# Private Authentication

## Hiding Name in the Applied Pi Calculus

Martín Abadi. Private authentication. In *Proceedings of the Workshop on Privacy Enhancing Technologies (PET 2002)*, LNCS. Springer-Verlag, 2002.

M. Abadi and C. Fournet. Hiding names: Private authentication in the applied pi calculus. In M. Okada, B. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *Software Security – Theories and Systems. Mext-NSF-JSPS International Symposium, Tokyo, Nov. 2002 (ISSS'02)*, volume 2609 of *LNCS*, pages 317–338. Springer, 2003.

# Session Establishment

- Two parties want to open a secure session; they need to
  - Generate a shared secret (the “session key”)
  - Agree on parameters
  - Verify each other’s identity
- Attackers may eavesdrop, delete, and insert messages, may impersonate principals,… in order to
  - gain information
  - confuse or hinder the participants
- This is a classical setting for cryptographic protocols


R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT–29(2):198–208, 1983.

# Session Establishment

- Protocol design and verification is still (surprisingly) active
  - Core secrecy and authentication now well-understood
  - New settings, e.g. mobility
  - New "secondary" requirements
    - Efficiency, DOS attacks
    - **Privacy**: a delicate concern, with no clear specification

- We discuss privacy issues in session establishment
  - we present a simple protocol for private authentication
  - we develop its model in the applied pi calculus
  - we express its properties using process equivalences for secrecy, authentication, and identity protection

# Private Communication

- Two or more principals wish to communicate securely, protecting their identities, movements, behaviours, communication patterns,… from third parties
  - Mobile telephony
  - Mobile computing
  - UPnP, home network
  - IPSEC, mobile IP
- Third parties? Other users + infrastructure
- Privacy may coexist with communication, but not by default
  - Effective communication requires routing
  - Traffic analysis reveals a lot of information, even if all traffic is encrypted (e.g. key identifiers linked to principals)
  - With some care, one can hide origin/destination of messages

# Private Authentication

- Protocols may help, but they are also part of the problem
  - Principal A may demand that B prove its identity before revealing anything
  - Protocols often pass names and credentials in cleartext
  - Protocols often provide evidence of session establishment

- Who should reveal one's identity first?
  - What is a good trade-off between authentication, performance, and anonymity?
  - In client-server systems, the server is seldom protected
  - In fluid, symmetric, peer-to-peer systems, privacy is more desirable and more problematic

- Privacy should be an explicit goal of the protocol

# The Problem

- Within a location (physical building, wireless LAN),
  A tries to contact B
  B is willing to respond (and prove his identity) to any $A \in S_B$
- The network and other participants are untrusted
- A and B do not share a long-term secret

- A and B should be able to establish authenticated, private communication channels
- A and B should not have to indicate their identity, presence, or willingness to communicate ($S_A$,$S_B$) to anyone else

# Assumptions

Network

- Each participant can broadcast messages
- Message headers don’t reveal identity information

Cryptography

- We rely on public-key encryption
- A and B each have a public/private key pair
- A and B know each other’s public key (offline PKI, SPKI,…)

- Only a principals that knowns the private key can recover an encrypted message encrypted with the public key
- The success or failure of a decryption is evident
- Encryption is which-key concealing

# The Protocol (informally)

1. A generates a fresh nonce $N_A$ and sends

$$\text{“hello”},\ \{\text{“hello”},\ N_A,\ K_A\}_{K_B}$$

2. B receives "hello" message, tries to decrypt, checks that $A \in S_B$, generates $N_B$, then sends

$$\text{“ack”},\ \{\text{“ack”},\ N_A,\ N_B,\ K_B\}_{K_A}$$

…or, in all other cases, sends a decoy

$$\text{“ack”},\ \{\ N_B\ \}_K$$

3. A receives B's message, decrypts, checks, gets $N_B$
Afterwards, A and B use $(N_A, N_B)$ as shared secrets

# Properties and Limitations

$$\text{“hello”},\ \{\text{“hello”},\ N_A,\ K_A\}_{K_B}$$
$$\text{“ack”},\ \{\text{“ack”},\ N_A,\ N_B,\ K_B\}_{K_A}$$
$$\text{“ack”},\ \{\ N_B\ \}_K$$

- Secrecy: ($N_A$,$N_B$) become shared secrets
  For instance, A and B can use h($N_A$,$N_B$) as shared key
- Responder authentication:
  A has evidence that it shares ($N_A$,$N_B$) with B
  B has no evidence so far, but it shares ($N_A$,$N_B$) at most with A
- Identity protection: without $K_A^{-1}$ or $K_B^{-1}$,
  the messages look the same for any sessions

# Extensions

- Efficiency
  - The protocol is quite inefficient, leading to potential DOS (messages, bandwidth, public-key decryptions)
  - The protocol does not scale well

  - We can include some (partial) principal identifier
  - We can include a session identifier,
    so that the second message can be routed
  - We can send a first message to numerous potential participants, sharing some message and encryption costs

- Groups
  - A and B don't know each other, but are member of some group, e.g. "network printers" or "Italians"

# Private Authentication (now in applied pi)

| | |
|---|---|
| $M, N ::=$ | **Terms** |
| $a, b, c, \ldots, k, \ldots, m, n, \ldots, s$ | name |
| $x, y, z$ | variable |
| $f(M_1, \ldots, M_l)$ | function application |
| $P, Q, R ::=$ | **Processes** |
| $\mathbf{0}$ | null process |
| $P \mid Q$ | parallel composition |
| $!P$ | replication |
| $\nu n.P$ | name restriction ("new") |
| $\mathit{if}\ \ M = N\ \ \mathit{then}\ \ P\ \ \mathit{else}\ \ Q$ | conditional |
| $u(x).P$ | message input |
| $\overline{u}\langle N \rangle.P$ | message output |

# Formatted Messages

- The protocol uses two messages, “hello” and “ack”
- We use an equational theory with
  - functions hello(_,_) and ack(_,_,_) as constructors
  - function hello.0(_), hello.1(_), … , ack.2(_) as selectors
  - equations

$$
\begin{aligned}
\mathsf{hello.0}\,(\mathsf{hello}(x_0, x_1)) &= x_0 \\
\mathsf{hello.1}\,(\mathsf{hello}(x_0, x_1)) &= x_1 \\
\mathsf{ack.0}\,(\mathsf{ack}(y_0, y_1, y_2)) &= y_0 \\
\mathsf{ack.1}\,(\mathsf{ack}(y_0, y_1, y_2)) &= y_1 \\
\mathsf{ack.2}\,(\mathsf{ack}(y_0, y_1, y_2)) &= y_2
\end{aligned}
$$

# Public-key Encryption

- The protocol relies on public-key encryption
- We use function symbols for decryption, encryption, and public-key derivation, with a single equation:

$$\mathsf{decrypt}(\mathsf{encrypt}(x, \mathsf{pk}(y)), y) \quad = \quad x$$

- There is no inverse for pk(_), so one can reveal a derived public key and keep the private key secret.
- We model a "signing" principal using a context and an active substitution

$$P_B[\_] \stackrel{\text{def}}{=} \nu s.(\{K_B = \mathsf{pk}(s)\} \mid [\_])$$

# Equational Theory (Signature)

| | |
|---|---|
| $T, U, V, V_0, \cdots ::=$ | terms |
| $A, B, x_1, x_2, \ldots$ | variable |
| $c_1, c_2, \mathit{init}_A, \mathit{accept}_B, \mathit{connect}_A, \ldots$ | name (channel) |
| $N, N_A, K_A^{-1}, \ldots$ | name (crypto) |
| $\mathsf{h}(U, V)$ | cryptographic hash |
| $\mathsf{pk}(U)$ | public-key derivation |
| $\{T\}_V$ | public-key encryption |
| $\mathsf{decrypt}(W, U)$ | private-key decryption |
| $\mathsf{hello}(U_0, U_1), \mathsf{ack}(V_0, V_1, V_2)$ | protocol message |
| $\mathsf{hello.0}\,(U)\,, \ldots, \mathsf{ack.2}\,(V)$ | field selector |
| $\emptyset$ | empty set |
| $U.V$ | set extension |

# Equational Theory (Axioms)

$$\begin{aligned}
\mathsf{decrypt}(\{x\}_{\mathsf{pk}(z)}, z) &= x \\
\mathsf{hello.j}\,(\mathsf{hello}(x_0, x_1)) &= x_j \\
\mathsf{ack.j}\,(\mathsf{ack}(x_0, x_1, x_2)) &= x_j \\
(\emptyset.x).x &= \emptyset.x \\
(x.y).z &= (x.z).y
\end{aligned}$$

- Encryption is implicitly which-key concealing; alternatively, we can add equations for the attacker:

$$\begin{aligned}
\mathsf{get\text{-}key}(\{x\}_z) &= z \\
\mathsf{test\text{-}key}(\{x\}_z, z) &= \mathsf{true}
\end{aligned}$$

Then, we retain secrecy and authentication, but not privacy

# Roles and Principals

The protocol has two roles:

- The initiator (A) sending the “hello” message
- The responder (B) sending “ack” messages upon request

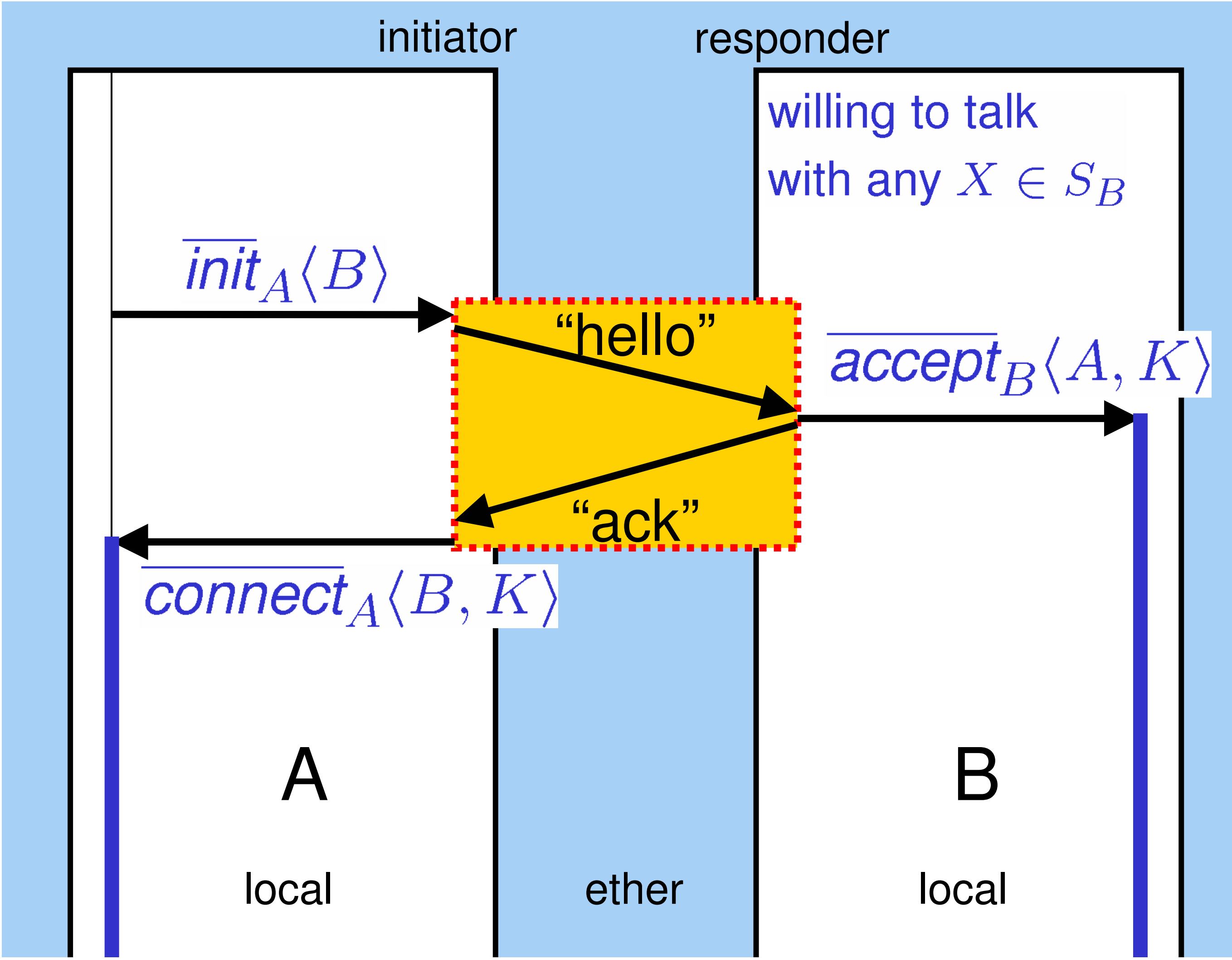
Each principal, X, consists of

- An instance of the protocol, $P_X$
- An (abstract) user process $U_X$ representing the application

It is essential to make explicit any interactions between protocols and users. We rely on control channels

# Roles and Principals (2)

An "API" for our private authentication protocol:



# Network and Attacker (broadcast)

- Communication on public channels models broadcast with an attacker that controls the network
- The attacker is the context; it may combine
  - Low-level attacks on the network
  - High-level attacks with any number of principals
- We sometimes represent passive attackers (eavesdroppers)

$A \xrightarrow{\nu\tilde{u}.[\widetilde{M}]} A'$ abbreviates $A \xrightarrow{\nu\tilde{u}.\langle\widetilde{M}\rangle}\xrightarrow{(\widetilde{M})} A'$.

$A \xrightarrow{\nu\tilde{u}.[\widetilde{M}]} A'$ implies $A \rightarrow \nu\tilde{u}.A'$

# The protocol (messages)

$$\begin{aligned}
\sigma_1 &\overset{\text{def}}{=} \{x_1 = \{\mathsf{hello}(N_A, A)\}_B\} \\
\sigma_2 &\overset{\text{def}}{=} \{x_2 = \{\mathsf{ack}(N_A, N_B, B)\}_A\} \\
\sigma_2^\circ &\overset{\text{def}}{=} \{x_2 = N_B\} \\
\sigma_K &\overset{\text{def}}{=} \{K = \mathsf{h}(N_A, N_B)\}
\end{aligned}$$

# The protocol (processes)

$$
\begin{array}{lcl}
P_A & \stackrel{\mathsf{def}}{=} & I_A \mid R_A \\
\\
I_A & \stackrel{\mathsf{def}}{=} & !\mathit{init}_A(B).\nu N_A.\left(\overline{c_1}\langle x_1\sigma_1\rangle \mid I'_A\right) \\
I'_A & \stackrel{\mathsf{def}}{=} & c_2(x_2). \\
 & & \mathit{if}\ x_2 = \{\mathsf{ack}(N_A, \nu N_B, B)\}_A\ \mathit{using}\ K_A^{-1} \\
 & & \mathit{then}\ \overline{\mathit{connect}}_A\langle B, K\sigma_K\rangle \\
R_B & \stackrel{\mathsf{def}}{=} & !c_1(x_1 \setminus \emptyset).\mathit{if}\ x_1\ \mathit{fresh} \\
 & & \mathit{and}\ x_1 = \{\mathsf{hello}(\nu N_A, \nu A)\}_B\ \mathit{using}\ K_B^{-1} \\
 & & \mathit{and}\ A \in S_B \\
 & & \mathit{then}\ \nu N_B.\left(\overline{c_2}\langle x_2\sigma_2\rangle \mid \overline{\mathit{accept}}_B\langle A, K\sigma_K\rangle\right) \\
 & & \mathit{else}\ \nu N_B.\overline{c_2}\langle x_2\sigma_2^\circ\rangle
\end{array}
$$

# The protocol (syntactic sugar)

For decryption, we use pattern matching, and write

$$\textit{if } x = \{\mathsf{ack}(N_A, \nu N_B, B)\}_A \textit{ using } K_A^{-1} \textit{ then } P \textit{ else } Q$$

for the process

$$\nu N_B. \left( \begin{array}{l} \{N_B = \mathsf{ack}.1\left(\mathsf{decrypt}(x, K_A^{-1})\right)\} \mid \\ \textit{if } x = \{\mathsf{ack}(N_A, N_B, B)\}_A \textit{ then } P \textit{ else } Q \end{array} \right)$$

For filtering duplicate messages, we write

$!c_1(x \setminus V).\textit{if } x \textit{ fresh then } P \textit{ else } Q$ for the process

$$\nu c. \, (\overline{c}\langle V\rangle \mid !c_1(x).c(s).(\overline{c}\langle s.x\rangle \mid \textit{if } x \in s \textit{ then } Q \textit{ else } P))$$

# Compliant configurations

- We need to make hypothesis on users
  - A principal is compliant when it uses its decryption key only according to our protocol
  - Access to the control channels is restricted to that principal

- A single compliant principal is of the form

$$Q_A \stackrel{\text{def}}{=} \nu\mathcal{V}_A.\big(U_A \mid \mathit{PK}_A\,[P_A]\big)$$

with $\mathcal{V}_A \stackrel{\text{def}}{=} \{\mathit{init}_X, \mathit{accept}_X, \mathit{connect}_X\}$

# Compliant configurations

- We need to make hypothesis on users
  - A principal is compliant when it uses its decryption key only according to our protocol
  - Access to the control channels is restricted to that principal

- A single compliant principal is of the form

$$Q_A \stackrel{\text{def}}{=} \nu \mathcal{V}_A.\big(U_A \mid \mathsf{PK}_A\,[P_A]\big)$$

- An assembly of compliant principals with a single compound user protocol is of the form

$$Q \stackrel{\text{def}}{=} \nu \mathcal{V}.\big(U \mid P\big)$$

$$P \stackrel{\text{def}}{=} \prod_{A \in \mathcal{C}} \mathsf{PK}_A\,[P_A]$$

# Authentication and Secrecy

**Theorem 1 [Complete runs]**

Let $A, B \in \mathcal{C}$.

If $P \xrightarrow{\rho} P'$ and $A \in S_B$, then $P' \xrightarrow{\omega} P'_{x1} \mid \varphi$.

If $P \xrightarrow{\rho} P'$ and $A \notin S_B$, then $P' \xrightarrow{\omega^-} P'_{x1} \mid \varphi^-$.

Conversely,

if $P \xrightarrow{\omega} P''$, then $A \in S_B$ and $P'' \equiv P_{x1} \mid \varphi$

$$\begin{aligned}
\xrightarrow{exch} &\stackrel{\text{def}}{=} \xrightarrow{\nu x_1.c_1[x_1]} \rightarrow^* \xrightarrow{\nu x_2.c_2[x_2]} \rightarrow \\
\xrightarrow{\omega} &\stackrel{\text{def}}{=} \xrightarrow{\mathit{init}_A(B)} \xrightarrow{exch} \xrightarrow{\nu K.\overline{\mathit{accept}}_B\langle A,K\rangle} \xrightarrow{\overline{\mathit{connect}}_A\langle B,K\rangle} \\
\xrightarrow{\omega^-} &\stackrel{\text{def}}{=} \xrightarrow{\mathit{init}_A(B)} \xrightarrow{exch}
\end{aligned}$$

$$\begin{aligned}
\varphi &\stackrel{\text{def}}{=} \nu N_A.\,(\sigma_1 \mid \nu N_B.(\sigma_2 \mid \sigma_K)) \\
\varphi^- &\stackrel{\text{def}}{=} (\nu N_A.\sigma_1) \mid (\nu N_B.\sigma_2^\circ) \\
\varphi^\circ &\stackrel{\text{def}}{=} (\nu N_A.\sigma_1^\circ) \mid (\nu N_B.\sigma_2^\circ)
\end{aligned}$$

$P_{x1}$ is $P$ with the message $x_1$ in $R_B$'s filter.

# Authentication and Secrecy

An "ideal result" with no IDs:
two fresh unrelated messages
+ a fresh session key

**Theorem 2 [Key fre**

For any $A, B \in \mathcal{C}$, if

$$
\begin{aligned}
P' \mid \varphi \;\; &\approx_l \;\; P' \mid \varphi^\circ \mid \nu N.\{K = N\} \\
&\approx_l \;\; P' \mid \varphi^- \mid \nu N.\{K = N\}
\end{aligned}
$$

The result of a "failed run":
two intercepted messages

The result of a
"successful run":
two intercepted messages
+ a computed session key

# Authentication and Secrecy

- We can reformulate these results for two principals, using transitions only for the network:

What can be observed by a passive attacker

$$P_A \mid P_B \mid \overline{\mathit{init}}_A\langle B\rangle$$

$$\to \xrightarrow{\nu x_1.c_1[x_1]} \to^* \xrightarrow{\nu x_2.c_2[x_2]} \to \approx_l$$

$$P_A \mid P_B \mid \varphi^\circ \mid$$
$$\begin{cases} \nu N.(\overline{\mathit{connect}}_A\langle B, N\rangle \mid \overline{\mathit{accept}}_B\langle A, N\rangle) \text{ when } A \in S_B \\ \mathbf{0} \text{ otherwise} \end{cases}$$

One of the two outcomes for the protocol run

# Authentication and Secrecy

**Theorem 3 [Responder authentication]**
Let $P \xrightarrow{\rho} P'$ such that (1) $\rho$ has no internal communication on $c_1$ or $c_2$; (2) $P'$ has no output on channel $\mathit{accept}_B$.

If $\overline{\mathit{connect}}_A\langle B, K\rangle$ occurs in $\rho$,
then $P \xrightarrow{\omega}\xrightarrow{\eta} P'$ for some permutation $\omega\eta$ of $\rho$.

- Intuitively, we have a correspondence assertion on control actions: whenever $U_A$ receives a $\mathrm{connect}_A$ message…
  - A initiated the session with B
  - B accepted the session with A
  - Both parties are sharing a key as good as a fresh name
  - Intercepted messages $x_1$, $x_2$ are unrelated to A, B and K.

# Privacy Properties?

- Previous results provide privacy guarantees for each run of the protocol
- We want to reason about the observational equivalence of arbitrary compliant user processes, running multiple sessions with compliant and non-compliant principals

$$P \stackrel{\text{def}}{=} \prod_{A \in \mathcal{C}} PK_A\,[P_A]$$

$$Q \stackrel{\text{def}}{=} \nu \mathcal{V}.\big(U \mid P\big)$$

- Overall, identity protection depends on both U and P
  - A can contact E (or accepts E’s session) on its own
  - If A contacts B then E, E can infer the presence of B

  …
- How to characterize the behaviour of U in this special context?

# Blinded Transitions (1)

We capture the "information leaks" of the protocol using abstract states and ad hoc transitions

- We write ρ:U for the user process U in state ρ
- We let ρ range over finite maps from integers to sessions:

$A\ B$: an offer from $A$ not yet considered by $B$.

$A\ B\ K_i$: an offer accepted by $B$ with key $K_i$ ($A \in S_B$).

$A\ B\ -$ : an offer rejected by $B$ ($A \notin S_B$).

$A\ E$: an offer from $A$ to some non-compliant $E$.

# Blinded Transitions (2)

INIT
$$\frac{U \xrightarrow{\overline{init}_A\langle B\rangle} U'}{\rho : U \xrightarrow{\overline{init}\;\nu i} \rho[i \mapsto A\ B] : U'}$$

The user protocol attempts a session from A to B.

The environment detects a new "opaque" session attempt (no A,B).

The session details are recorded into the abstract state.

ce

$\rangle$ if $A \in S_B$

if $A \notin S_B$

$$\begin{cases} \rho : U \mid \nu N.\overline{accept}_B\langle A, N\rangle & \text{if } A \in S_B \\ \rho : U & \text{if } A \notin S_B \end{cases}$$

CONNECT

$$\rho[i \mapsto A\ B\ K_i] : U \xrightarrow{connect\ i} \rho : \nu K_i.\left(U \mid \overline{connect}_A\langle B, K_i\rangle\right)$$

$$\rho[i \mapsto A\ B\ -\ ] : U \xrightarrow{connect\ i} \rho : U$$

# Blinded Transitions (2)

The environment enables some progress on session i

Actual progress depends on the hidden A and B, and may yield a new key & an accept message (or not)

$\rho[i \mapsto \ldots A\ B] : U'$

ACCEPT

$$\rho[i \mapsto A\ B] : U \xrightarrow{\textsf{accept}\ i} \begin{cases} \rho[i \mapsto A\ B\ K_i] : U \mid \overline{\textsf{accept}}_B\langle A, K_i\rangle & \text{if } A \in S_B \\ \rho[i \mapsto A\ B\ -\ ] : U & \text{if } A \notin S_B \end{cases}$$

The session details are updated in the session state

ACCEPT-FAKE

$$\rho : U \xrightarrow{\textsf{accept}_B(A)} \begin{cases} \rho : U \mid \nu \ldots \\ \rho : U \end{cases}$$

CONNECT

$$\rho[i \mapsto A\ B\ K_i] : U \xrightarrow{\textsf{connect}\ i} \rho : \nu K_i. \left(U \mid \overline{\textsf{connect}}_A\langle B, K_i\rangle\right)$$

$$\rho[i \mapsto A\ B\ -\ ] : U \xrightarrow{\textsf{connect}\ i} \rho : U$$

# An Equivalence for User Processes

*Private bisimilarity* ($\approx_{\mathcal{C}}$) is the largest symmetric relation $\mathcal{R}$ on extended processes with control state such that, whenever $T_1 \; \mathcal{R} \; T_2$ with $T_\ell = \rho_\ell : U_\ell$:

1. $\nu \mathcal{V}_\rho . U_1 \approx_s \nu \mathcal{V}_\rho . U_2$,
2. if $T_1 \rightarrow T_1'$, then $T_2 \rightarrow^* T_2'$ and $T_1' \; \mathcal{R} \; T_2'$
3. if $T_1 \xrightarrow{\gamma} T_1'$ and $fv(\gamma) \ldots$ then $T_2 \rightarrow^* \xrightarrow{\gamma} \rightarrow^* T_2'$ and $T_1' \; \mathcal{R} \; T_2'$

a standard definition of labelled bisimilarity, for blinded transitions

# An Equivalence for User Processes (2)

**Lemma [Privacy]** If $U_1 \approx_{\mathcal{C}}^{+} U_2$, then $Q(U_1) \approx_l Q(U_2)$.

- The hypothesis deals with arbitrary user processes
  It does not depend on the protocol (just its interface)
  and does not (necessarily) involve cryptography

- The resulting equivalence states that the compliant configurations are undistinguishable, for all contexts

# Some Derived Privacy Properties

- Consider user processes $U_1$, $U_2$
  that consist only of init messages.

  Informally, these user protocols attempt to open many sessions in parallel, and do nothing visible after key establishment.

  Such processes are (privately) equivalent when…

  1. They have the same number of messages
  2. They have the same messages to non-compliant principals
  3. They have the same non-compliant principals in $S_B$

- Two session attempts are privately equivalent as soon as their triggered processes are privately equivalent (optimal)

- We can add or remove silent compliant participants

# Private Authentication (Summary)

- Protocol designers define message formats, rather than protocol properties. Writing down precise statements for their intended properties is quite hard, but often reveals problems.
- There is a tension between privacy and authentication, with useful trade-offs in protocol design
- Privacy is more "global" than authentication and secrecy; it requires a fine model of user behaviour
- We studied a simple protocol with strong privacy properties
  - We used an applied pi calculus model
  - We relied on contexts & equivalences to reason on privacy
  - We related any user behaviours to their visible effect for the attacker using blinded transitions

# Questions on Privacy ?