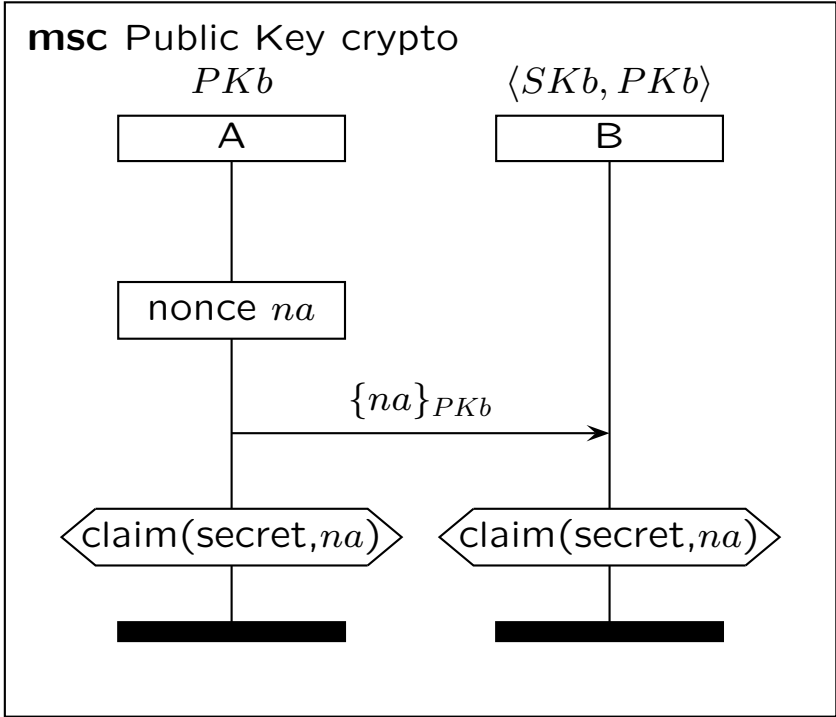

Operational Semantics of Security Protocols

Sjouke Mauw

joint work with Cas Cremers and Erik de Vink

Eindhoven University of Technology
The Netherlands

- Motivation and objectives.
- Derivation rules.
- Secrecy.
- Conclusions and ongoing research.



Is A's claim valid?

And B's claim?

“Security protocols are three-line programs that people still manage to get wrong”

(Roger Needham)

Security protocol = set of interaction rules to guarantee security property (plus some intended functionality).

Formal validation is imperative and feasible.

- BAN logic,
- Casper/FDR,
- Strand spaces,
- etc.

All have their pro's and con's.

- Only isolated protocols (no protocol sets).
- No proper definition of security properties (confidentiality, authentication, non-repudiation, etc). And where proper, no consensus.
- Can only deal with single intruder model (Dolev-Yao).
- No type flaw attacks.
- No forward secrecy/backward security.
- Model checking is incomplete.
- Trade-off between precision and useability.

Develop a modular general operational semantics for security protocol sets having the required merits.

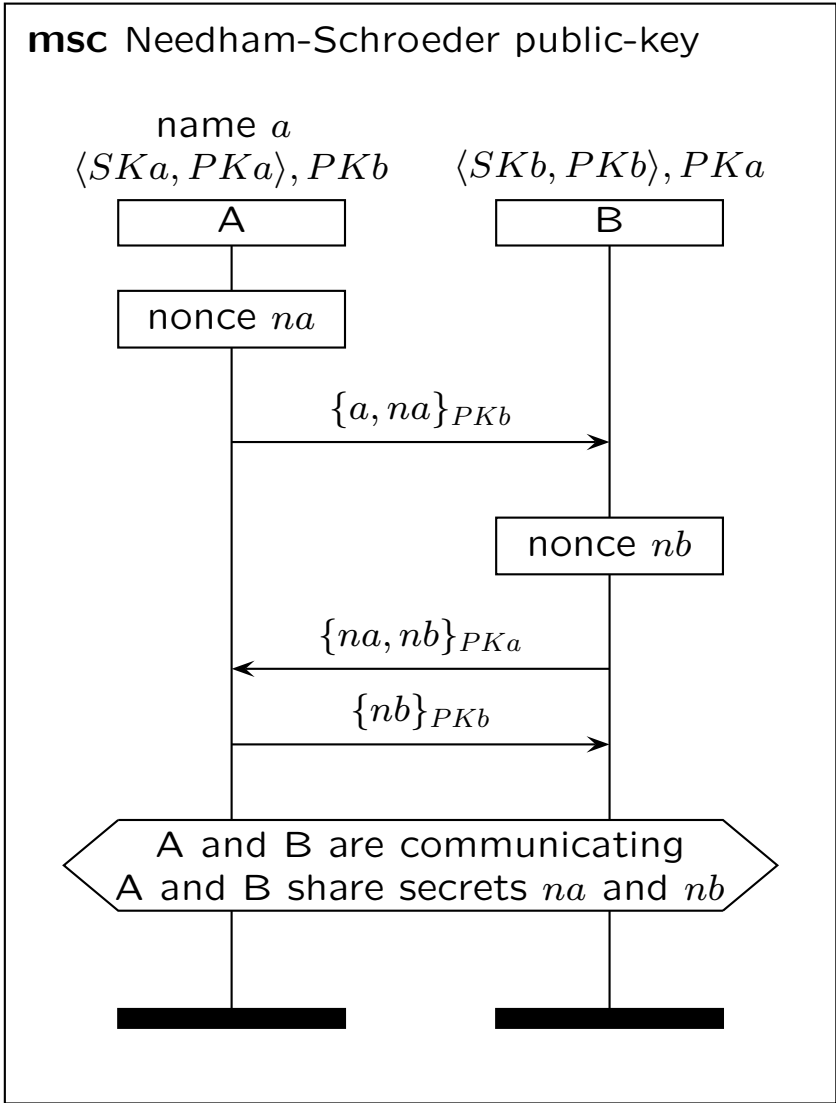
(Yes, this is ambitious, I know!)

Assume underlying set of cryptographic primitives, from which we only know the relevant properties.

Example: $\{\{m\}_{PK}\}_{SK} = \{\{m\}_{SK}\}_{PK} = m$

Advantages:

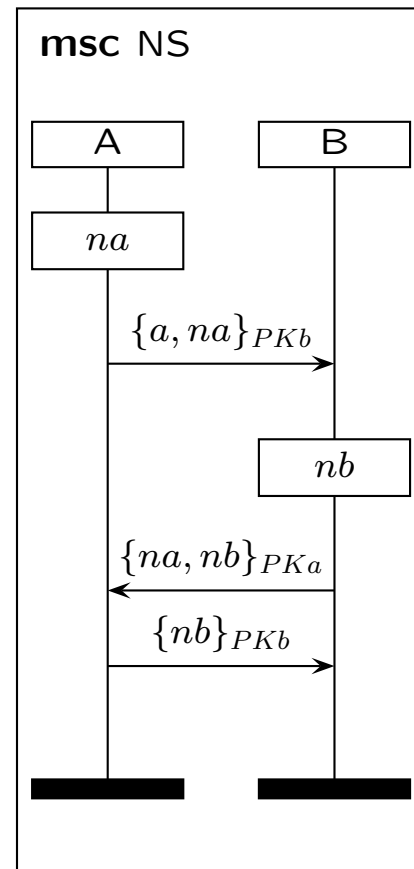
- Separation of concerns.
- Substitute any mathematical object satisfying the required properties.
- Develop protocols based on hypothetical primitives.



Role definitions for Needham-Schroeder:

$Initiator(i, r) =$
 $const\ na;$
 $var\ X;$
 $send(i, r, \{i, na\}_{PK(r)}) \cdot$
 $read(r, i, \{na, X\}_{PK(i)}) \cdot$
 $send(i, r, \{X\}_{PK(r)})$

$Responder(i, r) =$
 $const\ nb;$
 $var\ Y;$
 $read(i, r, \{i, Y\}_{PK(r)}) \cdot$
 $send(r, i, \{Y, nb\}_{PK(i)}) \cdot$
 $read(i, r, \{nb\}_{PK(r)})$



Run = Role instantiated for chosen agents, labeled with run identifier.

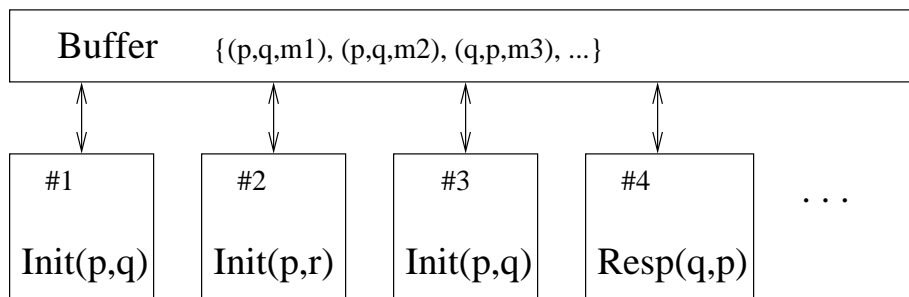
$$\begin{aligned} \text{Initiator}(a, b)\#rid1 = & \\ & \text{send}(a, b, \{a, na\#rid1\}_{PK(b)}) \cdot \\ & \text{read}(b, a, \{na\#rid1, X\}_{PK(a)}) \cdot \\ & \text{send}(a, b, \{X\}_{PK(b)}) \end{aligned}$$

$$\begin{aligned} \text{Responder}(a, b)\#rid2 = & \\ & \text{read}(a, b, \{a, Y\}_{PK(b)}) \cdot \\ & \text{send}(b, a, \{Y, nb\#rid2\}_{PK(a)}) \cdot \\ & \text{read}(a, b, \{nb\#rid2\}_{PK(b)}) \end{aligned}$$

Handle nonces symbolically. If all runid's are unique, then the pair (na,rid) is unique.

- Set of States S .
- Set of Events E .
- Transition relation $s \xrightarrow{e} s'$
Determine for every state $s \in S$ all possible executable events $e \in E$ and the resulting state $s' \in S$.
- Gives set of traces (or graph):
 $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} \dots$
- Defined by derivation rules $\frac{\textit{Hypothesis}}{s \xrightarrow{e} s'}$

Network = Dynamic collection of executing runs that communicate via buffer.



- Buffer:

$$\mathbb{B} = \mathcal{M}(\{(x, y, m) \mid x, y \in \text{Agent}, m \in \text{Term}\})$$

- Runs (partial function): $\mathbb{R} = \text{RunId} \rightarrow \text{Run}$

- Network State: $\Sigma^{\text{network}} = \mathbb{B} \times \mathbb{R}$

- Initial state: $\sigma_0^{\text{network}} = \emptyset \times \emptyset$

- Network transitions: $_ \dots \bar{_} \rightarrow _$

$$\frac{R(r_1, \dots, r_n) \in \text{RoleDef}, \text{rid} \notin \text{dom}(F), a_1, \dots, a_n \in \text{Agents}}$$

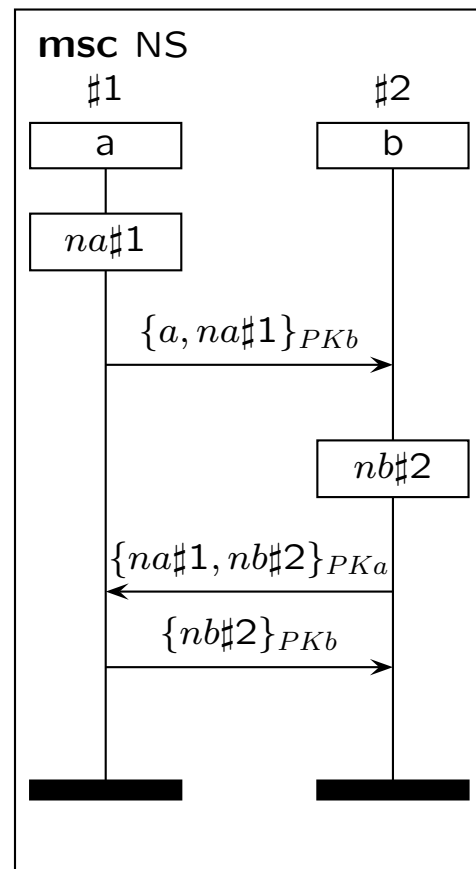
$$\langle B, F \rangle \xrightarrow{\text{create}(R, a_1, \dots, a_n) \# \text{rid}} \langle B, F \cup (\text{rid}, R(a_1, \dots, a_n) \# \text{rid}) \rangle$$

$$\frac{F(\text{rid}) = \text{send}(a, b, m) \cdot \sigma}{\langle B, F \rangle \xrightarrow{\text{send}(a, b, m) \# \text{rid}} \langle B \cup \{(a, b, m)\}, F[\text{rid} := \sigma] \rangle}$$

$$\frac{F(\text{rid}) = \text{read}(a, b, m) \cdot \sigma, (a, b, m') \in B, \text{match}(m, m') = \rho \neq \perp}{\langle B, F \rangle \xrightarrow{\text{read}(a, b, m') \# \text{rid}} \langle B \setminus \{(a, b, m')\}, F[\text{rid} := \rho(\sigma)] \rangle}$$

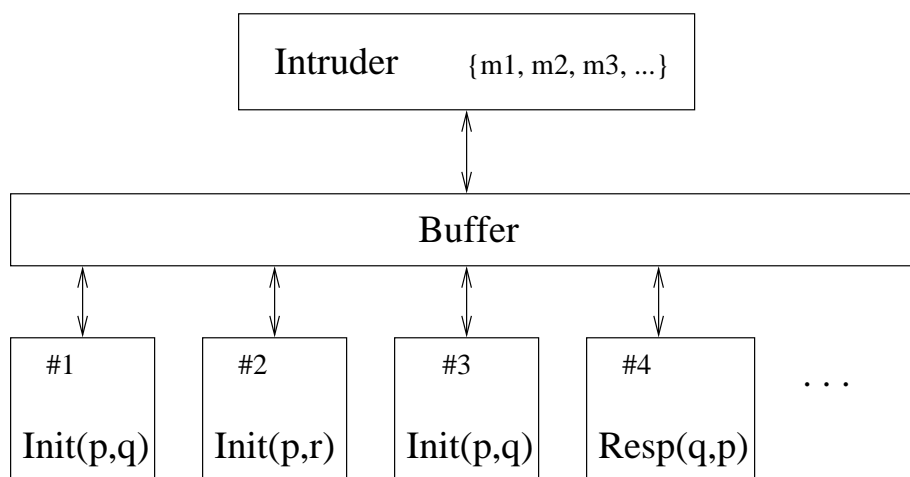
```

create(Initiator, a, b)#1 .
create(Responder, a, b)#2 .
send(a, b, {a, na#1}PKb)#1 .
read(a, b, {a, na#1}PKb)#2 .
send(b, a, {na#1, nb#2}PKa)#2 .
read(b, a, {na#1, nb#2}PKa)#1 .
send(a, b, {nb#2}PKb)#1 .
read(a, b, {nb#2}PKb)#2
    
```



- Intruder knowledge: \mathbb{K} = set of terms.
- Closedness: if $d_1, \dots, d_n \in K$ then also $f(d_1, \dots, d_n) \in K$ (for appropriately typed selected functions f).
- $K \vdash d$ means closure of $K \cup \{d\}$

Example: If $\{m\}_{PK} \in K$ and $SK \in K$ then $\{\{m\}_{PK}\}_{SK} = m \in K$



- State: $\Sigma^{inetrw} = \mathbb{K} \times \Sigma^{network}$
- Initial state: $K_0 \times \sigma_0^{network}$, where K_0 is the initial knowledge of the intruder

$$\frac{\langle B, F \rangle \dots \xrightarrow{\alpha} \langle B', F' \rangle}{\langle K, \langle B, F \rangle \rangle \xrightarrow{\alpha} \langle K, \langle B', F' \rangle \rangle}$$

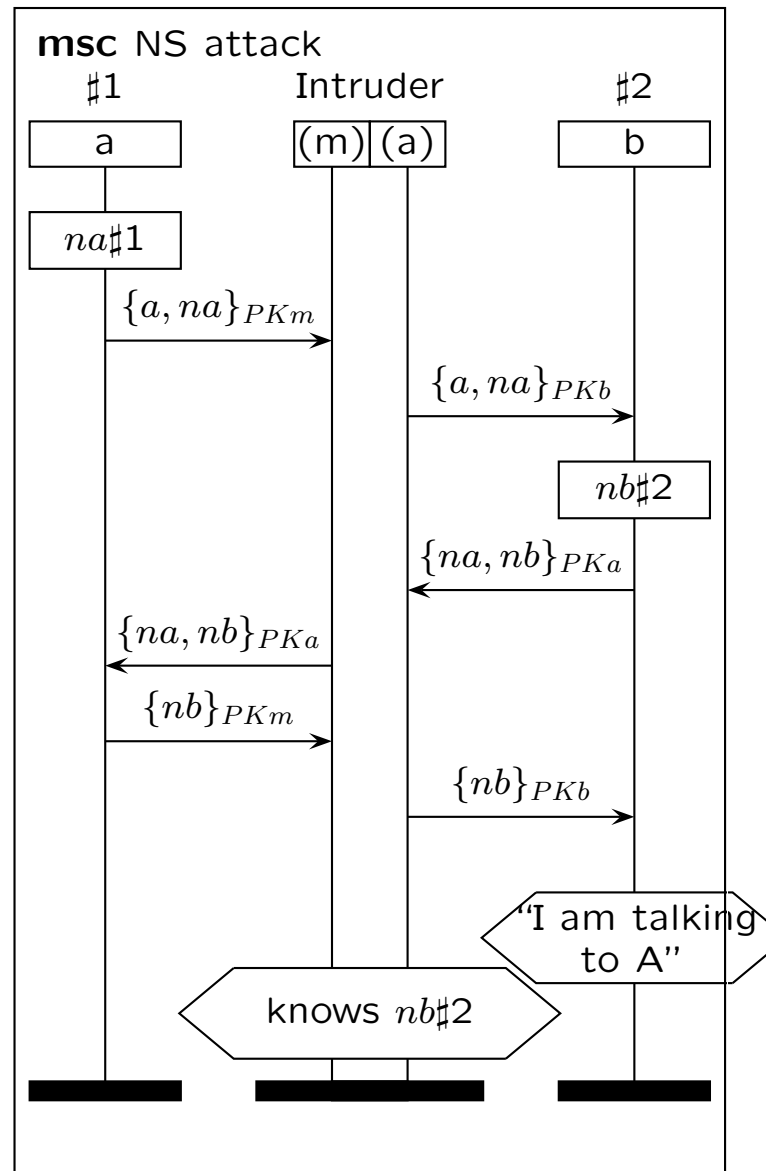
$$\frac{\langle B, F \rangle \xrightarrow{\text{send}(a,b,m)\#rid} \dots \langle B', F' \rangle}{\langle K, \langle B, F \rangle \rangle \xrightarrow{\text{send}(a,b,m)\#rid} \langle K + m, \langle B', F' \rangle \rangle}$$

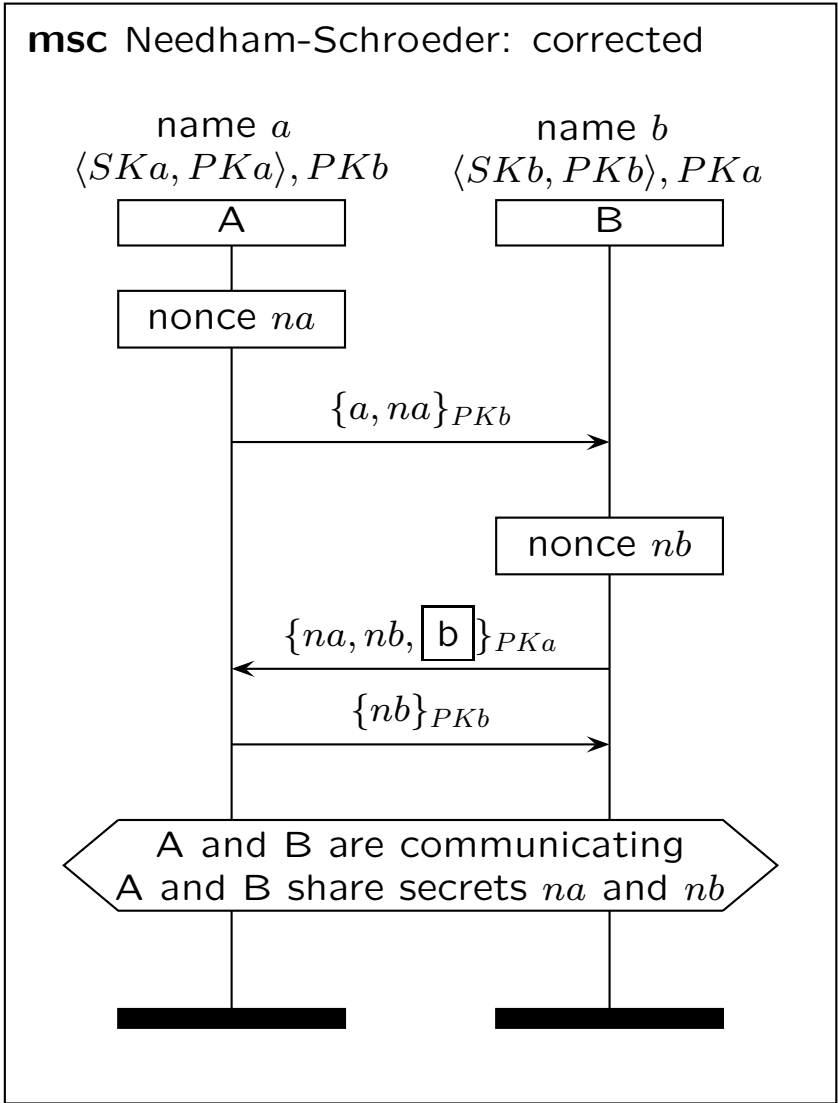
$$\frac{(a, b, m) \in B}{\langle K, \langle B, f \rangle \rangle \xrightarrow{\text{deflect}(a,b,m)} \langle K, \langle B \setminus \{(a, b, m)\}, F \rangle \rangle}$$

$$\frac{m \in K, a, b \in \text{Agents}}{\langle K, \langle B, F \rangle \rangle \xrightarrow{\text{inject}(a,b,m)} \langle K, \langle B \cup \{(a, b, m)\}, F \rangle \rangle}$$

```

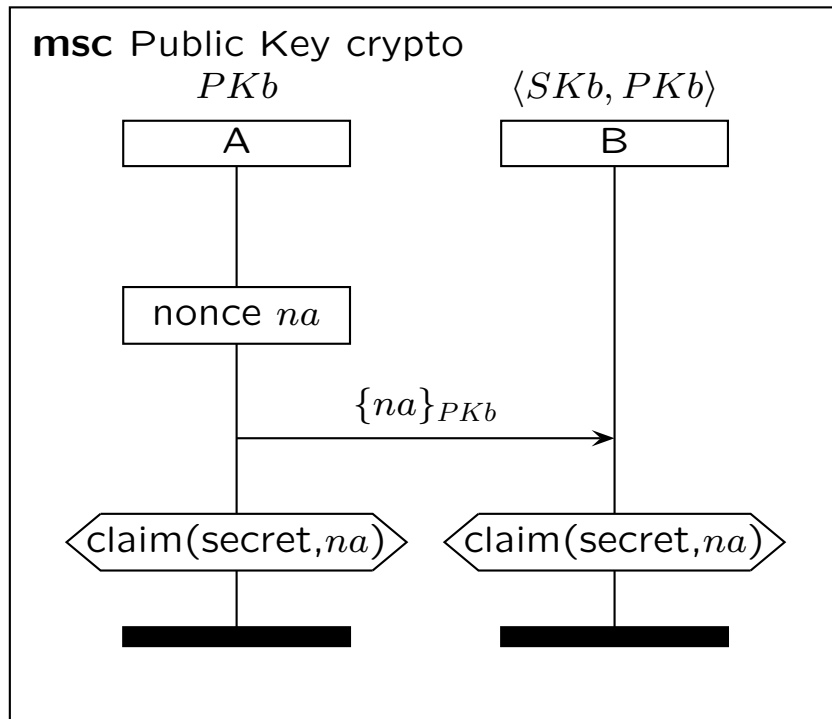
create(Initiator, a, m)#1 .
create(Responder, a, b)#2 .
send(a, m, {a, na#1}PKm)#1 .
deflect((a, m, {a, na#1}PKm)) .
inject((a, b, {a, na#1}PKb)) .
read(a, b, {a, na#1}PKb)#2 .
send(b, a, {na#1, nb#2}PKa)#2 .
deflect((b, a, {na#1, nb#2}PKa)) .
inject((m, a, {na#1, nb#2}PKa)) .
read(m, a, {na#1, nb#2}PKa)#1 .
send(a, m, {nb#2}PKm)#1 .
deflect((a, m, {nb#2}PKm)) .
inject((a, b, {nb#2}PKb)) .
read(a, b, {nb#2}PKb)
    
```





For all traces of protocol p , if a term t is claimed to be secret, the intruder will never learn (deduce) this term.

$\forall \alpha \in \text{Traces}(p)$ if $\alpha_i = \text{claim}(\text{secret}, t)$
then $\forall j > i$ $t \notin K_j$



Proof of A's claim (sketch):

1. $\forall_j SKb \notin K_j$.
2. Inspection of derivation rules learns that K is only extended with $\{na\}_{PKb}$.

B's claim can be refuted.

- Extending/finishing/simplifying the model
 - type flaw attacks
 - forward secrecy
 - several forms of authentication
 - other security claims
- Relation to other approaches (strand spaces, Casper/FDR, BAN logic)
- Case studies (experimenting with intruder models, protocol sets)
- Tool support
 - model checker
 - proof assistant

- Preliminary results are encouraging.
- Modular and general.
- Framework for proving
 - correctness of security protocols,
 - theoretical properties.
- Better formal basis than the methods mentioned earlier.
- Prototype of model checker.