02244 Language-Based Security
Part 1: Security Protocols: Syntax and Semantics

Sebastian Mödersheim

March 2018

Protocol Security

Date	Topic
12. March	Modeling Protocols: Syntax and Semantics I
	Announcement of the second project report.
19. March	Modeling Protocols: Syntax and Semantics II
9. April	Overview: Automated Reasoning for Security Protocol
16. April	Protocol Analysis: The Lazy Intruder
23. April	Protocol Analysis: Abstraction
30. April	Channels and Protocol Composition
7. May	Case Studies
	Hand in of second project report at noon

Roadmap

Introduction to:

- Black-box models of cryptography
- Security protocols
- AnB and OFMC

Programme:

1. Construction of a key-exchange protocol
2. The Syntax of AnB
3. From AnB to strands: intuition
4. Term Algebra and all that
5. The Dolev-Yao Intruder Model
6. Transition Systems
7. Security Goals

Textbook

- There are some textbooks on security protocols
  - ★ do not cover all topics of the course.
- There are some research papers on protocol verification
  - ★ tough to read.
- *Protocol Verification Tutorial*: an introduction to protocol verification that comes with the tool OFMC.
  - ★ We currently extend it to cover all topic of the protocol part of the course (and some topics we cannot cover in the course).
  - ★ Hopefully a nice way for you to read up on all topics.
  - ★ Questions, comments and feedback most welcome!

Construction of a key-exchange protocol

- Inspired by the first chapter in
  Boyd and Mathuria: *Protocols for Authentication and Key Establishment*
- We write the protocols in AnB and use OFMC to find attacks in them.
- Addition: using Diffie-Hellman in the key-exchange.

Roadmap

1. Construction of a key-exchange protocol
2. The Syntax of AnB
3. From AnB to strands: intuition
4. Term Algebra and all that
5. The Dolev-Yao Intruder Model
6. Transition Systems
7. Security Goals

AnB Syntax

```
Protocol:   NSSK
Types:      Agent A,B,s;
            Number NA,NB;
            Symmetric_key KAB;
            Function sk,pre
Knowledge:  A: A,B,s,sk(A,s),pre;
            B: A,B,s,sk(B,s),pre;
            s: A,B,s,sk(A,s),sk(B,s),pre
Actions:
  A->s: A,B,NA
  s->A: {| KAB,B,NA, {| KAB,A |}sk(B,s)  |}sk(A,s)
  A->B: {| KAB,A |}sk(B,s)
  B->A: {| NB |}KAB
  A->B: {| pre(NB) |}KAB
Goals:
  A authenticates s on KAB,B
  B authenticates s on KAB,A
  KAB secret between A,B,s
```

AnB: Things to Note

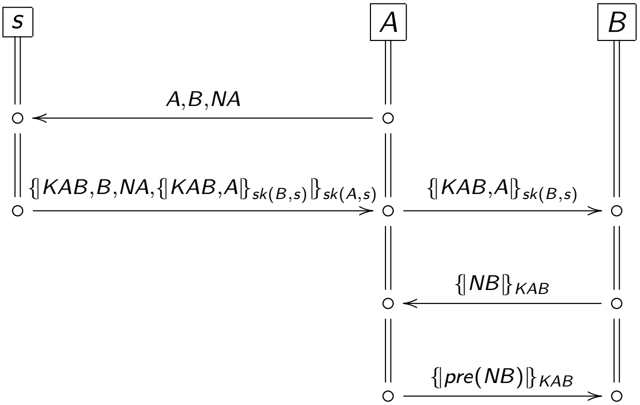
- Identifiers that start with uppercase: variables (E.g., A, B, KAB)
- Identifiers that start with lowercase: constants and functions (E.g., s, pre, sk)
- One should declare a type for all identifiers; OFMC can search for *type-flaw* attacks when using the option `-untyped` (in which case all types are ignored).
- The (initial) knowledge of agents MUST NOT contain variables of any type other than `Agent`.
  - ★ For long-term keys, passwords, etc. use functions like $sk(A, B)$.
- Each variable that does not occur in the initial knowledge is freshly created during the protocol by the first agent who uses it.
  - ★ In the NSSK example, A creates NA, s creates KAB, and B creates NB.

Roadmap

1. Construction of a key-exchange protocol
2. The Syntax of AnB
3. From AnB to strands: intuition
4. Term Algebra and all that
5. The Dolev-Yao Intruder Model
6. Transition Systems
7. Security Goals

AnB Protocol Specification

NSSK as Message Sequence Chart

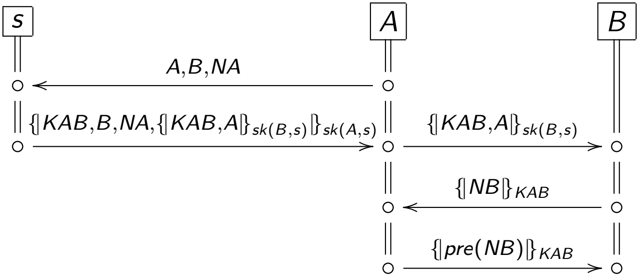


Strands and Roles

- Strand: sequence of send and receive events.
- Intuitive graphical representation, also very suitable for proofs.
- Represents one protocol execution ("session") from the point of view of one (honest) agent.
- Role: a strand with variables in messages
  - ★ Like a program (or process) for executing one protocol session.
  - ★ Representing what form of messages an agent is willing to receive, and how it will answer them.

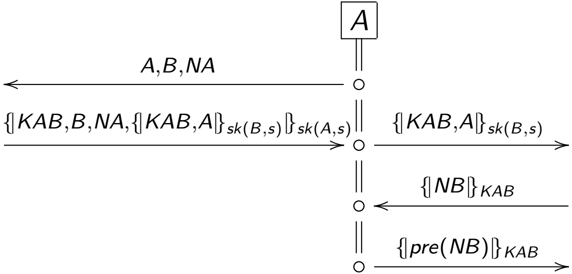
From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)



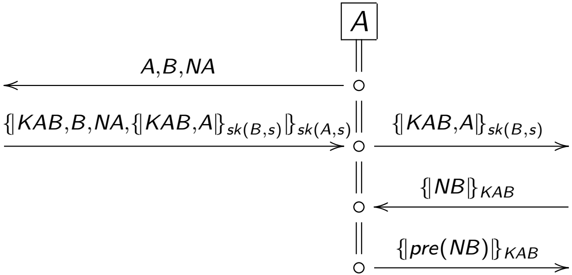
From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)



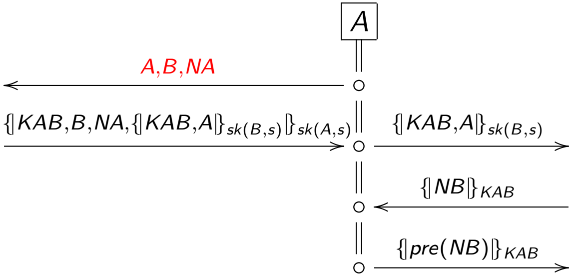
From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)



From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)

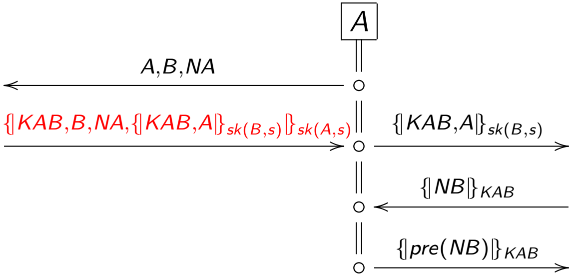


To run this, we must first choose any agent names A and B, and a fresh nonce NA (a unique constant). Replace all occurrences of these variables with the chosen values.

Then we can send the first message.

From AnB to Strands (Informal)

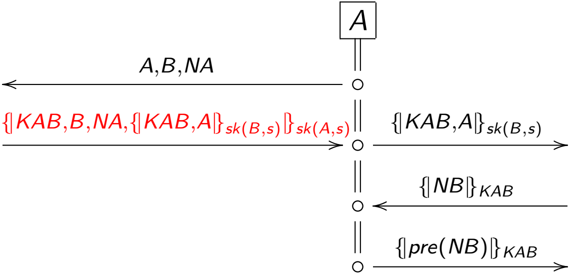
Split a message sequence chart into a set of roles: (Ex. NSSK role A)



Wait for an incoming message of the form $\{|\ \ldots\ |\}_{sk(A,s)}$.

From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)

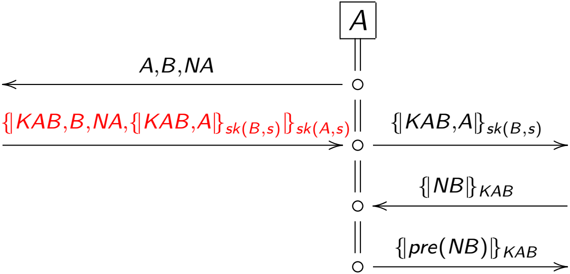


Wait for an incoming message of the form $\{\!| \dots |\!\}_{sk(A,s)}$.

- Accept only if encrypted with $sk(A,s)$

From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)

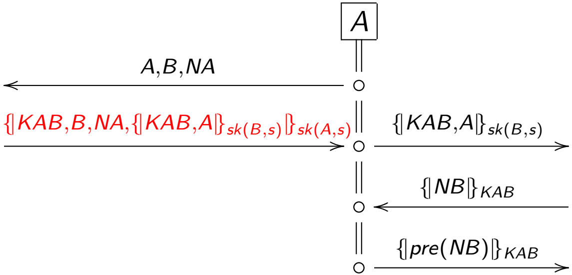


Wait for an incoming message of the form $\{\!|\ldots|\!\}_{sk(A,s)}$.

- Accept only if encrypted with $sk(A,s)$
- KAB can be anything, we learn it from this message.

From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)

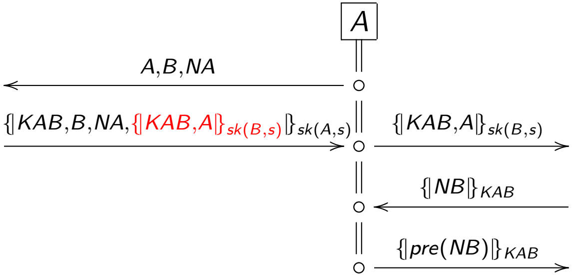


Wait for an incoming message of the form $\{\!| \ldots |\!\}_{sk(A,s)}$.

- Accept only if encrypted with $sk(A,s)$
- KAB can be anything, we learn it from this message.
- B and NA must be the same value as in the first message.

From AnB to Strands (Informal)

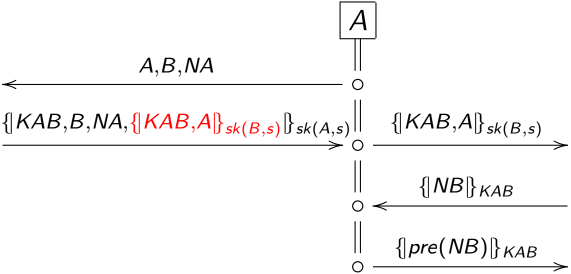
Split a message sequence chart into a set of roles: (Ex. NSSK role A)



What about the subterm $\{|KAB, A|\}_{sk(B,s)}$?

From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)

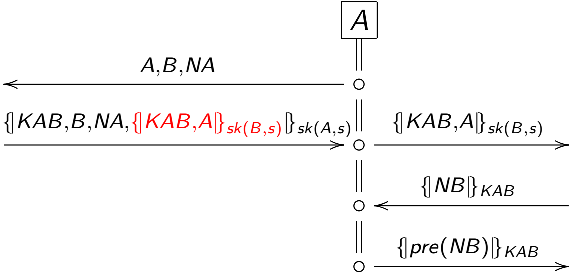


What about the subterm $\{\!|KAB, A|\!\}_{sk(B,s)}$?

- This is not right! Role A did not know the key $sk(B, s)$.

From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)

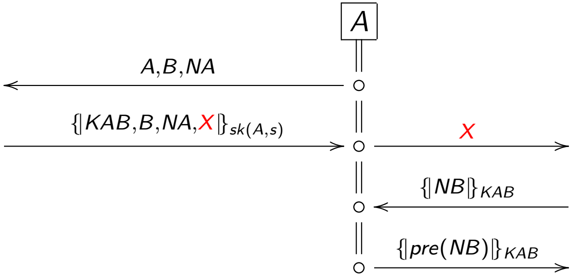


What about the subterm $\{|KAB, A|\}_{sk(B,s)}$?

- This is not right! Role A did not know the key $sk(B, s)$.
- There is no way that A can check the decryption!

From AnB to Strands (Informal)

Split a message sequence chart into a set of roles: (Ex. NSSK role A)



What about the subterm $\{|KAB, A|\}_{sk(B,s)}$?

- This is not right! Role A did not know the key $sk(B, s)$.
- There is no way that A can check the decryption!
- We model this by a new variable X that represents an arbitrary message at this place!

From AnB to Strands (Informal)

- Translation from AnB to roles must take care of
  - ★ what agents can actually check on incoming messages
  - ★ how they can generate outgoing messages

From AnB to Strands (Informal)

- Translation from AnB to roles must take care of
  - what agents can actually check on incoming messages
  - how they can generate outgoing messages
- This depends on the concrete knowledge that an agent has at a given point, as well as the properties of cryptographic operations (e.g. Diffie-Hellman is subtle).

From AnB to Strands (Informal)

- Translation from AnB to roles must take care of
  - ★ what agents can actually check on incoming messages
  - ★ how they can generate outgoing messages
- This depends on the concrete knowledge that an agent has at a given point, as well as the properties of cryptographic operations (e.g. Diffie-Hellman is subtle).
- For details: see Protocol Security Verification Tutorial, section 10.

AnB Roles: Instantiation

Roles can contain three kinds of variables:

1. Variables that occur in the knowledge section
   - These MUST be of type `agent`.
   - They are initially instantiated with concrete agent names, including the intruder i.
   - In attack traces of OFMC, you often may see variables here (when the concrete value does not matter for the attack).
2. Variables that represent freshly generated constants
   - Variables that first occur in a sent message by the role
   - Are initially instantiated in with a fresh constant (that did not occur before).
   - In OFMC they are denoted by $N(i)$ where N is the name in the AnB specification, and i is a (step) number.
3. Variables that represent received message parts
   - All remaining variables of a role.
   - Instantiated upon receiving a message.

Instantiation: Example

- Protocol with two roles A and B,
- consider two sessions, and
- let us fix the set of all existing agents to $\{a, b, i\}$

What instantiations can we have?

Instantiation: Example

- Protocol with two roles A and B,
- consider two sessions, and
- let us fix the set of all existing agents to $\{a, b, i\}$

What instantiations can we have?

- Choosing: $A_1, B_1, A_2, B_2 \in \{a, b, i\}$

Instantiation: Example

- Protocol with two roles A and B,
- consider two sessions, and
- let us fix the set of all existing agents to $\{a, b, i\}$

What instantiations can we have?

- Choosing: $A_1, B_1, A_2, B_2 \in \{a, b, i\}$
- That's 3^4 session instances to consider...

Instantiation: Example

- Protocol with two roles A and B,
- consider two sessions, and
- let us fix the set of all existing agents to $\{a, b, i\}$

What instantiations can we have?

- Choosing: $A_1, B_1, A_2, B_2 \in \{a, b, i\}$
- That's 3^4 session instances to consider...
- OFMC: leave variables uninstantiated as long as concrete name does not matter for the attack. (This is why you see variables in attack traces.)

Roadmap

1. Construction of a key-exchange protocol
2. The Syntax of AnB
3. From AnB to strands: intuition
4. Term Algebra and all that
5. The Dolev-Yao Intruder Model
6. Transition Systems
7. Security Goals

Message Term Algebra

for security protocols

Symbol	Arity	Meaning	Public
i	0	name of the intruder	yes
inv	1	private key of a given public key	no
crypt	2	asymmetric encryption in AnB: write $\{m\}_k$ for crypt(k, m)	yes
scrypt	2	symmetric encryption in AnB: write $\{\|m\|\}_k$ for scrypt(k, m)	yes
pair	2	pairing/concatenation in AnB: write m, n for pair(m, n)	yes
$\exp(\cdot, \cdot)$	2	exponentiation modulo fixed prime p	yes
$a, b, c, \ldots$	0	User-defined constants	User-def.
$f(\cdot)$	$\star$	User-defined function symbol f	User-def.

- Call Σ the set of all function symbols and Σ_p the public ones.
- Public functions can be applied by every agent
- inv is not public: the private key of a given public key.

Terms

Definition (Signature)

A signature Σ is a set of function symbols with an arity.
Constant: function symbol with arity 0

Terms

Definition (Signature)

A signature Σ is a set of function symbols with an arity.
Constant: function symbol with arity 0

Definition (Terms)

Let $V = \{X, Y, Z, \ldots\}$ be variable symbols.
Define the terms (over Σ and V), denoted $\mathcal{T}_\Sigma(V)$:

- All variables of V are terms
- If $t_1, \ldots, t_n$ are terms and $f/n \in \Sigma$ (for some $n \geq 0$), then also $f(t_1, \ldots, t_n)$ is a term.

Terms

Definition (Signature)

A signature Σ is a set of function symbols with an arity.
Constant: function symbol with arity 0

Definition (Terms)

Let $V = \{X, Y, Z, \ldots\}$ be variable symbols.
Define the terms (over Σ and V), denoted $\mathcal{T}_\Sigma(V)$:

- All variables of V are terms
- If $t_1, \ldots, t_n$ are terms and $f/n \in \Sigma$ (for some $n \geq 0$), then also $f(t_1, \ldots, t_n)$ is a term.

Example: $\Sigma = \{scrypt/2, pair/2, i/0, k/0\}$
$\mathcal{T}_\Sigma(\{X, Y\})$ contains the *atomic* terms X, Y, i, k; composed terms can be obtained with *scrypt* and *pair*, e.g. $scrypt(k, X)$, or $scrypt(k, pair(scrypt(X, Y), i))$.

Free Algebra

It is standard to interpret messages in the free algebra:

- Two terms are equal iff they are syntactically equal.
- We thus do not consider any algebraic equations like $f(a, b) = f(b, a)$.
- Terms have no "deeper meaning" attached.

Free Algebra

It is standard to interpret messages in the free algebra:

- Two terms are equal iff they are syntactically equal.
- We thus do not consider any algebraic equations like $f(a, b) = f(b, a)$.
- Terms have no "deeper meaning" attached.

This model has some desirable properties:

- Easy to compute with...

Free Algebra

It is standard to interpret messages in the free algebra:

- Two terms are equal iff they are syntactically equal.
- We thus do not consider any algebraic equations like $f(a, b) = f(b, a)$.
- Terms have no "deeper meaning" attached.

This model has some desirable properties:

- Easy to compute with...
- $a \neq b$ for any distinct constants a and b

Free Algebra

It is standard to interpret messages in the free algebra:

- Two terms are equal iff they are syntactically equal.
- We thus do not consider any algebraic equations like $f(a,b) = f(b,a)$.
- Terms have no “deeper meaning” attached.

This model has some desirable properties:

- Easy to compute with...
- $a \neq b$ for any distinct constants a and b
- If $m_1 \neq m_2$ then also $h(m_1) \neq h(m_2)$.

Free Algebra

It is standard to interpret messages in the free algebra:

- Two terms are equal iff they are syntactically equal.
- We thus do not consider any algebraic equations like $f(a, b) = f(b, a)$.
- Terms have no "deeper meaning" attached.

This model has some desirable properties:

- Easy to compute with...
- $a \neq b$ for any distinct constants a and b
- If $m_1 \neq m_2$ then also $h(m_1) \neq h(m_2)$.
  (Thus: hash functions are collision free)

Free Algebra

It is standard to interpret messages in the free algebra:

- Two terms are equal iff they are syntactically equal.
- We thus do not consider any algebraic equations like $f(a, b) = f(b, a)$.
- Terms have no "deeper meaning" attached.

This model has some desirable properties:

- Easy to compute with...
- $a \neq b$ for any distinct constants a and b
- If $m_1 \neq m_2$ then also $h(m_1) \neq h(m_2)$.
  (Thus: hash functions are collision free)

Some properties are not so nice:

- We cannot define a decryption function with the property $dcrypt(k, scrypt(k, m)) = m$.
- We cannot model algebraic properties of operators that we need for instance for Diffie-Hellman.

Substitutions

Definition (Substitution)

A substitution has the form

$$\sigma = [X_1 \mapsto t_1, \ldots, X_n \mapsto t_n]$$

where the X_i are variables and the t_i are terms.

We call the set $\{X_1, \ldots, X_n\}$ the domain of σ.

A substitution σ represents a function on terms in general:

- Every variable X_i in the domain is mapped to the respective t_i.
- For any variable X that is not in the domain: $\sigma(X) = X$.
- For a composed term $f(t_1, \ldots, t_n)$ we define:

$$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$$

Substitutions

Example

$$\sigma = [X \mapsto f(Z), Y \mapsto Z]$$

$$\sigma(g(Z, Y, f(X))) =$$

Roadmap

Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

- Every user has a public/private key pair.
- Every user knows the public key of every other user.

Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

- Every user has a public/private key pair.
- Every user knows the public key of every other user.
- The Dolev-Yao intruder:
  - ★ The intruder is also a user with his own key pair.
  - ★ The intruder can decrypt only messages that are "meant" for him, i.e., that are encrypted with his public key.
  - ★ The intruder controls the network (read, intercept, send)

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret
- Intruder controls entire communication medium. Realistic?

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret
- Intruder controls entire communication medium. Realistic?
  - ★ Worst-case assumption (what is not secured may be infected)

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret
- Intruder controls entire communication medium. Realistic?
  - ★ Worst-case assumption (what is not secured may be infected)
- The intruder can act as a participant. Why that?

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret
- Intruder controls entire communication medium. Realistic?
  - ★ Worst-case assumption (what is not secured may be infected)
- The intruder can act as a participant. Why that?
  - ★ Modeling a dishonest participant.

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret
- Intruder controls entire communication medium. Realistic?
  - ★ Worst-case assumption (what is not secured may be infected)
- The intruder can act as a participant. Why that?
  - ★ Modeling a dishonest participant.
  - ★ Some attacks do not work when all participants are honest.

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ There simply is no "break-crypto" or "flip some bits" operation.
- Kerckhoffs's principle
  - ★ Encryption and decryption algorithms are not secret
- Intruder controls entire communication medium. Realistic?
  - ★ Worst-case assumption (what is not secured may be infected)
- The intruder can act as a participant. Why that?
  - ★ Modeling a dishonest participant.
  - ★ Some attacks do not work when all participants are honest.
- Nowadays, many similar Dolev-Yao-style models are used:
  - ★ Term Algebra to model messages
  - ★ Sometimes considering some Algebraic Properties
  - ★ Intruder Deduction Relation $\vdash$ on terms

Intruder Deduction

The core of the Dolev-Yao model is a definition what the intruder can do with messages.

- We define a relation $M \vdash m$ where
  - ★ M is a set of messages
  - ★ m is a message
  - ★ expressing that the intruder can derive m, if his knowledge is M.

Example

$$M = \{\, k_1, \{\!|m_1|\!\}_{k_1}, m_2, \{\!|m_3|\!\}_{k_2} \,\}$$

Then for instance we should have:

- $M \vdash m_1$
- $M \nvdash m_3$
- $M \vdash \{\!|\langle m_1, m_2\rangle|\!\}_{k_1}$
- ...

Dolev-Yao Closure

Definition

We define $\vdash$ as the *least* relation that satisfies the following rules:

$$\frac{}{M \vdash m} \text{ if } m \in M \text{ (Axiom)}$$

$$\frac{M \vdash m_1 \quad \ldots \quad M \vdash m_n}{M \vdash f(m_1, \ldots, m_n)} \text{ if } f/n \in \Sigma_p \text{ (Compose)}$$

$$\frac{M \vdash \langle m_1, m_2 \rangle}{M \vdash m_i} \text{ (Proj}_i\text{)} \qquad \frac{M \vdash \{\!|m|\!\}_k \quad M \vdash k}{M \vdash m} \text{ (DecSym)}$$

$$\frac{M \vdash \{m\}_k \quad M \vdash \mathrm{inv}(k)}{M \vdash m} \text{ (DecAsym)} \qquad \frac{M \vdash \{m\}_{\mathrm{inv}(k)}}{M \vdash m} \text{ (OpenSig)}$$

Here, we write $\langle m_1, m_2 \rangle$ for $pair(m_1, m_2)$ to be close to the nice notation of AnB, but having an explicit symbol for the pair operation.

Example: Intruder Deduction

Example

$M = \{\ a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\langle na, a\rangle\}_{\mathsf{pk}(i)}\}$

Can the intruder derive $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(i)}} \quad \dfrac{}{M \vdash \mathsf{inv}(\mathsf{pk}(i))}}{M \vdash \langle na, a\rangle} \quad \dfrac{}{M \vdash \mathsf{pk}(b)}}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(b)}}$$

Example

$M = \{\, a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\langle na, a\rangle\}_{\mathsf{pk}(i)} \}$

Can the intruder derive $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(i)}} \quad \dfrac{}{M \vdash \mathsf{inv}(\mathsf{pk}(i))}}{M \vdash \langle na, a\rangle} \qquad \dfrac{}{M \vdash \mathsf{pk}(b)}}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(b)}}$$

Example

$M = \{\ a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\langle na, a\rangle\}_{\mathsf{pk}(i)}\}$

Can the intruder derive $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(i)}} \quad \dfrac{}{M \vdash \mathsf{inv}(\mathsf{pk}(i))}}{M \vdash \langle na, a\rangle} \quad \dfrac{}{M \vdash \mathsf{pk}(b)}}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(b)}}$$

Example: Intruder Deduction

Example

$M = \{\ a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\langle na, a\rangle\}_{\mathsf{pk}(i)}\}$

Can the intruder derive $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(i)}} \quad \dfrac{}{M \vdash \mathsf{inv}(\mathsf{pk}(i))}}{M \vdash \langle na, a\rangle} \qquad \dfrac{}{M \vdash \mathsf{pk}(b)}}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(b)}}$$

Example: Intruder Deduction

Example

$M = \{\ a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\langle na, a\rangle\}_{\mathsf{pk}(i)}\}$

Can the intruder derive $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(i)}} \quad \dfrac{}{M \vdash \mathsf{inv}(\mathsf{pk}(i))}}{M \vdash \langle na, a\rangle} \quad \dfrac{}{\color{red}{M \vdash \mathsf{pk}(b)}}}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(b)}}$$

Example: Intruder Deduction

Example

$M = \{\ a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i)), \{\langle na, a\rangle\}_{\mathsf{pk}(i)}\}$

Can the intruder derive $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(i)}} \quad \dfrac{}{M \vdash \mathsf{inv}(\mathsf{pk}(i))}}{M \vdash \langle na, a\rangle} \quad \dfrac{}{M \vdash \mathsf{pk}(b)}}{\color{red}{M \vdash \{\langle na, a\rangle\}_{\mathsf{pk}(b)}}}$$

Automation?

- How can we prove negative statements?
  - ★ in the previous example for instance that $M \nvdash \mathsf{inv}(\mathsf{pk}(a))$
- Can one build an algorithm that
  - ★ given a set M and a term t
  - ★ check whether or not $M \vdash t$?
- What would be the complexity of such an algorithm?

Diffie-Hellman

Development of a simple Diffie-Hellman based protocol in AnB.

- Exponentiation in a group g modulo prime p (we always omit modulus p in the AnB notation).
- A creates a fresh secret X, and computes public value $\exp(g, X)$.
- B creates Y and computes $\exp(g, Y)$.
- A and B somehow exchange the public values in an authentic way.
- A computes $K_A = \exp(\exp(g, Y), X)$
- B computes $K_B = \exp(\exp(g, X), Y)$
- They now have a shared secret key $K_A = K_B$.

This requires, however, that AnB/OFMC understands the algebraic property

$$\exp(\exp(g, X), Y) \approx \exp(\exp(g, Y), X)$$

Such an equation makes all reasoning much harder!

Dolev-Yao Closure

Definition

We define $\vdash$ as the *least* relation that satisfies the following rules:

$$\frac{}{M \vdash m} \text{ if } m \in M \text{ (Axiom)}$$

$$\frac{M \vdash m_1 \quad \dots \quad M \vdash m_n}{M \vdash f(m_1, \dots, m_n)} \text{ if } f/n \in \Sigma_p \text{ (Compose)}$$

$$\frac{M \vdash \langle m_1, m_2 \rangle}{M \vdash m_i} \text{ (Proj}_i\text{)} \qquad \frac{M \vdash \{\!|m|\!\}_k \quad M \vdash k}{M \vdash m} \text{ (DecSym)}$$

$$\frac{M \vdash \{m\}_k \quad M \vdash \mathrm{inv}(k)}{M \vdash m} \text{ (DecAsym)} \qquad \frac{M \vdash \{m\}_{\mathrm{inv}(k)}}{M \vdash m} \text{ (OpenSig)}$$

$$\frac{M \vdash s}{M \vdash t} \text{ if } s \approx_E t \text{ (Algebra)}$$

Example

$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}\,?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \dfrac{}{M \vdash k}}{M \vdash \langle b, \exp(g, y)\rangle}}{M \vdash \exp(g, y)} \quad \dfrac{}{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \quad \dfrac{}{M \vdash m}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}$$

Example

$M = \{\ x, \{|b, \exp(g,y)|\}_k, k, m\ \}$

$$M \vdash \{|m|\}_{\exp(\exp(g,x),y)}?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{|b,\exp(g,y)|\}_k} \quad \dfrac{}{M \vdash k}}{M \vdash \langle b,\exp(g,y)\rangle}}{M \vdash \exp(g,y)} \quad \dfrac{}{M \vdash x}}{M \vdash \exp(\exp(g,y),x)}}{M \vdash \exp(\exp(g,x),y)} \quad \dfrac{}{M \vdash m}}{M \vdash \{|m|\}_{\exp(\exp(g,x),y)}}$$

Intruder Deduction

Example

$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \dfrac{}{M \vdash k}}{\color{red}{M \vdash \langle b, \exp(g, y)\rangle}}}{M \vdash \exp(g, y)} \quad \dfrac{}{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \quad \dfrac{}{M \vdash m}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}$$

Intruder Deduction

Example

$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \dfrac{}{M \vdash k}}{M \vdash \langle b, \exp(g, y)\rangle}}{M \vdash \exp(g, y)} \qquad \dfrac{}{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \qquad \dfrac{}{M \vdash m}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}$$

Intruder Deduction

Example

$M = \{\ x, \{\!|b, \exp(g,y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}\,?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|b, \exp(g,y)|\!\}_k} \quad \dfrac{}{M \vdash k}}{M \vdash \langle b, \exp(g,y)\rangle}}{M \vdash \exp(g,y)} \quad \dfrac{}{\color{red}{M \vdash x}}}{M \vdash \exp(\exp(g,y),x)}}{M \vdash \exp(\exp(g,x),y)} \quad \dfrac{}{M \vdash m}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}}{}$$

Intruder Deduction

Example

$$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}\,?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \dfrac{}{M \vdash k}}{M \vdash \langle b, \exp(g, y)\rangle}}{M \vdash \exp(g, y)} \qquad \dfrac{}{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \qquad \dfrac{}{M \vdash m}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}$$

Intruder Deduction

Example

$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \overline{M \vdash k}}{M \vdash \langle b, \exp(g, y)\rangle}}{M \vdash \exp(g, y)} \qquad \overline{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \qquad \overline{M \vdash m}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}$$

Example

$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \dfrac{}{M \vdash k}}{M \vdash \langle b, \exp(g, y)\rangle}}{M \vdash \exp(g, y)} \qquad \dfrac{}{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \qquad \dfrac{}{\color{red}{M \vdash m}}}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}$$

Intruder Deduction

Example

$M = \{\ x, \{\!|b, \exp(g, y)|\!\}_k, k, m\ \}$

$$M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}?$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{M \vdash \{\!|b, \exp(g, y)|\!\}_k} \quad \overline{M \vdash k}}{M \vdash \langle b, \exp(g, y)\rangle}}{M \vdash \exp(g, y)} \quad \overline{M \vdash x}}{M \vdash \exp(\exp(g, y), x)}}{M \vdash \exp(\exp(g, x), y)} \quad \overline{M \vdash m}}{\color{red}{M \vdash \{\!|m|\!\}_{\exp(\exp(g,x),y)}}}$$

Roadmap

Strands

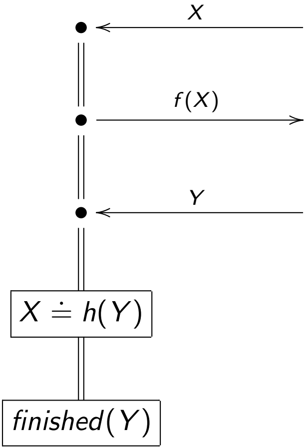
Definition

A strand is a sequence of steps where each step is either

- $\mathsf{Snd}(t)$ for sending a message t.
- $\mathsf{Rcv}(t)$ for receiving a message t.
- $\mathsf{s} \doteq \mathsf{t}$ for checking whether two terms s and t are equal.
- $\mathsf{Evt}(t)$ generates a special event t

Graphical Notation

E.g. the strand
$\mathrm{Rcv}(X).\mathrm{Snd}(f(X)).\mathrm{Rcv}(Y).X \doteq h(Y).\mathrm{Evt}(\mathit{finished}(Y))$:



From AnB to Strands

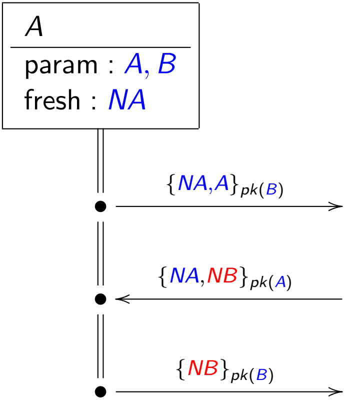
The AnB specification of a protocol describes the behavior of honest agents:

Input AnB specification:

$Knowledge:$
$A : A, B, pk(A), pk(B), inv(pk(A))$
$B : A, B, pk(A), pk(B), inv(pk(B))$

$Actions:$
$A \to B : \quad \{NA, A\}_{pk(B)}$
$B \to A : \quad \{NA, NB\}_{pk(A)}$
$A \to B : \quad \{NB\}_{pk(B)}$
$Goals:$
...

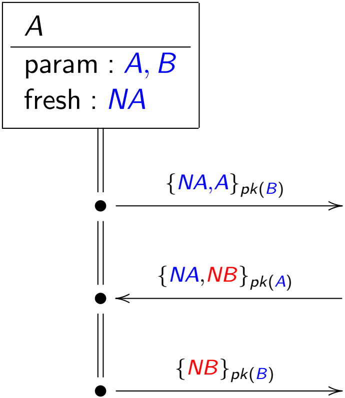
Output Strand for role A:



Instantiating Variables

For executing a role we need to

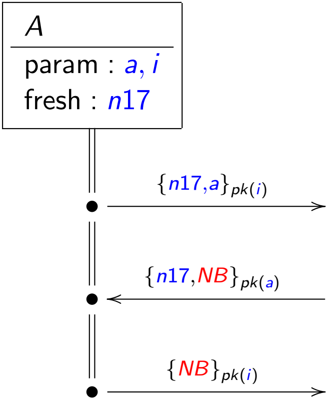
- Instantiate all parameters with agent names
  - ★ E.g. $\sigma = [A \mapsto a, B \mapsto i]$
- Instantiate all fresh variables with **unique** constants
  - ★ E.g. $\sigma = [NA \mapsto n17]$
- All remaining variables first occur in incoming messages.
  - ★ They are bound when this message is received.



Instantiating Variables

For executing a role we need to

- Instantiate all parameters with agent names
  - ★ E.g. $\sigma = [A \mapsto a, B \mapsto i]$
- Instantiate all fresh variables with **unique** constants
  - ★ E.g. $\sigma = [NA \mapsto n17]$
- All remaining variables first occur in incoming messages.
  - ★ They are bound when this message is received.



Free and Bound Variables

Definition

Let $S = S_1.\mathsf{Rcv}(t).S_2$ be a strand, and let X be a variable that occurs in t but not in S_1.

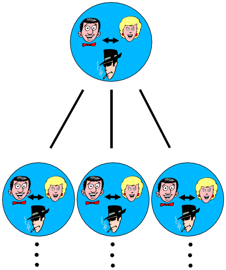
- Then we say X is bound in S by the receive step $\mathsf{Rcv}(t)$.
- We say that all variables that are not bound by such a receive step are free variables of S.
- We say that a strand is closed if it does not have free variables.

For all parts of the course – except for the lazy intruder technique – we will consider only closed strands.

State Transition Systems

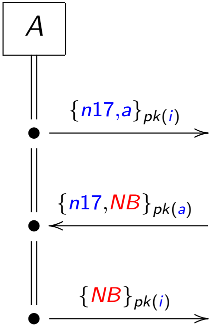
Evolution of an abstract model of the world:

- It has an initial state: Any number of instantiations of the protocol roles
- There are several transitions, i.e., ways the world can evolve from one state into a different state
  - ★ An honest agent sending a message
  - ★ An honest agent receiving a message
  - ★ An honest agent checking a condition
  - ★ An honest agent generating an event
  - ★ Actions of the intruder
- Every state consists of
  - ★ Local states of the honest agents
  - ★ The knowledge of the intruder
  - ★ Special events that we use to formulate the goals/attack states
- Define which states count as attack states



The Initial State

- For honest agents, we consider a number of role descriptions where agent names and fresh values are instantiated.
- There can be more any number of sessions between the same participants.
- There are thus infinitely many possible sessions.



The Initial Intruder Knowledge

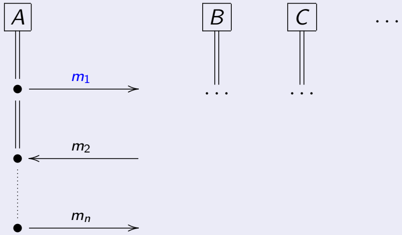
- In all sessions of the honest agents where the intruder plays one of the roles, he gets the appropriate instance of the role's knowledge.

$Knowledge\ B : A, B, pk(A), pk(B), inv(pk(B))$
Instance $[A \mapsto a, B \mapsto i]$.
Then the intruder gets $a, i, pk(a), pk(i), inv(pk(i))$.

Putting it all together

Transition: honest agents sending

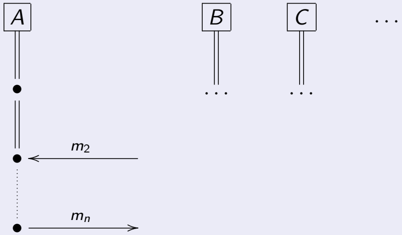
Current State



Intruder knowledge M

Events E

Possible Next State



Intruder knowledge $M \cup \{m_1\}$

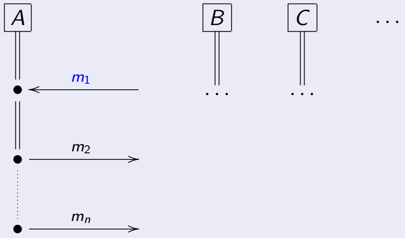
Events E

The intruder immediately learns the sent message m_1.

Putting it all together

Transition: honest agents receiving

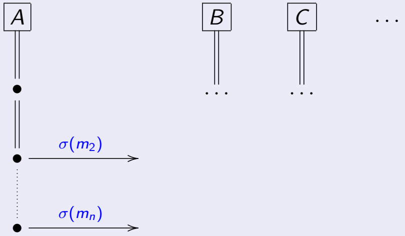
Current State



Intruder knowledge M
Events E

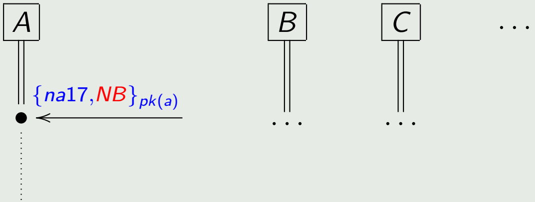
Possible Next State

if σ substitutes all variables of m_1 such that $M \vdash \sigma(m_1)$.



Intruder knowledge M
Events E

All messages that honest agents receive are chosen by the intruder.



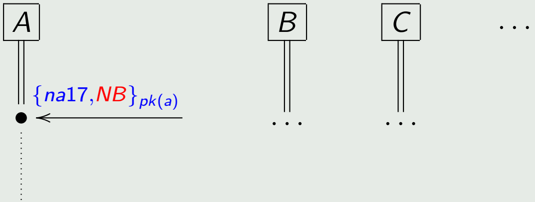
Intruder knowledge

$M = \{a, b, i, pk(a), na5, na17, \{na17, nb3\}_{pk(a)}\}$

Find all substitutions σ such that $M \vdash \sigma(\{na17, NB\}_{pk(a)})$!

$\sigma(NB) = ?$

Putting it all together



Intruder knowledge

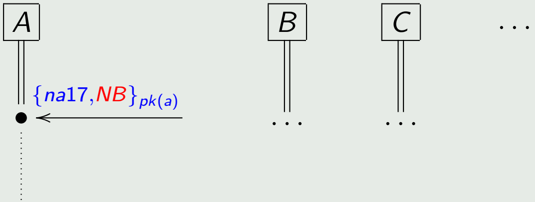
$M = \{a, b, i, pk(a), na5, na17, \{na17, nb3\}_{pk(a)}\}$

Find all substitutions σ such that $M \vdash \sigma(\{na17, NB\}_{pk(a)})$!

$\sigma(NB) = ?$

- $nb3$ (using the encrypted message)

Putting it all together



Intruder knowledge

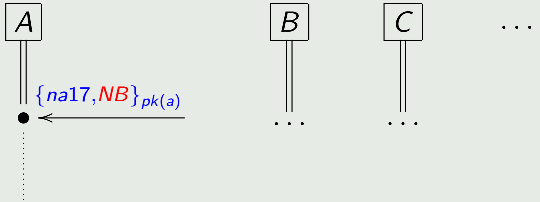
$M = \{a, b, i, pk(a), na5, na17, \{na17, nb3\}_{pk(a)}\}$

Find all substitutions σ such that $M \vdash \sigma(\{na17, NB\}_{pk(a)})$!

$\sigma(NB) = ?$

- $nb3$ (using the encrypted message)
- $na5$ or $nb17$ (construct himself)

Putting it all together



Intruder knowledge

$M = \{a, b, i, pk(a), na5, na17, \{na17, nb3\}_{pk(a)}\}$

Find all substitutions σ such that $M \vdash \sigma(\{na17, NB\}_{pk(a)})$!

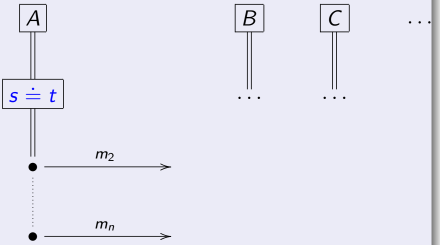
$\sigma(NB) = ?$

- $nb3$ (using the encrypted message)
- $na5$ or $nb17$ (construct himself)
- $a, b, i, pk(a), \{na17, nb3\}_{pk(a)}, \ldots$
  - ★ "ill-typed" messages, the intruder can actually use any message he can construct.
  - ★ infinite number of possible choices!

Putting it all together

Transition: checking equations

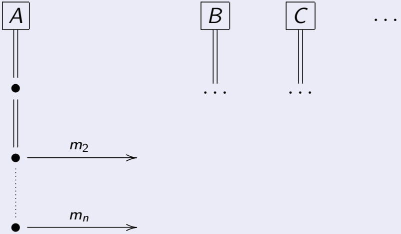
Current State



Intruder knowledge M
Events E

Possible Next State

if $s \approx_E t$



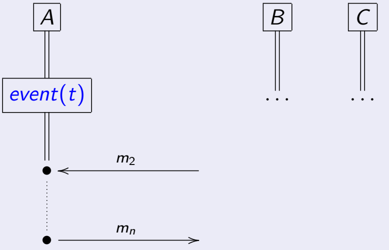
Intruder knowledge M
Events E

…otherwise this agent is stuck!

Putting it all together

Transition: Events

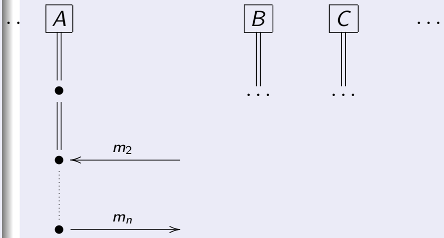
Current State



Intruder knowledge M

Events E

Possible Next State



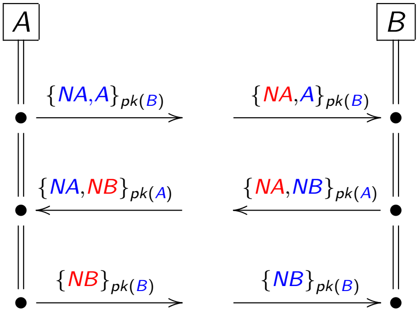
Intruder knowledge M

Events $E \cup \{t\}$

Events are simply collected when they occur.

Transitions – Example NSPK

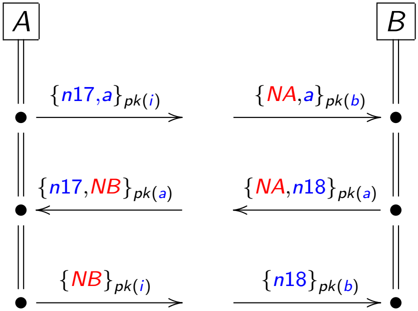
NSPK Protocol Roles:



Transitions – Example NSPK

One possible instance:

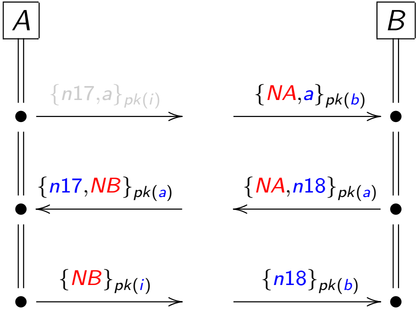
- For role A: $\sigma_A = [A \mapsto a, B \mapsto i, NA \mapsto n17]$.
- For role B: $\sigma_B = [B \mapsto b, A \mapsto a, NB \mapsto n18]$.



Initial intruder knowledge:
$M_0 = \{a, b, i, pk(a), pk(b), pk(i), \mathrm{inv}(pk(i))\}$

Transitions – Example NSPK

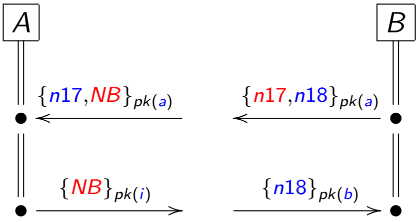
Possible transition: A sends out her first message



Intruder knowledge: $M = \{\ldots, \{n17, a\}_{pk(i)}\}$

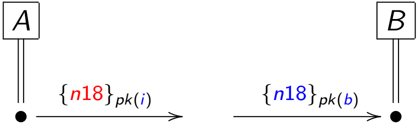
Transitions – Example NSPK

Possible transition: the intruder sends to B the message $\{n17, a\}_{pk(b)}$:



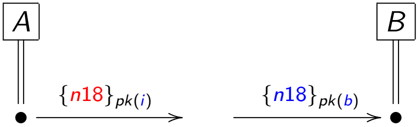
Transitions – Example NSPK

Now b can send out his reply, adding $\{n17, n18\}_{pk(a)}$ to the intruder knowledge. He cannot decrypt that, but send it to a:



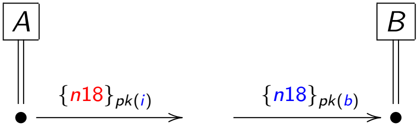
Transitions – Example NSPK

Now b can send out his reply, adding $\{n17, n18\}_{pk(a)}$ to the intruder knowledge. He cannot decrypt that, but send it to a:



Next, the intruder will learn $\{n18\}_{pk(i)}$, and thus get the secret $n18$.

Now b can send out his reply, adding $\{n17, n18\}_{pk(a)}$ to the intruder knowledge. He cannot decrypt that, but send it to a:



Next, the intruder will learn $\{n18\}_{pk(i)}$, and thus get the secret $n18$. The intruder can also complete the run with b, because he can produce $\{n18\}_{pk(b)}$.

Roadmap

Protocol Goals

Goals what the protocol should achieve, e.g

- Authenticate messages, binding them to their originator:
  `B weakly authenticates A on Key`
- Ensure timeliness of messages (recent, fresh, ...):
  `B authenticates A on Key`
- Guarantee secrecy of certain items (e.g. generated keys):
  `Key secret between A,B`

Other goals

- sender invariance, anonymity, non-repudiation (of receipt, submission, delivery), fairness, availability, ...

Events for Secrecy

Definition (Secrecy, informally)

The intruder cannot discover the data that is intended to be secret from him.

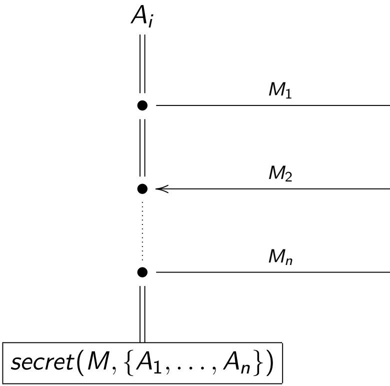
Events

- Goal `M secret between` $\mathtt{A_1}, \ldots, \mathtt{A_n}$
- Insert the signal event

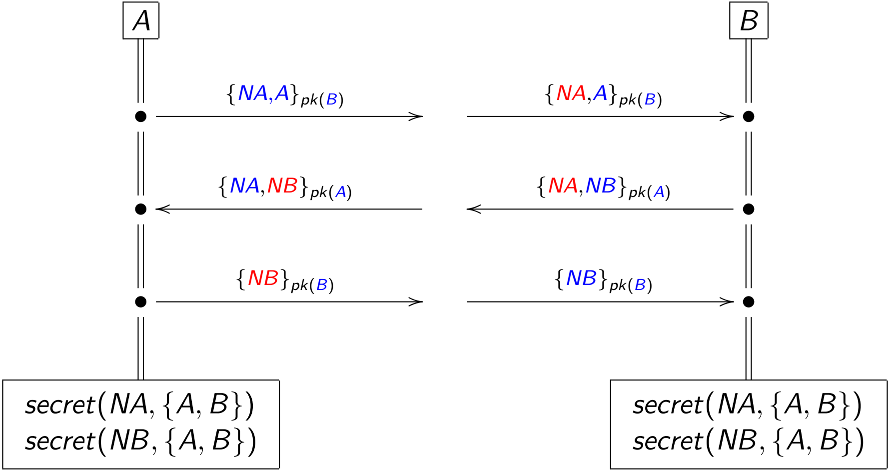
$$secret(M, \{A_1, \ldots, A_n\})$$

  at the end of each role A_i.

- Expressing: A_i believes at this point that M is a secret shared with $A_1, \ldots, A_n$.



For NSPK with the given secrecy goals we have:



Formalization of Secrecy

Definition (Secrecy)

An attack on secrecy is defined by a state

- where the signal $secret(M, \{A_1, \ldots, A_n\})$ has occurred
- the intruder knows M
- the intruder is none of the A_i.

Formalization of Secrecy

Definition (Secrecy)

An attack on secrecy is defined by a state

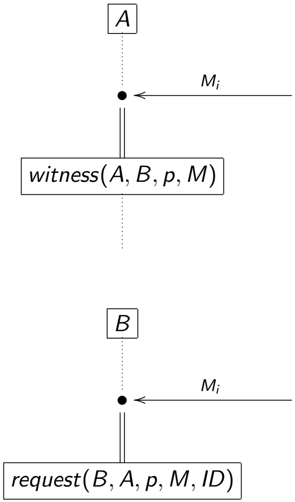
- where the signal $secret(M, \{A_1, \ldots, A_n\})$ has occurred
- the intruder knows M
- the intruder is none of the A_i.

Example: $secret(n_{n17}, \{a, i\})$ and $secret(n_{n18}, \{b, a\})$ occurred, then the intruder is clear to know n_{n17}, but not n_{n18}.

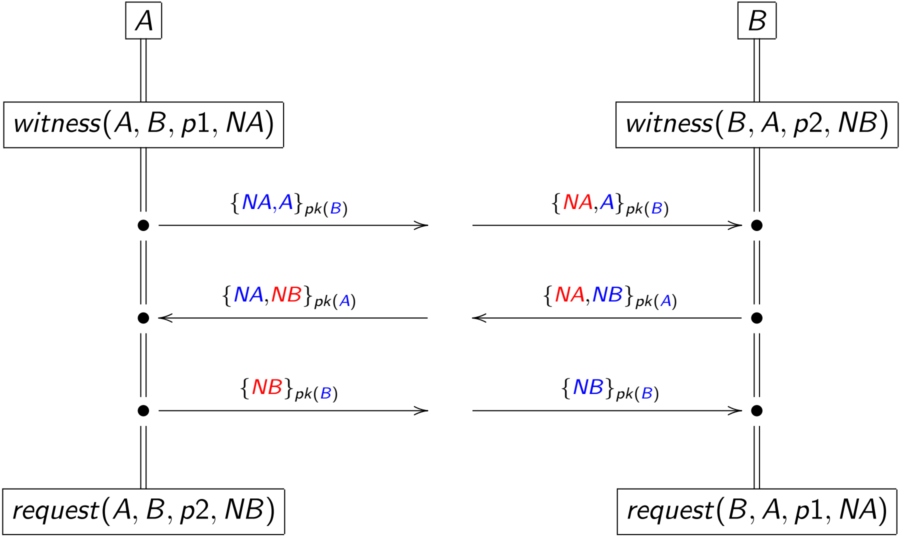
Events for Authentication

Two events for the goal `B (weakly) authenticates A on M`:

- In role A insert the event $witness(A, B, pABM, M)$.
- Position: as soon as A can construct M.
- In role B insert the event $request(B, A, pABM, M, ID)$.
- Position: at B's last send or receive event.
- The constant p identifies the goal (in case there are several authentication goals)
- The variable ID is a unique identifier for the request event.



Events for Authentication: Example



Formalizing Authentication

Definition

An attack on weak authentication is any state in which

- $request(B, A, p, M, ID)$ has occurred for some $A \neq i$ and
- the event $witness(A, B, p, M)$ has never occurred.

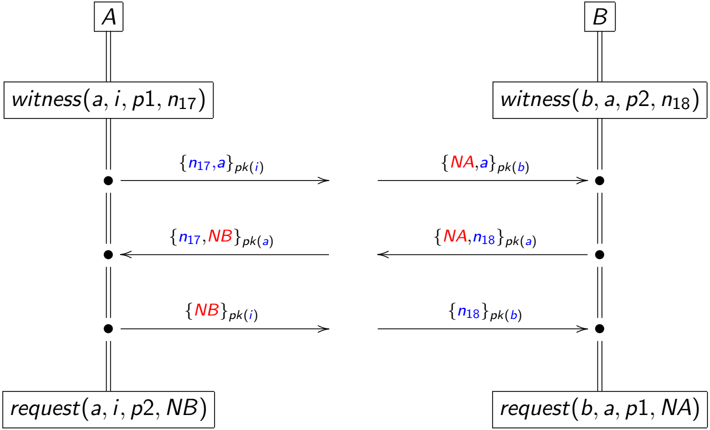
Definition

An attack on strong authentication is any state that is an attack on weak authentication or:

- Both $request(B, A, p, M, ID)$ and $request(B, A, p, M, ID')$ have occurred for $ID \neq ID'$ and $A \neq i$.

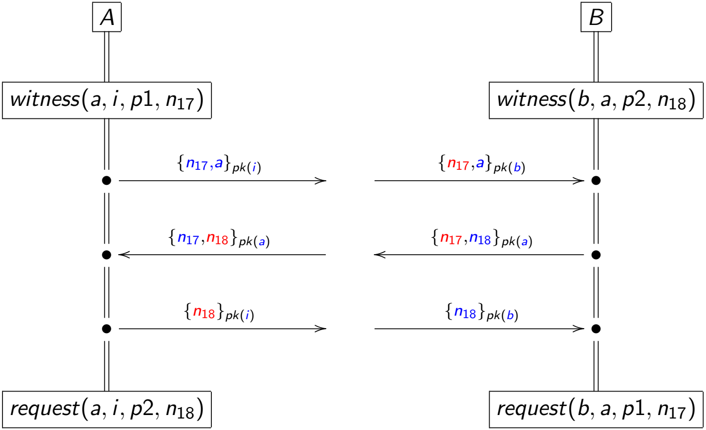
Example: NSPK

Consider this instantiation of the roles:



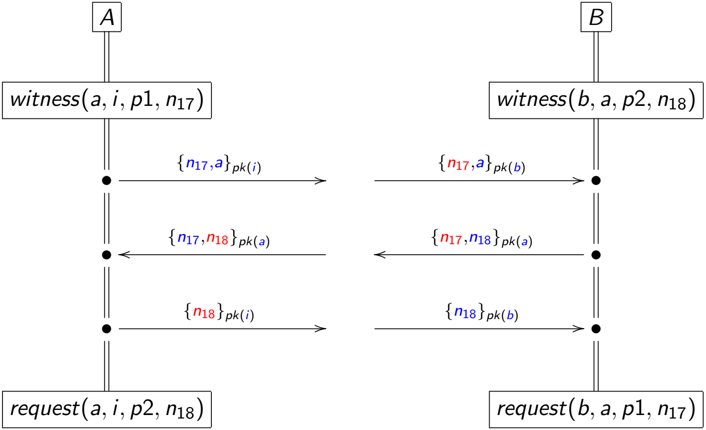
Example: NSPK

Our previous run gives $NA \mapsto n_{17}$ and $NB \mapsto n_{18}$:



- $request(a, i, p_2, n_{18})$ does not matter since sender is i.

Our previous run gives $NA \mapsto n_{17}$ and $NB \mapsto n_{18}$:



- $request(a, i, p_2, n_{18})$ does not matter since sender is i.
- $request(b, a, p1, n_{17})$ violates (weak) authentication.

Bibliography: Books

- Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*, Springer, 2003.
- John Clark and Jeremy Jacob. *A survey of authentication protocol literature*, 1997. http://www.cs.york.ac.uk/ jac/
- Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac
- Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- Claude Kirchner, Hélène Kirchner. *Rewriting, Solving, Proving*. http://www.loria.fr/˜ckirchne/=rsp/rsp.pdf, 1999.

Bibliography: Research Articles

- Iliano Cervesato, Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, Andre Scedrov. *A comparison between strand spaces and multiset rewriting for security protocol analysis.* Journal of Computer Security 13(2), 2005.
- Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *IEEE Trans. Inf. Th.*, 1983.
- Gavin Lowe. *A hierarchy of authentication specifications*. In Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97), pages 31–43. IEEE CS Press, 1997.
- Sebastian Mödersheim. *Algebraic Properties in Alice and Bob Notation*. Proceedings of Ares'09. IEEE Computer Society, 2009.
- Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, Bill Roscoe. *Modeling and Analysis of Security Protocols*. Addison-Wesley, 2000.
- F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. Journal of Computer Security, 7(2/3):191–230, 1999