Lecture 20

Entity Authentication Protocols

Nicola Laurenti December 4, 2020

Ok

Lecture 20— Contents

General model

Weak authentication protocols

Strong authentication protocols

General model for entity authentication

An entity A (called the prover) wants to prove his identity to another entity B (called the verifier), typically through an iteractive protocol. At the end of the protocol, the entity B must decide whether he trusts A (accept) or not (reject)

Attack scenario

Masquerade A malicious F wants to pose as A while interacting with B

Requirements

Correctness If A is honest, B accepts hime with high probability

Security / robustness (to false provers) If the prover is not honest (i.e., it is some F posing as A) it is hard for F to be accepted

Non transferability (against malicious verifiers) Even after the protocol has taken place between A and B it is hard for B to pose as A in an exchange with another entity C and be accepted

Proofs of identity

Proof of identity in general can be based on:

- something that only A possesses: token, smart card
- something that only A is: face, voice, biometrics
- something that only A knows: PIN, passwords, secret keys

An entity authentication protocol is called mutual if it allows A and B to prove each one's identity to each other

We only consider protocols based on something that only A knows

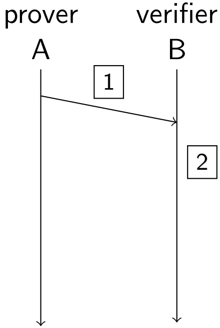
Password based protocols

A poassword based entity protocol is called strong if

- passwords are changed frequently
- it is hard to break with sequential guessing (brute force)
- the password is never transmitted or stored in the clear

If any of the above is not met, the protocol is called weak

Password authentication protocols



entities the prover A, the verifier B

setup A chooses a password $w_\mathsf{A} \in \mathcal{W}$ and securely delivers it to B
B stores a copy of $(\mathrm{id}_\mathsf{A}, w_\mathsf{A})$ in a database $\mathcal{D}$

$\boxed{1}$ $\mathsf{A} \to \mathsf{B}$: $u = (u_1, u_2) = (\mathrm{id}_\mathsf{A}, w_\mathsf{A})$

$\boxed{2}$ B : checks if $(u_1, u_2) \in \mathcal{D}$ and, if so, accepts A

Weaknesses

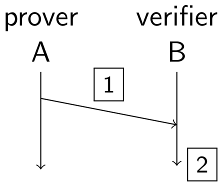
- transferability, since B learns w_A
- w_A is transmitted in the clear
- w_A is stored in the clear

Hashed password authentication protocols

entities the prover A, the verifier B

tools a hash function $h : \mathcal{W} \mapsto \mathcal{T}$

setup A chooses a password $w_\mathsf{A} \in \mathcal{W}$ and securely delivers it to B
B computes $t_\mathsf{A} = h(w_\mathsf{A})$ and stores a copy of $(\mathrm{id}_\mathsf{A}, t_\mathsf{A})$ in database $\mathcal{D}$



$\boxed{1}$ $\mathsf{A} \to \mathsf{B}$: $u = (u_1, u_2) = (\mathrm{id}_\mathsf{A}, w_\mathsf{A})$

$\boxed{2}$ B : computes $\tilde{t} = h(u_2)$ and checks if $(u_1, t) \in \mathcal{D}$ and, if so, accepts A

Improvement

- w_A no longer stored in clear

Weaknesses

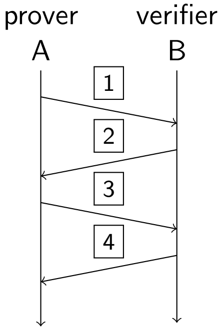
- transferability, as B learns w_A
- w_A still transmitted in clear
- a forger could carry on a 2nd preimage attack

Challenge-handshake authentication protocols

entities the prover A, the verifier B

tool a hash function $h : \mathcal{W} \times \mathcal{R} \mapsto \mathcal{X}$

setup A chooses a password $w_\mathsf{A} \in \mathcal{W}$ and securely delivers it to B



$\boxed{1}$ A $\rightarrow$ B : $u_1 = [\mathrm{id}_\mathsf{A}]$

$\boxed{2}$ B : generates a challenge $r \sim \mathcal{U}(\mathcal{R})$
B $\rightarrow$ A : $u_2 = [r]$

$\boxed{3}$ A : computes $x = h(w_\mathsf{A}, r)$
A $\rightarrow$ B : $u_3 = [x]$

$\boxed{4}$ B : retrieves w_A from $\mathcal{D}$
computes $x = h(w_\mathsf{A}, r)$
checks if $u_3 = x$ and, if so, accepts A

Observe that the password w_A is no longer transmitted in the clear, but it is still stored in the clear in $\mathcal{D}$

One time password protocols [Lamport, '81]

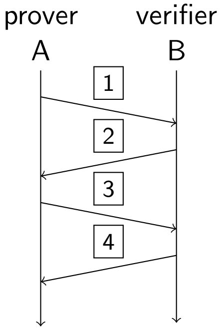
entities the prover A, the verifier B

tool two hash functions $h : \mathcal{X} \mapsto \mathcal{X}$ and $h' : \mathcal{W} \mapsto \mathcal{X}$

setup A chooses a password $w_{\mathsf{A}} \in \mathcal{W}$

A computes hash chain $x_N = h'(w_{\mathsf{A}}), x_{N-1} = h(x_N), \ldots, x_0 = h(x_1)$

A securely delivers x_0 to B



At the n-th protocol run, $n = 1, \ldots, N$

$\boxed{1}$ A $\rightarrow$ B : $u_1 = [\mathrm{id}_{\mathsf{A}}]$

$\boxed{2}$ B $\rightarrow$ A : $u_2 = [n]$

$\boxed{3}$ A $\rightarrow$ B : $u_3 = x_n$

$\boxed{4}$ B : computes $\tilde{x}_{n-1} = h(x_n)$
checks if $\tilde{x}_{n-1} = x_{n-1}$ and, if so, accepts A

Long-term w_{A} never stored or transmitted in clear

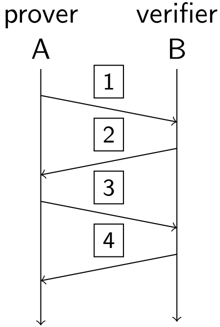
Temporary x_n transmitted in clear, but used only once

Vulnerable to man-in-the-middle attacks

Challenge-response protocols with symmetric A+IP

entities the prover A, the verifier B

tool a symmetric message A+IP mechanism, of the tag appending type with key k_A and tag function $T(\cdot,\cdot)$



$\boxed{1}$ A → B : $u_1 = \mathrm{id}_\mathsf{A}$

$\boxed{2}$ B : generates a random challenge $r \sim \mathcal{U}(\mathcal{R})$
B → A : $u_2 = r$

$\boxed{3}$ A : builds $u_3 = \mathrm{id}_\mathsf{A}, r$
signs $t_3 = T(k_\mathsf{A}, u_3)$
A → B : t_3

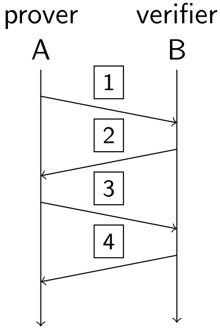
$\boxed{4}$ B : verifies whether $V(k_\mathsf{A}, u_3, t_3) = (r, \mathsf{ok})$ and, if so, accepts A

The challenge r must be changed at every run of the protocol, otherwise a dishonest prover F, pretending to be A, can replay $\boxed{1}$ and $\boxed{3}$ even without knowing k_A, and would be accepted

Challenge-response protocols with asymmetric A+IP

entities the prover A, the verifier B

tool a digital signature mechanism, with keys $k_{\mathsf{A}}, k'_{\mathsf{A}}$; a certificate c_{A} for k'_{A} and k'_{B}



$\boxed{1}$ A → B : $u_1 = \mathrm{id}_{\mathsf{A}}$

$\boxed{2}$ B : generates a random challenge $r_{\mathsf{B}} \sim \mathcal{U}(\mathcal{R})$
B → A : $u_2 = r_{\mathsf{B}}$

$\boxed{3}$ A : generates a random challenge $r_{\mathsf{A}} \sim \mathcal{U}(\mathcal{R})$
builds $u_3 = [\mathrm{id}_{\mathsf{B}}, r_{\mathsf{B}}, r_{\mathsf{A}}]$ signs $x_3 = S(k_{\mathsf{A}}, u_3)$
A → B : x_3

$\boxed{4}$ B : verifies whether $V(k'_{\mathsf{A}}, x_3) = ([\mathrm{id}_{\mathsf{B}}, r_{\mathsf{B}}, r_{\mathsf{A}}], \text{ok})$ and, if so, accepts A

The challenge r must be changed at every run of the protocol, otherwise an dishonest prover F, pretending to be A, can replay $\boxed{1}$ and $\boxed{3}$ even without knowing k_{A}, and would be accepted

Zero-knowledge protocols [Goldwasser-Micali, '85]

A general formulation, due to [Maurer, '09]

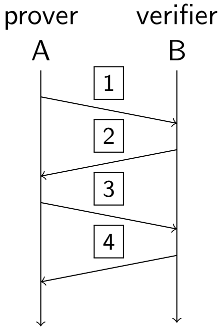
entities the prover A, the verifier B

tools two algebraic groups, $(\mathbb{G}, \circ)$ and $(\mathbb{H}, \star)$
the integer set $C = \{1, |\mathbb{G}| - 1\} \subset \mathbb{N}$
a function $f : \mathbb{G} \mapsto \mathbb{H}$ that is one-way and homomorphic, i.e.

$$f(x \circ y) = f(x) \star f(y) \quad , \quad \forall x, y \in \mathbb{G}$$

setup A generates a random $s_\mathsf{A} \sim \mathcal{U}(\mathbb{G})$
computes $t_\mathsf{A} = f(s_\mathsf{A}) \in \mathbb{H}$ and
securely delivers t_A to B

Zero-knowledge protocols



$\boxed{1}$ A $\rightarrow$ B : $u_1 = [\mathrm{id}_\mathsf{A}]$

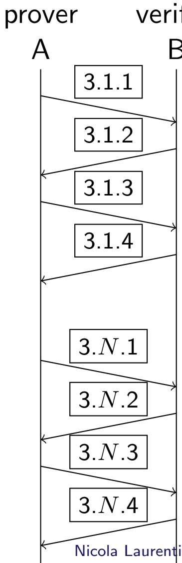
$\boxed{2}$ B $\rightarrow$ A : $u_2 =$ start

$\boxed{3}$ A $\leftrightarrow$ B : perform N iterations of the single check protocol

$\boxed{4}$ B : if all checks up to $n = N$ are correct, B accepts A

$f(x \circ y) = f(x) \star f(y)$

Zero-knowledge protocols, single check



$\boxed{3.n.1}$ A : generates a random $r_n \sim \mathcal{U}(\mathbb{G})$

computes $v_n = f(r_n)$

A $\rightarrow$ B : $u_{n,1} = [v_n]$

$\boxed{3.n.2}$ B : generates a random challenge $c_n \sim \mathcal{U}(C)$

B $\rightarrow$ A : $u_{n,2} = [c_n]$

$\boxed{3.n.3}$ A : computes $z_n = r_n \circ \left(s \overset{c_n}{\circ}\right)$

A $\rightarrow$ B : $u_{n,3} = [z_n]$

$\boxed{3.n.4}$ B : computes $y_n = f(z_n)$

checks whether $y_n = v_n \star \left(t \overset{c_n}{\star}\right)$, and if not, rejects A

A: r_n

$f(r_n)$

Summary

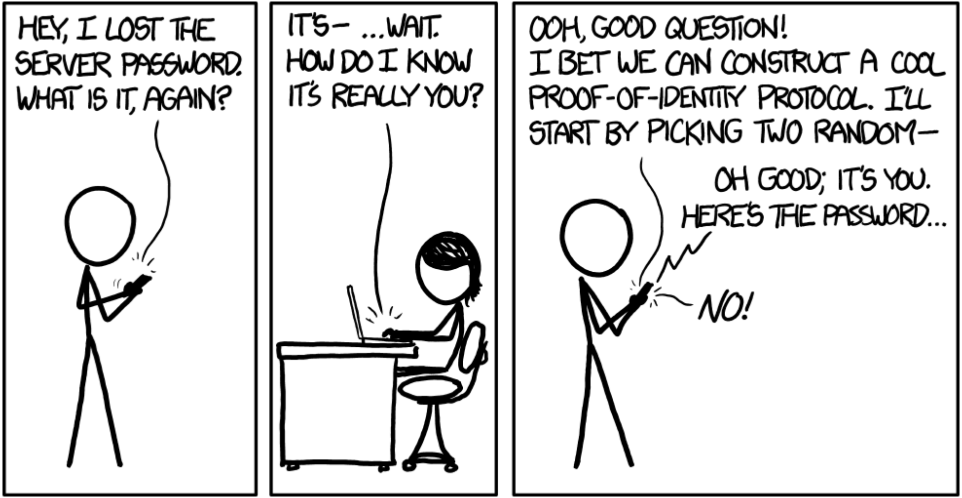
In this lecture we have:

- presented a general model for entity authentication adn described the target requirements
- introduced examples of weak password-based entity authentication protocols
- introduced examples of strong challenge-response entity authentication protocols

Assignment

- class notes
- textbook, §10.1 – §10.3

End of lecture



Identity, reproduced from xkcd URL: xkcd.com/1121