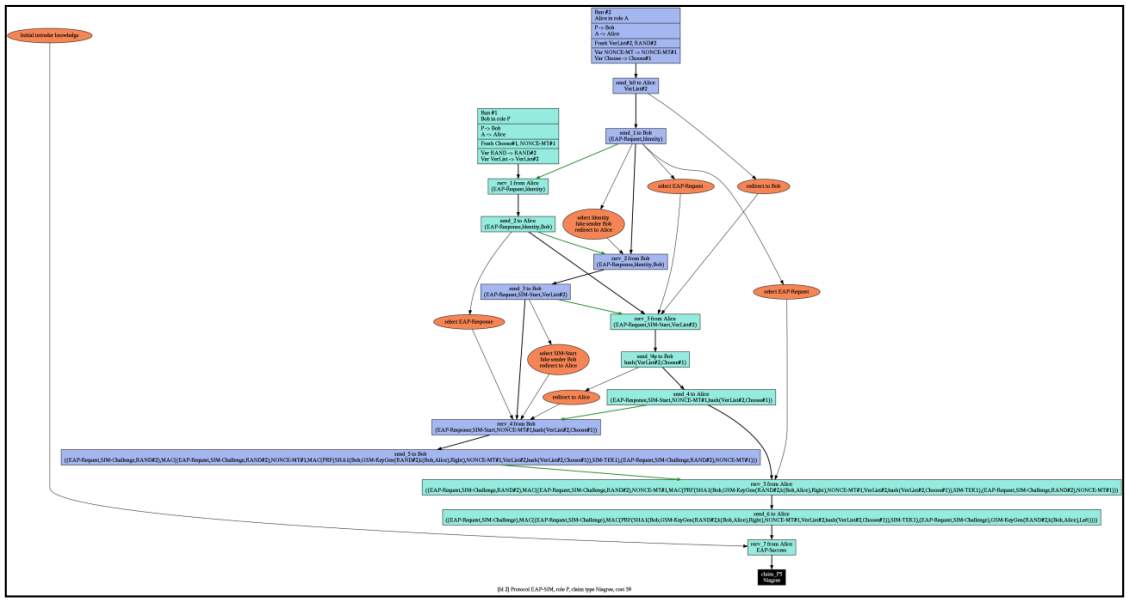

図 1.　攻撃に関する解説

上図では、EAP-SIM 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、最後のメッセージ(EAP-Success)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、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。

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-SIM 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-SIM 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の入力順序を区別していない。

MAY で記述されている個所はモデルに含めていない。

一般には、ロール A の AT_VERSION_LIST、ロール P の AT_SELECTED_VERSION は可変だが、ある程度予測可能な値である。Scyther では、予測可能な値という型の変数を持たないため、ナンスとして変数を宣言し、該当変数のみを含むメッセージを事前にネットワークに流すことで、予測可能な数値をモデル化した。なお、このようなモデル化は、メッセージのペイロードをそのまま記述できる簡明さはあるが、攻撃者が可能な処理が増加するため、攻撃グラフの解析が煩雑になるというデメリットがある。

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

プロトコル仕様を記述するにあたって、EAP-SIM では GSM の鍵生成機能について十分な記述がない。Scyther による評価では、SIM に事前共有鍵及び何らかの疑似乱数生成機能が搭

載されているものと仮定して評価を行った。

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、セッション鍵はマスター鍵 MK、ナンス NONCE-MT 及び RAND から生成されるため、ナンス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 0.16 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。