DECAP-Distributed Extensible Cloud Authentication Protocol

Andrea Huszti and Norbert Oláh
University of Debrecen
Nijmegen, Netherlands

Contents

Entity authentication methods

Scientific literature:

One-factor authentication solutions:

- 2000 M.S. Hwang, L.H. Li: smart card based, ElGamal encryption, impersonation attack
- 2002 Chien, Jan and Tsien: password based, several attacks

Two-factor authentication solutions:

- 2011 Amlan Jyoti Choudhury, Pardeep Kumar and Mangal Sain: two-factor authentication, smart card+password, Out of Band channel (OOB), impersonation attack
- 2014 Nan Chen and Rui Jiang: correcting the impersonation attack, no OOB

Entity authentication methods

In practice:
OpenStack is one of the most popular cloud computing software.

- User+password
- Lightweight Directory Access Protocol (LDAP)
- Kerberos

Centralized structure of authentication

One server authentication vulnerability:

- One target
- Lower attack cost
- Centralized responsibility
- Equipment is cheaper.

Distributed authentication

- Need to attack multiple servers simultaneously
- Increasing the attack cost
- Shared responsibility
- Equipment is more expensive

Security requirements of cloud computing

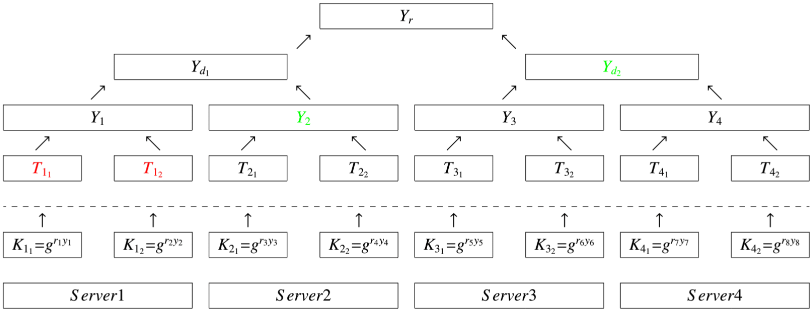
- Entity authentication
  Lack of strong authentication can lead to unauthorized access to users account on a cloud.
- Data integrity
  Data can be modified only by authorized parties.
- Secrecy, privacy
  Increased number of parties, devices and applications are involved, confidentiality of data should be protected against illegal access.
- Access control
  The data owner needs to make a flexible and scalable access control policy, so that only the authorized users can access.

Design's goals

Distributed Extensible Cloud Authentication Protocol

- Expand the participants of the server park
  Expansion algorithm for the Merkle tree
- Scalability
- Shared responsibility
- Two-factor authentication
  static password + one-time-password
- MAC key exchange
  providing data origin integrity
- Improving efficiency

Merkle-tree


Y_r
Y_{d_1}
Y_{d_2}
Y_1
Y_2
Y_3
Y_4
T_{1_1}
T_{1_2}
T_{2_1}
T_{2_2}
T_{3_1}
T_{3_2}
T_{4_1}
T_{4_2}
$K_{1_1}=g^{r_1y_1}$
$K_{1_2}=g^{r_2y_2}$
$K_{2_1}=g^{r_3y_3}$
$K_{2_2}=g^{r_4y_4}$
$K_{3_1}=g^{r_5y_5}$
$K_{3_2}=g^{r_6y_6}$
$K_{4_1}=g^{r_7y_7}$
$K_{4_2}=g^{r_8y_8}$
$Server1$
$Server2$
$Server3$
$Server4$

The proposed protocol

Registration

U	C_i	AS
ID, PW, X salt, i index	(y_i, z_i) secret key	
(r_i, s_i) secret	(g^{y_i}, g^{z_i}) public key	
(g^{r_i}, g^{s_i})		
$K_{i_1}=g^{r_i y_i}$, $K_{i_2}=g^{s_i z_i}$		
$T_i=(T_{i_1},T_{i_2})=(H(K_{i_1}), H(K_{i_2}))$		
building the Merkle tree		
$\xrightarrow{ID,(g^{r_i}, g^{s_i}),H(PW\|\|X\|\|i)}$	$K_i=(g^{r_i y_i}, g^{s_i z_i})$	
	$Y_i=H(H(K_{i_1})\|\|H(K_{i_2}))$	
	$\xrightarrow{Y_i,ID}$	Y_r, building the Merkle tree
	$\xleftarrow{Y_r,ID}$	
	$<ID,K_i,H(PW\|\|X\|\|i),Y_r>$	

Authentication

U		C_i
$v \in \{1, ..., i\}$ random gen.		$v \in \{1, ..., i\}$ random gen.
	$ID, T_{v_1}, Y_{d_1}, Y_{d_2}, H(Y_r \Vert H(PW \Vert X \Vert v))$ $\longrightarrow$	Checking:
		T_{v_1}
		$Y_{d_1}, Y_{d_2} \longrightarrow Y_r$ ver.
		$H(Y_r \Vert H(PW \Vert X \Vert v))$ ver.
	$\longleftarrow$ $H(SK), m$	SK= $H(Y_r \Vert T_{v_2})$, m rand.
		ID, SK, m storage
H(SK) verification		
	$ID, MAC(m, SK)$ $\longrightarrow$	
		MAC verification

Synchronization

U	C_v	AS
$K'_{v_1}=K_{v_1} * g=g^{r_v y_v+1}$	$K'_{v_1}=K_{v_1}*g$	
$K'_{v_2}=K_{v_2} * g=g^{s_v z_v+1}$	$K'_{v_2}=K_{v_2}*g$	
$T'_{v_1}=H(K'_{v_1})\ T'_{v_2}=H(K'_{v_2})$	$T'_{v_1}=H(K'_{v_1})\ T'_{v_2}=H(K'_{v_2})$	
T_v path update	$Y_v=H(H(T'_{v_1}) \Vert H(T'_{v_2}))$	
	$\xrightarrow{ID,Y_v}$	Y_r
	$\langle ID,K_v,H(PW \Vert X \Vert v),Y_r \rangle$ $\xleftarrow{Y_r}$	

Security analysis

Requirements:

- Authentication of the User
- Authentication of the Cloud server
- Secrecy of the key
- Key freshness
- Both parties should be assured that the other party knows the new key.

Model:

- Dolev-Yao adversary model: read, modify, delete, and inject messages (record communications, store values, synthesize messages etc.)

Security analysis

In case of an outsider adversary:

- Applied pi-calculus
- Proverif 1.93: Automatic verifier for cryptographic protocols.
- Cryptographic protocols are concurrent programs which interact using public channels.
- An arbitrary number of protocol executions

Main processes

- The main process controls activites between the User and the Server subprocesses
- We execute the User and the Server Processes in parallel infinitely many times

```
process
new y1: exponent; new y2: exponent; new y3: exponent; new y4: exponent;
new id: bitstring; new idS: bitstring; new x: passx;
let S1 =exp(g, y1) in let S2 =exp(g, y2) in let S3 =exp(g, y3) in let S4 =exp(g, y4) in
out(c,(S1,S2,S3,S4)); (*Public keys*)
new sskU:sskey;
new eskS:skey;
let spkU=spk(sskU) in out(c,spkU);
let epkS=pk(eskS) in out(c,epkS);
((!User(id,idS,S1,S2,S3,S4,sskU,spkU,epkS, x)) |
(!Server(id,idS,y1,y2,y3,y4,eskS,epkS,spkU)))
```

User process

```
(*User process*)
let User(id:bitstring, idS:bitstring,
S1U:G,S2U:G,S3U:G,S4U:G,sskU:sskey,spkU:spkey,epkS:pkey, S_synch:exponent)=

(*Authentication 1*)
let M1=H2((Y7,H_PW)) in
let USK=H2((Y7,TT2)) in
event User_auth_start(USK);
out(c,(id,TT1,Y5,Y6,M1));
in(c,(M2:bitstring,serverm:bitstring));
let E2=H2((USK)) in
if E2=M2 then
event Server_auth_end(USK);
let M3=mac(USK,serverm) in
out(c, (id,M3));

(*Synchronization*)
event User_sync_start;
(*Merkle-tree update*)
let T1_2=exp(T1,S_synch) in
let TT1_2= H(T1_2) in
let Y1_2= H2(TT1_2) in
let Y5_2= H2((Y1_2,Y2)) in
let Y7_2= H2((Y5_2,Y6)) in

(*Authentication 2*)
let M1_2=H2((Y7_2,H_PW)) in
let USK_2=H2((Y7_2,TT4)) in
event User_auth2_start(USK_2);
out(c,(id,TT3,Y5_2,Y6,M1_2));
in(c,(M2_2:bitstring,serverm_2:bitstring));
let CheckSK=H2((USK_2)) in
if CheckSK=M2_2 then
event Server_auth2_end(USK_2);
let M3_2=mac(USK_2,serverm_2) in
out(c, (id,M3_2)).
```

ProVerif events

Events

User		Server
	First Authentication	
Event User Auth Start		
	$\xrightarrow{ID, T_{v_1}, Y_{d_1}, Y_{d_2}, H(Y_r \| H(PW \| X \| v))}$	
		Event Server Auth Start
	$\xleftarrow{H(SK), m}$	
Event Server Auth end		
	$\xrightarrow{ID, MAC(m, SK)}$	
		Event User Auth end
	Synchronization	
Update Merkle-tree		Update Merkle-tree
	Second Authentication	
Event User Auth2 Start		
	$\xrightarrow{ID, T_{v_1}, Y_{d_1}*, Y_{d_2}, H(Y_r* \| H(PW \| X \| v))}$	
		Event Server Auth2 Start
	$\xleftarrow{H(SK*), m*}$	
Event Server Auth2 End		
	$\xrightarrow{ID, MAC(m*, SK*)}$	
		Event User Auth2 End

Security analysis

- Secrecy of the key and the password
  query attacker:SK.
  query attacker:pw.
- Authentication of the User and the Cloud server
  query sk:bitstring; inj-event(Server_auth_end(sk)) ==> inj-event(Server_auth_start(sk)).
  query sk:bitstring; inj-event(User_auth_end(sk)) ==> inj-event(User_auth_start(sk)).
  query sk:bitstring; inj-event(Server_auth2_end(sk)) ==> inj-event(Server_auth2_start(sk)).
  query sk:bitstring; inj-event(User_auth2_end(sk)) ==> inj-event(User_auth2_start(sk)).

- Key freshness
  – Yr changes after authentication - secret
  – T_{v1} changes - secret
  – v random
- Both parties should be assured that the other party knows the new key.
  If the following conditions occur, both parties know the new key:
  let E2=H2((USK)) in if E2=M2 then let M3=mac(USK,serverm)
  let Checkmac=mac(SK,serverm) in if M3=Checkmac then event second(SK).

Efficiency analysis

	Hash	Exp	Mult	Inv	Encryption/ Decryption	Interactions
Choudhury et al (2011)						
User	10		1	1		
Server	8	1				
Sum	**18**	**1**	**1**	**1**		**4**
Nan and Rui (2014)						
User	4+1				1	
Server	4+1				1	
Sum	**8+2**				**2**	**3**
Our						
User	3+1					
Server	3+1					
Sum	**6+2**					**3**

This work was supported by the construction EFOP-3.6.3-VEKOP-16. The project has been supported by the European Union, co-financed by the European Social Fund.

SZÉCHENYI 2020

HUNGARIAN GOVERNMENT

European Union
European Social Fund

INVESTING IN YOUR FUTURE

Thank you for your attention!