Kerberos PKINIT 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

  Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)

- ✧ 機能

  信頼できる第三者機関（TTP, Trusted Third Party）の存在を前提とする、オープンなネットワークにおけるサーバ・クライアント間でのネットワーク認証・鍵交換プロトコル。特徴は公開鍵暗号を利用し、最初のリクエスト時にデータの一部に署名を付与すること。

- ✧ 関連する標準

  RFC4556 (https://tools.ietf.org/html/rfc4556)

2. Scyther の文法による記述

2.1. プロトコル仕様

```
usertype Time;
usertype String;
usertype SessionKey;
const hash: Function;
protocol Kerberos-PKINIT(C, A, G, S)
{
        role C
        {
        fresh U: String;
        fresh T2: Time;
        fresh T1: Time;
        fresh T0: Time;
        fresh N2: Nonce;
        fresh N1: Nonce;
```

```
        var Ktemp: Nonce;
        var Texpire2: Time;
        var Texpire1: Time;
        var Tstart2: Time;
        var Tstart1: Time;
                var Kcg, Kcs: SessionKey;
        var Tcg, Tcs: Ticket; //data that C does not check.

        send_1(C, A, (U, G, N1, {pk(C), T0, N1, hash(U, G, N1)}sk(C)));
        recv_2(A, C,
                (U, Tcg, {G, Kcg, Tstart1, Texpire1, N1}Ktemp,
           {{Ktemp}pk(C)}sk(A)));
        send_3(C, G, (S, N2, Tcg,  {C, T1}Kcg));
                recv_4(G, C, (U, Tcs, {S, Kcs, Tstart2, Texpire2,
N2}Kcg));
                send_5(C, S, (Tcs, {C, T2}Kcs));
        recv_6(S, C, {T2}Kcs);
        }

        role A
        {
        fresh Ktemp: Nonce;
        fresh Texpire1: Time;
        fresh Tstart1: Time;
        fresh Kcg: SessionKey;
        var U: String;
        var T0: Time;
        var N1: Nonce;

                recv_1(C, A, (U, G, N1, {pk(C), T0, N1, hash(U, G,
N1)}sk(C)));
        send_2(A, C,
```

```
            (U, {U, C, G, Kcg, Tstart1, Texpire1}k(A, G),
            {G, Kcg, Tstart1, Texpire1, N1}Ktemp,
            {{Ktemp}pk(C)}sk(A)));
}

role G
{
fresh Texpire2: Time;
fresh Tstart2: Time;
fresh Kcs: SessionKey;
var U: String;
var T1: Time;
var Texpire1: Time;
var Tstart1: Time;
var N2: Nonce;
var Kcg: SessionKey;

recv_3(C, G,
                    (S, N2,
                    {U, C, G, Kcg, Tstart1, Texpire1}k(A, G),
                    {C, T1}Kcg));
send_4(G, C,
            (U,
            {U, C, S, Kcs, Tstart2, Texpire2}k(G, S),
            {S, Kcs, Tstart2, Texpire2, N2}Kcg));
}

role S
{
            var U: String;
            var T2: Time;
var Texpire2: Time;
```

```
        var Tstart2: Time;
        var Kcs: SessionKey;

        recv_5(C, S,
                        ({U, C, S, Kcs, Tstart2, Texpire2}k(G, S),
                        {C, T2}Kcs));
        send_6(S, C, {T2}Kcs);
        }
}
```

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 C のセキュリティ要件
claim_C1(C, Secret, Kcg);
claim_C2(C, Secret, Kcs);
claim_C3(C, Secret, Ktemp);
claim_C4(C, Weakagree);
// ロール A のセキュリティ要件
claim_A3(A, Weakagree);
// ロール G のセキュリティ要件
claim_G1(G, Secret, Kcg);
claim_G2(G, Secret, Kcs);
claim_G3(G, Weakagree);
// ロール S のセキュリティ要件
claim_S1(S, Secret, Kcs);
claim_S2(S, Weakagree);
```

3. Scyther による評価結果

3.1. 出力

Scyther での評価結果では、セッション鍵の漏洩の可能性が指摘されているが、通常 Kerberos ではサーバ間の階層構造は固定されており、また、サーバが不正な行動をとることは想定されていない。そのため、サーバの不正が無い限りにおいては問題にならない。

```
claim id [Kerberos-PKINIT,C1], Secret(Kcg)    : At least 19 attacks.
claim id [Kerberos-PKINIT,C2], Secret(Kcs)    : At least 20 attacks.
claim id [Kerberos-PKINIT,C3], Secret(Ktemp)  : At least 19 attacks.
claim id [Kerberos-PKINIT,C4], Weakagree      : At least 1 attack.
claim id [Kerberos-PKINIT,A3], Weakagree      : At least 1 attack.
claim id [Kerberos-PKINIT,G1], Secret(Kcg)    : At least 11 attacks.
claim id [Kerberos-PKINIT,G2], Secret(Kcs)    : At least 11 attacks.
claim id [Kerberos-PKINIT,G3], Weakagree      : At least 12 attacks.
claim id [Kerberos-PKINIT,S1], Secret(Kcs)    : At least 11 attacks.
claim id [Kerberos-PKINIT,S2], Weakagree      : At least 12 attacks.
```

3.2. 攻撃の解説

以下の図 1 では、Kerberos PKINIT において、ロール C とロール S で共有されるセッション鍵 Kcs がロール S について Secret を満たさない例を示している。Kerberos プロトコルでは、サーバ間の階層構造は固定されており、また、サーバが不正な行動をとることは想定されていない。この結果は Kerberos の安全性がサーバの信頼性に依拠しているという設計者の意図に合致したものとなっており、通常の運用の際は問題ない。なお、図では攻撃者がロール A を演じている。ロール A はセッション鍵 Kcs を暗号化している鍵 Kcg を自分で生成できるので、ロール G が生成するセッション鍵 Kcs を入手できる。

4. 形式化

4.1. 方針

Kerberos では、チケットの有効期限を時刻情報で記述している。形式化では、Time という型を宣言し、時刻情報に関する変数を Time 型として記述した。

4.2. 妥当性

Kerberos では、サーバの不正行為を想定していないと思われる。Scyther では、攻撃者の能力を制限することができないため、仕様が目的としている安全性を評価できていない可能性がある。ただし、上述の結果では、Kerberos の安全性がサーバの信頼性に依拠しているという設計者の意図に合致したものとなっており問題ない。

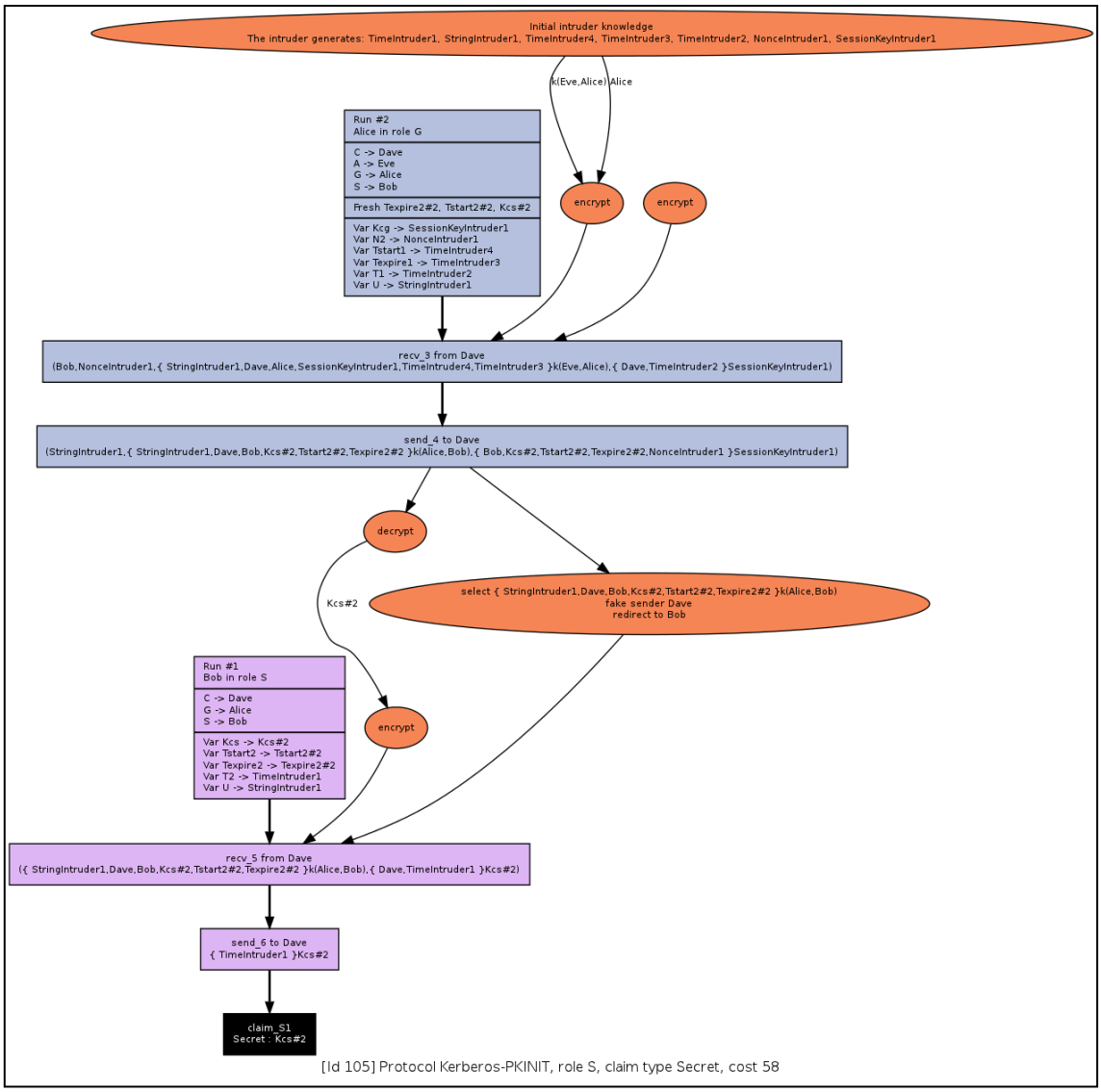

図. 1.　攻撃に関する解説

4.3. 検証ツールとの相性

プロトコル仕様の記述にあたって、形式化では時刻情報について Time という型を宣言しているが、時刻の整合性をチェックしているわけではなく、基本的にナンスとして取り扱われる。

セキュリティ要件を記述するにあたって、暗号プロトコルの目的は、サーバによるクライアントの片側認証及びセッション鍵（チケット）の交換である。したがって、Kcs に関する秘匿性及びクライアント - サーバ間で non-injective agreement が成立していれば良いと思われる。また、本プロトコルの基となる RFC4120 中では、時刻情報を盛り込むことで、リプレイ攻撃を防げるとしている。したがって、クライアント - サーバ間で injective

agreement が満たされていることが望ましい。

Scyther では、injectivity を評価することができない。しかし、評価の結果、Kerberos PKINIT は Weakagreement を満たさないため、これが問題となることはなかった。

4.4. 検証ツール適用時の性能

検証時間は約 5 時間だった。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。