IKE-SIG 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 Internet Key Exchange (IKE)

- ✧ 機能

  IPsec の前に実行する鍵交換プロトコル。認証に電子署名を使用。

- ✧ 関連する標準

  RFC2409 (https://tools.ietf.org/html/rfc2409)

2. Scyther の文法による記述

2.1. プロトコル仕様

```
////////////////////////////////////////////////////////////////////////
// data type
usertype String;
usertype SecurityAssociation;

////////////////////////////////////////////////////////////////////////
// constants (or easily expectable variables)

const SAi, SAr: SecurityAssociation;

////////////////////////////////////////////////////////////////////////
// (cryptographic) functions
hashfunction mac, prf;
const DHg1, DHg2: Function; //finite field exponentital

////////////////////////////////////////////////////////////////////////
// packet specification
```

```
// message 1: I->R-----------------------------

macro ikev1sig-1-header = (Ci);
macro ikev1sig-1-payload = (SAi);
macro ikev1sig-1 = (ikev1sig-1-header, ikev1sig-1-payload);

// message 2: R->I-----------------------------

macro ikev1sig-2-header = (Ci, Cr);
macro ikev1sig-2-payload = (SAr);
macro ikev1sig-2 = (ikev1sig-2-header, ikev1sig-2-payload);

// message 3: I->R-----------------------------

macro KEi = DHg1(Xi);
macro ikev1sig-3-header = (Ci, Cr);

//send
macro ikev1sig-3-payload-I = (KEi, Ni);
macro ikev1sig-3-I = (ikev1sig-3-header, ikev1sig-3-payload-I);

//recv
macro ikev1sig-3-payload-R = (Gi, Ni);
macro ikev1sig-3-R = (ikev1sig-3-header, ikev1sig-3-payload-R);



// message 4: R->I-----------------------------

macro KEr = DHg1(Xr);
macro ikev1sig-4-header = (Ci, Cr);
```

```
//recv
macro ikev1sig-4-payload-I = (Gr, Nr);
macro ikev1sig-4-I = (ikev1sig-4-header, ikev1sig-4-payload-I);

//send
macro ikev1sig-4-payload-R = (KEr, Nr);
macro ikev1sig-4-R = (ikev1sig-4-header, ikev1sig-4-payload-R);



// key derivation-----------------------------

macro SharedSecret-I = DHg2(Gr, Xi);
macro SKEYID-I = prf(Ni, Nr, SharedSecret-I); //for signature
macro SKEYIDd-I = prf(SKEYID-I, SharedSecret-I, Ci, Cr);
macro SKEYIDa-I = prf(SKEYID-I, SKEYIDd-I, SharedSecret-I, Ci, Cr);
macro SKEYIDe-I = prf(SKEYID-I, SKEYIDa-I, SharedSecret-I, Ci, Cr);

macro SharedSecret-R = DHg2(Gi, Xr);
macro SKEYID-R = prf(Ni, Nr, SharedSecret-R); //for signature
macro SKEYIDd-R = prf(SKEYID-R, SharedSecret-R, Ci, Cr);
macro SKEYIDa-R = prf(SKEYID-R, SKEYIDd-R, SharedSecret-R, Ci, Cr);
macro SKEYIDe-R = prf(SKEYID-R, SKEYIDa-R, SharedSecret-R, Ci, Cr);



//For executable
macro SharedSecret-Ix = DHg2(DHg1(Xr), Xi);
macro SKEYID-Ix = prf(Ni, Nr, SharedSecret-Ix); //for signature
macro SKEYIDd-Ix = prf(SKEYID-Ix, SharedSecret-Ix, Ci, Cr);
macro SKEYIDa-Ix = prf(SKEYID-Ix, SKEYIDd-Ix, SharedSecret-Ix, Ci, Cr);
macro SKEYIDe-Ix = prf(SKEYID-Ix, SKEYIDa-Ix, SharedSecret-Ix, Ci, Cr);

macro SharedSecret-Rx = DHg2(DHg1(Xi), Xr);
```

```
macro SKEYID-Rx = prf(Ni, Nr, SharedSecret-Rx); //for signature
macro SKEYIDd-Rx = prf(SKEYID-Rx, SharedSecret-Rx, Ci, Cr);
macro SKEYIDa-Rx = prf(SKEYID-Rx, SKEYIDd-Rx, SharedSecret-Rx, Ci, Cr);
macro SKEYIDe-Rx = prf(SKEYID-Rx, SKEYIDa-Rx, SharedSecret-Rx, Ci, Cr);

// message 5: I->R-----------------------------

macro IDii = I;

//send
macro ikev1sig-5-header-I = ({Ci,Cr}SKEYIDe-I, mac((Ci,Cr), SKEYIDe-I));
macro HASHi-I = prf(SKEYID-I, KEi, Gr, Ci, Cr, SAi, IDii);
macro SIGi-I = {HASHi-I}sk(I);
macro ikev1sig-5-payload-I = (IDii, SIGi-I);
macro ikev1sig-5-I = (ikev1sig-5-header-I, ikev1sig-5-payload-I);

//recv
macro ikev1sig-5-header-R = ({Ci,Cr}SKEYIDe-R, mac((Ci,Cr), SKEYIDa-R));
macro HASHi-R = prf(SKEYID-R, Gi, KEr, Ci, Cr, SAi, IDii);
macro SIGi-R = {HASHi-R}sk(I);
macro ikev1sig-5-payload-R = (IDii, SIGi-R);
macro ikev1sig-5-R = (ikev1sig-5-header-R, ikev1sig-5-payload-R);



//For executable
//send
macro   ikev1sig-5-header-Ix   =   ({Ci,Cr}SKEYIDe-Ix,   mac((Ci,Cr),
SKEYIDe-Ix));
macro HASHi-Ix = prf(SKEYID-Ix, KEi, KEr, Ci, Cr, SAi, IDii);
macro SIGi-Ix = {HASHi-Ix}sk(I);
macro ikev1sig-5-payload-Ix = (IDii, SIGi-Ix);
macro ikev1sig-5-Ix = (ikev1sig-5-header-Ix, ikev1sig-5-payload-Ix);
```

```
//recv
macro   ikev1sig-5-header-Rx   =   ({Ci,Cr}SKEYIDe-Rx,   mac((Ci,Cr),
SKEYIDa-Rx));
macro HASHi-Rx = prf(SKEYID-Rx, KEi, KEr, Ci, Cr, SAi, IDii);
macro SIGi-Rx = {HASHi-Rx}sk(I);
macro ikev1sig-5-payload-Rx = (IDii, SIGi-Rx);
macro ikev1sig-5-Rx = (ikev1sig-5-header-Rx, ikev1sig-5-payload-Rx);



// message 6: R->I-----------------------------

macro IDir = R;

//recv
macro ikev1sig-6-header-I = ({Ci, Cr}SKEYIDe-I, mac((Ci,Cr), SKEYIDa-I));
macro HASHr-I = prf(SKEYID-I, Gr, KEi, Cr, Ci, SAi, IDir);
macro SIGr-I = {HASHr-I}sk(R);
macro ikev1sig-6-payload-I = (IDir, SIGr-I);
macro ikev1sig-6-I = (ikev1sig-6-header-I, ikev1sig-6-payload-I);

//send
macro ikev1sig-6-header-R = ({Ci, Cr}SKEYIDe-R, mac((Ci,Cr), SKEYIDa-R));
macro HASHr-R = prf(SKEYID-R, KEr, Gi, Cr, Ci, SAi, IDir);
macro SIGr-R = {HASHr-R}sk(R);
macro ikev1sig-6-payload-R = (IDir, SIGr-R);
macro ikev1sig-6-R = (ikev1sig-6-header-R, ikev1sig-6-payload-R);



//For executable
//recv
macro   ikev1sig-6-header-Ix   =   ({Ci,   Cr}SKEYIDe-Ix,   mac((Ci,Cr),
```

```
SKEYIDa-Ix));
macro HASHr-Ix = prf(SKEYID-Ix, KEr, KEi, Cr, Ci, SAi, IDir);
macro SIGr-Ix = {HASHr-Ix}sk(R);
macro ikev1sig-6-payload-Ix = (IDir, SIGr-Ix);
macro ikev1sig-6-Ix = (ikev1sig-6-header-Ix, ikev1sig-6-payload-Ix);

//send
macro ikev1sig-6-header-Rx = ({Ci, Cr}SKEYIDe-Rx, mac((Ci,Cr),
SKEYIDa-Rx));
macro HASHr-Rx = prf(SKEYID-Rx, KEr, KEi, Cr, Ci, SAi, IDir);
macro SIGr-Rx = {HASHr-Rx}sk(R);
macro ikev1sig-6-payload-Rx = (IDir, SIGr-Rx);
macro ikev1sig-6-Rx = (ikev1sig-6-header-Rx, ikev1sig-6-payload-Rx);

////////////////////////////////////////////////////////////////
// helper protocols
/*
  An adversary can ask these oracles to translate a packet to
  another form.
  I.e., these oracles help the adversarial model to be more precise.
  */

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, Ci, Cr: Nonce;
      var I, R: Agent;

      recv_!msg51(MSG5,MSG5, ikev1sig-5-Ix);
      send_!msg52(MSG5,MSG5, ikev1sig-5-Rx);
    }

  /*****************************************************/
  // message 6
  role MSG6
    {
      var Xi, Xr, Ni, Nr, Ci, Cr: Nonce;
      var I, R: Agent;

      recv_!msg61(MSG6,MSG6, ikev1sig-6-Rx);
      send_!msg62(MSG6,MSG6, ikev1sig-6-Ix);
    }

}

////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////
```

```
protocol IKEv1-sig(I, R)
{
  /*************************************************************/
  role I
    {
      /*************************************/
      // variables

      fresh Xi, Ni, Ci: Nonce;//Ci is I's cookie
      var Nr, Cr: Nonce;
      var Gr: Ticket; //g^Xr

      /*************************************/
      // sequence

      send_1(I,R, ikev1sig-1);
      recv_2(R,I, ikev1sig-2);
      send_3(I,R, ikev1sig-3-I);
      recv_4(R,I, ikev1sig-4-I);

      claim(I, Running, R, Ni, Nr, Ci, Cr);
      claim(I, Running, R, KEi, Gr);
      send_!5(I,R, ikev1sig-5-I);
      recv_!6(R,I, ikev1sig-6-I);

    }

  /*************************************************************/
  role R
    {
      /*************************************/
      // variables
```

```
        var Ni, Ci: Nonce;//Ci is I's cookie
        fresh Xr, Nr, Cr: Nonce;
        var Gi: Ticket; //g^Xi

        /*************************************/
        // sequence

        recv_1(I,R, ikev1sig-1);
        send_2(R,I, ikev1sig-2);
        recv_3(I,R, ikev1sig-3-R);
        send_4(R,I, ikev1sig-4-R);
        recv_!5(I,R, ikev1sig-5-R);

        claim(R, Running, I, Ni, Nr, Ci, Cr);
        claim(R, Running, I, Gi, KEr);
        send_!6(R,I, ikev1sig-6-R);

    }
}
```

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, SKEYID-I);
        claim(I, SKR, SKEYIDa-I);
        claim(I, SKR, SKEYIDe-I);
        claim(I, Weakagree);
        claim(I, Commit, R, Ni, Nr, Ci, Cr);
        claim(I, Commit, R, KEi, Gr);
// ロールＲのセキュリティ要件
```

```
    claim(R, SKR, SKEYID-R);
    claim(R, SKR, SKEYIDa-R);
    claim(R, SKR, SKEYIDe-R);
    claim(R, Weakagree);
    claim(R, Commit, I, Ni, Nr, Ci, Cr);
    claim(R, Commit, I, Gi, KEr);
```

3. Scyther による評価結果

3.1. 出力

Scyther v1.1 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。reflection attack の可能性が指摘された。IKEv1 Phase 1: authenticated with pre-shared encryption プロトコルでは、ペイロードにロール名などの通信相手を特定するための情報を含まないためである。したがって、通信相手を確認する実装であれば、この問題は生じない。

```
claim id [IKEv1-sig,I3], SKR(prf(Ni,Nr,DHg2(Gr,Xi))) : No attacks within bounds.
claim id [IKEv1-sig,I4], SKR(prf(prf(Ni,Nr,DHg2(Gr,Xi)),prf(prf(Ni,Nr,DHg2(Gr,Xi)),DHg2(Gr,Xi),Ci,Cr),DHg2(Gr,Xi),Ci,Cr)) : No attacks within bounds.
claim id [IKEv1-sig,I5], SKR(prf(prf(Ni,Nr,DHg2(Gr,Xi)),prf(prf(Ni,Nr,DHg2(Gr,Xi)),prf(prf(Ni,Nr,DHg2(Gr,Xi)),DHg2(Gr,Xi),Ci,Cr),DHg2(Gr,Xi),Ci,Cr),DHg2(Gr,Xi),Ci,Cr)) : No attacks within bounds.
claim id [IKEv1-sig,I6], Weakagree : No attacks within bounds.
claim id [IKEv1-sig,I7], Commit(R,Ni,Nr,Ci,Cr) : At least 2 attacks.
claim id [IKEv1-sig,I8], Commit(R,DHg1(Xi),Gr) : At least 2 attacks.
claim id [IKEv1-sig,R3], SKR(prf(Ni,Nr,DHg2(Gi,Xr))) : No attacks within bounds.
claim id [IKEv1-sig,R4], SKR(prf(prf(Ni,Nr,DHg2(Gi,Xr)),prf(prf(Ni,Nr,DHg2(Gi,Xr)),DHg2(Gi,Xr),
```

```
Ci,Cr),DHg2(Gi,Xr),Ci,Cr)) : No attacks within bounds.
claim                          id                          [IKEv1-sig,R5],
SKR(prf(prf(Ni,Nr,DHg2(Gi,Xr)),prf(prf(Ni,Nr,DHg2(Gi,Xr)),prf(prf(Ni,N
r,DHg2(Gi,Xr)),DHg2(Gi,Xr),Ci,Cr),DHg2(Gi,Xr),Ci,Cr),DHg2(Gi,Xr),Ci,Cr
)) : No attacks within bounds.
claim id [IKEv1-sig,R6], Weakagree  : At least 1 attack.
claim id [IKEv1-sig,R7], Commit(I,Ni,Nr,Ci,Cr)        :  At  least  1
attack.
claim id [IKEv1-sig,R8], Commit(I,Gi,DHg1(Xr))        :  At  least  1
attack.
```

3.2. 攻撃の解説

次ページの図1は、IKEv1-SIGにおいてロールIがデータの共有(non-injective agreement)を満たしていないことを表す例である。

Scyther Ver.1.1で攻撃グラフを生成すると、変数が多すぎるために図が見にくくなっている。ここでは Scyther Compromized Ver.0.8 を用いて攻撃グラフの生成を行った。２つのツール間で評価結果に違いはないが、Compromized版の攻撃グラフではマクロ定義を利用しているため、可読性が高くなっている。

攻撃グラフで注目すべきは、Run#1でAliceがロールIを演じているが、同時にAliceはロールRを演じていると思っている点である。すなわち、上図は reflection attack を表している。IKEv1 Phase 1: authenticated with signatures プロトコルでは、ペイロードにロール名などの通信相手を特定するための情報を含まない。したがって、通信相手を確認しないような実装であれば、Aliceは通信相手が自分自身であることを気にせず暗号プロトコルを終了してしまう。

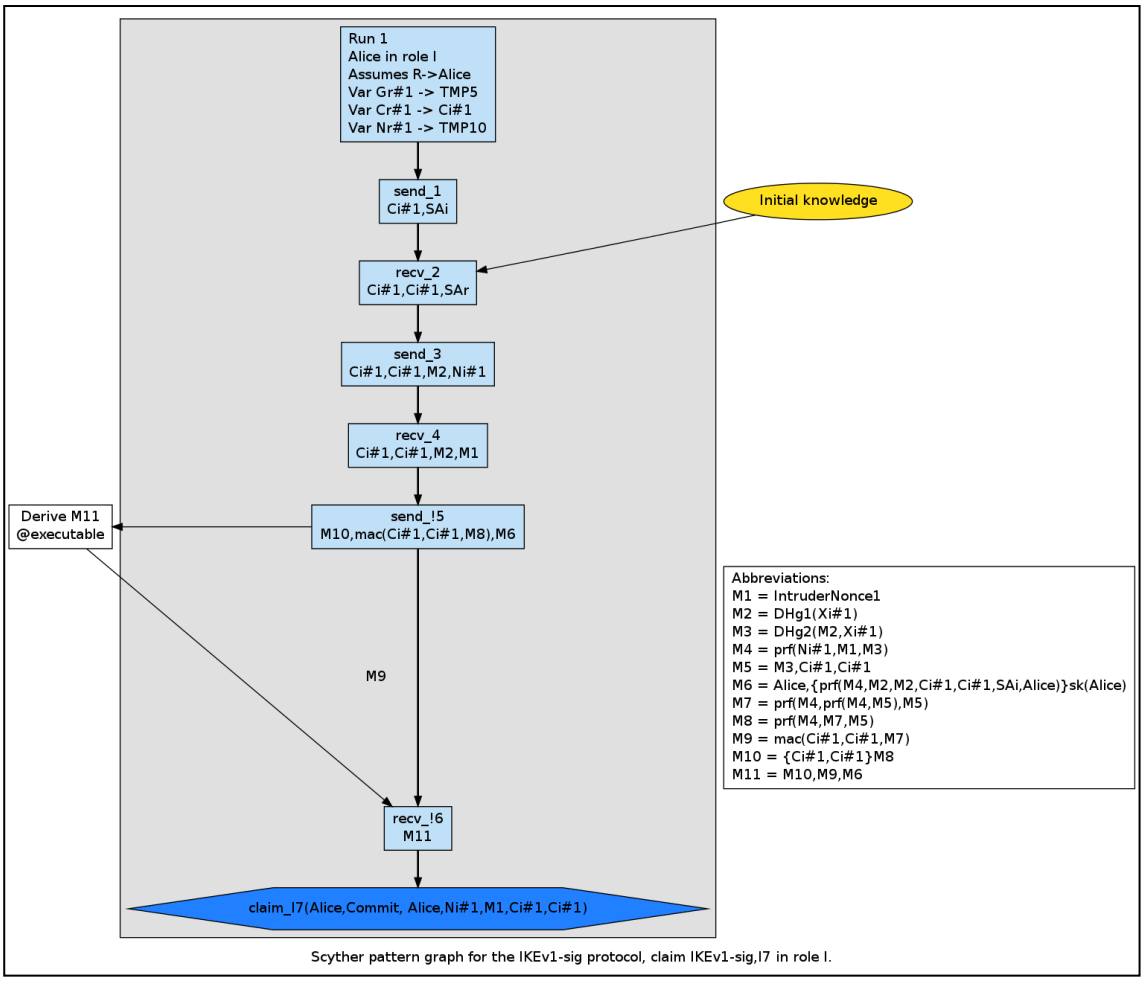

図.1　攻撃に関する解説

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情報はモデルから削除した。

Security parameter index, security association など、予測可能である値は定数とみなした。

IKE では Diffie-Hellman (DH)鍵交換を用いて鍵共有を行い、さらに共有したセッション鍵と事前共有情報（事前共有鍵、通信相手の公開鍵など）を用いて相互認証を行う。DH 鍵

交換は、指数演算処理が可換であることを利用しているが、Scyther では処理の可換性を記述することができない。このため、通信メッセージを途中で意図的に書き換え、送信パケットと受信パケットが異なるように記述することで、送信者、受信者にとって意味のあるデータとなるモデル化を行っている。

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るようなモデル化は行っていない。Scyther の開発者らは、DH 鍵交換を上記の記述のようにモデル化することを提案しているが、このモデル化がどの程度正確なのかについては分かっていない。ただし、既存の結果では、上記モデルで得られた結果と他のツールでの評価結果が食い違っているケースはないと思われる。

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（DH 鍵交換の形式化については、「方針」「妥当性」の項を参照)。

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。

ただし、Scyther では injective な性質について評価できない。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。そこで、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。

Scyther で記述できないプロセスはないが、上記のように DH 鍵交換のモデル化は直感的とはいえない。また、送受信データの対応がないようなモデル化を行っているため、そのままでは Scyther で評価することができない。このため、上記のモデルでは、データの対応付けを行う補助的な暗号プロトコル(helper protocol)を定義し、攻撃者による暗号プロトコルの実行をサポートすることで、評価を可能としている。これらの helper protocol のメッセージに関する記述が大量に発生するため、モデル記述が肥大化し、バグが混入しやすくなっている。

4.4. 検証ツール適用時の性能

検証時間は約 940 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。