Security and Trust II: Information Assurance
Section 4: Authentication and Key Distribution

Peter-Michael Seidel

February 27, 2017

Outline

Basic ideas of authentication

Challenge-Response Authentication

Impersonation Attacks

What did we learn?

Outline

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 terminology

Information security

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

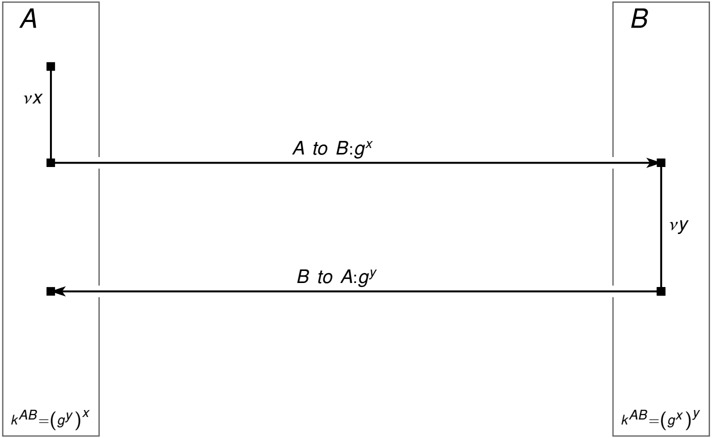
Although not synonymous

- secrecy, confidentiality and privacy
- authenticity and integrity

are often used interchanteably

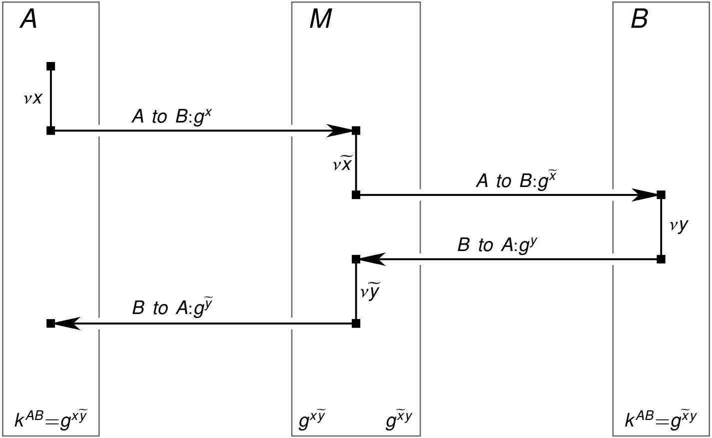
Recall from Part 3

It is easy to generate a shared secret



Recall from Part 3

It is hard to know who is it shared with



Problem of authentication

"On the Internet, nobody knows you're a dog."

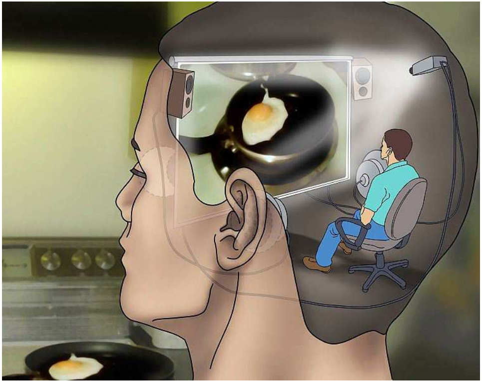
Problem of authentication

"There is no logical impossibility in the hypothesis that the world sprang into being five minutes ago, exactly as it then was, with a population that 'remembered' a wholly unreal past."

Bertrand Russell, The Analysis of Mind

Logics of authentication

Derive global facts from local observations



René Descartes: "I think, therefore I exist."

Tools of authentication

You authenticate yourself by leveraging over:

- what you know: secrets, digital keys
- what you have: tokens, smart cards, physical keys
- what you are: biometric properties, handwriting

Tools of authentication

You authenticate yourself by leveraging over:

- what you know: secrets, digital keys
  - can be copied and given away
- what you have: tokens, smart cards, physical keys
  - can be given away, but not copied
- what you are: biometric properties, handwriting
  - cannot be given away, or copied

Tools of authentication

In cyberspace[1] there are only messages...

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

[1] space with no distance, inhabitants with no body, "Satan's computer"

Tools of authentication

In cyberspace[1] there are only messages...

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

...authentication is just responding to challenges

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

[1] space with no distance, inhabitants with no body, "Satan's computer"

Tools of authentication

In cyberspace[1] there are only messages. . .

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

. . . 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.

[1] space with no distance, inhabitants with no body, "Satan's computer"

Outline

First authentication protocol

Turing
test

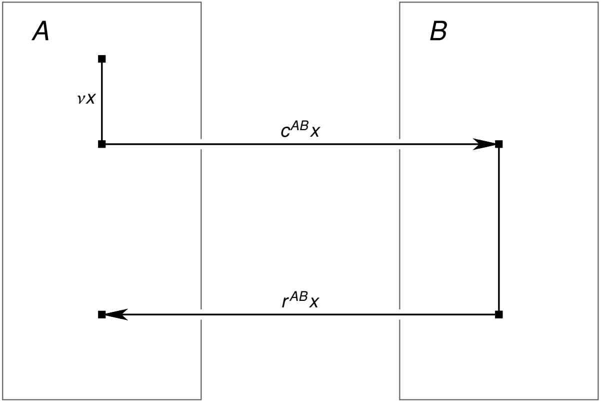
First authentication protocol

Turing
test

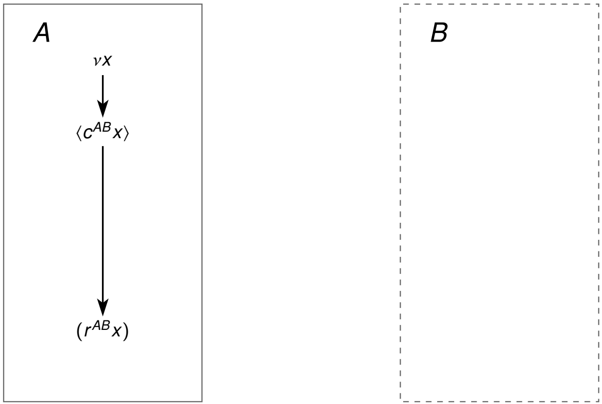
First authentication protocol

Turing test

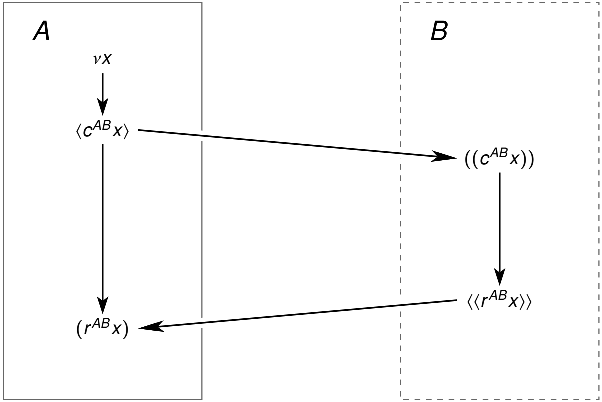
The Challenge-Response Protocol Pattern



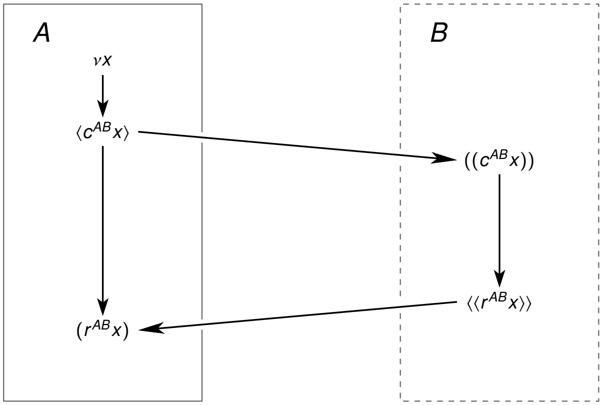
The Challenge-Response Protocol Pattern



The Challenge-Response Protocol Pattern



The Challenge-Response Protocol Pattern



$$A: \quad (\nu x)_A \triangleright \langle c^{AB} x\rangle_A \qquad \triangleright \qquad (r^{AB} x)_A$$

$$\Longrightarrow \Big((\nu x)_A \triangleright \langle c^{AB} x\rangle_A \triangleright ((c^{AB} x))_B \triangleright \langle\langle r^{AB} x\rangle\rangle_{\vec{B}} \triangleright (r^{AB} x)_A\Big) \qquad \text{(cr)}$$

The Challenge-Response Protocol Pattern

Notation

- νx — generate fresh nonce (into) x

The Challenge-Response Protocol Pattern

Notation

- νx — generate fresh nonce (into) x
- $\langle t \rangle$ — send a message t
  - $\langle\langle t \rangle\rangle$ — send a message containing t

The Challenge-Response Protocol Pattern

Notation

- νx — generate fresh nonce (into) x
- $\langle t \rangle$ — send a message t
  - $\langle\langle t \rangle\rangle$ — send a message containing t
- (t) — receive a message (into) t
  - $((t))$ — receive a message containing t

The Challenge-Response Protocol Pattern

Notation

- νx — generate fresh nonce (into) x
- $\langle t \rangle$ — send a message t
  - $\langle\langle t \rangle\rangle$ — send a message containing t
- (t) — receive a message (into) t
  - $((t))$ — receive a message containing t
- a_{Alice} — the action a is performed by *Alice*
  - $\langle\langle t \rangle\rangle_{\overrightarrow{Alice}}$ — *Alice* is the originator of t

The Challenge-Response Protocol Pattern

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?

The Challenge-Response Protocol Pattern

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}

$$\mathrm{V}^{AB}(y, x) \quad \Longleftrightarrow \quad y = r^{AB}x$$

The Challenge-Response Protocol Pattern

Remark

For simplicity, we are glossing over some subtle details.

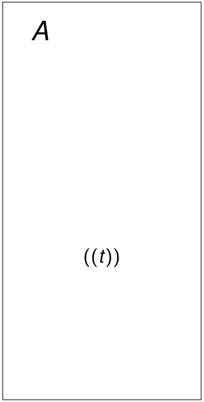
E.g., Alice is often not capable to produce the term $r^{AB}x$. How does she verify that she has received a valid response to her challenge?

She is given a *verification algorithm* V^{AB}

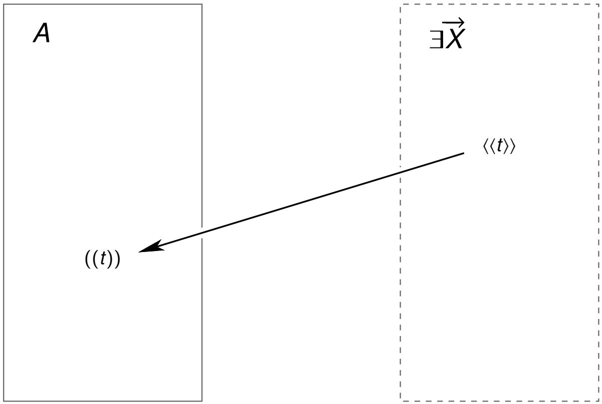
$$\mathsf{V}^{AB}(y, x) \quad \Longleftrightarrow \quad y = r^{AB}x$$

We shall soon see an instance of this.

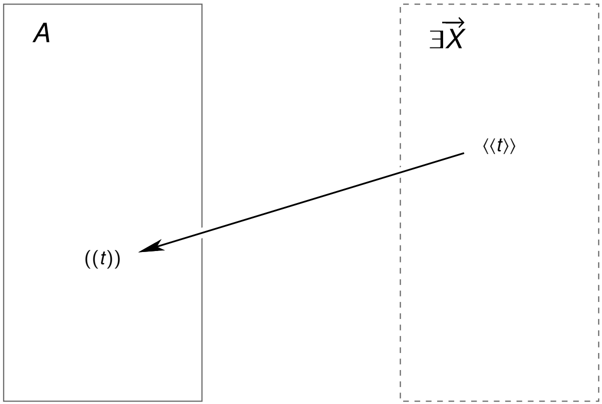
Origination axiom



Origination axiom

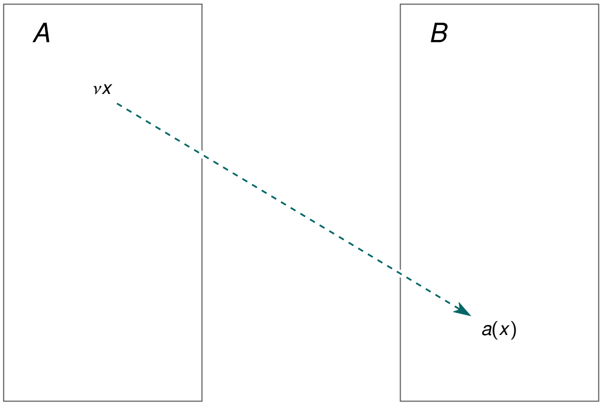


Origination axiom

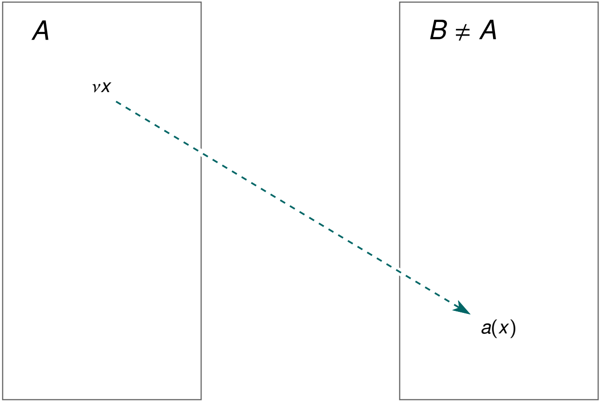


$$A:\ ((t))_A \implies \exists X.\ \langle\langle t \rangle\rangle_{\vec{X}} \triangleright ((t))_A \qquad \text{(rcv)}$$

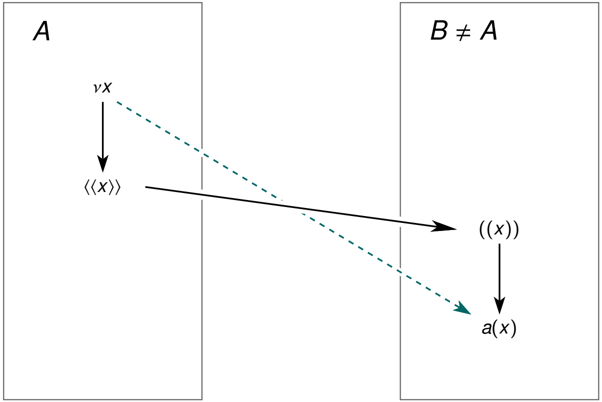
Freshness axiom



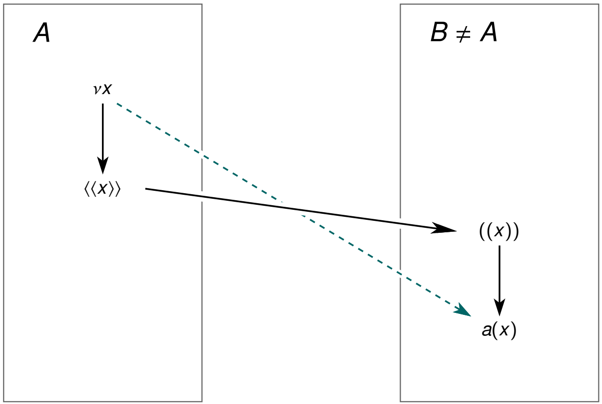
Freshness axiom



Freshness axiom



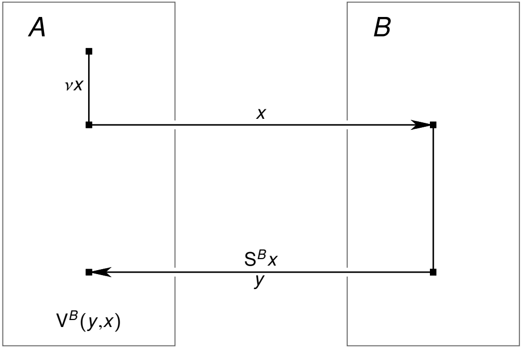
Freshness axiom



$$
\begin{aligned}
(\nu x)_A \;\wedge\; x \in FV(a_B) &\implies \Big(\big((\nu x)_A \qquad\qquad \rhd \qquad\qquad a_B\big) \\
\wedge \quad A \neq B &\implies \big((\nu x)_A \rhd \langle\langle x\rangle\rangle_A \rhd ((x))_B \rhd a_B\big)\Big) \quad \text{(new)}
\end{aligned}
$$

Challenge-Response with Signature

$(\mathrm{CRS}_0) = (\mathrm{CR})\left[c^{AB}x = x\ ,\ r^{AB}x = \mathrm{S}^B x\right]$



$$\mathrm{S}^B t = \mathrm{S}^B u \implies t = u \qquad \text{(sig1)}$$

$$\langle\langle \mathrm{S}^B t \rangle\rangle_{\vec{X}} \implies X = B \qquad \text{(sig2)}$$

$$\mathrm{V}^B(y, t) \iff y = \mathrm{S}^B t \qquad \text{(sig3)}$$

Challenge-Response with Signature

$(\mathrm{CRS}_0) = (\mathrm{CR})\left[c^{AB}x = x \ , \ r^{AB}x = \mathrm{S}^{B}x\right]$

Proposition

(CRS) is an implementation of (CR), for $A \neq B$.

Challenge-Response with Signature

$(\text{CRS}_0) = (\text{CR})\left[c^{AB}x = x \ , \ r^{AB}x = \text{S}^Bx\right]$

Proposition

(CRS) is an implementation of (CR), for $A \neq B$.

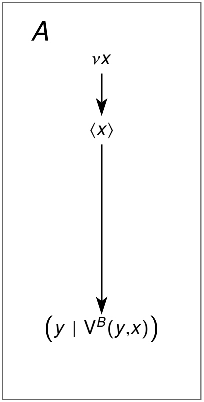
More precisely, if axioms (rcv) *and* (new) *are satisfied, then*

$$(\text{sig1}) \wedge (\text{sig2}) \wedge (\text{sig3}) \quad \Longrightarrow \quad (\text{cr})_{[c^{AB}x=x \ , \ r^{AB}x=\text{S}^Bx]}$$

whenever $A \neq B$.

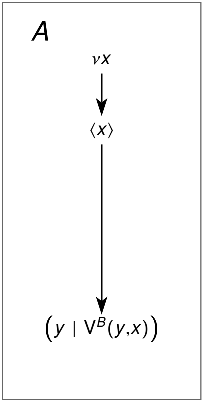
Proof

Suppose that Alice sees



Proof

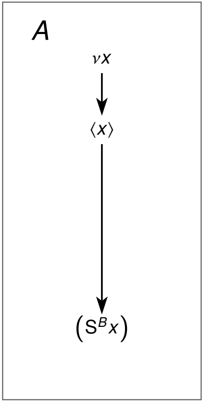
Suppose that Alice sees



where $\big(y \mid \mathsf{V}^B(y,x)\big)$ means that y passes the test $\mathsf{V}^B(y,x)$.

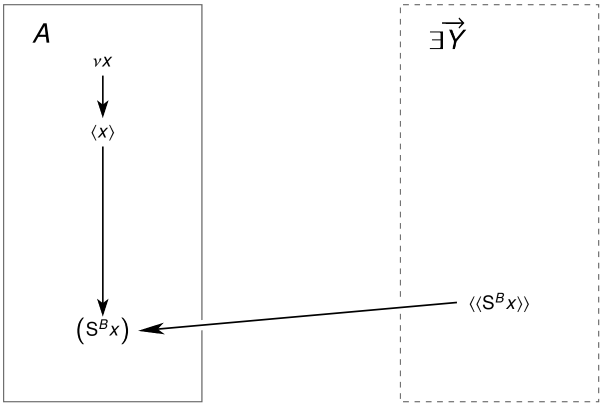
Proof (continued)

Since (sig3) tells $\mathrm{V}^B(y, x) \iff y = \mathrm{S}^B x$, we have



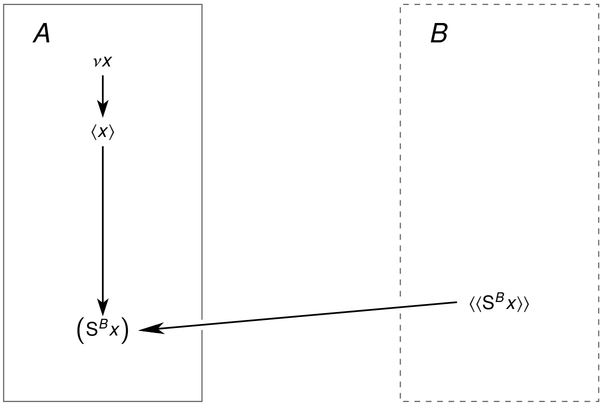
Proof (continued)

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



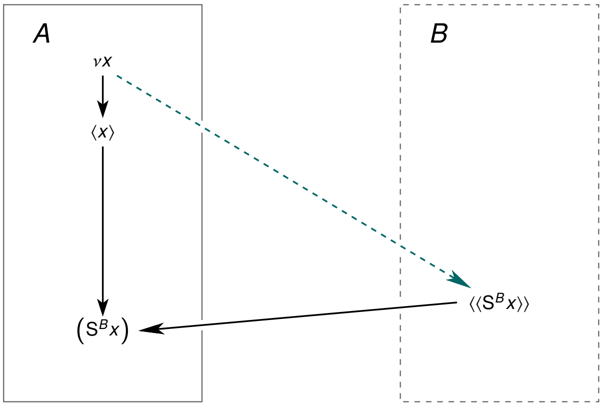
Proof (continued)

$\ldots$ and by (sig2), $\langle\langle S^B t\rangle\rangle_{\vec{Y}} \Longrightarrow Y = B$.



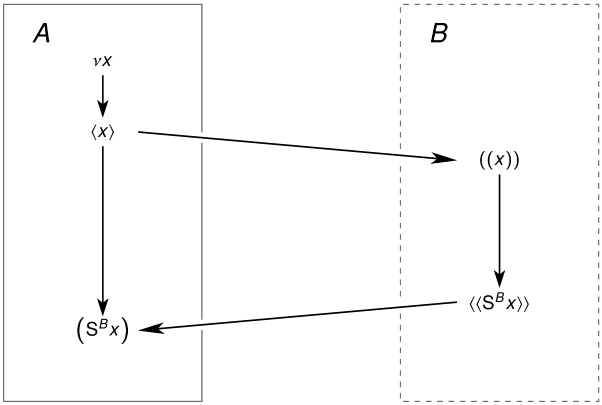
Proof (continued)

Since (sig1) implies $x \in FV\langle\langle S^B x\rangle\rangle$, (new) implies



Proof (completed)

. . . and finally $A \neq B$ and the second part of (new) yield



Simple signature system

Definition

Given the types

- $\mathcal{M}$ of *plaintexts*
- $\mathcal{S}$ of *signatures*
- $\mathcal{K}$ of *keys*

Simple signature system

Definition

…a simple signature system is a triple of algorithms:

- key generation $\langle \mathrm{K_S}, \mathrm{K_V} \rangle : \mathcal{K} \times \mathcal{K}$
- signing $\mathrm{S} : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{S}$, and
- verification $\mathrm{V} : \mathcal{K} \times \mathcal{S} \times \mathcal{M} \longrightarrow \{0, 1\}$

Simple signature system

Definition

...that together provide

- signature verification:

$$\mathsf{V}(\mathsf{K}_\mathsf{V}, s, m) \iff s = \mathsf{S}(\mathsf{K}_\mathsf{S}, m)$$

- unforgeability:

$$\left(\forall m.\ \mathsf{V}\left(\mathsf{K}_\mathsf{V}, \mathsf{A}(m), m\right)\right) \implies \mathsf{A}(m) = \mathsf{S}(\mathsf{K}_\mathsf{S}, m)$$

for all feasible attackers $\mathsf{A} : \mathcal{M} \longrightarrow \mathcal{S}$

Example of a simple signature system: RSA

- $\mathcal{M} = \mathcal{C} = \mathbb{Z}_n$, where $n = pq$, p, q prime
- $\mathcal{K} = \mathbb{Z}_{\varphi(n)}$
- $\mathrm{K_S} = d$ $\rightsquigarrow$ **private** key
- $\mathrm{K_V} = d^{-1} \mod \varphi(n)$ $\rightsquigarrow$ **public** key
- $\mathrm{S}(d, m) = m^d \mod n$
- $\mathrm{V}(e, s, m) \iff s^e = m \mod n$

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.

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

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 $g \in \mathbb{F}^*$.

$$\mathcal{M} = \mathbb{F} \qquad \mathsf{K}_\mathsf{V}(a) = g^a$$

$$\mathcal{S} = \mathbb{F}^* \times \mathbb{F} \qquad \mathsf{K}_\mathsf{S}(a) = a$$

$$\mathcal{K} = \mathbb{F}^* \times \mathbb{F}^* \qquad \mathsf{S}(r, \overline{k}, m) = \left\langle g^r, (m - \overline{k} \cdot g^r) \cdot r^{-1} \right\rangle$$

$$\mathcal{R} = \mathbb{F}^* \qquad \mathsf{V}\left(k, \langle c_1, c_2 \rangle, m\right) \iff \left(k^{c_1} \cdot c_1^{c_2} = g^m\right)$$

Example of a probabilistic signature system: El Gamal

Fix a finite field $\mathbb{F}$ and $g \in \mathbb{F}^*$.

$$
\begin{aligned}
\mathcal{M} &= \mathbb{F} & \mathrm{K_V}(a) &= g^a \\
\mathcal{S} &= \mathbb{F}^* \times \mathbb{F} & \mathrm{K_S}(a) &= a \\
\mathcal{K} &= \mathbb{F}^* \times \mathbb{F}^* & \mathrm{S}(r, \overline{k}, m) &= \left\langle g^r, (m - \overline{k} \cdot g^r) \cdot r^{-1} \right\rangle \\
\mathcal{R} &= \mathbb{F}^* & \mathrm{V}(k, \langle c_1, c_2 \rangle, m) &\iff \left(k^{c_1} \cdot c_1^{c_2} = g^m \right)
\end{aligned}
$$

Signature verification

$$
\begin{aligned}
\mathrm{V}(\mathrm{K_V}(a), \mathrm{S}(r, \mathrm{K_S}(a), m), m) &\iff \mathrm{V}(g^a, \mathrm{S}(r, a, m), m) \\
&\iff \mathrm{V}\left(g^a, \left\langle g^r, (m - ag^r) r^{-1} \right\rangle, m\right) \\
&\iff \left(g^{ag^r} \cdot g^{r(m - ag^r) r^{-1}} = g^m \right)
\end{aligned}
$$

Signature systems

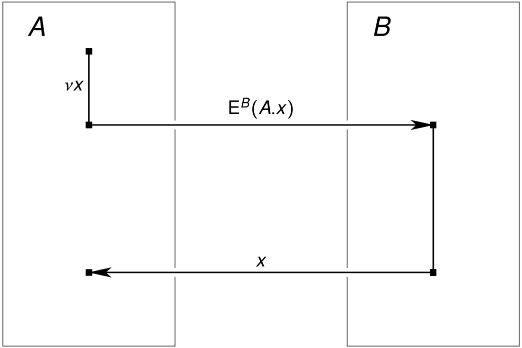
Homework

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 $r \in \mathcal{R}$ in unforgeability?

CR with Public Key Encryption

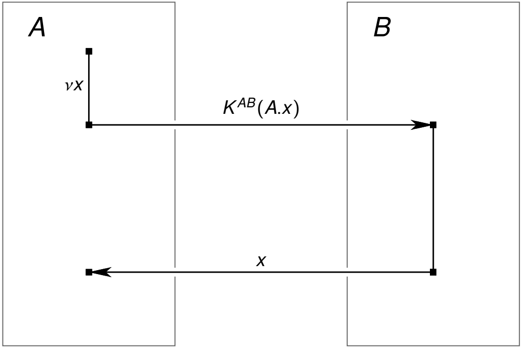
$(\mathrm{CRE}) = (\mathrm{CR})\left[c^{AB}x = \mathrm{E}^B(A.x)\ ,\ r^{AB}x = x\right]$



$$A: \quad (\nu x)_A \triangleright \left\langle\!\left\langle \mathrm{E}^B t(x) \right\rangle\!\right\rangle_A \triangleright \langle\langle x \rangle\rangle_{\overrightarrow{X}} \implies X = A \vee X = B \quad (\mathrm{enc})$$

CR with Shared Key at the Input

$$(\mathrm{CRKI}) = (\mathrm{CR})\left[c^{AB}x = K^{AB}(A.x)\ ,\ r^{AB}x = x\right]$$



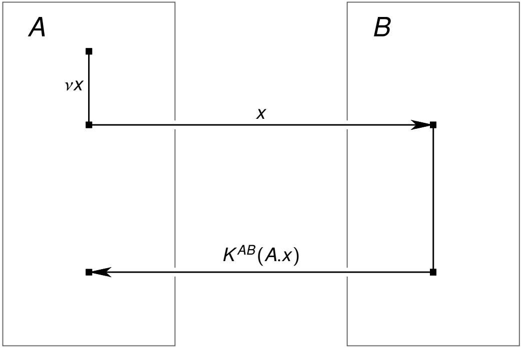
$$K^{AB}t = K^{AB}u \implies t = u \qquad \text{(hk1)}$$

$$\langle\langle K^{AB}t\rangle\rangle_{\vec{X}} \implies X = A \vee X = B \qquad \text{(hk2)}$$

$$K^{AB} = K^{BA} \qquad \text{(hk3)}$$

CR with Shared Key at the Output

$$(\mathrm{CRKO}) = (\mathrm{CR})\left[c^{AB}x = x\ ,\ r^{AB}x = K^{AB}(A.x)\right]$$

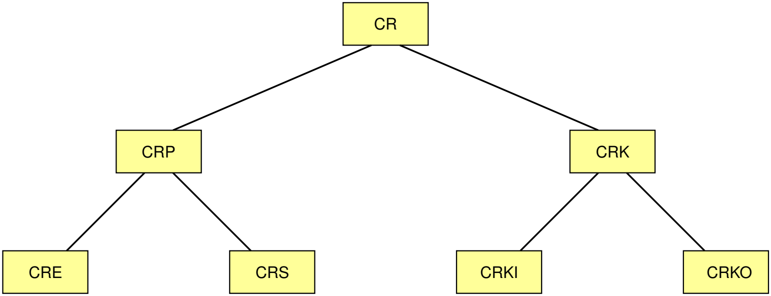


$$K^{AB}t = K^{AB}u \implies t = u \qquad \text{(hk1)}$$

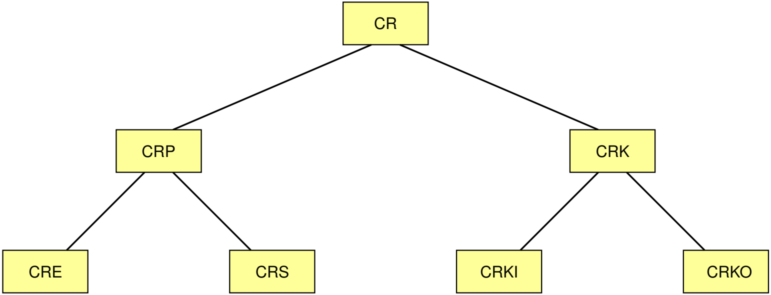
$$\langle\langle K^{AB}t\rangle\rangle_{\overrightarrow{X}} \implies X = A \vee X = B \qquad \text{(hk2)}$$

$$K^{AB} = K^{BA} \qquad \text{(hk3)}$$

Basic CR-implementations



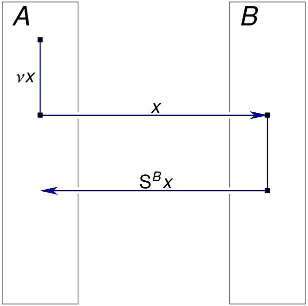
Basic CR-implementations

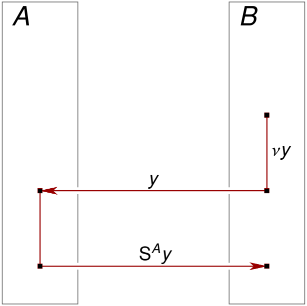


(The difference between (CRS) and (CRS_0) is discussed later.)

Mutual authentication

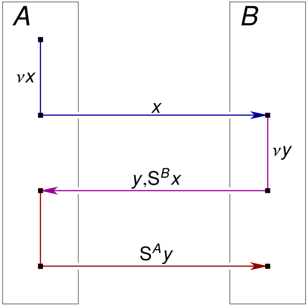
To establish a session, Alice and Bob authenticate each other





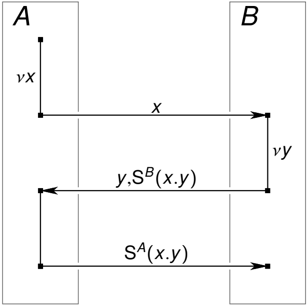
Mutual authentication: (CRS_0-seq)

. . . and Bob responds eagerly. . .



Mutual authentication: $(\mathrm{CRS}_0\text{-seq})^2$

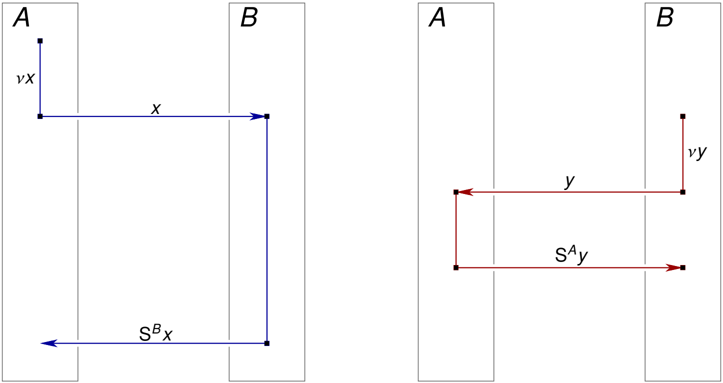
…binding the two authentications together…



[2] This protocol is better known as (ISO-9897-3).

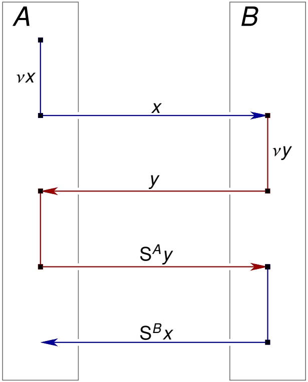
Mutual authentication: (CRS_0-nest)

...or Bob may respond lazily...



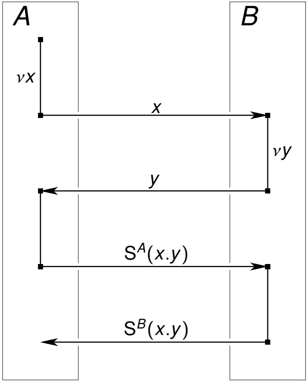
Mutual authentication: (CRS_0-nest)

. . . first authenticates Alice. . .



Mutual authentication: (CRS_0-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

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

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 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

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.

Ditto for Bob.

Matching conversations in (CRS$_0$-nest)

Proposition

Protocol (CRS$_0$-nest) *realizes mutual authentication*

Matching conversations in (CRS_0-nest)

Proposition

Protocol (CRS_0-nest) *realizes mutual authentication, provided that both principals are* honest.

Matching conversations in (CRS_0-nest)

Proposition

Protocol (CRS_0-nest) *realizes mutual authentication, provided that both principals are* honest*.*

Formal notion of honesty

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

Matching conversations in (CRS_0-nest)

Proposition

Protocol (CRS_0-nest) *realizes mutual authentication, provided that both principals are* honest.

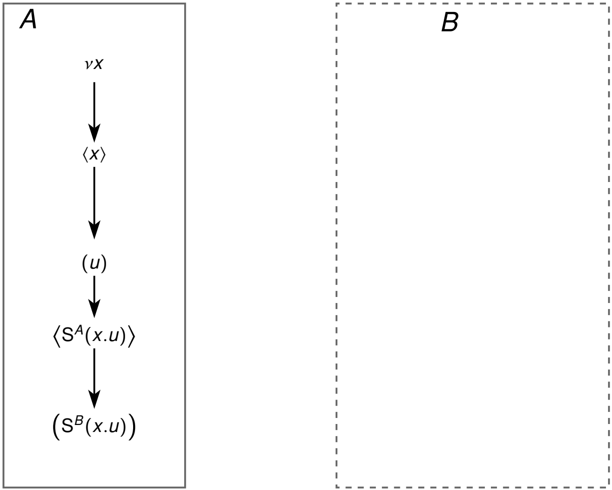
Formal notion of honesty

We say that a principal in a protocol is *honest* within a protocol if she acts according to the protocol. This means that she only performs her actions

- in the prescribed order, and
- with the prescribed data.

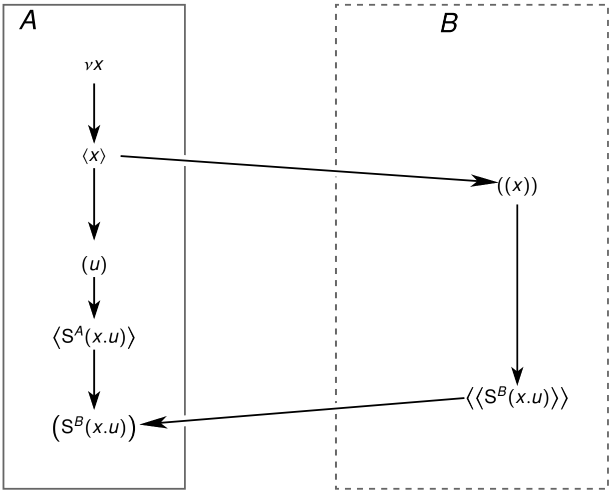
Matching conversations in (CRS$_0$-nest): Alice

Initially, Alice only sees her own actions:



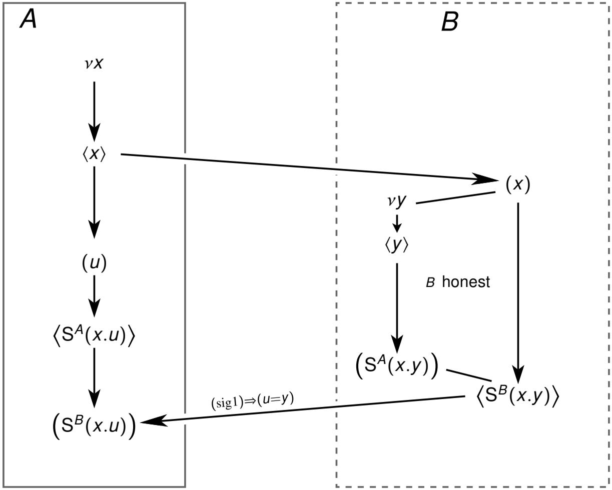
Matching conversations in (CRS_0-nest): Alice

By (cr)$\left[c^{AB} = \mathrm{id},\ r^{AB} = \mathsf{S}^B\right]$ (proved earlier)



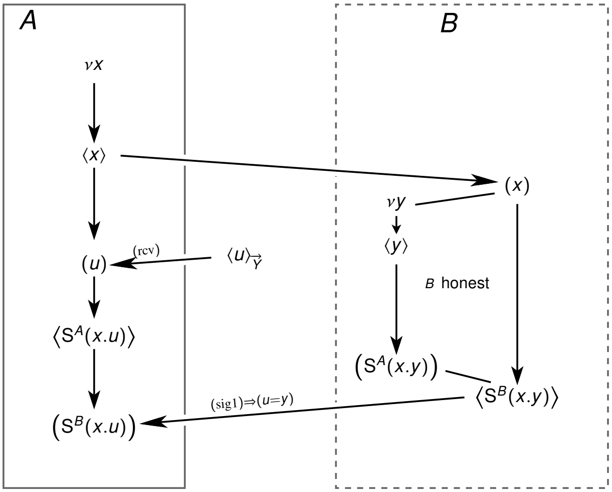
Matching conversations in (CRS_0-nest): Alice

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



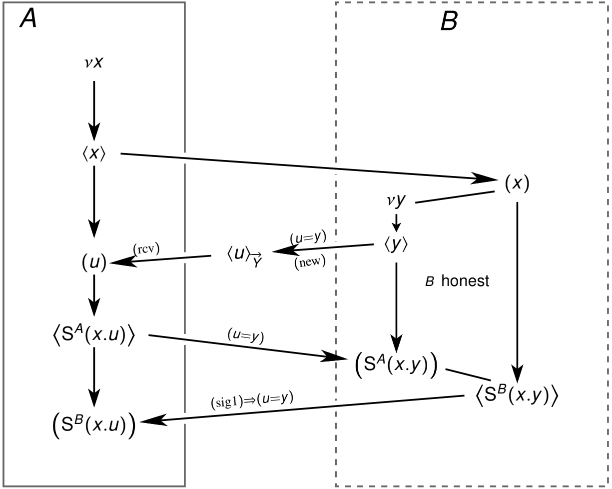
Matching conversations in (CRS_0-nest): Alice

By (rcv), someone must have sent u



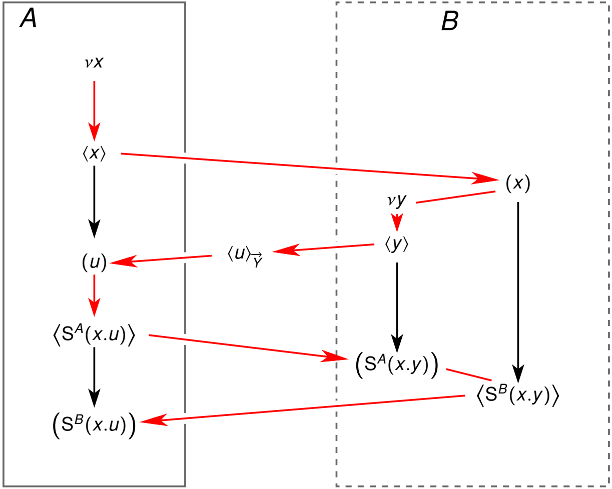
Matching conversations in (CRS_0-nest): Alice

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



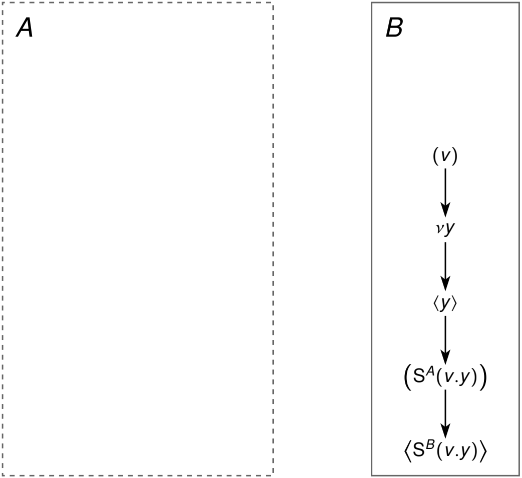
Matching conversations in (CRS_0-nest): Alice

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



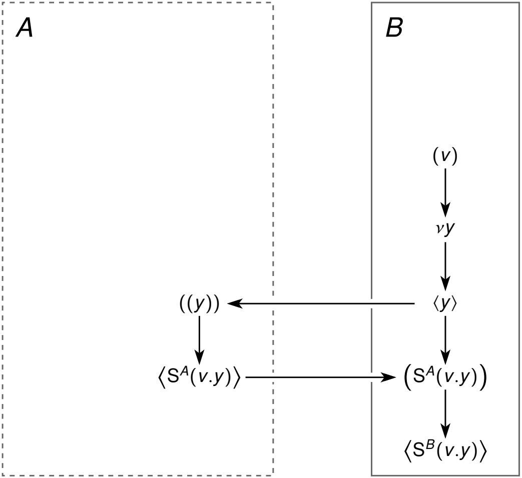
Matching conversations in (CRS_0-nest): Bob

Initially, Bob only sees his own actions:



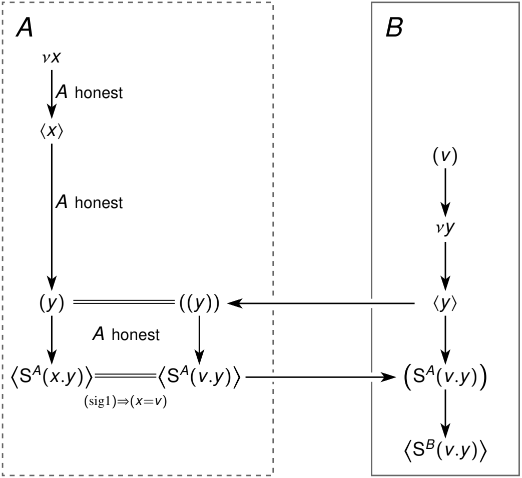
Matching conversations in (CRS_0-nest): Bob

By the (cr)-axiom, he concludes that Alice must be on-line.



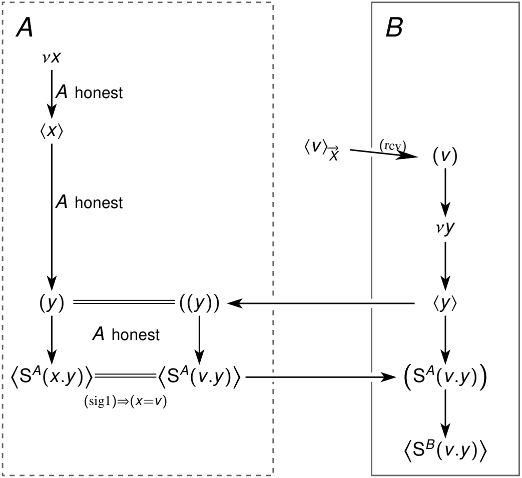
Matching conversations in (CRS_0-nest): Bob

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



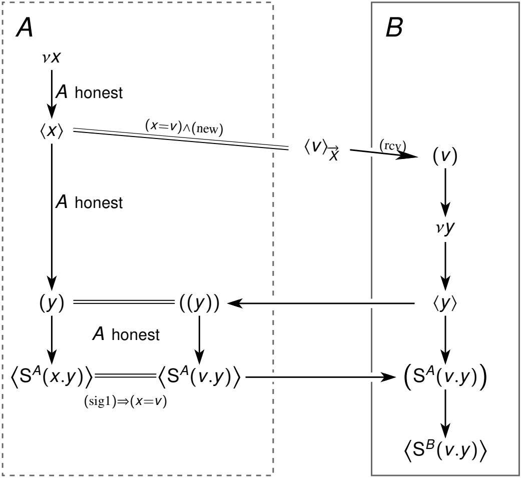
Matching conversations in (CRS_0-nest): Bob

By (rcv), someone must have sent the first message.



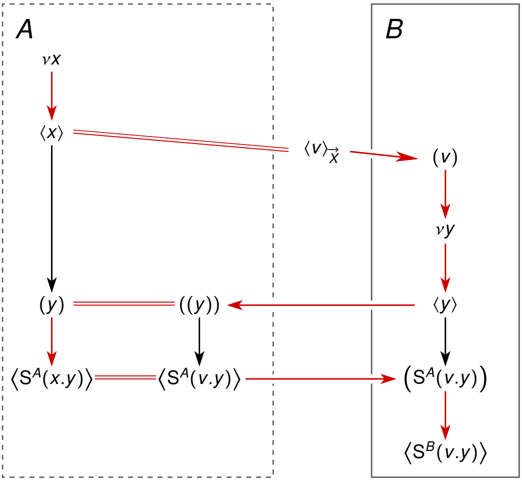
Matching conversations in (CRS$_0$-nest): Bob

By (sig1) and (new), that must have been Alice.



Matching conversations in (CRS_0-nest): Bob

Bob has derived the total order of his and Alice's actions.



Matching conversations in (CRS_0-nest)

Both Alice and Bob have thus recorded
the following conversation

$$(\nu x)_A < \langle x \rangle_A < (x)_B < (\nu y)_B < \langle y \rangle_B < (y)_A < \\ \left\langle S^A(x.y) \right\rangle_A < \left(S^A(x.y) \right)_B < \left\langle S^B(x.y) \right\rangle_B$$

Their records match, and the mutual authentication is achieved.

Mutual authentication by (CRE)

Homework

Compose two instances of the (CRE) protocol, to build a protocol (CRE-seq) for mutual authentication.

Analyze the difference between (CRE-seq) and the (NSPK) protocol, introduced in Sec. 5 of Part 3? Is (CRE-seq) vulnerable to the same attack as (NSPK)?

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)?

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.

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 Server

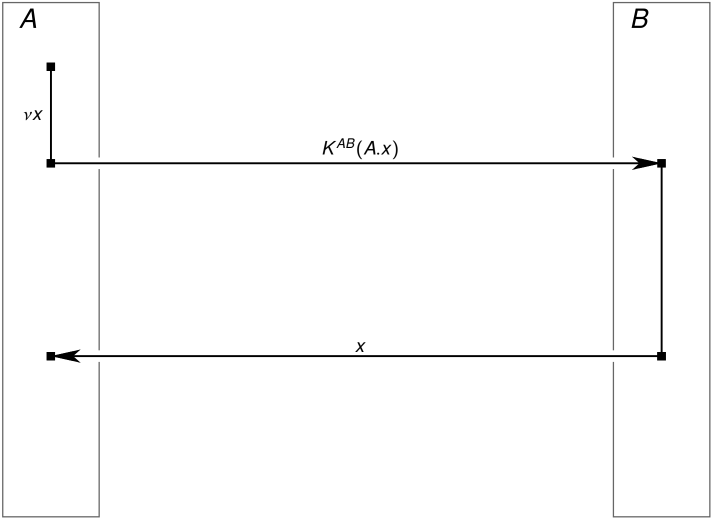
An Authentication Server S shares a symmetric key k^{BS} with every principal B.

Authentication service proceeds as follows

- A authenticates S, using $K^{AS}m = \mathrm{E}(k^{AS}, m)$
- S authenticates B using $K^{BS}m = \mathrm{E}(k^{BS}, m)$
- if S is honest, then A thus authenticates B.

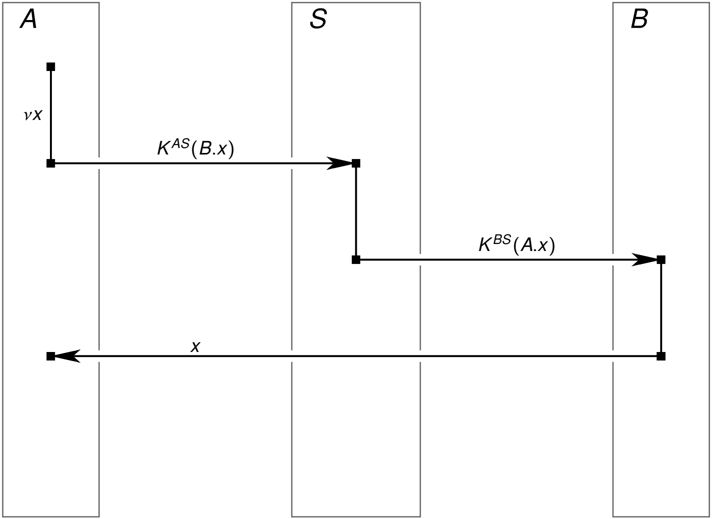
Authentication services

CRKI



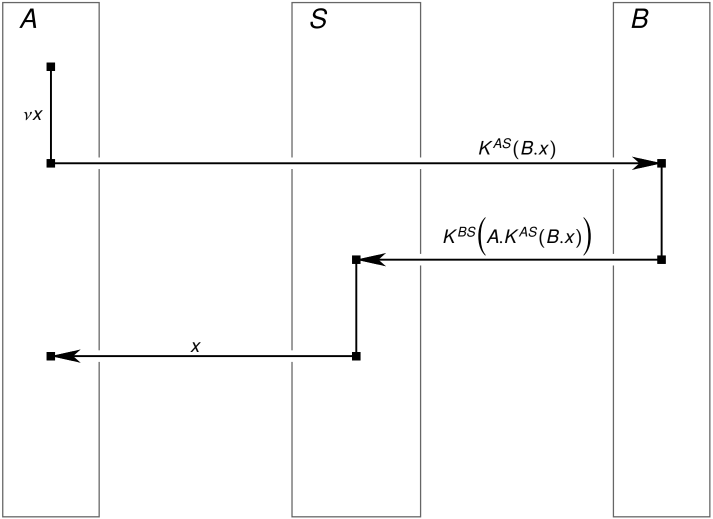
Authentication services

CRKII



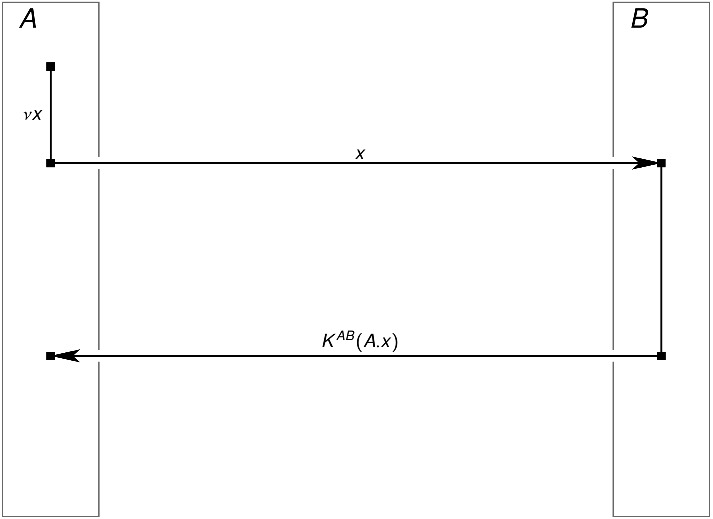
Authentication services

CRKIO



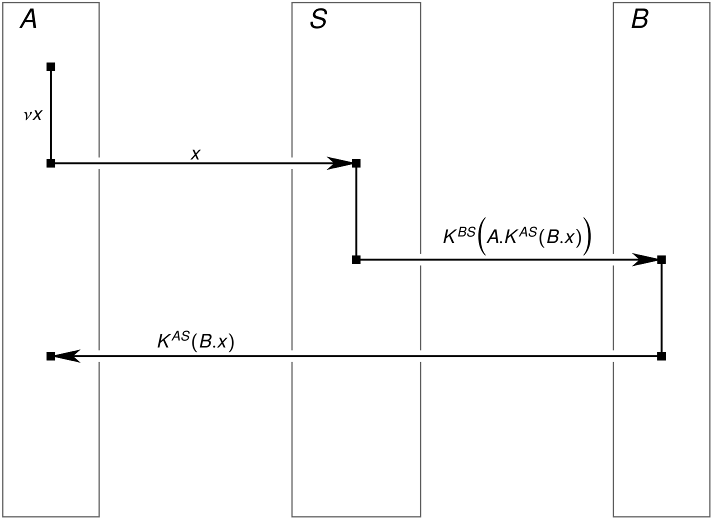
Authentication services

CRKO



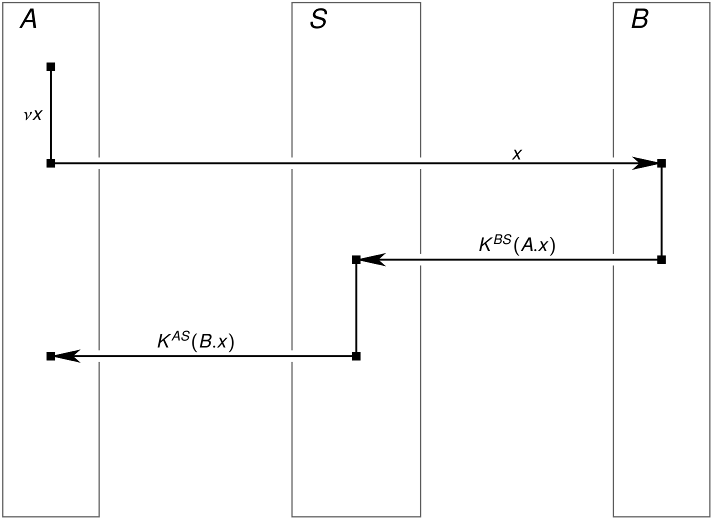
Authentication services

CRKOI



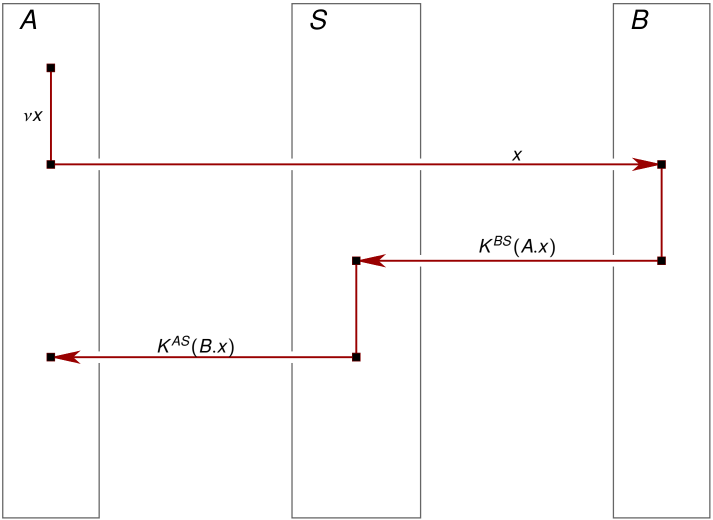
Authentication services

CRKOO



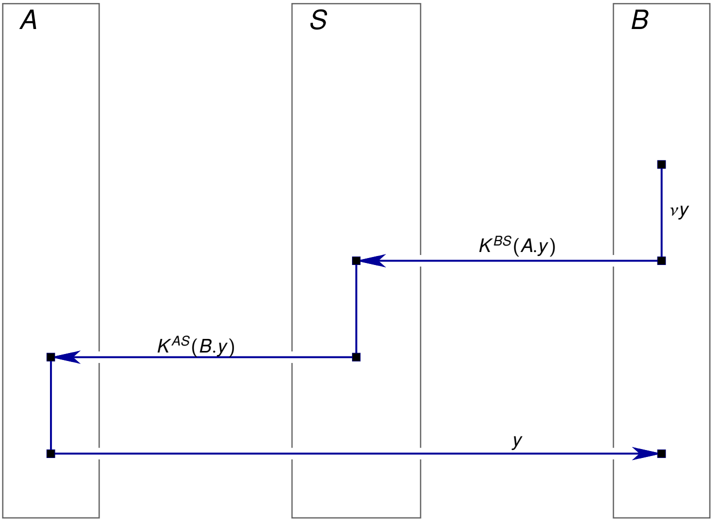
Towards the Yahalom protocol

Component 1: CRKOO



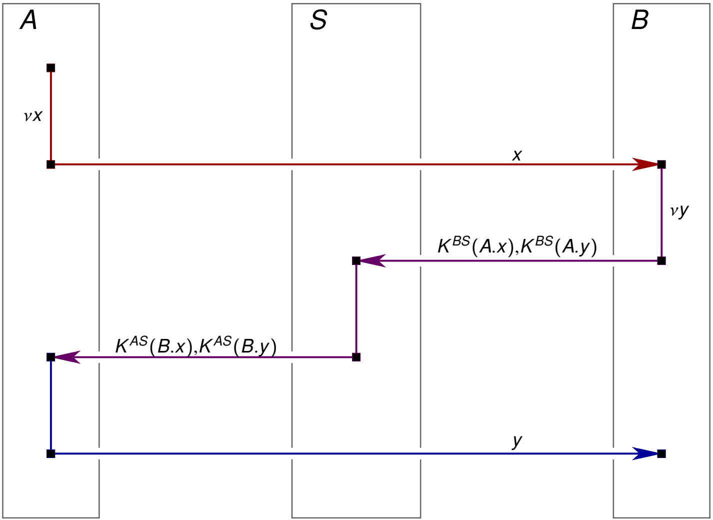
Towards the Yahalom protocol

Component 2: CRKII



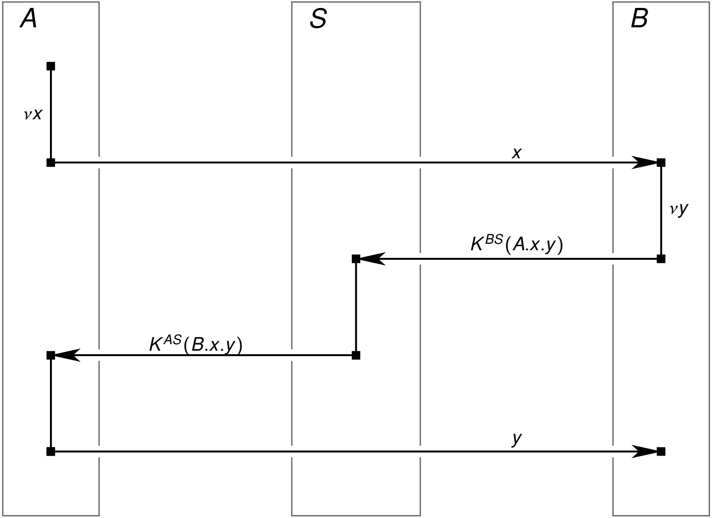
Towards the Yahalom protocol

Step 3: Composition



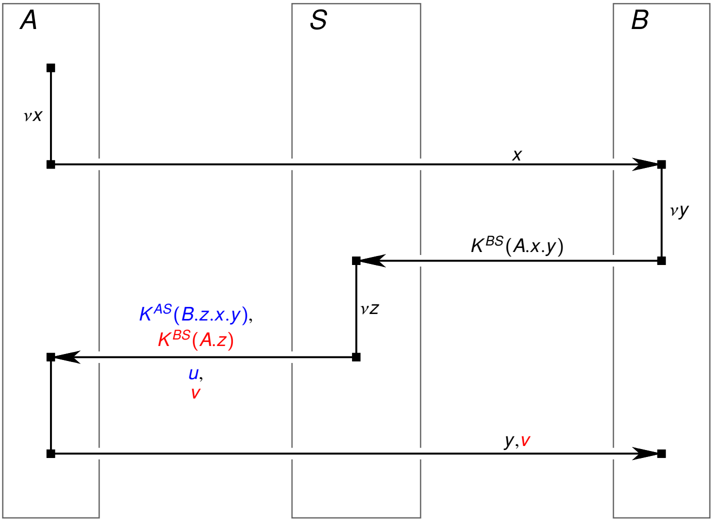
Towards the Yahalom protocol

Step 4: Binding



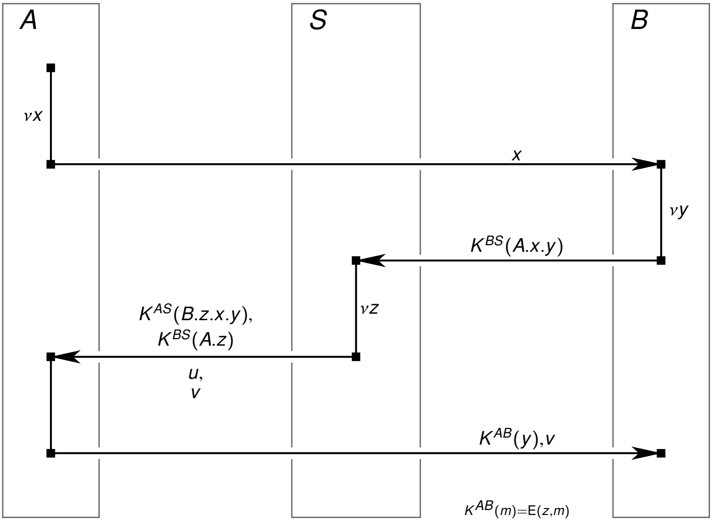
Towards the Yahalom protocol

Step 5: Key distribution



Yahalom protocol

Step 5: Key confirmation



Outline

Basic ideas of authentication

Challenge-Response Authentication

Impersonation Attacks

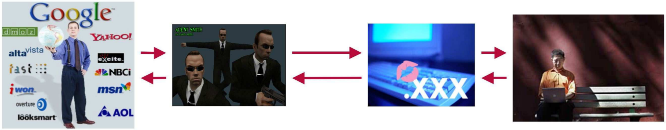
- Examples of impersonation
- Attack on (CRS_0)
- Attack on (CRS_0-nest)

What did we learn?

Recall from Part 1: CAPTCHA

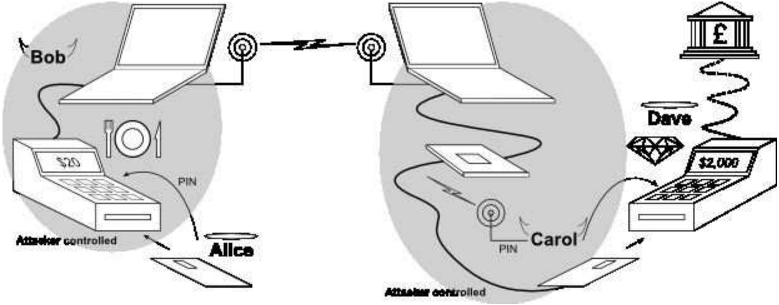
X6K5p

Man-in-the-Middle (MitM) Attack



Man-in-the-Middle (MitM) Attack

Smart card relay



. . . much easier with NFC phones!

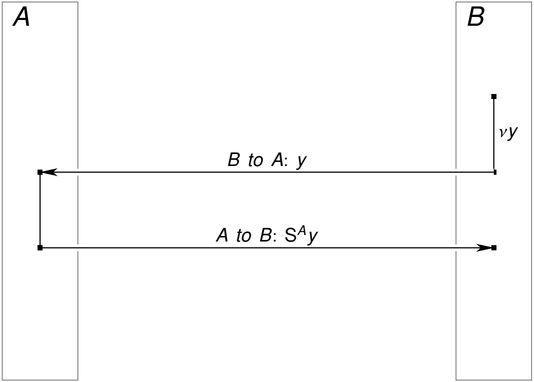
Refining authentication to capture MitM attacks

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.

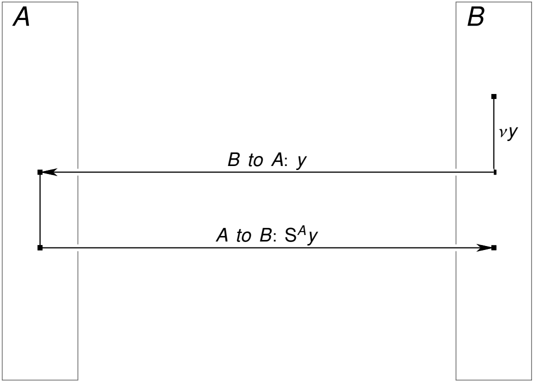
(CRS_0) authentication

Here is the protocol (CRS_0), initiated by Bob.



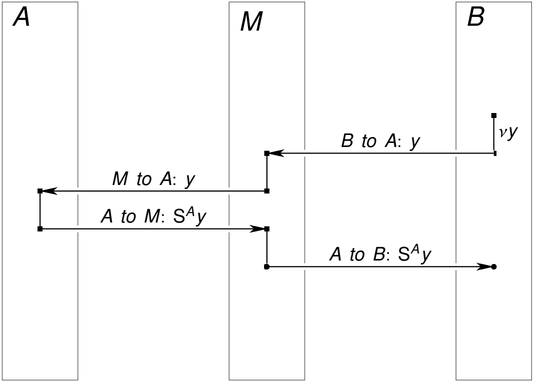
(CRS_0) authentication

Here is the protocol (CRS_0), initiated by Bob.
We proved that it correctly implements (CR).



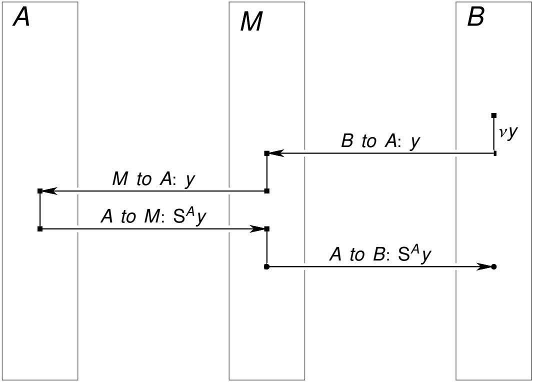
(CRS_0) authentication

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



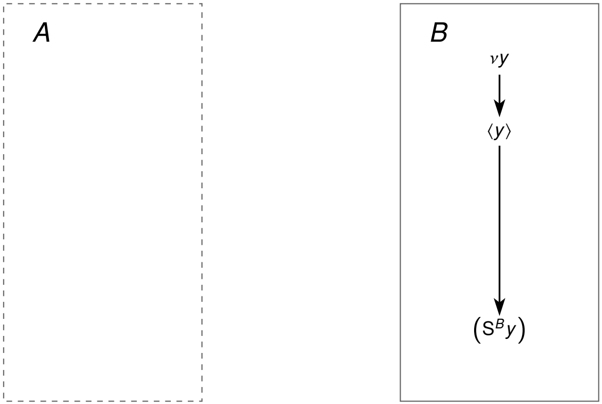
(CRS_0) authentication

But here is a Man-in-the-Middle attack on it.
(CRS_0) does not guarantee *agreement* about the identities.



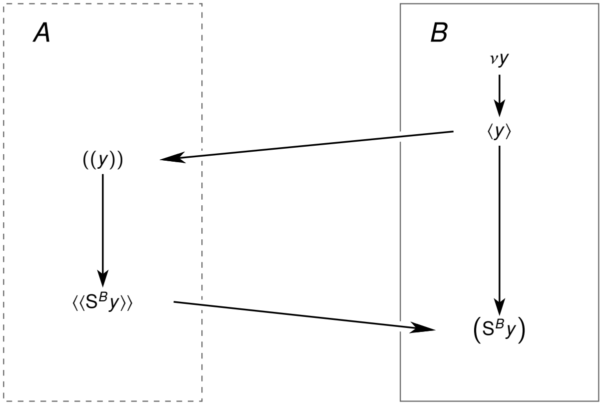
Ping authentication in (CRS_0)

We proved that from Bob's actions



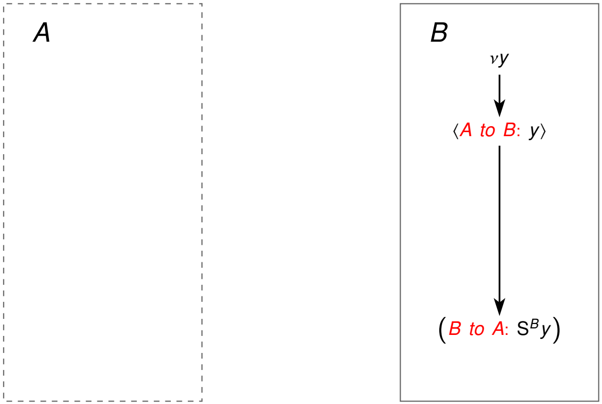
Ping authentication in (CRS_0)

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



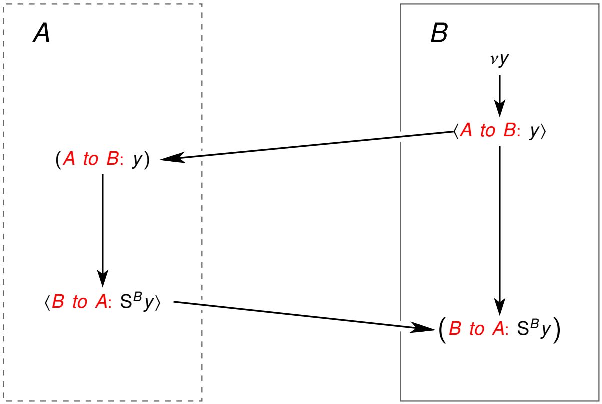
Ping authentication in (CRS_0)

We did **not** prove that from Bob's intent to challenge Alice



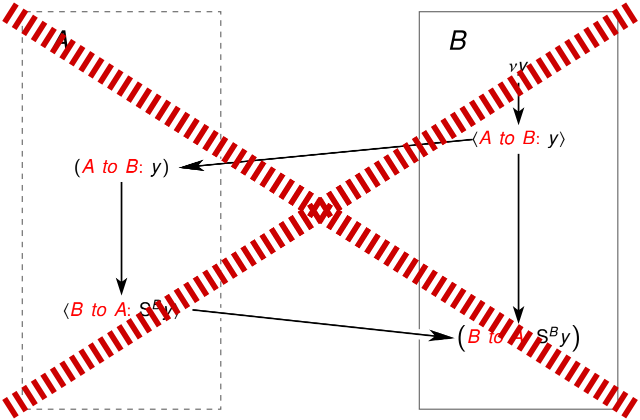
Ping authentication in (CRS_0)

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



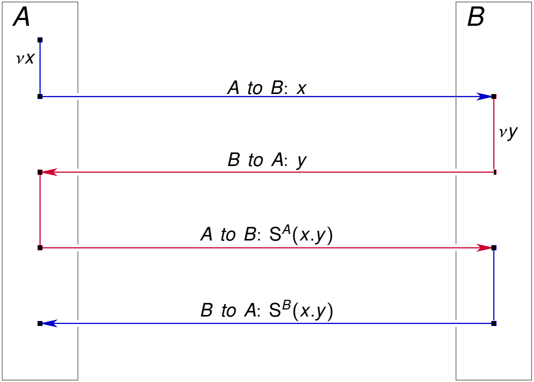
No agreement in (CRS_0)

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



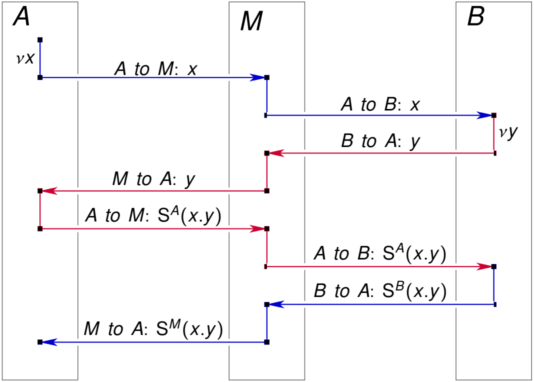
Mutual authentication: (CRS$_0$-nest)

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



Mutual authentication: (CRS_0-nest)

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



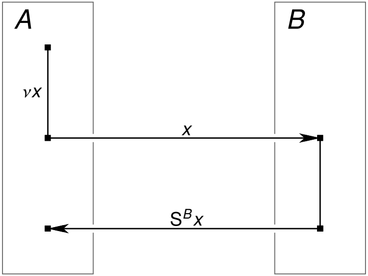
Moral

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

One-way authentication with Signature

$(\mathrm{CRS}_0) = (\mathrm{CR})\left[c^{AB}x = x\ ,\ r^{AB}x = \mathrm{S}^{B}x\right]$

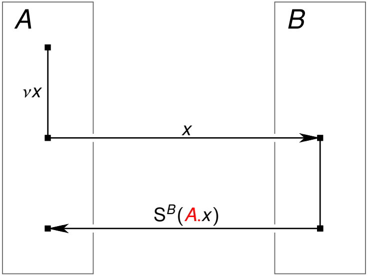
NOT



One-way authentication with Signature

$(\text{CRS}) = (\text{CR})\left[c^{AB}x = x \ , \ r^{AB}x = \text{S}^B(A.x)\right]$

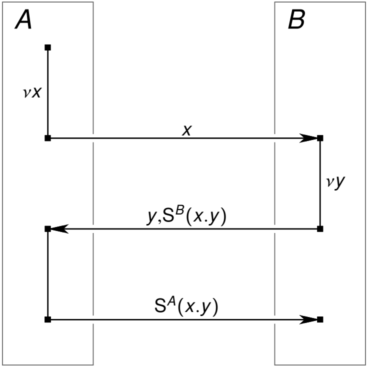
BUT



Mutual authentication with Signatures

(CRS_0-seq) = (ISO-9798-3)

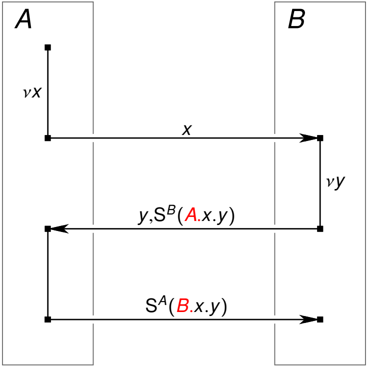
NOT



Mutual authentication with Signatures

(CRS-seq)

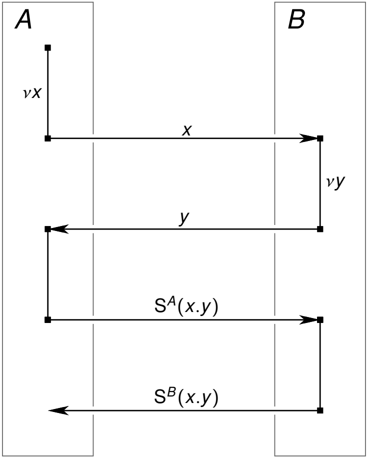
BUT



Mutual authentication with Signatures

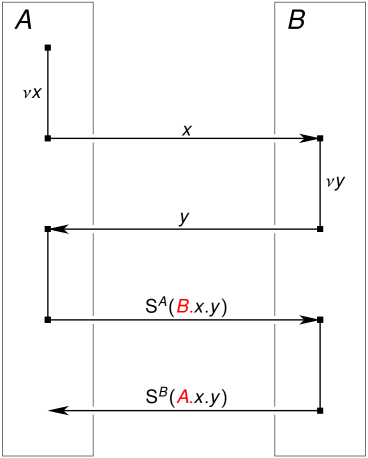
(CRS_0-nest)

NOT



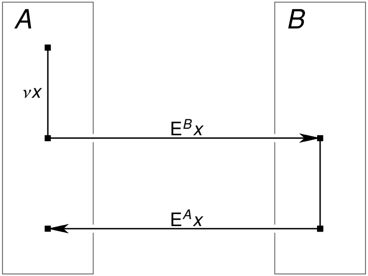
Mutual authentication with Signatures
(CRS-nest)

BUT



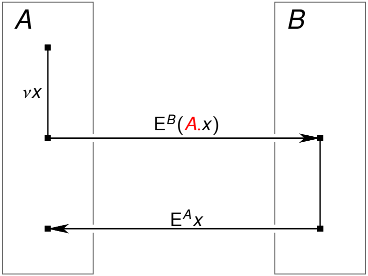
One-way authentication with Encryptions
(CREE$_0$)

NOT



One-way authentication with Encryptions
(CREE)

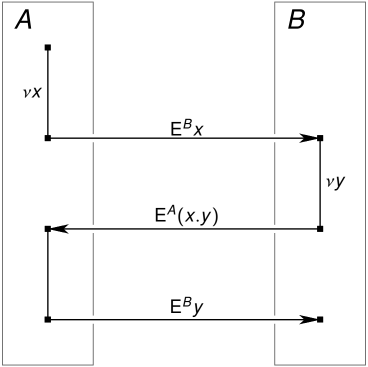
BUT



Mutual authentication with Encryptions

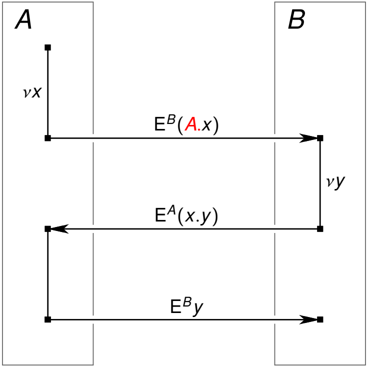
(CREE$_0$-seq)

NOT



Mutual authentication with Encryptions (NSPK)

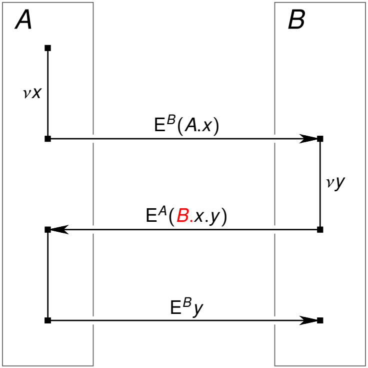
. . . and NOT



Mutual authentication with Encryptions

(CREE-seq) = (NSL)

BUT



Discussion

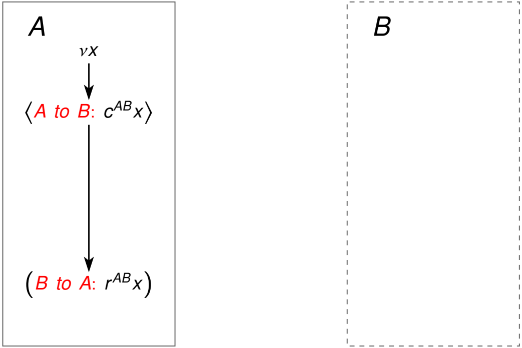
The definitions of

- 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.

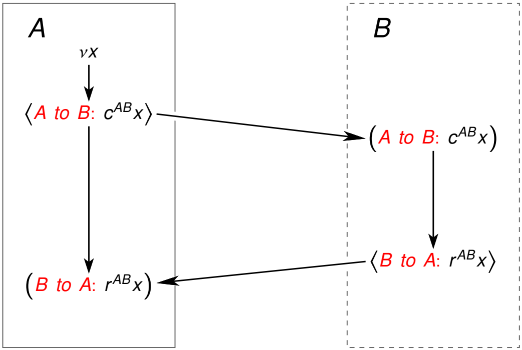
Strong one-way authentication

Intended authentication



Strong one-way authentication

Intended authentication



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.

Strong authentication with signatures

Proposition

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

Strong authentication with signatures

Proposition

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

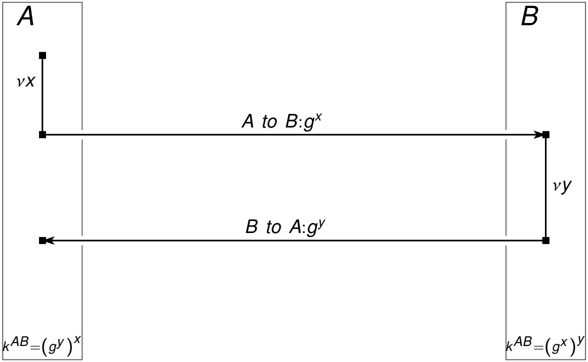
Homework

Prove this.

Outline

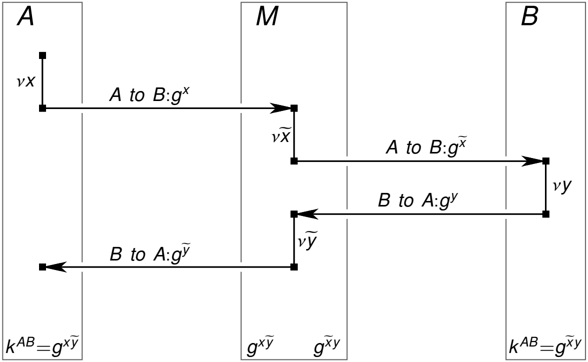
Secure key generation

Can we now generate keys **securely**. . .



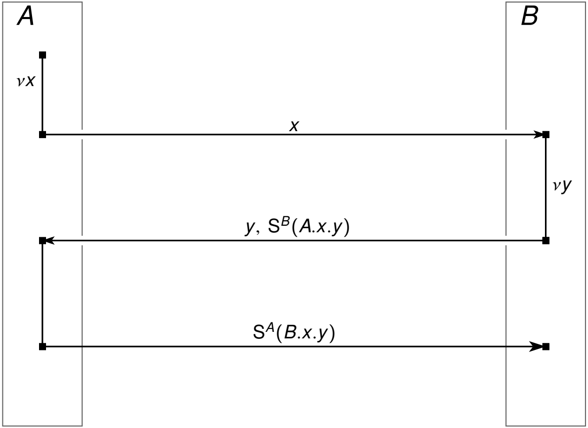
Secure key generation

...while avoiding the MitM-attacks?



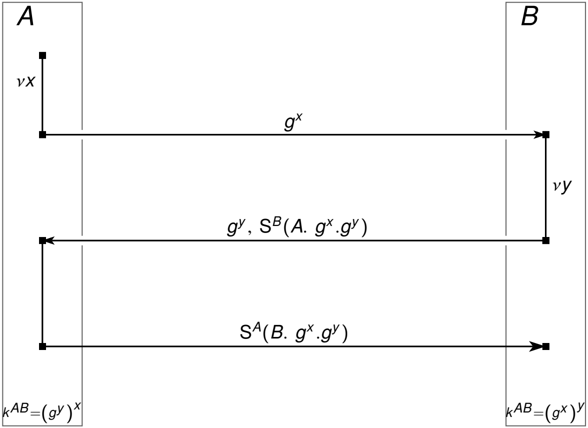
Secure key generation

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



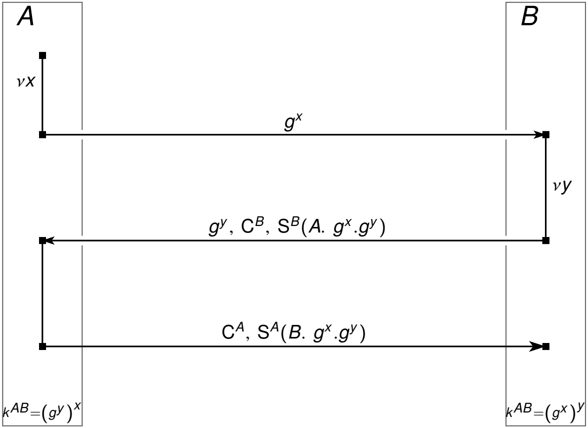
Secure key generation

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



Secure key generation

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



where $C^X = S^S\left(X.V^X\right)$

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.

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. (Recall Yahalom.)

A symmetric key Authentication Server is often called a Key Distribution Center (KDC).

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. (Recall Yahalom.)

A symmetric key Authentication Server is often called a Key Distribution Center (KDC).

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^A and C^B.

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 $A \to B$ is bootstrapped over $A \to S$ and $S \to B$.

KDCs and CAs

Similarities

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

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

KDCs and CAs

Disadvantages of KDC

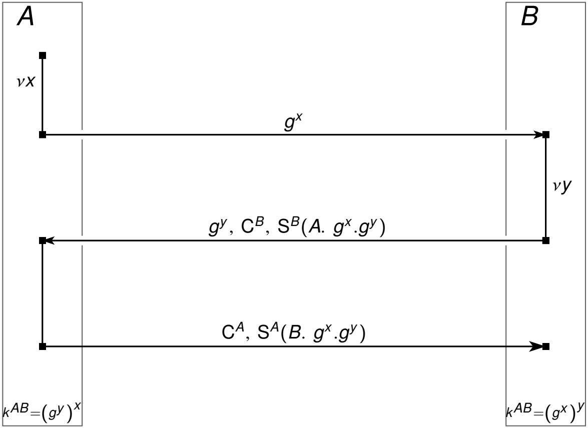
- can impersonate everyone to everyone
- single point of failure, performance bottleneck
- must be on-line, otherwise the network halts

Disadvantage of CA

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

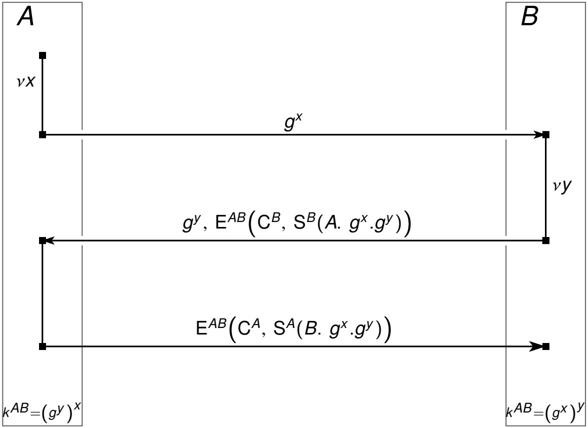
Secure key generation

Adding key confirmation and identity protection to



Secure key generation

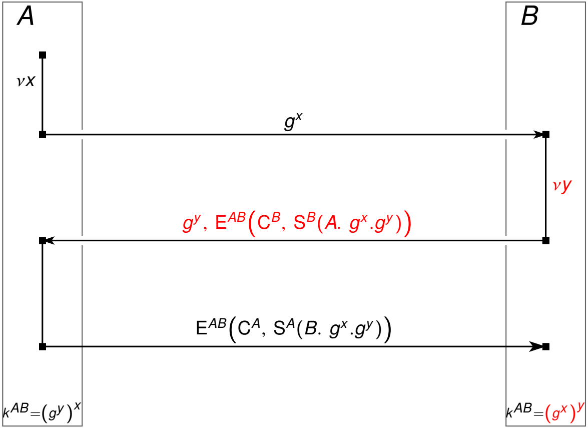
. . . we get in the realm of practical protocols:



where $\mathrm{E}^{AB}(u) = \mathrm{E}\left(k^{AB}, u\right)$

Secure key generation

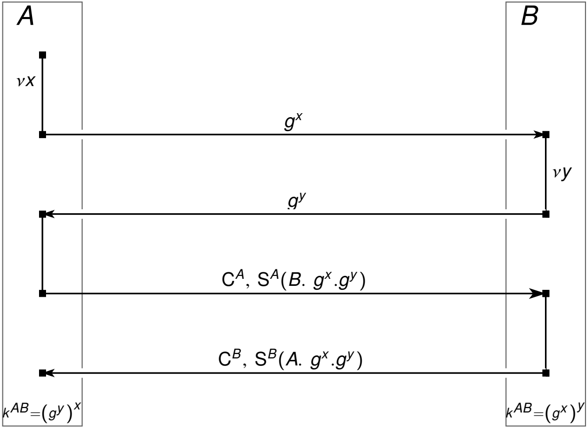
Problem: Bob exposed to DoS attack!



where $E^{AB}(u) = E\left(k^{AB}, u\right)$

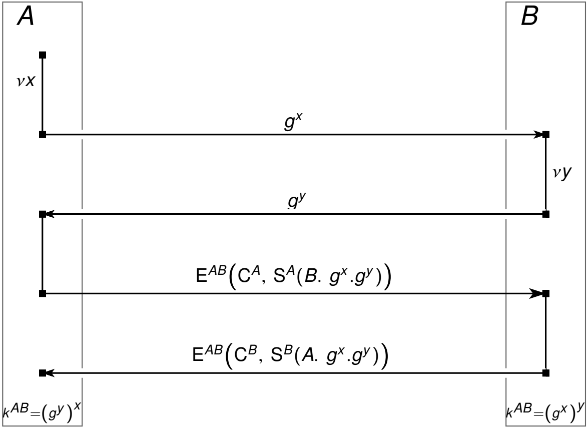
Secure key generation

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



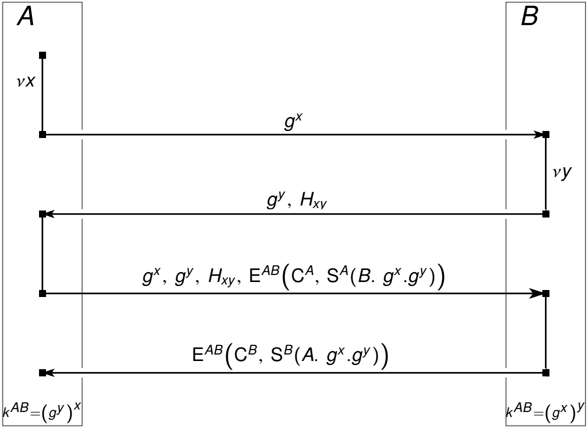
Secure key generation

. . . just like before to



Secure key generation

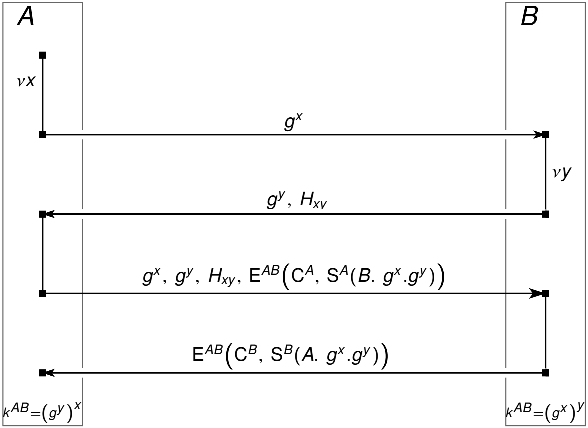
If Bob is a busy CA, he can use cookies H_{xy}...



where $H_{xy} = H(g^x.g^y)$

Secure key generation

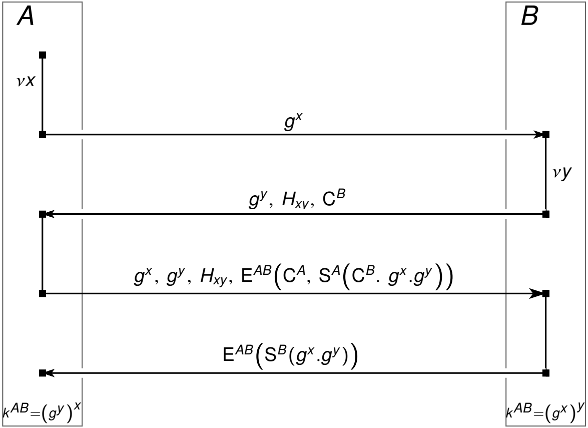
. . . and needn't keep the state at all!



where $H_{xy} = H(g^x.g^y)$

Secure key generation

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



where $H_{xy} = H\left(g^x.g^y\right)$

Secure key generation

Homework

What are the security consequences of replacing $\mathrm{S}^A(B.\ g^x.g^y)$ by $\mathrm{S}^A\left(\mathrm{C}^B.\ g^x.g^y\right)$ in the third message in the preceding protocol?

Is this protocol open for a MitM-attack because of $\mathrm{S}^B(g^x.g^y)$ instead of $\mathrm{S}^B(A.\ g^x.g^y)$ in the final message?

What kind of attacks would become possible if the encryptions by E^{AB} were removed?

Summary: Questions of authentication

Why is it that

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

Summary: Questions of authentication

Why is it that

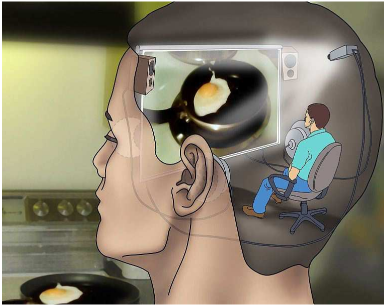
- it is easy to establish a secure channel, but
- it is hard to know with whom?

Why is it that

- crypto systems are broken once in a while, but
- authentications fail every day?

Old answer: Authentication is a deep problem

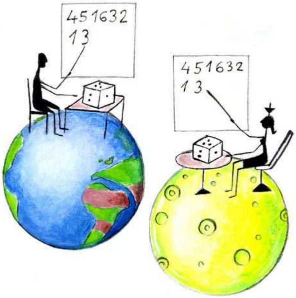
From local observations to global conclusions — through reflection



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: "Noone 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?

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?

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.