02244 Logic for Security
Security Protocols
Week 2:
Modeling Protocols—Dolev and Yao

Sebastian Mödersheim

February 9, 2026

Plan for Today

- Hand out of the mandatory assignment.
- Recap of last week
- The Dolev-Yao intruder model
- A decision procedure for Dolev-Yao deductions

Sixth Version

```
Protocol: KeyExchange
Types: Agent A,B,s;
       Number X,Y,g,Payload;
       Function sk;
Knowledge: A: A,B,s,sk(A,s),g;
           B: A,B,s,sk(B,s),g;
           s: A,B,s,sk(A,s),sk(B,s),g;
Actions:
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
Goals:
exp(exp(g,X),Y) secret between A,B;
Payload secret between A,B;
A authenticates B on exp(exp(g,X),Y);
B authenticates A on exp(exp(g,X),Y),Payload;
```

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

Diffie-Hellman:

- every agent generates a random X and Y
- they exchange $\exp(g, X) \mod p$ and $\exp(g, Y) \mod p$
  - ★ p is a large fixed prime number – we omit in OFMC
  - ★ g is a fixed generator of the group $\mathbb{Z}_p^\star$
  - ★ Both p and g are public
  - ★ we omit writing $\mod p$ in OFMC
- It is computationally hard to obtain X from $\exp(g, X) \mod p$
- However A and B have now a shared key $\exp(\exp(g, X), Y) \mod p = \exp(\exp(g, Y), X) \mod p$

Diffie-Hellman and ECDH

	Classic	
Group	$\mathbb{Z}_p^\star = \{1, \ldots, p-1\}$	
Group Op.	$\times : \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star \to \mathbb{Z}_p^\star$ (Mult. modulo p)	
Generator	$g \in \mathbb{Z}_p^\star$	
Secrets	$X, Y \in \{1, \ldots, p-1\}$	
Half keys	$g^X := \underbrace{g \times \ldots \times g}_{X \text{ times}}$ $g^Y := \ldots$	
Full key	$(g^X)^Y = (g^Y)^X$	

Diffie-Hellman and ECDH

	Classic	Elliptic Curve (ECDH)
Group	$\mathbb{Z}_p^\star = \{1, \ldots, p-1\}$	Finite field $\mathbb{F}$ of order n
Group Op.	$\times : \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star \to \mathbb{Z}_p^\star$ (Mult. modulo p)	$+ : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ (not quite so intuitive...)
Generator	$g \in \mathbb{Z}_p^\star$	g on curve
Secrets	$X, Y \in \{1, \ldots, p-1\}$	$X, Y \in \{1, \ldots, n-1\}$
Half keys	$g^X := \underbrace{g \times \ldots \times g}_{X \text{ times}}$	$X \cdot g := \underbrace{g + \ldots + g}_{X \text{ times}}$
	$g^Y := \ldots$	$Y \cdot g := \ldots$
Full key	$(g^X)^Y = (g^Y)^X$	$X \cdot Y \cdot g = Y \cdot X \cdot g$

Diffie-Hellman and ECDH

	Classic	Elliptic Curve (ECDH)
Group	$\mathbb{Z}_p^\star = \{1, \ldots, p-1\}$	Finite field $\mathbb{F}$ of order n
Group Op.	$\times : \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star \to \mathbb{Z}_p^\star$ (Mult. modulo p)	$\times : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ (not quite so intuitive...)
Generator	$g \in \mathbb{Z}_p^\star$	g on curve
Secrets	$X, Y \in \{1, \ldots, p-1\}$	$X, Y \in \{1, \ldots, n-1\}$
Half keys	$g^X := \underbrace{g \times \ldots \times g}_{X \text{ times}}$ $g^Y := \ldots$	$g^X := \underbrace{g \times \ldots \times g}_{X \text{ times}}$ $g^Y := \ldots$
Full key	$(g^X)^Y = (g^Y)^X$	$(g^X)^Y = (g^Y)^X$

Trick: write $\times$ for the group operation also in ECDH.

Diffie-Hellman and ECDH

	Classic	Elliptic Curve (ECDH)
Group	$\mathbb{Z}_p^\star = \{1, \ldots, p-1\}$	Finite field $\mathbb{F}$ of order n
Group Op.	$\times : \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star \to \mathbb{Z}_p^\star$ (Mult. modulo p)	$\times : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ (not quite so intuitive...)
Generator	$g \in \mathbb{Z}_p^\star$	g on curve
Secrets	$X, Y \in \{1, \ldots, p-1\}$	$X, Y \in \{1, \ldots, n-1\}$
Half keys	$g^X := \underbrace{g \times \ldots \times g}_{X \text{ times}}$ $g^Y := \ldots$	$g^X := \underbrace{g \times \ldots \times g}_{X \text{ times}}$ $g^Y := \ldots$
Full key	$(g^X)^Y = (g^Y)^X$	$(g^X)^Y = (g^Y)^X$
Typical size	thousand of bits	hundreds of bits

Trick: write $\times$ for the group operation also in ECDH.

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

- Why is this version better than the fifth version?

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

- Why is this version better than the fifth version?
  - ★ Both A and B contribute something fresh to the key

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

- Why is this version better than the fifth version?
  - ★ Both A and B contribute something fresh to the key
  - ★ The trusted party s does not even get to know the key
    - ▶ An honest but curious s cannot read messages between A and B.

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

- Why is this version better than the fifth version?
  - ★ Both A and B contribute something fresh to the key
  - ★ The trusted party s does not even get to know the key
    - An honest but curious s cannot read messages between A and B.
  - ★ Perfect Forward Secrecy: The intruder cannot read `Payload` even when learning `sk(A,s)` and `sk(B,s)` after the exchange.

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

- Why is this version better than the fifth version?
  - ★ Both A and B contribute something fresh to the key
  - ★ The trusted party s does not even get to know the key
    - ▶ An honest but curious s cannot read messages between A and B.
  - ★ Perfect Forward Secrecy: The intruder cannot read `Payload` even when learning `sk(A,s)` and `sk(B,s)` after the exchange.
- Do we even need the trusted party s then?

Sixth Version

```
A->B: exp(g,X)
B->s: {| A,B,exp(g,X),exp(g,Y) |}sk(B,s)
s->A: {| A,B,exp(g,X),exp(g,Y) |}sk(A,s)
A->B: {| Payload |}exp(exp(g,X),Y)
```

- Why is this version better than the fifth version?
  - ★ Both A and B contribute something fresh to the key
  - ★ The trusted party s does not even get to know the key
    - ▶ An honest but curious s cannot read messages between A and B.
  - ★ Perfect Forward Secrecy: The intruder cannot read `Payload` even when learning `sk(A,s)` and `sk(B,s)` after the exchange.
- Do we even need the trusted party s then? Yes!
  - ★ `exp(g,X)` and `exp(g,Y)` are public
    - ▶ you may call them public keys (with X and Y the private keys)
  - ★ but they need to be authenticated (like public keys):
    - ▶ that `exp(g,X)` really comes from A
    - ▶ and `exp(g,Y)` really comes from B

Modeling Agents and Fixed Key-Infrastructures

- Normally variables (uppercase) like A,B,C,...
  - ★ can be played by any concrete (lowercase) agent like a,b,c,...,i
- Special agent: i – the intruder
- Honest agent: constant like s for a trusted server
  - ★ Cannot be instantiated (especially the intruder), fixed in all protocol runs
- Given key infrastructures: use functions e.g.
  - ★ sk(A,B) the shared key of A and B
  - ★ pw(A,B) the password of A at server B
  - ★ pk(A) the public key of A
    - ▶ inv(K) is the private key that belongs to public key K.
    - ▶ Note inv and exp are a built-in function (do not declare as a function).
  - ★ Give every role the necessary initial knowledge

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`, `B` creates `NB`.

Message Term Algebra

The syntax of Messages (Msg) in AnB can be described by the following context-free grammar:

Msg	::=	Constant	
	|	Variable	
	|	Msg , Msg	concatenation
	|	{| Msg |} Msg	symmetric encryption
	|	{ Msg } Msg	asymmetric encryption
	|	inv (Msg)	inverse
	|	exp (Msg , Msg)	modular exponentiation
	|	Function (Msg)	user-defined functions
	|	(Msg)	

where Constant and Function are user-chosen identifiers that start with a lower-case letter, and Variable with an upper-case letter.

A distinguished constant is the name of the intruder: i.

Message Term Algebra

Definition (Signature)

A signature Σ is a set of function symbols.
Constants are a special case of functions: they take 0 arguments.

Message Term Algebra

Definition (Signature)

A signature Σ is a set of function symbols.
Constants are a special case of functions: they take 0 arguments.

Definition (Terms)

Let $V = \{X, Y, Z, \ldots\}$ be a set of variable symbols.
$\mathcal{T}_\Sigma(V)$ is the set of terms (over Σ and V), defined as follows:

- All variables of V are terms
- If $f \in \Sigma$ is a function symbol that takes n arguments and if $t_1, \ldots, t_n$ are terms, then also $f(t_1, \ldots, t_n)$ is a term.

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.

Syntax and Semantics

So far we have covered the syntax of messages:

- What constitutes a syntactically correct message.

Syntax and Semantics

So far we have covered the syntax of messages:

- What constitutes a syntactically correct message.

Now we define the semantics of messages:

- What does a message actually mean?
- How can OFMC make a meaningful analysis of a protocol?

Syntax and Semantics

So far we have covered the syntax of messages:

- What constitutes a syntactically correct message.

Now we define the semantics of messages:

- What does a message actually mean?
- How can OFMC make a meaningful analysis of a protocol?
- Like a set of rules of a game.

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
  - ★ The intruder simply cannot break the crypto (or flip some bits)
  - ★ He can only use encryption/decryption algorithms with known keys like everybody else

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ The intruder simply cannot break the crypto (or flip some bits)
  - ★ He can only use encryption/decryption algorithms with known keys like everybody else
- Kerckhoffs's principle:
  - ★ Encryption and decryption algorithms are not secret.

Danny Dolev & Andrew C. Yao

Properties

- Black-box model of cryptography
  - ★ The intruder simply cannot break the crypto (or flip some bits)
  - ★ He can only use encryption/decryption algorithms with known keys like everybody else
- 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
  - ★ The intruder simply cannot break the crypto (or flip some bits)
  - ★ He can only use encryption/decryption algorithms with known keys like everybody else
- 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
  - ★ The intruder simply cannot break the crypto (or flip some bits)
  - ★ He can only use encryption/decryption algorithms with known keys like everybody else
- 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
  - ★ The intruder simply cannot break the crypto (or flip some bits)
  - ★ He can only use encryption/decryption algorithms with known keys like everybody else
- 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 (or compromized) participant.
  - ★ Some attacks do not work when all participants are honest.

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 we should have for instance:

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

Idea: Syllogism

Trivial example from Aristoteles:

Premise 1	All humans are mortal
Premise 2	Sokrates is a human
Conclusion	Sokrates is mortal

Idea: Syllogism

Trivial example from Aristoteles:

Premise 1	All humans are mortal
Premise 2	Sokrates is a human
Conclusion	Sokrates is mortal

Things to note:

- Independent of the precise definitions of **human** and **mortal**.
- Rather, human and mortal are characterized by the premises.
- Really logical: if you accept the premises, you cannot really reject the conclusion.
- Aristoteles regarded syllogisms as the most primitive building blocks of logical reasoning. (How to "prove" those?)

Modern View

Premise 1	$\forall X.\ \text{human}(X) \implies \text{mortal}(X)$
Premise 2	human(sokrates)
Conclusion	mortal(sokrates)

Modern View

Premise 1	$\forall X.\ \text{human}(X) \implies \text{mortal}(X)$
Premise 2	$\text{human}(\text{sokrates})$
Conclusion	$\text{mortal}(\text{sokrates})$

Things to note:

- Independent of the precise definitions of **human** and **mortal**.
- In every interpretation that fulfills the premises, also the conclusion is fulfilled.
- One can regard it as even more basic reasoning steps, e.g. natural deduction:

$$\dfrac{\dfrac{\forall X.h(X) \implies m(X)}{h(s) \implies m(s)}\,\forall E \qquad h(s)}{m(s)}\,{\to}E$$

Dolev-Yao Closure

We define $M \vdash t$ as a proof calculus with rules of the form

$$\frac{Premise_1 \quad \ldots \quad Premise_n}{Conclusion} \; Side\text{-}Condition$$

meaning:

- if we have proved all the premisses
- and the side-condition holds,
- then we have a proof of the conclusion.

The simplest rule is Axiom:

Axiom

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

The intruder can derive every message m that is directly in his knowledge M.

Dolev-Yao Closure: Symmetric Crypto

Symmetric Cryptography

$$\frac{M \vdash m \quad M \vdash k}{M \vdash \{\!|m|\!\}_k}\ (\text{EncSym}) \qquad \frac{M \vdash \{\!|m|\!\}_k \quad M \vdash k}{M \vdash m}\ (\text{DecSym})$$

- The intruder can encrypt any message m he knows with any key k he knows.
- The intruder can decrypt any message $\{\!|m|\!\}_k$ to which he knows the decryption key k.

Example

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

$$\dfrac{\dfrac{}{M \vdash \{\!|m_1|\!\}_{k_1}}\ \text{Axiom} \qquad \dfrac{}{M \vdash k_1}\ \text{Axiom}}{M \vdash m_1}\ \text{DecSym}$$

Note: $M \nvdash m_3$

Infinity

Note that with the three rules given so far the set of derivable terms is already infinite:

Example

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

$$\dfrac{\dfrac{\dfrac{}{M \vdash m_2}\,\text{Axiom} \quad \dfrac{}{M \vdash k_1}\,\text{Axiom}}{M \vdash \{\!|m_2|\!\}_{k_1}}\,\text{EncSym} \quad \dfrac{}{M \vdash k_1}\,\text{Axiom}}{M \vdash \{\!|\{\!|m_2|\!\}_{k_1}|\!\}_{k_1}}\,\text{EncSym}$$

Dolev-Yao Closure: Concatenation

Concatenation

$$\frac{M \vdash m_1 \quad M \vdash m_2}{M \vdash \langle m_1, m_2 \rangle}\ (\mathrm{Cat}) \qquad \frac{M \vdash \langle m_1, m_2 \rangle}{M \vdash m_i}\ (\mathrm{Proj}_i)$$

- The intruder can concatenate and split messages.

Example

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{M \vdash \{\!|\langle a, m_1 \rangle|\!\}_{k_1}}\ \text{Axiom} \quad \dfrac{}{M \vdash k_1}\ \text{Axiom}}{M \vdash \langle a, m_1 \rangle}\ \text{DecSym}}{M \vdash m_1}\ \mathrm{Proj}_2 \qquad \dfrac{}{M \vdash m_2}\ \text{Axiom}}{M \vdash \langle m_1, m_2 \rangle}\ \text{Cat}$$

Dolev-Yao Closure: Asymmetric Crypto

Asymmetric Cryptography

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

- The intruder can encrypt any message m he knows with any public key k he knows.
- The intruder can decrypt any message $\{m\}_k$ if he knows the private key $\text{inv}(k)$ to the public key k.

Example

$$M = \{\, k_1, \text{inv}(k_1), k_2, \{m_1\}_{k_1}, m_2, \{m_3\}_{k_2} \,\}$$

$$\dfrac{\dfrac{}{M \vdash \{m_1\}_{k_1}}\ \text{Axiom} \quad \dfrac{}{M \vdash \text{inv}(k_1)}\ \text{Axiom}}{M \vdash m_1}\ \text{DecAsym}$$

Note: $M \nvdash m_3$

Dolev-Yao Closure: Signatures

Signatures

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

- The intruder can sign any message m he knows with any private key $\mathrm{inv}(k)$ he knows.
- The intruder can open any message $\{m\}_{\mathrm{inv}(k)}$ that was signed with a private key $\mathrm{inv}(k)$.

Example

$$M = \{\, k_1, \mathrm{inv}(k_1), k_2, \{m_1\}_{\mathrm{inv}(k_2)}, m_2 \,\}$$

$$\dfrac{\dfrac{\dfrac{}{M \vdash \{m_1\}_{\mathrm{inv}(k_2)}}\ \mathrm{Axiom}}{M \vdash m_1}\ \mathrm{OpenSig} \qquad \dfrac{}{M \vdash \mathrm{inv}(k_1)}\ \mathrm{Axiom}}{M \vdash \{m_1\}_{\mathrm{inv}(k_1)}}\ \mathrm{Sign}$$

Dolev-Yao Closure: Public Functions

Public Functions

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

- The intruder can apply any public function f to terms t_i that he knows.

Example

$$M = \{\, k_1, k_2, \{\!|m_1|\!\}_{kdf(k_1,k_2)} \,\} \text{ where } kdf \text{ is public}$$

$$\dfrac{\dfrac{}{M \vdash \{\!|m_1|\!\}_{kdf(k_1,k_2)}}\ \text{Axiom} \quad \dfrac{\dfrac{}{M \vdash k_1}\ \text{Axiom} \quad \dfrac{}{M \vdash k_2}\ \text{Axiom}}{M \vdash kdf(k_1, k_2)}\ \text{Compose}}{M \vdash m_1}\ \text{DecSym}$$

Note: the rules EncSym, EncAsym, and Cat are just special cases of Compose.

Dolev-Yao Closure: Summary

Dolev-Yao rules

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

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

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

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

The compose rule is for all public functions Σ_p, including $\{\!|\cdot|\!\}$. $\{\cdot\}$. $\langle \cdot, \cdot \rangle$

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

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: 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: 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: 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: 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 $\{\langle na, a\rangle\}_{\mathsf{pk}(b)}$?

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

Automation

Goal: design (in pseudocode) a decision procedure for Dolev-Yao:

- Given a finite set M of messages (the intruder knowledge)
- and given a message m (the goal)
- Output whether $M \vdash m$ holds.
  - ★ additionally, in the positive case, give the proof.

Automating Dolev-Yao

Step 1: Composition only

Consider first the following simpler problem:

- $M \vdash_c m$ are those deductions where the intruder does not apply any analysis steps ("composition only"):

Composition Only

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

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

Example

$M = \{k_1, k_2, \{\!|m|\!\}_{h(k_1,k_2)}\}$ where $h \in \Sigma_p$

- $M \vdash_c h(k_1, k_2)$
- $M \nvdash_c m$
- $M \vdash m$

Automating Dolev-Yao

Step 1: Composition only—Solution

Composition Only

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

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

Challenge: decision procedure for $\vdash_c$:

- Input: M and m
- Output yes if $M \vdash_c m$, and no otherwise.

Idea: backwards search for a derivation.

Automating Dolev-Yao

Step 1: Composition only—Solution

Composition Only

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

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

Example

$M = \{k_1, k_2, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \{\!|n_2|\!\}_{k_3}\}$ where $h \in \Sigma_p$

$$\frac{??}{M \vdash_c h(k_1, k_2)}$$

Check Axiom: $h(k_1, k_2) \in M$? (No: we cannot apply Axiom)

Automating Dolev-Yao

Step 1: Composition only—Solution

Composition Only

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

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

Example

$M = \{k_1, k_2, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \{\!|n_2|\!\}_{k_3}\}$ where $h \in \Sigma_p$

$$\frac{??}{M \vdash_c h(k_1, k_2)}$$

Check Compose: h public? (Yes: so we can try Compose)

Automating Dolev-Yao

Step 1: Composition only—Solution

Composition Only

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

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

Example

$M = \{k_1, k_2, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \{\!|n_2|\!\}_{k_3}\}$ where $h \in \Sigma_p$

$$\frac{\dfrac{??}{M \vdash_c k_1} \quad \dfrac{??}{M \vdash_c k_2}}{M \vdash_c h(k_1, k_2)} \textit{ Compose}$$

Try to find proofs for the new sub-goals – recursively.

Automating Dolev-Yao

Step 1: Composition only—Solution

Composition Only

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

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

Example

$M = \{k_1, k_2, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \{\!|n_2|\!\}_{k_3}\}$ where $h \in \Sigma_p$

$$\dfrac{\dfrac{}{M \vdash_c k_1}\ \textit{Axiom} \quad \dfrac{}{M \vdash_c k_2}\ \textit{Axiom}}{M \vdash_c h(k_1, k_2)}\ \textit{Compose}$$

$k_1, k_2 \in M$, so we can solve that with Axiom – done!

Automating Dolev-Yao

Step 1: Composition only—Solution

Decision procedure for $M \vdash_c m$

1. Check if $m \in M$; if so return yes.
2. Otherwise, let $m = f(t_1, \ldots, t_n)$
   - ★ If f is not public return no.
   - ★ Otherwise recursively check whether:

$$M \vdash_c t_1 \text{ and } \ldots \text{and } M \vdash_c t_n$$

Return yes if all these return yes, and no otherwise.

Automating Dolev-Yao

Step 2: Analysis—solution

Analysis Steps

- If $\{\!|m|\!\}_k \in M$ and $M \vdash_c k$ then add m to M.
- If $\{m\}_k \in M$ and $M \vdash_c \mathrm{inv}(k)$ then add m to M.
- if $\{m\}_{\mathrm{inv}(k)} \in M$ then add m to M.
- If $\langle m_1, m_2 \rangle \in M$ then add m_1 and m_2 to M.
- Repeat until no new messages can be added.

Example

$M = \{k_1, k_2, \{\!|n_2|\!\}_{k_3}, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}\}$

Automating Dolev-Yao

Step 2: Analysis—solution

Analysis Steps

- If $\{\!|m|\!\}_k \in M$ and $M \vdash_c k$ then add m to M.
- If $\{m\}_k \in M$ and $M \vdash_c \mathrm{inv}(k)$ then add m to M.
- if $\{m\}_{\mathrm{inv}(k)} \in M$ then add m to M.
- If $\langle m_1, m_2 \rangle \in M$ then add m_1 and m_2 to M.
- Repeat until no new messages can be added.

Example

$M = \{k_1, k_2, \{\!|n_2|\!\}_{k_3}, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}\}$

- $\{\!|n_2|\!\}_{k_3}$: check $M \vdash_c k_3$? No. (Maybe later.)

Automating Dolev-Yao

Step 2: Analysis—solution

Analysis Steps

- If $\{\!|m|\!\}_k \in M$ and $M \vdash_c k$ then add m to M.
- If $\{m\}_k \in M$ and $M \vdash_c \mathrm{inv}(k)$ then add m to M.
- if $\{m\}_{\mathrm{inv}(k)} \in M$ then add m to M.
- If $\langle m_1, m_2 \rangle \in M$ then add m_1 and m_2 to M.
- Repeat until no new messages can be added.

Example

$M = \{k_1, k_2, \{\!|n_2|\!\}_{k_3}, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \langle n_1, k_3 \rangle\}$

- $\{\!|n_2|\!\}_{k_3}$: check $M \vdash_c k_3$? No. (Maybe later.)
- $\{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}$: $M \vdash_c h(k_1, k_2)$? Yes, add $\langle n_1, k_3 \rangle$ to M.

Automating Dolev-Yao

Step 2: Analysis—solution

Analysis Steps

- If $\{\!|m|\!\}_k \in M$ and $M \vdash_c k$ then add m to M.
- If $\{m\}_k \in M$ and $M \vdash_c \mathrm{inv}(k)$ then add m to M.
- if $\{m\}_{\mathrm{inv}(k)} \in M$ then add m to M.
- If $\langle m_1, m_2 \rangle \in M$ then add m_1 and m_2 to M.
- Repeat until no new messages can be added.

Example

$M = \{k_1, k_2, \{\!|n_2|\!\}_{k_3}, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \langle n_1, k_3 \rangle, n_1, k_3\}$

- $\{\!|n_2|\!\}_{k_3}$: check $M \vdash_c k_3$? No. (Maybe later.)
- $\{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}$: $M \vdash_c h(k_1, k_2)$? Yes, add $\langle n_1, k_3 \rangle$ to M.
- $\langle n_1, k_3 \rangle$: add n_1 and k_3 to M

Automating Dolev-Yao

Step 2: Analysis—solution

Analysis Steps

- If $\{\!|m|\!\}_k \in M$ and $M \vdash_c k$ then add m to M.
- If $\{m\}_k \in M$ and $M \vdash_c \mathrm{inv}(k)$ then add m to M.
- if $\{m\}_{\mathrm{inv}(k)} \in M$ then add m to M.
- If $\langle m_1, m_2 \rangle \in M$ then add m_1 and m_2 to M.
- Repeat until no new messages can be added.

Example

$M = \{k_1, k_2, \{\!|n_2|\!\}_{k_3}, \{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}, \langle n_1, k_3 \rangle, n_1, k_3, n_2\}$

- $\{\!|n_2|\!\}_{k_3}$: check $M \vdash_c k_3$? Now yes! add n_2
- $\{\!|\langle n_1, k_3 \rangle|\!\}_{h(k_1,k_2)}$: $M \vdash_c h(k_1, k_2)$? Yes, add $\langle n_1, k_3 \rangle$ to M.
- $\langle n_1, k_3 \rangle$: add n_1 and k_3 to M

Automating Dolev-Yao

To check $M \vdash m$:

- Perform the Analysis Steps procedure, augmenting M with all derivable messages.
- Now it suffices to check $M \vdash_c m$.

Properties of the algorithm for checking $M \vdash m$:

- Soundness: if algorithm says "yes", then $M \vdash m$.
- Completeness: if $M \vdash m$, then the algorithm says "yes".
  - ★ This is quite tricky to prove.
- Termination: the algorithm never runs into an infinite loop.

Automating Dolev-Yao

To check $M \vdash m$:

- Perform the Analysis Steps procedure, augmenting M with all derivable messages.
- Now it suffices to check $M \vdash_c m$.

Properties of the algorithm for checking $M \vdash m$:

- Soundness: if algorithm says "yes", then $M \vdash m$.
- Completeness: if $M \vdash m$, then the algorithm says "yes".
  - ★ This is quite tricky to prove.
- Termination: the algorithm never runs into an infinite loop.
  - ★ Procedure for $\vdash_c$ terminates since it goes to smaller terms

Automating Dolev-Yao

To check $M \vdash m$:

- Perform the Analysis Steps procedure, augmenting M with all derivable messages.
- Now it suffices to check $M \vdash_c m$.

Properties of the algorithm for checking $M \vdash m$:

- Soundness: if algorithm says "yes", then $M \vdash m$.
- Completeness: if $M \vdash m$, then the algorithm says "yes".
  - ★ This is quite tricky to prove.
- Termination: the algorithm never runs into an infinite loop.
  - ★ Procedure for $\vdash_c$ terminates since it goes to smaller terms
  - ★ Analysis terminates because it only adds proper subterms of an existing term. This cannot go on forever, since there are only finitely many subterms.

The Completeness Proof

Task of the proof: given a Dolev-Yao proof tree for $M \vdash m$, show that the procedure will say "yes", i.e., the procedure finds this derivation.

Example

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

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

The Completeness Proof

Task of the proof: given a Dolev-Yao proof tree for $M \vdash m$, show that the procedure will say "yes", i.e., the procedure finds this derivation.

Example

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

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

- Argument: the step labeled A would have been found by our analysis procedure, augmenting M with $\langle na, a\rangle$.

The Completeness Proof

Task of the proof: given a Dolev-Yao proof tree for $M \vdash m$, show that the procedure will say "yes", i.e., the procedure finds this derivation.

Example

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

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

- Argument: the step labeled A would have been found by our analysis procedure, augmenting M with $\langle na, a\rangle$.
- So we could replace $[A]$ by axiom afterwards.

The Completeness Proof

Task of the proof: given a Dolev-Yao proof tree for $M \vdash m$, show that the procedure will say "yes", i.e., the procedure finds this derivation.

Example

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

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

- Argument: the step labeled A would have been found by our analysis procedure, augmenting M with $\langle na, a\rangle$.
- So we could replace $[A]$ by axiom afterwards.
- The remainder has only composition steps, and the completeness of $\vdash_c$ is straightforward.

The Completeness Proof

Task of the proof: given a Dolev-Yao proof tree for $M \vdash m$, show that the procedure will say "yes", i.e., the procedure finds this derivation.

Example

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

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

- Argument: the step labeled A would have been found by our analysis procedure, augmenting M with $\langle na, a\rangle$.
- So we could replace $[A]$ by axiom afterwards.
- The remainder has only composition steps, and the completeness of $\vdash_c$ is straightforward.
- However, this works only if the message we analyze is an axiom!

Completeness Proof

What if the message is composed that we want to analyze?

$$\dfrac{\dfrac{\dfrac{\Pi_1}{M \vdash k} \quad \dfrac{\Pi_2}{M \vdash m}}{M \vdash \{\!|m|\!\}_k} \quad \dfrac{\Pi_3}{M \vdash k}}{M \vdash m}$$

for some arbitrary subproofs $\Pi_1, \ldots, \Pi_3$.

Completeness Proof

What if the message is composed that we want to analyze?

$$\dfrac{\dfrac{\dfrac{\Pi_1}{M \vdash k} \quad \dfrac{\Pi_2}{M \vdash m}}{M \vdash \{\!|m|\!\}_k} \quad \dfrac{\Pi_3}{M \vdash k}}{M \vdash m}$$

for some arbitrary subproofs $\Pi_1, \ldots, \Pi_3$.

- Here the intruder decrypts a term that he has encrypted himself.

Completeness Proof

What if the message is composed that we want to analyze?

$$\dfrac{\dfrac{\dfrac{\Pi_1}{M \vdash k} \quad \dfrac{\Pi_2}{M \vdash m}}{M \vdash \{\!|m|\!\}_k} \quad \dfrac{\Pi_3}{M \vdash k}}{M \vdash m}$$

for some arbitrary subproofs $\Pi_1, \ldots, \Pi_3$.

- Here the intruder decrypts a term that he has encrypted himself.
- This can be simplified!

Completeness Proof

What if the message is composed that we want to analyze?

$$\dfrac{\dfrac{\dfrac{\Pi_1}{M \vdash k} \quad \dfrac{\Pi_2}{M \vdash m}}{M \vdash \{\!|m|\!\}_k} \quad \dfrac{\Pi_3}{M \vdash k}}{M \vdash m}$$

for some arbitrary subproofs $\Pi_1, \ldots, \Pi_3$.

- Here the intruder decrypts a term that he has encrypted himself.
- This can be simplified!
- It is without loss of generality to forbid the application of analysis to a term that was obtained by composition.

Completeness Proof

What if the message is itself the result of an analysis?

$$\dfrac{\dfrac{\overline{M \vdash \{\langle m_1, m_2\rangle\}_{\mathsf{pk}(a)}} \quad \overline{M \vdash \mathsf{inv}(\mathsf{pk}(a))}}{M \vdash \langle m_1, m_2\rangle}\, A_2}{M \vdash m}\, A_1$$

Completeness Proof

What if the message is itself the result of an analysis?

$$\dfrac{\dfrac{\overline{M \vdash \{\langle m_1, m_2\rangle\}_{\mathsf{pk}(a)}} \quad \overline{M \vdash \mathsf{inv}(\mathsf{pk}(a))}}{M \vdash \langle m_1, m_2\rangle}\, A_2}{M \vdash m}\, A_1$$

- Just always consider the inner-most analysis step first:
  - ★ an analysis step that has no analysis step in the subproofs.
  - ★ then the message being analyzed must be obtained by axiom, because analysis of composition we have already ruled out.
  - ★ thus we can eliminate one by one all analysis steps in the proof until we have a proof with only composition steps, and then completeness follows from $\vdash_c$.

Negative Question

Can we thus prove also statements of the form $M \not\vdash m$
. . . that a m cannot be derived from M?

Example

$$M = \{\, k_1, \{\!|m_1|\!\}_{k_1}, m_2, \{\!|m_3|\!\}_{k_2} \,\} \not\vdash m_3$$

Negative Question

Can we thus prove also statements of the form $M \not\vdash m$
. . . that a m cannot be derived from M?

Example

$$M = \{\, k_1, \{\!|m_1|\!\}_{k_1}, m_2, \{\!|m_3|\!\}_{k_2} \,\} \not\vdash m_3$$

- Yes, due to completeness when our algorithm answers "no", we know there is no derivation for m.

Relevant Research Papers

- Danny Dolev and Andrew C. Yao. *On the security of public key protocols*, IEEE Transactions on Information Theory, 29(2), 1983
- David Basin, Sebastian Mödersheim, and Luca Viganò. *OFMC: A symbolic model checker for security protocols.* International Journal of Information Security, 4(3), 2005.
- Gavin Lowe. *Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR*. Software Concepts Tools, 17(3), 1996.
- Jonathan K. Millen and Vitaly Shmatikov. *Constraint solving for bounded-process cryptographic protocol analysis*. Computer and Communications Security, 2001,
- Roger Needham and Michael Schroeder. *Using Encryption for Authentication in Large Networks of Computers.* Communications of the ACM, 21(12), 1978.
- Michaël Rusinowitch and Mathieu Turuani. *Protocol Insecurity with Finite Number of Sessions is NP-complete.* Computer Security Foundations Workshop, 2001.