

Identification and Entity Authentication

Vivek Haldar

Vhaldar@ics.uci.edu

Outline of Talk

- Definitions
- Passwords
- Challenge-response techniques

Definition

- A *claimant* tries to show a *verifier* that the claimant is as declared
- Different from *message authentication*
 - Message authentication has no *timeliness*
 - Entity authentication happens in *real time*

A good identification scheme is...

- ***Sound***: an honest party can successfully authenticate herself
- ***Non-transferable***
- ***No impersonation***
- All this is true even when
 - A large number of authentications are observed
 - Eve is able to spoof/eavesdrop
 - Multiple instances are run simultaneously

Basis of identification

- Something *known* - passwords, PINs, keys...
- Something *possessed* - cards, handhelds...
- Something *inherent* - biometrics

Passwords - weak authentication

- Usually fixed
- Stored either in the clear, or "encrypted" with a OWF
- *Rules* reduce the chance of easy passwords
- *Salt* increases search space for a dictionary attack
- *Pass **phrases*** - more security

Attacks on password schemes

- ***Replay*** of fixed passwords
- Exhaustive ***search***
 - 8 character password has 40-50 bits
- More directed ***dictionary attacks***
 - Crack - widely available tool for doing this`

UNIX passwords

- User password serves as key to encrypt known plaintext (64 bit zeroes)
- Encryption - modification of DES, iterated 25 times
- 12 bit salt added - total $64 + 12 = 76$ bits
 - Salt taken from system clock
 - Alters expansion function of DES

PINs and keys

- Long key on physical device (card), short PIN to remember
- PIN unlocks long key
- Need possession of both card and PIN
- Provides *two-level* security

One time passwords

- Avoids *replay attacks*
- ***Shared lists*** - pre-distribute list
- ***Sequentially updated*** - create next password while entering current password
- ***Based on one way functions*** - Lamport's scheme...

Lamport's One Time Passwords

- User has a secret w
- Using a OWF h , create the password sequence:

$$w, h(w), h(h(w)), \dots, h^t(w)$$

- Bob knows only $h^t(w)$
- Password for i^{th} identification is:

$$w_i = h^{t-i}(w)$$

Attacks on OTPs..

- ***Pre-play attack*** - Eve intercepts an unused password and uses it later
- Make sure you're giving password to the right party
- Bob must be *authenticated*

Another one-time password scheme

- Stores actual passwords on system side
- Alice and Bob share a password P
- Alice: generate r ,
send to Bob: $(r, h(r, P))$
- Check: Bob computes $h(r, P)$, from given r , and local copy of P .
- Works only if r is something that will only be accepted once (else replay attack!)

Challenge-response authentication

- Alice is identified by a *secret* she possesses
- *Bob* needs to know that Alice does indeed possess this secret
- *Alice* provides ***response*** to a time-variant ***challenge***
- Response depends on ***both*** secret and challenge

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Challenge Response using Symmetric Key Encryption

- Alice and Bob share a key K
- ***Unidirectional*** authentication using ***timestamps***
- ***Unidirectional*** authentication using ***random numbers***
- ***Mutual*** authentication using ***random numbers***

Unilateral authentication using timestamps

- Alice \rightarrow Bob: $E_K(t_A, B)$
- Bob decrypts and verified that timestamp is OK
- Parameter B prevents replay of same message in $B \rightarrow A$ direction

Unilateral authentication using random numbers

- Bob \rightarrow Alice: r_b
- Alice \rightarrow Bob: $E_K(r_b, B)$
- Bob checks to see if r_b is the one it sent out
 - Also checks " B " - prevents reflection attack
- r_b must be ***non-repeating***

Mutual authentication using random numbers

- Bob \rightarrow Alice: r_b
- Alice \rightarrow Bob: $E_K(r_{a'}, r_{b'}, B)$
- Bob \rightarrow Alice: $E_K(r_{a'}, r_b)$
- Alice checks that $r_{a'}$, r_b are the ones used earlier

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Challenge-response based on keyed OWFs

- Instead of encryption, used keyed MAC h_K
- Check: compute MAC from *known quantities*, and check with message
- SKID2 (unilateral), and SKID3 (mutual)

Mutual authentication using keyed MAC – SKID3

- Bob \rightarrow Alice: r_b
- Alice \rightarrow Bob: $r_{a'}, h_K(r_{a'}, r_{b'}, B)$
- Bob \rightarrow Alice: $h_K(r_{a'}, r_{b'}, A)$

Unilateral authentication using keyed MAC – SKID2

- Bob \rightarrow Alice: r_b
 - Alice \rightarrow Bob: $r_{a'}, h_K(r_{a'}, r_{b'}, B)$
-
- Same as SKID3 without last exchange

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Authentication based on public key decryption

Witness to chosen random r

Challenge to Alice – encrypted with her public key

- Bob \rightarrow Alice: $h(r), B, P_A(r, B)$
- Alice \rightarrow Bob: r

Alice decrypts challenge to get r . Checks with $h(r)$. Sends r back for Bob to check.

Mutual Authentication based on PK decryption

- Alice \rightarrow Bob: $P_B(r_{A'}, B)$
- Bob \rightarrow Alice: $P_A(r_{A'}, r_B)$
- Alice \rightarrow Bob: r_B

Challenge-response authentication

Using

- Symmetric encryption
- One way functions
- Public key encryption
- Digital signatures

Unilateral Authentication using Signatures

Alice \rightarrow Bob: $cert_A, t_A, B, S_A(t_A, B)$

Bob checks:

- Timestamp OK
- Identifier "B" is its own
- Signature is valid (after getting public key of Alice using certificate)

Unilateral Authentication using Signatures

Bob \rightarrow Alice: r_B

Alice \rightarrow Bob: $cert_A, r_A, B, S_A(r_A, r_B, B)$

Bob checks:

- Identifier "B" is its own
- Signature is valid (after getting public key of Alice using certificate)
- Signed r_A prevents chosen-text attacks

Mutual Authentication using Signatures

Bob \rightarrow Alice: r_B

Alice \rightarrow Bob: $cert_{A'}, r_{A'}, B, S_A(r_{A'}, r_{B'}, B)$

Bob \rightarrow Alice: $cert_{B'}, A, S_B(r_{A'}, r_{B'}, A)$

What I didn't cover

- Zero knowledge identification – another talk

Thank You!