Outline

Basic ideas of authentication

Challenge-Response Authentication

Impersonation Attacks

What did we learn?

Recall from Lecture 1

Information security

- secrecy: "bad information flows don't happen"
- authenticity: "good information flows do happen"

In network computation

- all information flow constraints are security properties

Recall from Part 3

It is easy to generate a shared secret

Recall terminology

Information security

- confidentiality: "bad information flows don't . . . ."
- integrity: "good information flows do . . . ."

Although not synonymous

- secrecy, confidentiality and privacy
- authenticity and integrity

are often used interchangeably
Recall from Part 3

It is hard to know who is it shared with

\[
\begin{align*}
A & \rightarrow B \\
\text{what you know} & : \text{secrets, digital keys} \\
\text{what you have} & : \text{tokens, smart cards, physical keys} \\
\text{what you are} & : \text{biometric properties, handwriting} \\
\end{align*}
\]
Tools of authentication
In cyberspace¹ there are only messages...

▶ you have no biometric properties
▶ no smart cards
▶ you only know your secrets

¹space with no distance, inhabitants with no body, “Satan’s computer”

...authentication is just responding to challenges

▶ you must prove that you know your secrets
▶ without disclosing all of them

Everyone who knows all your secrets is you.

¹space with no distance, inhabitants with no body, “Satan’s computer”

Outline
Basic ideas of authentication

Challenge-Response Authentication
Challenge-Response protocols
Origination and freshness
Basic implementations of CR
Signatures
Implementing signatures
Other implementations
Mutual authentication
Authentication Server
Example: Yahalom protocol

First authentication protocol

Turing test

challenge

First authentication protocol

Turing test

challenge
First authentication protocol

The Challenge-Response Protocol Pattern

The Challenge-Response Protocol Pattern

The Challenge-Response Protocol Pattern

The Challenge-Response Protocol Pattern

The Challenge-Response Protocol Pattern

Notation

\[ \forall x \rightarrow \text{generate fresh nonce (into) } x \]
Remark
For simplicity, we are glossing over some subtle details.
E.g., Alice is often not capable to produce the term $r_{AB}^x$. How
does she verify that she has received a valid response to her
challenge?

She is given a verification algorithm $V_{AB}$

$$V_{AB}(y, x) \iff y = r_{AB}^x$$

We shall soon see an instance of this.
Origination axiom

\[ A : ((t)A) \implies \exists X. ((tX)A) \]  

\((\text{cv})\)

Freshness axiom

\[ AB \neq A \implies \nu x.a(x) \]
Freshness axiom

\[(v x)_A \land x \in FV(\alpha_B) \implies ((v x)_A \rightarrow \alpha_B) \land A \neq B \implies ((v x)_A \land (x))_A \land (x)_A \land \alpha_B) \quad (\text{new})\]

Challenge-Response with Signature

\[(\text{CRS}_B) = (\text{CR})_B[\nu x = x \cdot r_B x = S_B x] \]

Proposition

\((\text{CRS})\) is an implementation of \((\text{CR})\), for \(A \neq B\).

More precisely, if axioms \((\text{rcv})\) and \((\text{new})\) are satisfied, then

\[\text{(sig1)} \land \text{(sig2)} \land \text{(sig3)} \implies (\text{cr})_B[\nu x = x \cdot r_B x = S_B x] \]

whenever \(A \neq B\).

Proof

Suppose that Alice sees

\[ \{(y \mid \forall B (y, x))\} \]

where \(\{y \mid \forall B (y, x)\}\) means that \(y\) passes the test \(\forall B (y, x)\).
Proof (continued)

Since (sig3) tells \( y = S^a x \), we have

\[
A \leftarrow y \rightarrow B \quad (S^a x)
\]

Proof (continued)

By (rcv), everything that is received must have been sent.

\[
A \leftarrow (S^a x) \rightarrow B
\]

Proof (continued)

\[\text{... and by (sig2), } (S^a x) \rightarrow Y = B.\]

Proof (continued)

Since (sig1) implies \( x \in \text{FV}(S^a x) \), (new) implies

\[
A \leftarrow (S^a x) \rightarrow B
\]

Proof (completed)

\[\text{... and finally } A \neq B \text{ and the second part of (new) yield}\]

Simple signature system

Definition

Given the types

- \( M \) of plaintexts
- \( S \) of signatures
- \( K \) of keys
Simple signature system

Definition

...a simple signature system is a triple of algorithms:

- key generation \((K_s, K_v) : \mathcal{K} \times \mathcal{K}\)
- signing \(S : \mathcal{K} \times M \rightarrow S\), and
- verification \(V : \mathcal{K} \times S \times M \rightarrow \{0, 1\}\)

Example of a simple signature system: RSA

- \(M = \mathbb{C} = \mathbb{Z}_n\), where \(n = pq, p, q\) prime
- \(\mathcal{K} = \mathbb{Z}_\phi(n)\)
- \(K_s = d \quad \text{← private key}\)
- \(K_v = d^{-1} \mod \phi(n) \quad \text{← public key}\)
- \(S(d, m) = m^e \mod n\)
- \(V(e, s, m) \iff s^e = m \mod n\)

Probabilistic signature systems

Remark

While the signature verification condition defines the basic functionality of the signatures, the unforgeability condition is a logical approximation of the desired security.

Going beyond the simple signature systems, we refine the unforgeability condition to various probabilistic versions — just like the trapdoor encryption condition on crypto systems was refined to the various notions of secrecy.

Simple signature system

Definition

...that together provide

- signature verification:
  \[ V(K_v, s, m) \iff s = S(K_s, m) \]
- unforgeability:
  \[ \forall m, V(K_v, A(m), m) \implies A(m) = S(K_s, m) \]
  for all feasible attackers \(A : M \rightarrow S\)

Probabilistic signature systems

Remark

While the signature verification condition defines the basic functionality of the signatures, the unforgeability condition is a logical approximation of the desired security.

Going beyond the simple signature systems, we refine the unforgeability condition to various probabilistic versions — just like the trapdoor encryption condition on crypto systems was refined to the various notions of secrecy.
Example of a probabilistic signature system: El Gamal

Fix a finite field \( \mathbb{F} \) and \( g \in \mathbb{F} \).
\[
M = \mathbb{F} \\
S = \mathbb{F}^* \times \mathbb{F} \\
K = \mathbb{F}^* \times \mathbb{F} \\
R = \mathbb{F}^* \\
V(t, v_1, v_2, m) \iff (k^m \cdot c_1^v = g^m)
\]

Signature systems

Homework

Prove that the RSA system satisfies the signature verification and the unforgeability conditions. Which assumptions do you need?

Prove that the El Gamal system satisfies the signature verification condition. What is the role of the random seeds \( r \in \mathbb{K} \) in unforgeability?

CR with Public Key Encryption

\( CRFE = (CR)[c^{\mathbb{K}}s = E^P(A, x) \cdot r^{\mathbb{K}}x = x] \)

CR with Shared Key at the Input

\( CRKI = (CR)[c^{\mathbb{K}}s = K^{SA}(A, x) \cdot r^{\mathbb{K}}s = x] \)

CR with Shared Key at the Output

\( CRKO = (CR)[c^{\mathbb{K}}s = x \cdot r^{\mathbb{K}}x = K^{SA}(A, x)] \)
Mutual authentication

To establish a session, Alice and Bob authenticate each other.

Mutual authentication: (CRS<sub>0</sub>-seq)<sup>2</sup>

... binding the two authentications together...

Mutual authentication: (CRS<sub>0</sub>-nest)

... or Bob may respond lazily...

<sup>2</sup>This protocol is better known as (ISO-9897-3).
Mutual authentication: (CRS₀-nest)

...first authenticates Alice...

Mutual authentication: (CRS₀-nest)

...but the two authentications still need to be bound together.

Formalizing mutual authentication

Matching conversation records

We say that a protocol realizes mutual authentication if

▶ each of the participants can prove
▶ all participants’ send and receive actions

and all principals’ views of

▶ the content of the messages sent and received, and
▶ the ordering in which they were sent and received coincide (i.e. match).

Formalizing mutual authentication

Matching conversation records

We say that a protocol realizes mutual authentication if

▶ each of the participants can prove
▶ all participants’ send and receive actions
▶ except the last send-receive pair

and therefore

▶ if Alice’s view of what Bob said
▶ matches Bob’s view of Bob said,
▶ then Alice’s view of what Bob said is true.

Ditto for Bob.

Formalizing mutual authentication

Remark

Suppose that Alice’s and Bob’s views of their conversation match. This implies that their view of their conversation is true, because

▶ Alice’s view of what she said is true, and
▶ Bob’s view of what he said is true,

and therefore

▶ if Alice’s view of what Bob said
▶ matches Bob’s view of Bob said,
▶ then Alice’s view of what Bob said is true.
Matching conversations in $\text{(CRS}_0\text{-nest)}$

**Proposition**

Protocol $\text{(CRS}_0\text{-nest)}$ realises mutual authentication.

**Formal notion of honesty**

We say that a principal in a protocol is **honest** within a protocol if she acts according to the protocol.

Initially, Alice only sees her own actions:

\[
\begin{align*}
\text{Alice:} & \quad x \xrightarrow{u} (S_B(x, u)) \\
\text{Bob:} & \quad S_B(x, u) \xrightarrow{} (S_A(x, u))
\end{align*}
\]

Matching conversations in $\text{(CRS}_0\text{-nest)}$: Alice

By $(\text{cr})[r^\text{id} \rightleftharpoons \text{id}, x \leftarrow r^\text{id} = S^3]$ (proved earlier)
Matching conversations in (CRS₀-nest): Alice

But Bob is honest, so he only sends $S^0(x,y)$ for a fresh $y$.

Initially, Bob only sees his own actions:

Matching conversations in (CRS₀-nest): Alice

Finally, using $u = y$, derived before, Alice concludes:

Matching conversations in (CRS₀-nest): Bob

Initially, Bob only sees his own actions:

By (rcv), someone must have sent $u$.

Matching conversations in (CRS₀-nest): Alice

Alice has derived the ordering of her and Bob's actions:

Matching conversations in (CRS₀-nest): Bob

By the (cr)-axiom, he concludes that Alice must be on-line.
Matching conversations in (CRS₀-nest): Bob

Since she is honest, she acted according to the protocol:

Both Alice and Bob have thus recorded the following conversation

\[(v, x)_{A} < (x)_{A} < (x)_{B} < (y)_{B} < (y)_{A} < (v)_{A} < (v, x)_{A} < (v, x)_{B} < (v)_{A} < (v, x)_{A} < (v, x)_{B} < (y)_{B} < (y)_{A} < (v)_{A} < (v, x)_{A} < (v, x)_{B} < (y)_{B} < (y)_{A} < (v)_{A} < (v, x)_{A} < (v, x)_{B} \]

Their records match, and the mutual authentication is achieved.
Mutual authentication by (CRK)?

The main shortcoming of both (CRKO) and (CRKI) protocols is that Alice and Bob are required to share a secret $k_{AB}$ to run these protocols.

Mutual authentication by (CRK)?

This defeats the purpose of authentication, because generating a shared secret $k_{AB}$ is usually the whole point.

Authentication Server

An Authentication Server $S$ shares a symmetric key $k_{BS}$ with every principal $B$.

Authentication services

CRKI

$A$ $S$ $B$

$K_{BS}(x)$

$K_{AB}(A,x)$

$S$ authenticates $B$ using $K_{BS}m = E(k_{BS}, m)$

If $S$ is honest, then $A$ thus authenticates $B$.

Authentication services

CRKII

$A$ $S$ $B$

$K_{BS}(x)$

$K_{AB}(A,x)$

$A$ authenticates $S$, using $K_{AB}m = E(k_{AB}, m)$
Towards the Yahalom protocol

**Step 3: Composition**

<table>
<thead>
<tr>
<th>A</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
<td></td>
<td><code>y</code></td>
</tr>
</tbody>
</table>

```
K_{BS}(A.x), K_{BS}(A.y)

K_{AS}(B.x), K_{AS}(B.y)
```

**Step 4: Binding**

<table>
<thead>
<tr>
<th>A</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
<td></td>
<td><code>y</code></td>
</tr>
</tbody>
</table>

```
K_{BS}(A.x), K_{BS}(A.y)

K_{AS}(B.x), K_{AS}(B.y)
```

**Step 5: Key distribution**

<table>
<thead>
<tr>
<th>A</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
<td></td>
<td><code>y</code></td>
</tr>
</tbody>
</table>

```
K_{BS}(A.x.y), K_{BS}(B.x.y)

K_{AS}(B.x.y)
```

**Step 5: Key confirmation**

<table>
<thead>
<tr>
<th>A</th>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
<td></td>
<td><code>y</code></td>
</tr>
</tbody>
</table>

```
K_{BS}(A.x.y)

K_{AS}(B.x.y)
```

**Outline**

Basic ideas of authentication

Challenge-Response Authentication

Impersonation Attacks

Examples of impersonation

Attack on (CRS₀)

Attack on (CRS₀-nest)

What did we learn?

Recall from Part 1: CAPTCHA

```
Google

XKDr
```
Refining authentication to capture MitM attacks

The definition of authentication needs to be strengthened to capture not only

▶ the challenge and the response messages, but also
▶ principals’ intent to respond to a challenge.

(CRS₀) authentication

Here is the protocol (CRS₀), initiated by Bob.

We proved that it correctly implements (CR).

(CRS₀) authentication

But here is a Man-in-the-Middle attack on it.

...much easier with NFC phones!
**Security 4: Authentication**

**Basics**

**CR-reasoning**

**Impersonation**

**Examples**

**Solutions**

---

**Ping authentication in (CRS) 0**

We proved that from Bob's actions, it follows that Alice must have been online recently.

**Ping authentication in (CRS) 0**

We did not prove that from Bob's intent to challenge Alice follows Alice's intent to respond to Bob.

---

**No agreement in (CRS) 0**

We did not prove that from Bob's intent to challenge Alice follows Alice's intent to respond to Bob.

---

**Ping authentication in (CRS) 0**

But here is a Man-in-the-Middle attack on it. (CRS) 0 does not guarantee agreement about the identities.
Mutual authentication: (CRS₀-nest)

Here is a protocol that we proved secure, assuming that Alice and Bob are honest, and that they both know it.

```
A  ν
A to B  x
B to A  y
A to B  S₀(x, y)
B to A  S₀(x, y)
```

But here is what may happen if Alice tries to talk to Mallory, who is not honest.

```
A  ν
A to M  x
B  ν
B to A  y
M to A  x
A to M  S₀(x, y)
B to A  S₀(x, y)
M to A  S₀(x, y)
B to M  S₀(x, y)
M to B  S₀(x, y)
```

Moral

To avoid impersonation, always specify the participants of the challenge-response exchange in the protected message.

```
A  ν
A to B  x
B  ν
B to A  y
S₀(B, x)
```

One-way authentication with Signature

```
(CRS) = (CR) \{ c^{AB}x = x, r^{AB}x = S₀(A, x) \}
```

But

```
A  ν
A to B  x
B  ν
B to A  y
S₀(B, x)
```

Mutual authentication with Signatures

```
(CRS₀-seq) = (ISO-9798-3)
```

NOT
Mutual authentication with Signatures

(CRS-seq)

BUT

Mutual authentication with Signatures

(CRS₀-nest)

Solutions:

Mutual authentication with Signatures

(CRS₀-nest)

BUT

One-way authentication with Encryptions

(CREE₀)

Solutions:

One-way authentication with Encryptions

(CREE₀)}
Discussion

The definitions of

- one-way authentication in terms of the challenge-response pattern,
- mutual authentication in terms of the matching conversation records

still allow confusion about who is talking to whom.

Strong one-way authentication

Intended authentication

\begin{align*}
\text{A: } &c_{AB}(x) \\ \text{B: } &c_{AB}(x) \\
\end{align*}

\begin{align*}
\text{A: } &\leftarrow c_{AB}(x) \\ 
\text{B: } &\leftarrow c_{AB}(x) \\
\end{align*}

Strong mutual authentication

Agreement

Strong mutual authentication requires not only matching conversation records: all principals’ records of

- the content and
- the order of all messages must coincide, but also matching views of the intent: all principals’ views of

- the purported sources and
- the intended destinations of all messages should also coincide.
Proposition

The protocols (CRS), (CRS-seq) and (CRS-nest) all realize strong authentication.

Homework

Prove this.

Outline

Basic ideas of authentication

Challenge-Response Authentication

Impersonation Attacks

What did we learn?

Back to key setup

More on Authentication Servers

What has been achieved?

Secure key generation

...while avoiding the MiM-attacks?

Proposition

The protocols (CRS), (CRS-seq) and (CRS-nest) all realize strong authentication.

Secure key generation

Yes! Take (CRS-seq) for authentication...

Outline

Basic ideas of authentication

Challenge-Response Authentication

Impersonation Attacks

What did we learn?

Back to key setup

More on Authentication Servers

What has been achieved?

Secure key generation

...while avoiding the MiM-attacks?

Proposition

The protocols (CRS), (CRS-seq) and (CRS-nest) all realize strong authentication.

Secure key generation

Yes! Take (CRS-seq) for authentication...
Secure key generation

...and plug in (DHKA) for key agreement.

\[
\begin{align*}
A & \rightarrow B \\
& \rightarrow S \\
& \rightarrow B
\end{align*}
\]

Secure key generation

The signatures \( S \) are bound to their owners by certificates \( C \).

\[
\begin{align*}
A & \rightarrow B \\
& \rightarrow S \\
& \rightarrow B
\end{align*}
\]

where \( C^X = S^X(X^{\text{VX}}) \)

Bootstrapping authentication

Symmetric Key Authentication Servers

Authentication \( A \rightarrow B \) using symmetric keys is piped \( A \rightarrow S \rightarrow B \) through an Authentication Server \( S \).

Public Key Authentication Servers

Authentication \( A \rightarrow B \) using public keys goes directly, but an Authentication Server \( S \) must certify public keys in advance, and issue \( C^X \) and \( C^Y \).

A public key Authentication Server is often called a Certifying Authority (CA).

KDCs and CAs

Similarities

- An Authentication Server \( S \) shares a key with every principal \( A, B \) in its range.
- Authentication \( A \rightarrow B \) is bootstrapped over \( A \rightarrow S \) and \( S \rightarrow B \).
KDCs and CAs

**Similarities**

- An Authentication Server $S$ shares a key with every principal $A, B$ in its range.
- Authentication $A \rightarrow B$ is bootstrapped over $A \rightarrow S$ and $S \rightarrow B$.

**Differences**

- A KDC directly participates in every authentication session between every $A$ and $B$.
- A CA authenticates each $A$ in advance, and issues a certificate $C_A$, which can be used at any time, for any session with any $B$.

KDCs and CAs

**Disadvantages of KDC**

- can impersonate everyone to everyone
- single point of failure, performance bottleneck
- must be on-line, otherwise the network halts

**Disadvantage of CA**

- revocation
  - CA distributes Certificate Revocation Lists (CRL)
  - every certificate should be checked against CRL
  - often omitted

Secure key generation

...we get in the realm of practical protocols:

$$\begin{align*}
    E^{A}(u) &= E(k_{AG}, u) \\
    E^{A}(C, S(B, g^y)) &= E^{A}(C, S(B, g^y)) \\
    E^{A}(g^x, S(B, g^y)) &= E^{A}(g^x, S(B, g^y)) \\
    E^{A}(g^x, g^y) &= E^{A}(g^x, g^y) \\
    E^{A}(C, S(B, g^y)) &= E^{A}(C, S(B, g^y)) \\
    E^{A}(g^x, S(B, g^y)) &= E^{A}(g^x, S(B, g^y)) \\
    E^{A}(g^x, g^y) &= E^{A}(g^x, g^y)
\end{align*}$$

where $E^{A}(u) = E(k_{AG}, u)$
Secure key generation

Solution: Expand (CRS-nest) by (DHKA)

\[ g' = H(g^a g^b) \]

Secure key generation

If Bob is a busy CA, he can use cookies \( H_y \)...

where \( H_y = H(g^g) \)

Secure key generation

The core of IKEv2 (and JFK), the basic IPSec protocol:

\[ g' = H(g^a g^b) \]

where \( H_y = H(g^g) \)

Secure key generation

\[ \text{...just like before to} \]

\[ g' = H(g^a g^b) \]

where \( H_y = H(g^g) \)

Secure key generation

\[ \text{...and needn’t keep the state at all!} \]

\[ g' = H(g^g) \]
Summary: Questions of authentication

Why is it that

- it is easy to establish a secure channel, but
- it is hard to know with whom?

Old answer: Authentication is a deep problem

From local observations to global conclusions — through reflection

René to himself: “I think, therefore I exist.”

New answer: Authentication is a technical problem

From local observations to global conclusions — by cryptography

Alice to Bob: “No one else could decrypt this, therefore you exist.”

Authentication in Cyberspace

Assumptions

- the network is controlled by the Adversary
  - “Satan’s computer”
- the Adversary is computationally limited
  - the same algorithmics like everyone else

But computational limitations are relative to the available computational resources

Traveling Salesman Problem

- unfeasible for standard computers
- NP-hard for Turing machines
But computational limitations are relative to the available computational resources.

Traveling Salesman Problem

- easy for the ants in your yard
  - they use pheromones as a computational resource
  - pheromone evaporates at a steady rate
  - new paths are generated at random
  - each ant leaves a pheromone trail behind it
  - old paths are marked and amplified by pheromone
  - the stronger the marking, the more attractive the path
  - shorter paths become more attractive
  - shorter time for evaporation

Beyond Cyberspace

What if computation is not limited to cyberspace?

What if Alice, Bob, Mallory and Satan, besides computers, also use smart cards, mobile phones, fly planes, shoot guns and even talk to each other?

They do all that in pervasive computation. Next part.