EAP-IKEv2 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

  The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method

- ✧ 機能

  AP (Access Point) peer と EAP server 間の相互認証・鍵交換プロトコル。

- ✧ 関連する標準

  RFC5106 (https://www.ietf.org/rfc/rfc5106.txt)

2. Scyther の文法による記述

2.1. プロトコル仕様

```
////////////////////////////////////////////////////////////////////
/* constants (or easily expectable variables) */
usertype String;
usertype Number;
usertype SecurityAssociation;
usertype SecurityParameterIndex;
//Code field constants in EAP
const EAP-Request, EAP-Response, EAP-Success, EAP-Failure;
//Type field constants in EAP
const Identity, Notification, Nak: String;
//IKEv2 constants
const ZERO, ONE: Number;
const SAi, SAr: SecurityAssociation;
const SPIi, SPIr: SecurityParameterIndex;
////////////////////////////////////////////////////////////////////
/* (cryptographic) functions */
```

```
hashfunction prf, mac, prfplus;
const KeyPad4EAPIKEv2: String;
macro Kaut = prf(k(I,R), KeyPad4EAPIKEv2); //pre-shared key version
const DHg1, DHg2: Function; //finite field exponentials
const MacKey, EncKey, PrfKey;
////////////////////////////////////////////////////////////////
/* messages */
// message 1:I->R ---------------------------------
//EAP-Request/Identity
macro eap01 = (EAP-Request,  Identity);
// message 2:R->I ---------------------------------
//EAP-Response/Identity
macro eap02 = (EAP-Response, Identity, R);
// message 3:I->R ---------------------------------
//EAP-Req (HDR, SAi, KEi, Ni)
macro ike03hdr = (SPIi, ZERO);
macro KEi = DHg1(Xi);
//send
macro ike03pl-I = (SAi, KEi, Ni);
macro eap03-I = (EAP-Request, ike03hdr, ike03pl-I);
//recv
macro ike03pl-R = (SAi, Gi, Ni);
macro eap03-R = (EAP-Request, ike03hdr, ike03pl-R);
//For executable
//send
macro ike03pl-Ix = ike03pl-I;
macro eap03-Ix = (EAP-Request, ike03hdr, ike03pl-Ix);
//recv
macro ike03pl-Rx = ike03pl-I;
macro eap03-Rx = (EAP-Request, ike03hdr, ike03pl-Rx);
// message 4:R->I ---------------------------------
//EAP-Res (HDR, SAr, KEr, Nr)
```

```
macro ike04hdr = (SPIi, SPIr);
macro KEr = DHg1(Xr);
//recv
macro ike04pl-I = (SAr, Gr, Nr);
macro eap04-I = (EAP-Response, ike04hdr, ike04pl-I);
//send
macro ike04pl-R = (SAr, KEr, Nr);
macro eap04-R = (EAP-Response, ike04hdr, ike04pl-R);
//For executable
//recv
macro ike04pl-Ix = ike04pl-R;
macro eap04-Ix = (EAP-Response, ike04hdr, ike04pl-Ix);
//send
macro ike04pl-Rx = ike04pl-R;
macro eap04-Rx = (EAP-Response, ike04hdr, ike04pl-Rx);
// key derivation --------------------------------
macro SKEYSEEDi = prf(Ni, Nr, DHg2(Gr, Xi));
macro SKai-I = prfplus(SKEYSEEDi, MacKey, I, R, SPIi, SPIr);
macro SKar-I = prfplus(SKEYSEEDi, MacKey, R, I, SPIi, SPIr);
macro SKei-I = prfplus(SKEYSEEDi, EncKey, I, R, SPIi, SPIr);
macro SKer-I = prfplus(SKEYSEEDi, EncKey, R, I, SPIi, SPIr);
macro SKpi-I = prfplus(SKEYSEEDi, PrfKey, I, R, SPIi, SPIr);
macro SKpr-I = prfplus(SKEYSEEDi, PrfKey, R, I, SPIi, SPIr);
macro SKEYSEEDr = prf(Ni, Nr, DHg2(Gi, Xr));
macro SKai-R = prfplus(SKEYSEEDr, MacKey, I, R, SPIi, SPIr);
macro SKar-R = prfplus(SKEYSEEDr, MacKey, R, I, SPIi, SPIr);
macro SKei-R = prfplus(SKEYSEEDr, EncKey, I, R, SPIi, SPIr);
macro SKer-R = prfplus(SKEYSEEDr, EncKey, R, I, SPIi, SPIr);
macro SKpi-R = prfplus(SKEYSEEDr, PrfKey, I, R, SPIi, SPIr);
macro SKpr-R = prfplus(SKEYSEEDr, PrfKey, R, I, SPIi, SPIr);
//For executable
macro SKEYSEEDix = prf(Ni, Nr, DHg2(DHg1(Xr), Xi));
```

```
macro SKai-Ix = prfplus(SKEYSEEDix, MacKey, I, R, SPIi, SPIr);
macro SKar-Ix = prfplus(SKEYSEEDix, MacKey, R, I, SPIi, SPIr);
macro SKei-Ix = prfplus(SKEYSEEDix, EncKey, I, R, SPIi, SPIr);
macro SKer-Ix = prfplus(SKEYSEEDix, EncKey, R, I, SPIi, SPIr);
macro SKpi-Ix = prfplus(SKEYSEEDix, PrfKey, I, R, SPIi, SPIr);
macro SKpr-Ix = prfplus(SKEYSEEDix, PrfKey, R, I, SPIi, SPIr);
macro SKEYSEEDrx = prf(Ni, Nr, DHg2(DHg1(Xi), Xr));
macro SKai-Rx = prfplus(SKEYSEEDrx, MacKey, I, R, SPIi, SPIr);
macro SKar-Rx = prfplus(SKEYSEEDrx, MacKey, R, I, SPIi, SPIr);
macro SKei-Rx = prfplus(SKEYSEEDrx, EncKey, I, R, SPIi, SPIr);
macro SKer-Rx = prfplus(SKEYSEEDrx, EncKey, R, I, SPIi, SPIr);
macro SKpi-Rx = prfplus(SKEYSEEDrx, PrfKey, I, R, SPIi, SPIr);
macro SKpr-Rx = prfplus(SKEYSEEDrx, PrfKey, R, I, SPIi, SPIr);
// message 5:I->R ---------------------------------
//EAP-Req (HDR, SK{IDi, [CERT], [CERTREQ], [NFID], AUTH})
macro ike05hdr = (SPIi, SPIr);
macro IDi = I;
//send
macro ike05auth-I = prf(Kaut, eap03-I, Nr, prf(SKpi-I, IDi));
macro ike05ae-I = ({IDi, ike05auth-I}SKei-I, mac(IDi, ike05auth-I,
SKai-I));
macro eap05-I = (EAP-Request, ike05hdr, ike05ae-I);
//recv
macro ike05auth-R = prf(Kaut, eap03-R, Nr, prf(SKpi-R, IDi));
macro ike05ae-R = ({IDi, ike05auth-R}SKei-R, mac(IDi, ike05auth-R,
SKai-R));
macro eap05-R = (EAP-Request, ike05hdr, ike05ae-R);
//For executable
//send
macro ike05auth-Ix = prf(Kaut, eap03-Ix, Nr, prf(SKpi-Ix, IDi));
macro ike05ae-Ix = ({IDi, ike05auth-Ix}SKei-Ix, mac(IDi, ike05auth-Ix,
SKai-Ix));
```

```
macro eap05-Ix = (EAP-Request, ike05hdr, ike05ae-Ix);
//recv
macro ike05auth-Rx = prf(Kaut, eap03-Rx, Nr, prf(SKpi-Rx, IDi));
macro ike05ae-Rx = ({IDi, ike05auth-Rx}SKei-Rx, mac(IDi, ike05auth-Rx,
SKai-Rx));
macro eap05-Rx = (EAP-Request, ike05hdr, ike05ae-Rx);
// message 6:R->I --------------------------------
//EAP-Res (HDR, SK{IDr, [CERT], AUTH})
macro ike06hdr = (SPIi, SPIr);
macro IDr = R;
//recv
macro ike06auth-I = prf(Kaut, eap04-I, Ni, prf(SKpr-I, IDr));
macro ike06ae-I = ({IDr, ike06auth-I}SKer-I, mac(IDr, ike06auth-I,
SKar-I));
macro eap06-I = (EAP-Response, ike06hdr, ike06ae-I);
//send
macro ike06auth-R = prf(Kaut, eap04-R, Ni, prf(SKpr-R, IDr));
macro ike06ae-R = ({IDr, ike06auth-R}SKer-R, mac(IDr, ike06auth-R,
SKar-R));
macro eap06-R = (EAP-Response, ike06hdr, ike06ae-R);
//For executable
//recv
macro ike06auth-Ix = prf(Kaut, eap04-Ix, Ni, prf(SKpr-Ix, IDr));
macro ike06ae-Ix = ({IDr, ike06auth-Ix}SKer-Ix, mac(IDr, ike06auth-Ix,
SKar-Ix));
macro eap06-Ix = (EAP-Response, ike06hdr, ike06ae-Ix);
//send
macro ike06auth-Rx = prf(Kaut, eap04-Rx, Ni, prf(SKpr-Rx, IDr));
macro ike06ae-Rx = ({IDr, ike06auth-Ix}SKer-Rx, mac(IDr, ike06auth-Rx,
SKar-Rx));
macro eap06-Rx = (EAP-Response, ike06hdr, ike06ae-Rx);
// message 7:I->R --------------------------------
```

```
//EAP-Success
macro eap07 = (EAP-Success);
////////////////////////////////////////////////////////////////
/* helper protocols */
////////////////////////////////////////////////////////////////
protocol @oracles (DH-naked)
{
  /*************************************************/
  // (g^x)^y -> (g^y)^x
  role DH-naked
    {
      var x, y: Ticket;

      recv_!dh1(DH-naked,DH-naked, DHg2(DHg1(x),y));
      send_!dh2(DH-naked,DH-naked, DHg2(DHg1(y),x));
    }
}
////////////////////////////////////////////////////////////////
protocol @executable (MSG5, MSG6)
{
  /*************************************************/
  // message 5
  role MSG5
    {
      var Xi, Xr, Ni, Nr: Nonce;
      var I, R: Agent;

      recv_!msg51(MSG5,MSG5, eap05-Ix);
      send_!msg52(MSG5,MSG5, eap05-Rx);
    }
  /*************************************************/
  // message 6
```

```
  role MSG6
    {
      var Xi, Xr, Ni, Nr: Nonce;
      var I, R: Agent;

      recv_!msg61(MSG6,MSG6, eap06-Rx);
      send_!msg62(MSG6,MSG6, eap06-Ix);
    }
}
////////////////////////////////////////////////////////////////
protocol EAP-IKEv2(I, R)
{
  /*****************************************************/
  role I
    {
      /*-------------------------------------------*/
      //variables
      fresh Xi, Ni: Nonce;
      var Xr, Nr: Nonce;
      var Gr: Ticket;

      /*-------------------------------------------*/
      //sequence
      send_1( I,R,  eap01 );
      recv_2( R,I,  eap02 );
      //---- cryptographic protocol starts from here
      send_3( I,R,  eap03-I );
      recv_4( R,I,  eap04-I );
      claim(I, Running, R, Ni, Nr, KEi, Gr);
      send_!5( I,R,  eap05-I );
      recv_!6( R,I,  eap06-I );
      send_7( I,R,  eap07 );
```

```
    }

  /*************************************************/
  role R
    {
      /*-----------------------------------------*/
      //variables
      var Xi, Ni: Nonce;
      fresh Xr, Nr: Nonce;
      var Gi: Ticket;

      /*-----------------------------------------*/
      //sequence
      recv_1( I,R,  eap01 );
      send_2( R,I,  eap02 );
      //---- cryptographic protocol starts from here
      recv_3( I,R,  eap03-R );
      send_4( R,I,  eap04-R );
      recv_!5( I,R,  eap05-R );
      claim(R, Running, I, Ni, Nr, Gi, KEr);
      send_!6( R,I,  eap06-R );
      recv_7( I,R,  eap07 );
    }
}
```

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 I のセキュリティ要件
claim(I, SKR, SKEYSEEDi);
claim(I, Weakagree);
claim(I, Commit, R, Ni, Nr, KEi, Gr);
// ロール R のセキュリティ要件
```

```
claim(R, SKR, SKEYSEEDr);
claim(R, Weakagree);
claim(R, Commit, I, Ni, Nr, Gi, KEr);
```

3. Scyther による評価結果

3.1. 出力

攻撃は発見されなかった。

```
claim id [EAP-IKEv2,I2], SKR(prf(Ni,Nr,DHg2(Gr,Xi)))   :  No  attacks
within bounds.
claim id [EAP-IKEv2,I3], Weakagree   : No attacks within bounds.
claim id [EAP-IKEv2,I4], Commit(R,Ni,Nr,DHg1(Xi),Gr)   :  No  attacks
within bounds.
claim id [EAP-IKEv2,R2], SKR(prf(Ni,Nr,DHg2(Gi,Xr)))   :  No  attacks
within bounds.
claim id [EAP-IKEv2,R3], Weakagree   : No attacks within bounds.
claim id [EAP-IKEv2,R4], Commit(I,Ni,Nr,Gi,DHg1(Xr))   :  No  attacks
within bounds.
```

3.2. 攻撃の解説

攻撃は発見されなかった。

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-IKEv2 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-IKEv2 プロトコルでは、利用する暗号化方式を IKEv2 (RFC 5106)に一任している。

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暗号化方式やハッシュ関数を特定せず、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。

DH 鍵交換では、受信者のデータを Ticket 型変数 Gi, Gr で定義している。

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、Scyther では処理の可換性を厳密に記述する方法がないため、暗号プロトコルで送受信するメッセージの一部について、送信データと受信データが異なるような記述となっている。

攻撃者モデルを記述するにあたって、これらの送受信データが一致することが明確になるように、補助的な暗号プロトコル（helper protocol）を記述し、送信データから受信データを生成する手段を提供することで、できるだけ攻撃者に不利にならないようなモデル化を行っている。また、EAP-IKEv2 では、IKEv2 の AUTH ペイロードで MAC を計算する際に入力となるメッセージの範囲を指定していない。IKEv2 ではメッセージ全体を指定している。今回のモデル化では、EAP ヘッダを含めて MAC 値を計算するようモデル化している。

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、マスタ鍵に相当する SKEYSEED について non-injective agreement を満たすか評価を行った。

形式化における注意点として以下を挙げる。EAP-IKEv2 のモデル化は、helper protocol なしでは評価が十分に行われない(第 5 メッセージ以降をトレースできない)。したがって、Diffie-Hellman 鍵交換を近似するため helper protocol の記述が必須であり、結果として Scyther 用のモデルは冗長な記述となっている。また、これによりバグが混入しやすくなってしまった。Scyther tool は通常の開発環境に比べるとデバッグ環境が貧弱であり、コー

ドの肥大化に伴い、記述の誤りを見つけるためのコストが指数関数的に増大する傾向がある。

4.4. 検証ツール適用時の性能

検証時間は約 1 分だった。実行環境は以下のとおり。

- CPU：Intel Xeon E5502 1.87GHz x 4CPU(SMP)
- メモリ：48GB

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。