Kerberos cross realm version 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 Kerberos Network Authentication Service (V5), Kerberos cross realm version

- ✧ 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。特徴は異なる管理ドメイン間で認証情報を共有可能であること。暗号として共通鍵暗号を利用。

- ✧ 関連する標準

  RFC4120 (https://www.ietf.org/rfc/rfc4120.txt)

2. Scyther の文法による記述

2.1. プロトコル仕様

```
usertype Time;
usertype SessionKey;
protocol Kerberos-cross-realm(C, AS-l, TGS-l, TGS-r, S-r)
{
        role C
        {
                fresh T3: Time;
                fresh T2: Time;
                fresh T1: Time;
                fresh N3: Nonce;
                fresh N2: Nonce;
                fresh N1: Nonce;
                var Texpire3: Time;
```

```
                var Texpire2: Time;
                var Texpire1: Time;
                var Tstart3: Time;
                var Tstart2: Time;
                var Tstart1: Time;
                var Kcgsl: SessionKey; // C <-> TGS_local
                var Kcgsr: SessionKey; // C <-> TGS_remote
                var Kcsr: SessionKey;  // C <-> S_remode
                var Tck1, Tck2, Tck3: Ticket;  // Data that role C
does not check
                send_1(C, AS-l, (C, TGS-l, N1));
                recv_2(AS-l, C,
                        (C, Tck1,
                        {TGS-l, Kcgsl, Tstart1, Texpire1, N1}k(C,
AS-l)));
                send_3(C, TGS-l, (TGS-r, N2, Tck1, {C, T1}Kcgsl));
                recv_4(TGS-l, C,
                        (C, Tck2, {TGS-r, Kcgsr, Tstart2, Texpire2,
N2}Kcgsl));
                send_5(C, TGS-r, (S-r, N3, Tck2, {C, T2}Kcgsr));
                recv_6(TGS-r, C, (C, Tck3, {S-r, Kcsr, Tstart3,
Texpire3}Kcgsr));
                send_7(C, S-r, (Tck3, {C, T3}Kcsr));
                recv_8(S-r, C, {T3}Kcsr);
        }
        role AS-l
        {
                fresh Texpire1: Time;
                fresh Tstart1: Time;
                fresh Kcgsl: SessionKey; // C <-> TGS_local
                var N1: Nonce;
                recv_1(C, AS-l, (C, TGS-l, N1));
```

```
                send_2(AS-l, C,
                        (C,
                        {C, TGS-l, Kcgsl, Tstart1, Texpire1}k(AS-l,
TGS-l),
                        {TGS-l, Kcgsl, Tstart1, Texpire1, N1}k(C,
AS-l)));
        }
        role TGS-l
        {
                fresh Texpire2: Time;
                fresh Tstart2: Time;
                var Texpire1: Time;
                var T1: Time;
                var N2: Nonce;
                var Tstart1: Time;
                var Kcgsl: SessionKey; // C <-> TGS_local
                fresh Kcgsr: SessionKey; // C <-> TGS_remote

                recv_3(C, TGS-l,
                        (TGS-r, N2,
                        {C, TGS-l, Kcgsl, Tstart1, Texpire1}k(AS-l,
TGS-l),
                        {C, T1}Kcgsl));
                send_4(TGS-l, C,
                        (C,
                        {C,     TGS-r,     Kcgsr,     Tstart2,
Texpire2}k(TGS-l, TGS-r),
                        {TGS-r,    Kcgsr,    Tstart2,    Texpire2,
N2}Kcgsl));
        }
        role TGS-r
        {
```

```
                fresh Texpire3: Time;
                fresh Tstart3: Time;
                var Texpire2: Time;
                var T2: Time;
                var N3: Nonce;
                var Tstart2: Time;
                var Kcgsr: SessionKey; // C <-> TGS_remote
                fresh Kcsr: SessionKey; // C <-> S_remote
                recv_5(C, TGS-r,
                        (S-r, N3,
                        {C,     TGS-r,     Kcgsr,     Tstart2,
Texpire2}k(TGS-1, TGS-r),
                        {C, T2}Kcgsr));
                send_6(TGS-r, C,
                        (C,
                        {C, S-r, Kcsr, Tstart3, Texpire3}k(TGS-r,
S-r),
                        {S-r, Kcsr, Tstart3, Texpire3}Kcgsr));
        }
        role S-r
        {
                var Texpire3: Time;
                var T3: Time;
                var Tstart3: Time;
                var Kcsr: SessionKey; // C <-> S_remote
                recv_7(C, S-r,
                        ({C, S-r, Kcsr, Tstart3, Texpire3}k(TGS-r,
S-r),
                        {C, T3}Kcsr));
                send_8(S-r, C, {T3}Kcsr);
        }
}
```

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, Kcgsl);
claim_C2(C, Secret, Kcgsr);
claim_C3(C, Secret, Kcsr);
claim_C4(C, Alive);
claim_C5(C, Weakagree);
// ロール AS－l のセキュリティ要件
claim_ASL2(AS-l, Alive);
claim_ASL3(AS-l, Weakagree);
// ロール TGS-ｌのセキュリティ要件
claim_TGSL1(TGS-l, Secret, Kcgsl);
claim_TGSL3(TGS-l, Alive);
claim_TGSL4(TGS-l, Weakagree);
// ロール TGS-ｒのセキュリティ要件
claim_TGSR1(TGS-r, Secret, Kcgsr);
claim_TGSR2(TGS-r, Secret, Kcsr);
claim_TGSR3(TGS-r, Alive);
claim_TGSR4(TGS-r, Weakagree);
// ロール S-r のセキュリティ要件
claim_SR1(S-r, Secret, Kcsr);
claim_SR2(S-r, Alive);
claim_SR3(S-r, 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-cross-realm,C1], Secret(Kcgsl)      :  No  attacks
```

```
within bounds.
claim id [Kerberos-cross-realm,C2], Secret(Kcgsr)      :  No  attacks
within bounds.
claim id [Kerberos-cross-realm,C3], Secret(Kcsr)       :  No  attacks
within bounds.
claim id [Kerberos-cross-realm,C4], Alive     :  No  attacks  within
bounds.
claim id [Kerberos-cross-realm,C5], Weakagree : At least 1 attack.
claim id [Kerberos-cross-realm,ASL2], Alive   : Exactly 1 attack.
claim id [Kerberos-cross-realm,ASL3], Weakagree        :  Exactly  1
attack.
claim id [Kerberos-cross-realm,TGSL1], Secret(Kcgsl)   : At least 10
attacks.
claim id [Kerberos-cross-realm,TGSL3], Alive  : At least 12 attacks.
claim id [Kerberos-cross-realm,TGSL4], Weakagree       : At least 12
attacks.
claim id [Kerberos-cross-realm,TGSR1], Secret(Kcgsr)   : At least 10
attacks.
claim id [Kerberos-cross-realm,TGSR2], Secret(Kcsr)    : At least 10
attacks.
claim id [Kerberos-cross-realm,TGSR3], Alive  : At least 12 attacks.
claim id [Kerberos-cross-realm,TGSR4], Weakagree       : At least 12
attacks.
claim id [Kerberos-cross-realm,SR1], Secret(Kcsr)      : At least 10
attacks.
claim id [Kerberos-cross-realm,SR2], Alive    : At least 12 attacks.
claim id [Kerberos-cross-realm,SR3], Weakagree         : At least 12
attacks.
```

3.2. 攻撃の解説

以下の図では、Kerberos cross realm version において、鍵 Kcsr がロール TGS-1 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 2 つのロールが持っている配役表（水色と紫色の、それぞれ一番上のノードの 2 列目）でサーバの配役が全く異なっていることに注意すべきである。すなわち、攻撃者は、2 つの Kerberos システムを行き来しながら、Dave が演じるロール TGS-l に自らが入手した鍵 Kcsr（モデル記述上では鍵 Kcgsl に当たる）を秘密鍵だと思い込ませている。

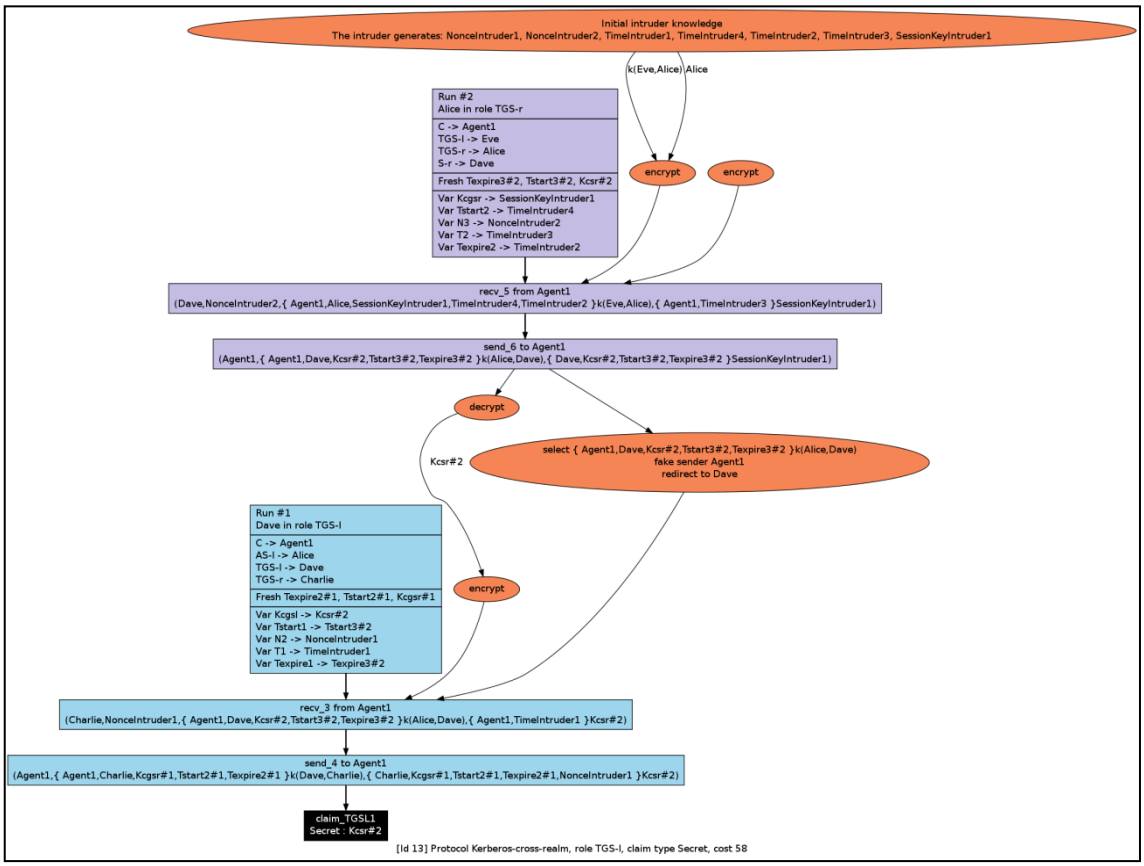

図. 1.　攻撃に関する解説

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。

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）の交換であることより、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、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 cross realm version 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 4 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。