Assurance and formal models

Chaire Informatique et sciences numériques
Collège de France, cours du 18 mai 2011

Assurance

Specification and implementation
(review)

For any system:

- Specification: *What is it supposed to do?*
- Implementation: *How does it do it?*
- Correctness: *Does it really work?*

In security:

- Specification: *Policy*
- Implementation: *Mechanism*
- Correctness: *Assurance*

Assurance vs. security by obscurity

- In many systems, obscurity (not correctness) is a goal. E.g.,
  - spam filters,
  - censorship systems,
  - computer games,
  - military systems.

- Obscurity may *sometimes* help, at least *for a while*, *in combination* with other precautions.

The enemy knows the system.

[C. Shannon]

For example, motivated attackers typically can learn how cryptosystems work.
It is much easier to protect only the keys.
(See Kerckhoff's principle in cryptography.)

⇒ **The security of a system cannot depend on secrecy of specification or implementation.**

⇒ **Policies must be appropriate, mechanisms must actually be correct.**

Assurance

Some strategies and techniques:

- open design (maybe open source?),
- specifications and proofs,
- testing,
- processes,
- certification,
- economy of mechanism, and
  the trusted computing base (TCB).

The TCB

Trusted Computing Base: *the collection of hardware, software, and set-up information on which the security of the system depends.*

Also:

- The part of the system that has to be right.
- The part of the system that may appear to violate its security policy.

The TCB (cont.)

Ideally:

- The TCB should be precisely defined, small, and simple.
- The TCB should be specified, tested, and verified.

In practice:

- Often, lots of dubious code is put in the TCB.
- The TCB gets big and not trustworthy.

cnet news

March 31, 2010 9:50 AM PDT

Vietnamese dissidents targeted by botnet attacks

by Tom Krazit

Malware that was disguised as a popular Vietnamese-language keyboard driver for Windows users was used to create a botnet, according to blog posts from **Google's Neel Mehta** and **McAfee Chief Technical Officer George Kurtz**. That botnet was then used to target blogs rallying against a bauxite mining project in Vietman, employing DDoS (Distributed Denial of Service) attacks to shut down those blogs, according to the posts.

Formal models and proofs
(in particular for security protocols)

What is different about security (1)

- Wish for some guarantees despite lucky, powerful, and persistent attackers.
  - Even if the attacker controls the network.
  - Even if a session key is compromised.
  - Even if an insider is dishonest.
  - …

What is different about security (2)

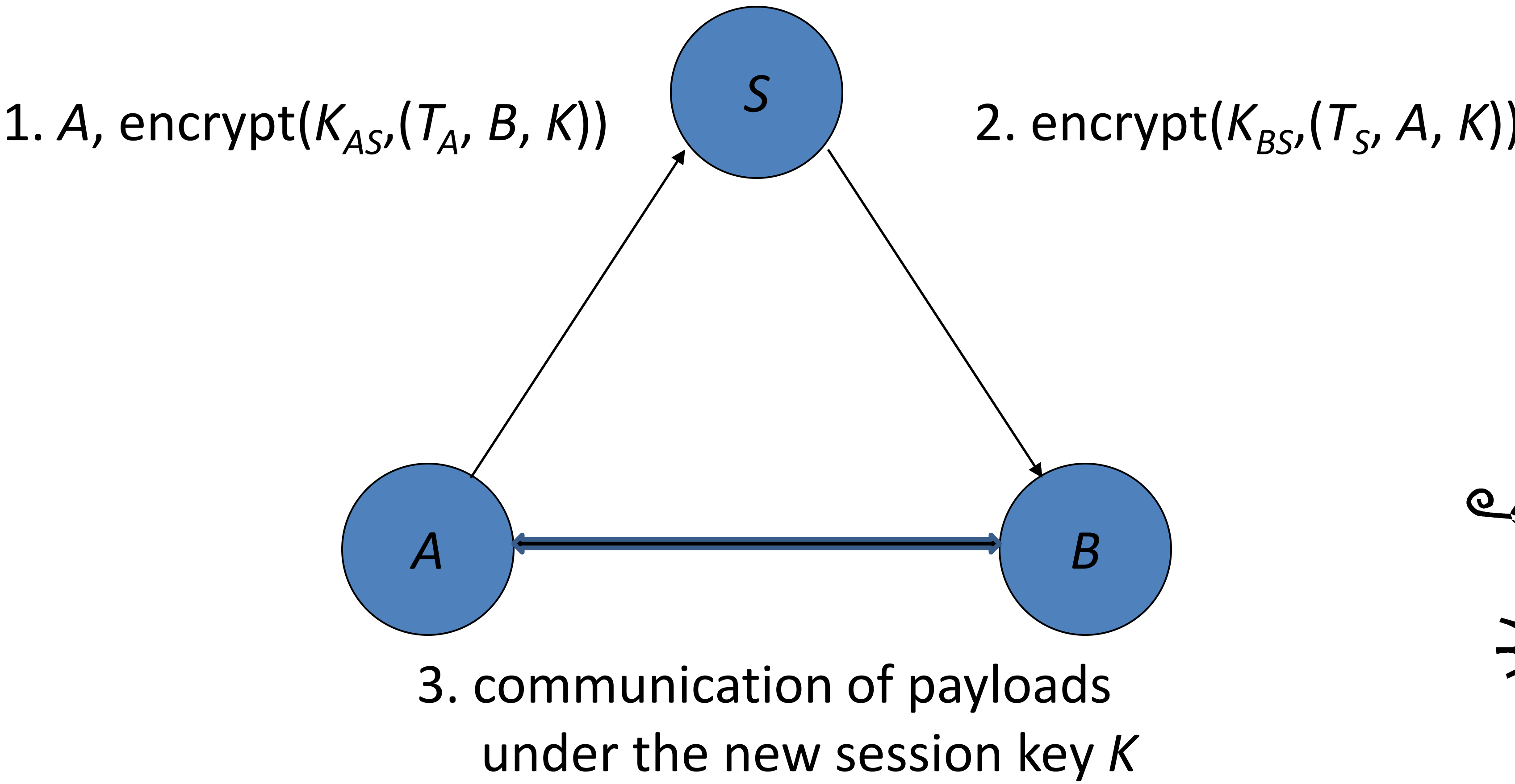
- Attacks that exploit the limitations of models.
  - Binary-level exploits despite "secure" languages.
  - Power analysis on "secure" cryptography.
  - …

- Doing without full functional correctness:

  Message authenticity and secrecy,
  not message correctness.

These characteristics impact models and proofs.

The WMF protocol (reminder)



T_A, T_S are timestamps.

Here encrypt is symmetric encryption. It may include authentication.

What the messages actually mean

1) *K* is a good key for *A* and *B* around time T_A
2) *A* says that *K* is a good key for *A* and *B* around time T_S

Understanding the meaning of messages is central to designing and analyzing protocols.

Even imprecise, informal meanings can be extremely helpful.

A first analysis in a logic of authentication (late 1980s)

- Replace messages with formal representations of their meanings.
- Set out assumptions:
  S believes (K_{AS} is a good key for A and S)
  S believes fresh(T_A)
  B believes (A controls (K is a good key for A and B))
  …
- Reason with a few general rules.
- Conclude:
  A believes (K is a good key for A and B)
  B believes (K is a good key for A and B)

Comments on a logic of authentication

- Served for finding many subtleties and errors.
- Explained protocols.
- Highlighted assumptions and conclusions.
- Used by many people, including protocol designers.

- Lacked a clear link with operational models of protocols or clear cryptographic justification (but see *PCL*).
- Required more creativity as one moved away from the classic key-exchange protocols.

Some other approaches

(not a complete or orthogonal list)

- Informal but rigorous frameworks based on probabilities and complexity theory.
- Theorem proving, e.g., with Coq or Isabelle.
- Finite-state model checking, e.g., with FDR.
- Type systems and other static analyses for programming languages (and process calculi).

Some observations

- Most security protocols have ambiguities, subtleties, and flaws.
- Many of these have to do with cryptography.
- Many of these don't have to do with the details of cryptography.

$\Rightarrow$ *For design, implementation, and analysis, abstract views of cryptography are practical.*

Other low-hanging fruit

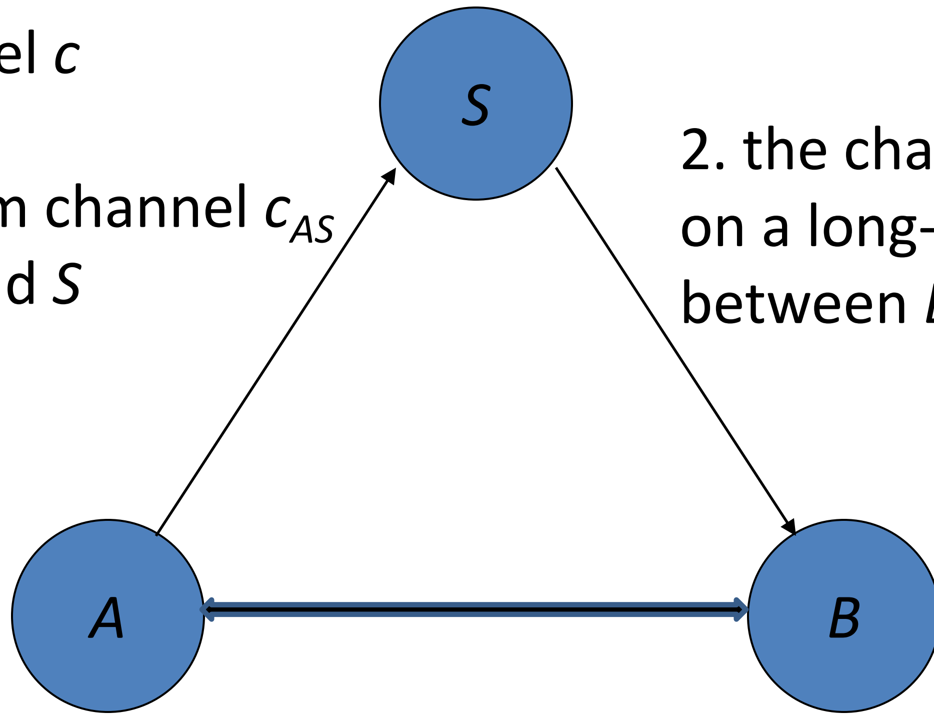
An example of a mundane ambiguity:

The simplest fix is to require that a SSL implementation receive a change cipher spec message before accepting a finished message. (Indeed, there is some language in the specification which could be interpreted to mandate this restriction, although it is not entirely clear.) [. . .] at least one implementation has fallen for this pitfall.

(Wagner and Schneier)

The WMF protocol, more abstractly

1. new channel *c*
for *A* and *B*,
on a long-term channel c_{AS}
between *A* and *S*



2. the channel *c*
on a long-term channel c_{BS}
between *B* and *S*

3. communication of payloads
on new private channel *c*
(e.g., *A* sends *M* to *B* on *c*)

Towards a language for protocols

- The pi calculus is a general, simple language for concurrent processes that communicate by sending messages on named channels.
- It includes an operator ν ("new") for generating fresh channels.

$$(\nu\ c)(\overline{c}\langle M\rangle \mid c(x).\ldots)$$

"(new c)(send M on c and, in parallel, receive x on c...)"

  - Here two processes run in parallel.
  - One sends *M* to the other on a fresh channel *c*.
  - x is a bound variable.

Syntax

M, N	$::=$		terms (i.e., data)
	$\mid$	x	variable
	$\mid$	n	name

$P, Q ::=$		processes (i.e., programs)
$\mid$	nil	nil process (may be omitted)
$\mid$	$\overline{M}\langle N \rangle.P$	sending
$\mid$	$M(x).P$	receiving
$\mid$	$(\nu n)P$	restriction ("new")
$\mid$	$P \mid Q$	parallelism
$\mid$	$!P$	replication

An abstract version of the protocol

$$A(M) \quad = \quad (\nu c)\overline{c_{AS}}\langle c\rangle.\overline{c}\langle M\rangle$$

"With new c, send c on c_{AS}, send M on c."

An abstract version of the protocol

$$A(M) \quad = \quad (\nu c)\overline{c_{AS}}\langle c\rangle.\overline{c}\langle M\rangle$$

$$S \quad = \quad c_{AS}(x).\overline{c_{SB}}\langle x\rangle$$

"Receive x on c_{AS}, forward it on c_{SB}."

An abstract version of the protocol

$$\begin{aligned} A(M) &= (\nu c)\overline{c_{AS}}\langle c\rangle.\overline{c}\langle M\rangle \\ S &= c_{AS}(x).\overline{c_{SB}}\langle x\rangle \\ B &= c_{SB}(x).x(y).nil \end{aligned}$$

"Receive x on c_{SB}, receive y on x."

An abstract version of the protocol

$$
\begin{aligned}
A(M) &= (\nu c)\overline{c_{AS}}\langle c\rangle.\overline{c}\langle M\rangle \\
S &= c_{AS}(x).\overline{c_{SB}}\langle x\rangle \\
B &= c_{SB}(x).x(y).nil \\
P(M) &= (\nu c_{AS})(\nu c_{SB})(S \mid A(M) \mid B)
\end{aligned}
$$

"With new c_{AS} and c_{SB},
run S, $A(M)$, and B in parallel."

Secrecy as equivalence

- Secrecy properties can be phrased as equivalences between processes.
  - For example, $P(M)$ and $P(N)$ are equivalent, for all M and N.

Secrecy as equivalence

- Secrecy properties can be phrased as equivalences between processes.
  - For example, $P(M)$ and $P(N)$ are equivalent, for all M and N.

Here, many notions of "equivalence" will do.

Secrecy as equivalence

- Secrecy properties can be phrased as equivalences between processes.
  - For example, $P(M)$ and $P(N)$ are equivalent, for all M and N.

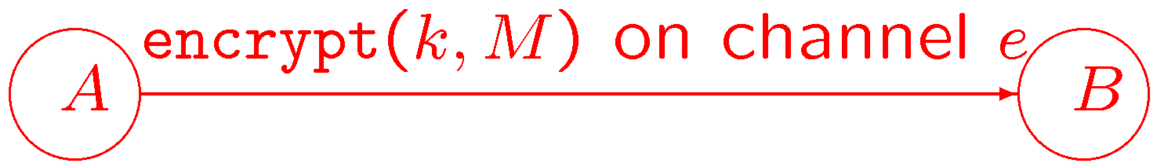
Here, many notions of "equivalence" will do.

- Other security properties can also be presented and proved formally, as equivalences or as properties of executions.

Extending the pi calculus

- In the pure pi calculus, we can easily represent systems like

- But it is much harder (or impossible) to represent the use of cryptography, as in:



Work with Blanchet, Fournet, Gordon, and others

The applied pi calculus

We add function symbols, as in:

$$(\nu\ k)(\ldots \mathtt{encrypt}(k, M) \ldots \mathtt{decrypt}(k, x) \ldots)$$

- Here the operator ν generates a key.
- Encryption and decryption are function symbols, with equations.

$$\mathtt{decrypt}(\mathtt{encrypt}(x, y), x) \ = \ y$$

Expressiveness

- Representing protocols such as WMF is just "a matter of programming".
- For example, we may write:

$$(\nu k).\overline{c}\langle \mathtt{encrypt}(k, 0)\rangle$$

Here, a process reveals a term that uses a fresh name k without revealing k itself.

  – This does not arise in the pure pi calculus.

  – It is a source of expressiveness and complications.

Syntax for terms

M, N	$::=$		terms
	$\vert$	x	variable
	$\vert$	n	name
	$\vert$	$f(M_1, \ldots, M_k)$	function application

where f is a function symbol of arity k (and optionally also with conditions on types)

Syntax for processes

$P, Q ::=$		processes
$\mid$	nil	nil process
$\mid$	$\overline{M}\langle N \rangle.P$	sending
$\mid$	$M(x).P$	receiving
$\mid$	$(\nu n)P$	restriction
$\mid$	$P \mid Q$	parallelism
$\mid$	$!P$	replication
$\mid$	$\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q$	conditional

Equality

Equality is defined by an equational theory (basically, a set of equations). For example:

- For pairs:

$$\texttt{fst}(\texttt{pair}(x, y)) = x$$
$$\texttt{snd}(\texttt{pair}(x, y)) = y$$

Equality

Equality is defined by an equational theory (basically, a set of equations). For example:

- For pairs:

$$\texttt{fst}(\texttt{pair}(x, y)) = x$$
$$\texttt{snd}(\texttt{pair}(x, y)) = y$$

- For symmetric encryption:

$$\texttt{decrypt}(\texttt{encrypt}(x, y), x) \;=\; y$$

Equality (cont.)

- For asymmetric encryption:

$$\mathtt{adecrypt}(\mathtt{sk}(x), \mathtt{aencrypt}(\mathtt{pk}(x), y)) = y$$

Equality (cont.)

- For asymmetric encryption:

$$\mathtt{adecrypt}(\mathtt{sk}(x), \mathtt{aencrypt}(\mathtt{pk}(x), y)) \; = \; y$$

and optionally with other equations, e.g.,

$$\mathtt{pdecrypt}(x, \mathtt{pencrypt}(y, z)) = \mathtt{pencrypt}(y, \mathtt{pdecrypt}(x, z))$$

Equality (cont.)

- For asymmetric encryption:

$$\mathtt{adecrypt}(\mathtt{sk}(x), \mathtt{aencrypt}(\mathtt{pk}(x), y)) \ = \ y$$

and optionally with other equations, e.g.,

$$\mathtt{pdecrypt}(x, \mathtt{pencrypt}(y, z)) = \mathtt{pencrypt}(y, \mathtt{pdecrypt}(x, z))$$

- For probabilistic encryption:

$$\mathtt{adecrypt}(\mathtt{sk}(x), \mathtt{aencrypt}(\mathtt{pk}(x), y, z)) \ = \ y$$

Other examples

- MACs
- Digital signatures
- One-way hash functions
- XOR
- Exponentiation as used in Diffie-Hellman
- Errors
- …

Semantics: reduction

$$\overline{c}\langle M\rangle.P \mid c(x).Q \;\rightarrow\; P \mid Q[M/x]$$

where $Q[M/x]$ is the result of replacing x with M in Q

$$\mathit{if}\ M = M\ \mathit{then}\ P\ \mathit{else}\ Q \;\rightarrow\; P$$

$$\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q \;\rightarrow\; Q$$

if $M \neq N$

(In addition, some other trivial rules allow rearranging processes, e.g., by commutativity and associativity of parallel composition.)

Equivalence

- Two processes P and Q are *testing equivalent* if no context R can distinguish them.
  - For a given channel n, $P \mid R$ may output on n if and only if $Q \mid R$ may output on n.
  - The context R may represent an attacker.

Equivalence

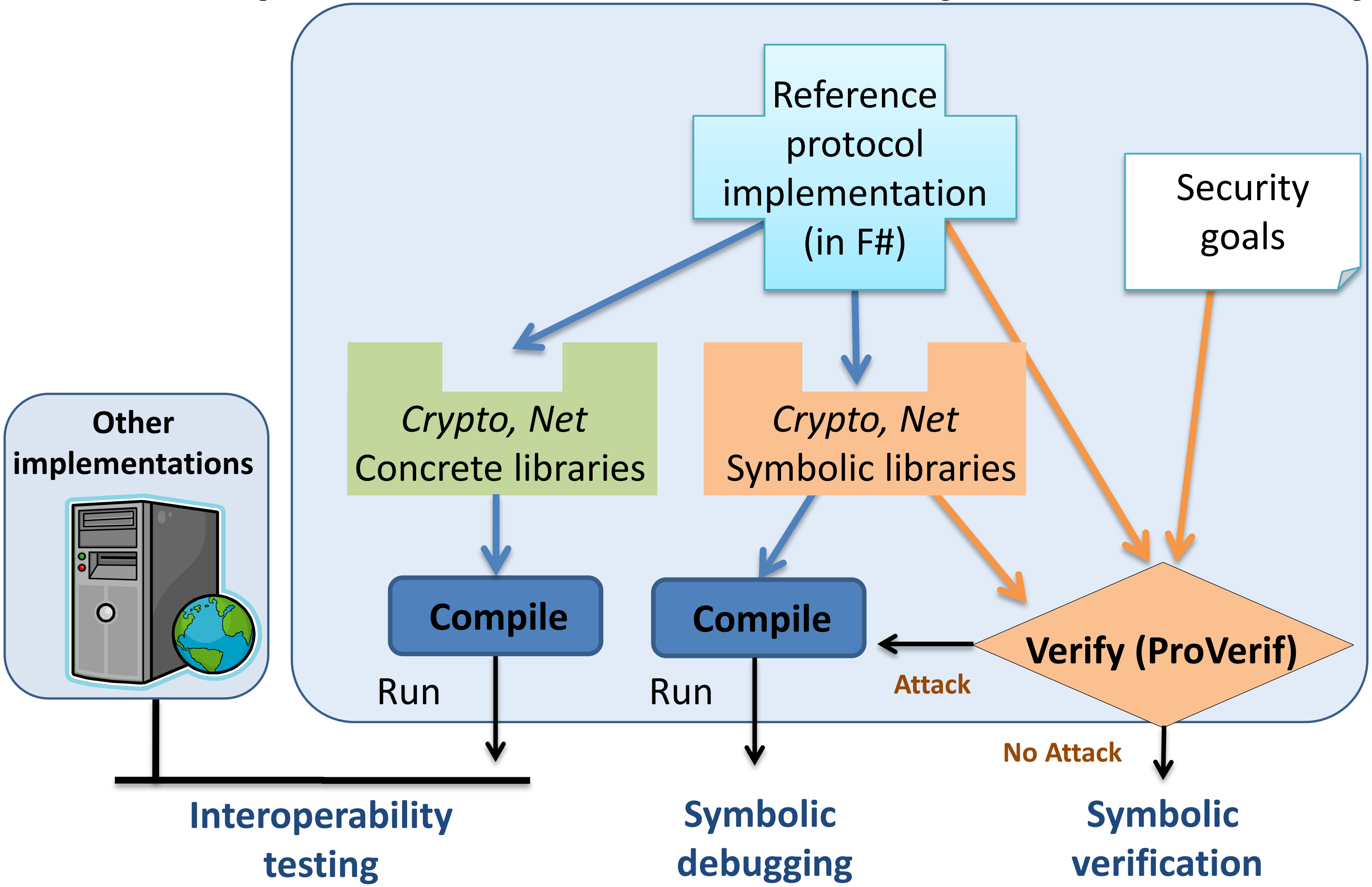
- Two processes P and Q are *testing equivalent* if no context R can distinguish them.
  - For a given channel n, $P \mid R$ may output on n if and only if $Q \mid R$ may output on n.
  - The context R may represent an attacker.
- This equivalence is coarse enough that, for example, it relates the two processes:

$$(\nu k).\overline{c}\langle \mathtt{encrypt}(k, 0)\rangle \qquad (\nu k).\overline{c}\langle \mathtt{encrypt}(k, 1)\rangle$$

ProVerif demo

(Bruno Blanchet)

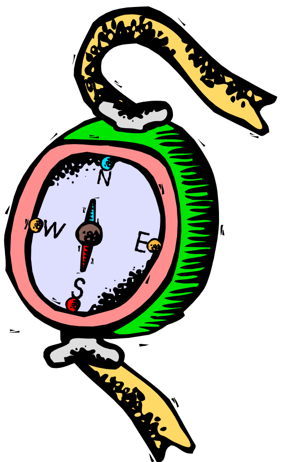
Analyzing reference implementations (symbolically)



Work at MSR Cambridge and MSR-INRIA Joint Centre [Bhargavan et al.]

Four current research directions

1. Models, proof techniques, and tools (e.g., type systems).
2. Analysis of particular protocols.
3. Analysis of actual implementations.
4. Relating and combining symbolic and computational approaches.



Some reading

- The chapter on assurance and evaluation in Anderson’s book.
- My “Security Protocols: Principles and Calculi (Tutorial Notes)”, and its references.
- “Modular Verification of Security Protocol Code by Typing”, by Bhargavan, Fournet, and Gordon.


SÉCURITÉ
POUR LA PATRIE
EGALITÉ DE DEVOIRS
Dugourc inv. l'an II de la Rép. par Brevet d'Invention
G163270

Fin