Verificarea protocoalelor de autentificare de grup prin Scyther

Andrei Cristian
andrei.cristian1@info.uaic.ro

May 19, 2021

Introducere

Lucrarea "Verifying Group Authentication Protocols by Scyther", Huihui Yang, Vladimir Oleshchuk, and Andreas Prinz(University of Agder, Kristiansand, Norway) prezinta analiza a doua protocoale complexe de autentificare de grup folosind Scyther.
Din cauza limitarii utilitarului, doar un subset de proprietati de securitate au fost verificate:

- autentificare mutuala;
- autentificare cu cheie implicita [1];
- siguranta impotriva atacurilor de impersonare si adversarilor pasivi.

[1] proprietatea in care una dintre parti este asigurata ca nicio alta parte in afara de o a doua parte identificata in mod specific nu poate avea acces la o anumita cheie secreta

Scyther

- Pentru verificarea securitatii protocoalelor exista doua abordari principale: securitatea demonstrabila (eng. *provable security*) si metodele formale (eng. *formal methods*).
- **Scyther** este un utilitar de verificare formala si este conceput pentru verificarea automata a protocoalelor de securitate.
- Modelul adversarial este predefinit si anume modelul Dolev-Yao. Aceasta abordare simplifica formalizarea protocoalelor de securitate si il face mai usor de folosit pentru utilizatorii noi.
- Poate oferi clase de comportament de protocol spre deosebire de doar urmele de atac (furnizate in cazul altor utilitare).

Protocoale de autentificare de grup

- Scopul principal este imbunatatirea eficientei autentificarii pentru grupuri mari
- Relatia dintre autentificator si utilizatorii care urmeaza sa fie autentificati este unu la unu.
- In acest tip de protocol de autentificare de grup, autentificatorul **poate autentifica mai multi utilizatori in acelasi timp**.
- Daca in protocol autentificarea are acces:
  - autentificarea mutuala ar trebui sa fie satisfacuta;
  - se va stabili o cheie de sesiune de grup.

Ce urmareste lucrarea

- Extinderea lucrarii [1];
- In lucrarea mentionata, marimea grupului era de 3, iar in aceasta lucrare sunt analizate cazurile pentru grupuri ce contin doi, trei si patru membri.
- Formalizarea protocoalele bazate pe DLP[2] de tipurile I si II (tip II = autentificatorul are certificat bazat pe PKI[3]; tip I = autentificatorul nu are certificat) cand numarul de membri din grup este $N(N \geq 3)$.
- Analiza unor noi proprietati ale protocoalelor, precum "Alive" si "Nisynch"

[2] discrete logharitm problem
[3] public key infrastructure

Scenarii de utilizare

1.
   - Asa cum este prezentat in figura 8, autentificatorul de tip I are o lista de prieteni, dar membrii din aceasta lista se pot sau nu cunoaste intre ei.
   - De fiecare data inainte de intalnirea grupului, autentificatorul intai selecteaza membrii grupului si apoi trebuie sa autentifice fiecare membru din acest grup.
   - Cum toti membrii s-au inregistrat deja ca prieteni ai autentificatorului, presupunem ca acestia partajeaza niste secrete cu autentificatorul inainte de autentificare.

Scenariul de utilizare pentru protocoale de tip I

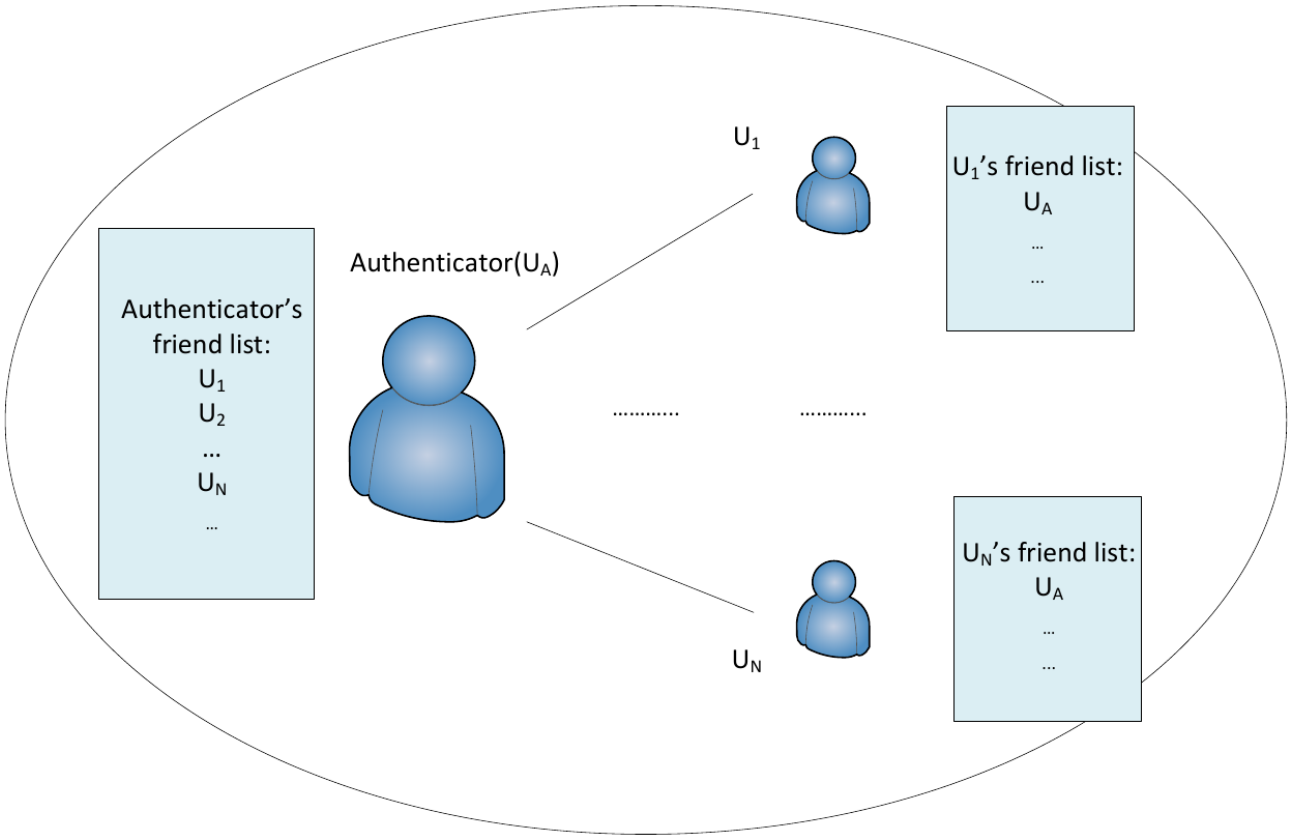

Figure: Scenariul de utilizare 1: Tipul I

Scenarii de utilizare

2.
   - In protocoalele de tip II (figura 10), autentificatorul este un server.
   - Trebuie sa autentifice utilizatori pe care nu ii cunoaste neaparat dinainte.
   - In acest caz, serverul trebuie sa detina un certificat pentru a realiza autentificarea grupului.

Scenariul de utilizare pentru protocoale de tip II

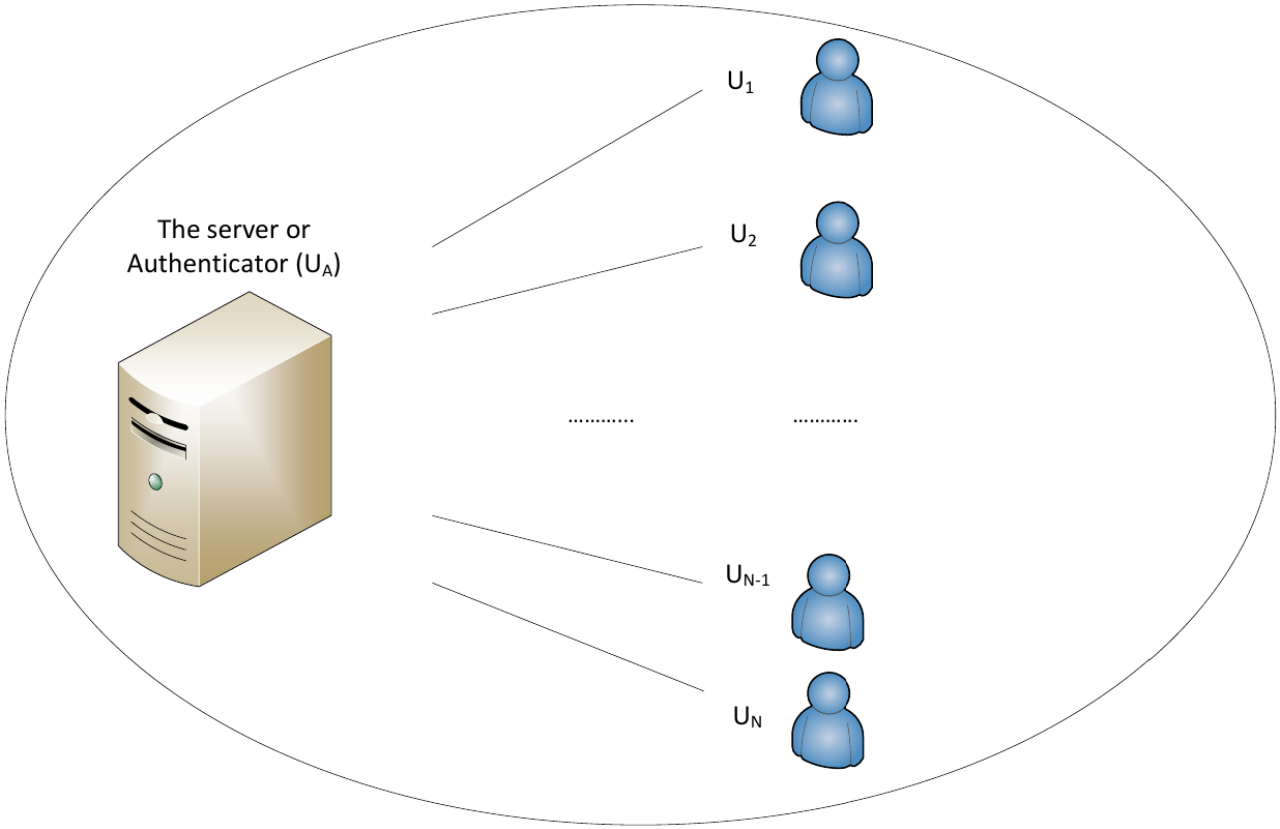

Figure: Scenariul de utilizare 2: Tipul II

Un framework general

Presupunem ca sunt N membri in grupul de utilizatori $\mathbb{U}$. Fluxul de mesaje al frameworkului general propus in [1] poate fi descris in urmatorii patru pasi:

1. $U_A \rightarrow U_1 : ID_A, UID, X, C_0, MAC_A$.
2. $U_i \rightarrow U_{i+1} : ID_i, UID, X, KP_U, C_i, MAC_i$, unde $1 \leq i \leq N-1$.
3. $U_N \rightarrow U_A : ID_N, KP_U, C_N, MAC_N$.
4. $U_A \rightarrow \mathbb{U} : Y, MAC'_A$.

Pentru $j \in \{A, N, U_i\}$, ID_j este identitatea lui j, UID este setul de identitati al tuturor utilizatorilor din $\mathbb{U}$, X este o informatie importanta pe care U_A vrea sa o transmita la tot grupul, C_k este utilizat pentru a calcula C_{k+1}, pt. $k = 0..N-1$, MAC_j este codul de autentificare al mesajului, KP_U este setul de parametri cheie al grupului de utilizatori $\mathbb{U}$, iar Y contine parametrii cheie generati de $\mathbb{U}$.

Protocoale bazate pe problema logaritmului discret I

Calcularea parametrilor $C_i(0 \leq i \leq N)$, X si Y ai protocolului bazat pe DLP pentru ambele tipuri Tipul I si Tipul II.

1. C_0 este calculat de catre U_A prin $C_0 = \xi(r) = \xi(g_A^r)$, unde $r_A \in [1, p-1]$ este un numar generat aleator, ξ este un mesaj ce va fi criptat prin algoritmul de criptare Elgamal. Similar, $U_i(2 \leq i \leq N)$ calculeaza $C_i = C_{i-1} \times r^{x_i} = \xi(r^{\sum_{t=1}^{i} x_t})$.
2. X este calculat ca solutie a $X \equiv V_i mod k_i (1 \leq i \leq N)$, folosind teorema chineza a resturilor (CRT), unde k_i este un secret partajat intre U_A si U_i.

Protocoale bazate pe problema logaritmului discret II

- Tipul I: $V_i = \{y_i \oplus K_G, y_i \oplus t_i, g^{m_i}, h_i\}$, iar $h_i = H(ID_A \oplus ID_i \oplus y_A \oplus t_i)$ si este folosit pentru autentificarea lui U_A cu U_i. Aici, y_i este un secret predistribuit intre U_A si U_i, K_G este cheia de sesiune a grupului generata de U_A, t_i este un nonce si g^{m_i} este parametrul cheie generat de U_A pentru a calcula cheia partajata intre U_A si U_i.
- Tipul II: $V_i = SIGN_{SK_A}\{ID_A, ID_i, K_G, G^{m_i}, t_i\}$. Parametrii g^{m_i} si t_i au aceeasi semnificatie ca in cazul Tipului I. Autentificarea lui U_A este realizata prin verificarea folosind semnatura sa in loc de utilizarea lui h_i ca in cazul Tipului I.

3. U_A calculeaza Y prin rezolvarea $Y \equiv W_i mod k_i (1 \leq i \leq N)$, unde $W_i = \{ID_A, ID_i, KP_i\}$ si $KP_i = KP_U - \{G^{n_i}\}$.

Protocoale bazate pe problema logaritmului discret III

4. Cheia sesiunii dintre U_A si U_I este calculata ca find $g^{m_i n_j}$, in timp ce cheia sesiunii dintre U_i si U_j $(1 \leq i, j \leq N, i \neq j)$ este calculata ca fiind $g^{n_i n_j}$.

Modelul adversarului in Scyther I

- Modelul adversarului in Scyther este predefinit si se bazeaza pe modelul *Dolev-Yao* [2].
- Nu trebuia sa formalizam abilitatile adversarului cand analizam protocoale.
- Adversarul (notat cu **A**) poate **intercepta mesaje** de pe canalul de comunicare si poate **invata** din mesajele pe care le are.
- Presupunem ca M este setul de cunostinte al adversarului si f este o functie prin care se exprima relatiile intre diferite elemente din M.
- k poate reprezenta atat o cheie simetrica, dar si asimetrica, iar k^{-1} este inversul acesteia ($k^{-1} = k$ in cazul cheii simetrice).

Modelul adversarului in Scyther II

- Fie (t_i, t_j) reprezentarea concatenarii intre termenii t_i si t_j.
  - $t \in M \Rightarrow M \vdash t$: daca t este un element al lui M, atunci A cunoaste t.
  - $M \vdash (t_1, t_2) \Rightarrow \{M \vdash t_1, M \vdash t_2\}$: daca A cunoaste (t_1, t_2), atunci A cunoaste ambii termeni t_1 si t_2.
  - $\{M \vdash t_1, M \vdash t_2\}$ $RightarrowM \vdash (t_1, t_2)$: daca A cunoaste ambii termeni t_1 si t_2, atunci A cunoaste (t_1, t_2).
  - $\wedge_{1 \leq i \leq n} M \vdash t_i \Rightarrow M \vdash f(t_1, ..., t_n)$: daca A cunoaste toti $t_i (1 \leq i \leq n)$ si f este o functie publica, atunci A poate calcula rezultatul functiei f cu datele de intrare $t_1, ..., t_n$.
  - $\{M \vdash t, M \vdash k\} \Rightarrow M \vdash \{t\}_k$: daca A cunoaste mesajul t si cheia k, atunci A poate calcula mesajul criptat $\{t\}_k$.

Modelul adversarului in Scyther III

- $\{M \vdash \{t\}_k, M \vdash k^{-1}\} \Rightarrow M \vdash t$: daca A cnoaste mesajul criptat $\{t\}_k$ si cheia de decriptare k^{-1}, atunci A poate decripta criptotextul si obtine astfel plaintextul t.

- In plus, adversarul A poate sa stearga, sa creeze noi mesaje si sa le insereze in canalul de comunicare.

Specificarea cerintelor de securitate in Scyther I

- Evenimentul **match** poate fi folosit in doua moduri diferite:
  1. Specificarea constrangerilor de egalitate - de exemplu codurile dupa evenimentul $match(p_1, p_2)$ pot fi executate doar daca p_1 este egal cu p_2.
  2. Asemanator cu '=' din limbajul de programare **C**, daca p este o variabila, iar v este o valoare, atunci $match(p, v)$ semnifica asignarea valorii v variabilei p.
- **claim** este folosit pentru specificarea cerintelor de securitate **Alive, Nisynch, secret** si **commitment**.
  - **Alive** este o forma de autentificare care are ca scop asigurarea ca intr-adevar partea de comunicare destinata (R) a executat niste evenimente - $claim(R, \textbf{Alive})$.

Specificarea cerintelor de securitate in Scyther II

- **Nisynch** semnifica faptul ca toate mesajele primite de R sunt intr-adevar trimise de catre partenerul de comunicare (sender) si au fost primite de carte celalalt partener de comunicare (receiver) - $claim(R, \mathbf{Nisynch})$.
- $claim(R, \mathbf{secret}, rt)$ inseamna ca R pretinde ca termenul rt sa nu fie stiut de catre adversar.
- daca rt este o cheie de sesiune, folosim $claim(R, \mathbf{SKR}, rt)$ pentru a specifica acest lucru.
- **Commitment** este o promisiune a unei parti din comunicare catre alta parte. De exemplu, $claim(R, \mathbf{Commit}, R', t)$ inseamna ca rolul R face o promisiune t catre rolul R'. **Commitment** este folosit pentru a verifica protocoalele impotriva **atacurilor de uzurpare** (en. impersonation attacks).

Bibliografie

 H. Yang, V. Oleshchuk, and A. Prinz, "Verifying group authentication protocols by scyther," *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 7, pp. 3–19, 2016.

 D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983. DOI: 10.1109/TIT.1983.1056650.