Axis: SECSI

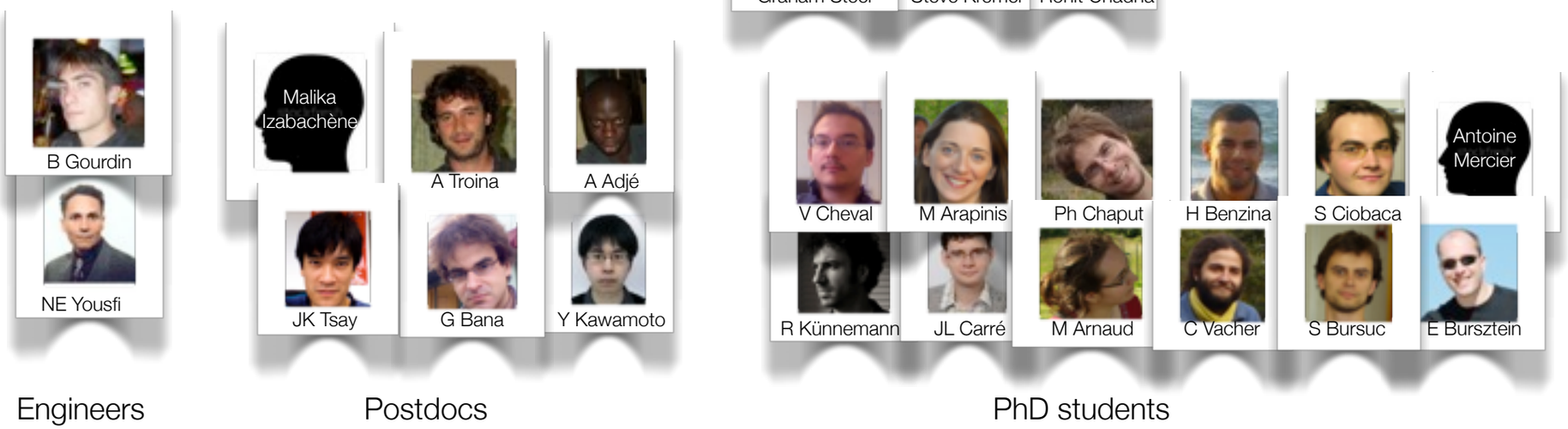
«security of information systems»

AERES evaluation of LSV – 2 December 2013

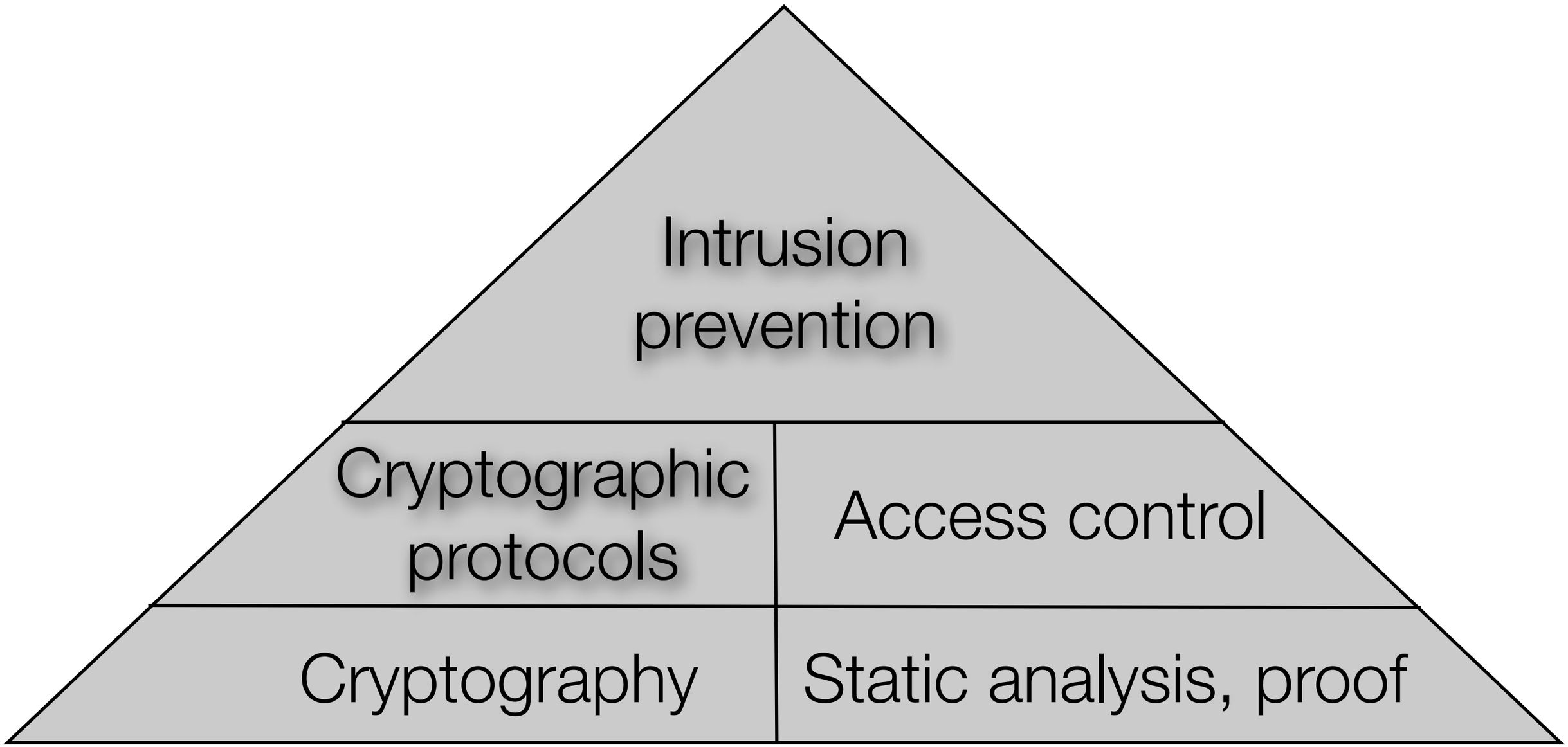
The SECSI Team 2008-2013

- SECSI = SECurité des Systèmes d'Information
- Founded in 2000, INRIA team 2002-

Former members:



The many facets of computer security

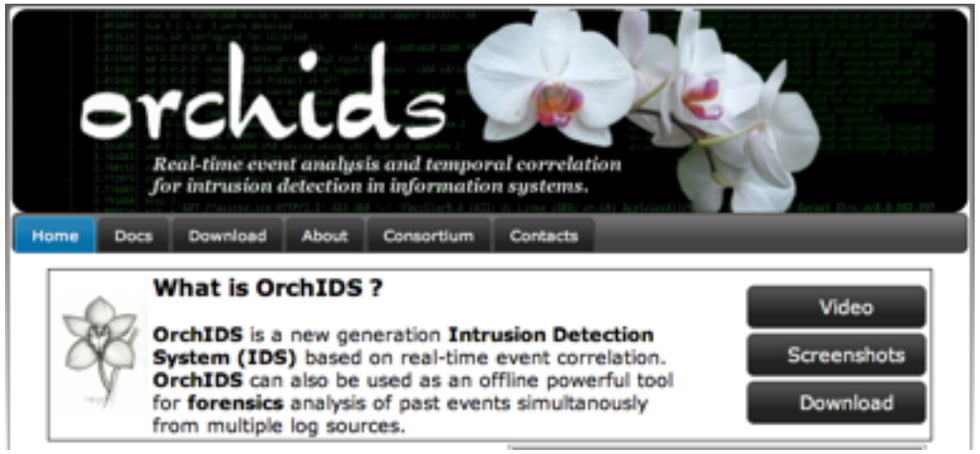


SECSI activities

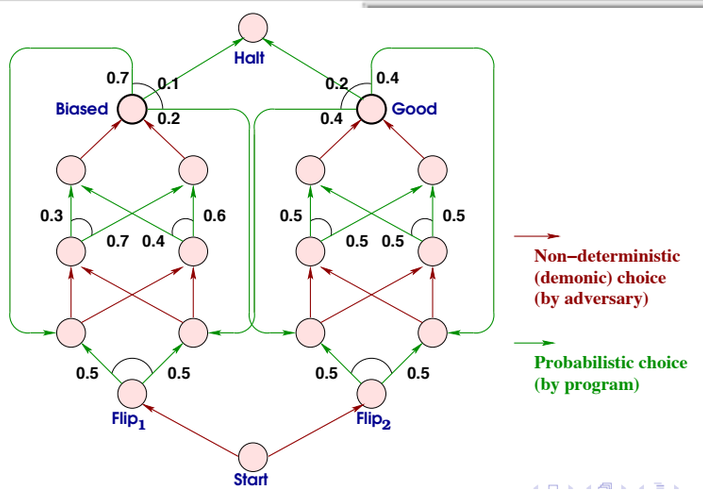
logic-based
verification techniques for
security properties

- Verification of cryptographic protocols
  see next slides

- Intrusion prevention
  Intrusion detection as **model-checking**
  The **ORCHIDS** tool, currenly valorized
  (DGA convention 2013-2016)
  Analyzing **network resilience** through
  timed games



- Semantic foundations of
  mixed probabilistic/non-deterministic choice
  Anonymity and view-consistent **schedulers**
  Belief functions, **previsions**
  Noetherian spaces and **WSTS** (spin-off)



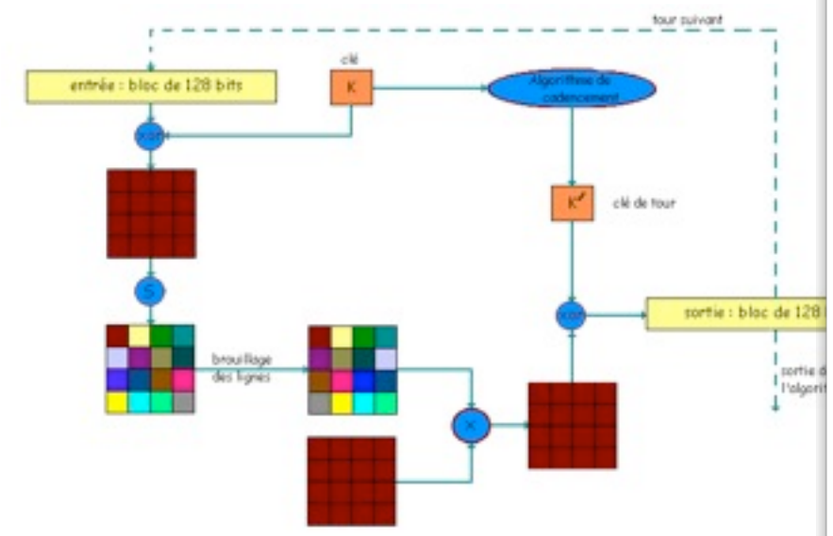
Outline

1. **Verification of cryptographic protocols**
2. Focus I: verification of electronic voting protocols
3. Focus II: computational models of security
4. Scientific project

Cryptography

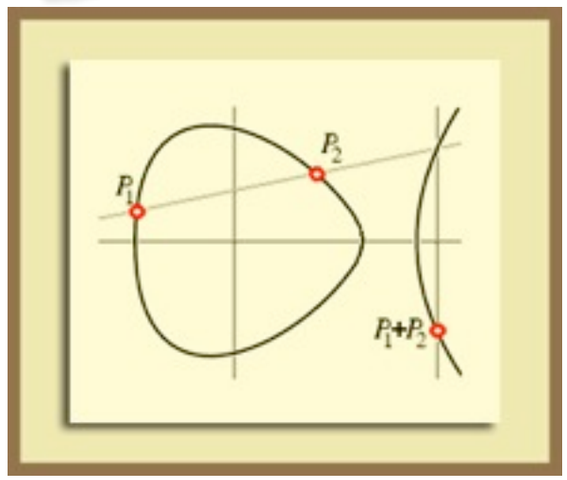
- Basic functionalities:
  encryption
  signature
  hashing
  ...

- Not our business:
  we assume some cryptography,
  with given security properties




```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG/MacGPG2 v2.0.18 (Darwin)
Comment: GPGTools - http://gpgtools.org

mQENBE/teyoBCADXBnYRCs+sddsIu+38u4pJk6AlCyJrY5uVfGl6Z4K4rwYFlnEN
Jy0pxpsJj2OV4++SuhI+wlDOHr45kXAJNh11h+jGQsICF1xgya0hCagXBnmjmFbN
/3mlNeF42HOXPOy/2xKp6orGuvl9pW+SbMo3zTOAjzcyC3RnfWm72HEMesAlnXKz
fczZGOzsmKbt5cU6Vl6A9S66NWjq3QZ/Y8lEgAsI2mYNtcbzuzFUpxlltIjw9uDb
h1lZXeNJC7pgOtPNx2YfNzDl75z/UBp+2pJ2GZImf9gk3gLiydXxILsgeMhALsWF
wekdgsXkU1zohF5tpg1wF4W6oy5vH+vZ2G5RABEBAAG0QUplYW4gR291YmF1bHQt
TGFycmVjcSAoQ29uY291cnMgRU5TKSA8Z291YmF1bHRAbHN2LmVucy1jYWNoYW4u
ZnI+iQE+BBMBAgAoBQJP7XsqAhsDBQkDwmcABgsJCAcDAgYVCAIJCgsEFgIDAQIe
AQIXgAAKCRATtyehwe0YdfYoB/4yPTYDf1X1Yv/aNS4+RiC/hlqRdIvEgfAPJs7X
5QT1ZKoG6ZxQd6UFEpiAAl6drdsLSXBi4PPrN456v2ZANxwTe16mDjU0P0RN9as6
XrgVTlXAfENEYOukWDukCzhVqe8/6JkIA2kRXfdoMEd7aDuwEy/cs4luk9+0PGh/
VFIpkTJS99ORS/GQMzmtquGBAkTGSESLP0hmoB8DcCN2cHiCuCffpLjZ9phwiDDJ
zKhN59hwCAr47p0Mv6nHzhFjKVX8R4+Z6AKy8GYCsKjJvUudf+ZiR1OrZnLBlyP1
IbckFQj6QEEYLKp+PCgltJfELswPWITdEpJwHq858rzgNqy1uQENBE/teyoBCACh
8POEMbHPRbsqYRr/+w0+rCJD0WAOjj0dpvMg9wbytgMtnsL6p9CzIou3GePBPqRt
314XB1T5yKYNOXUGGrJrC/pKRl/tfRnWajE4zJLnFwkRV55XrznqELEkHfpg7cr6
YX7lzSfrtTSVwUuQrrbEhKDX9k9qOiujc6ciau5FvMGgUBemeCjVejzo3fvF71Xm
ylvc23vJmhY2IjeO8PRddpr52GY0FbZ0C78xdfKcAOOu9WC3TnrKP59YK1fGNFEj
ddlnDhS1aRWIK4taLPH23Z6i8HmkDU6rydazIMfydaL+ojE3jEUE6UFc+YNreTYE
sf17Q3cSETMBsab4DOfLABEBAAGJASUEGAECAA8FAk/teyoCGwwFCQPCZwAACgkQ
E7cnocHtGHWZ3wf+KsG+iNMo0aCeQ/L0nhUA7Rqjl3OB0Rp2y2++Lb1qEAh+unIQ
9jQXLhQm6v3qltVYH9yvwujL2Z0RpR9tWMRG+XeHXh2l1gvvMFdWBXwl7VwbP85M
lbQBJP4/SkHdzYx5EcmAj01y0TjtMzwQjCdGPBM6vNLc7y0yuBoQ747d+RHvqxSn
Ctpu4R21LC/hekI/3zEsi//KdsAJnyk+RrX10Nnck7tjlYn4G150KTOvIzAVnMvu
FbQw+2deJkOGlZjw+KE55BXGj2piqoR5BHvxbZf3u5p8SbMOhqpHnm6eTaQqj/Yj
ZNACXVvHqlP64WVCxpa3Co9mng8Sysrso8pWkQ==
=TyrM
-----END PGP PUBLIC KEY BLOCK-----
```



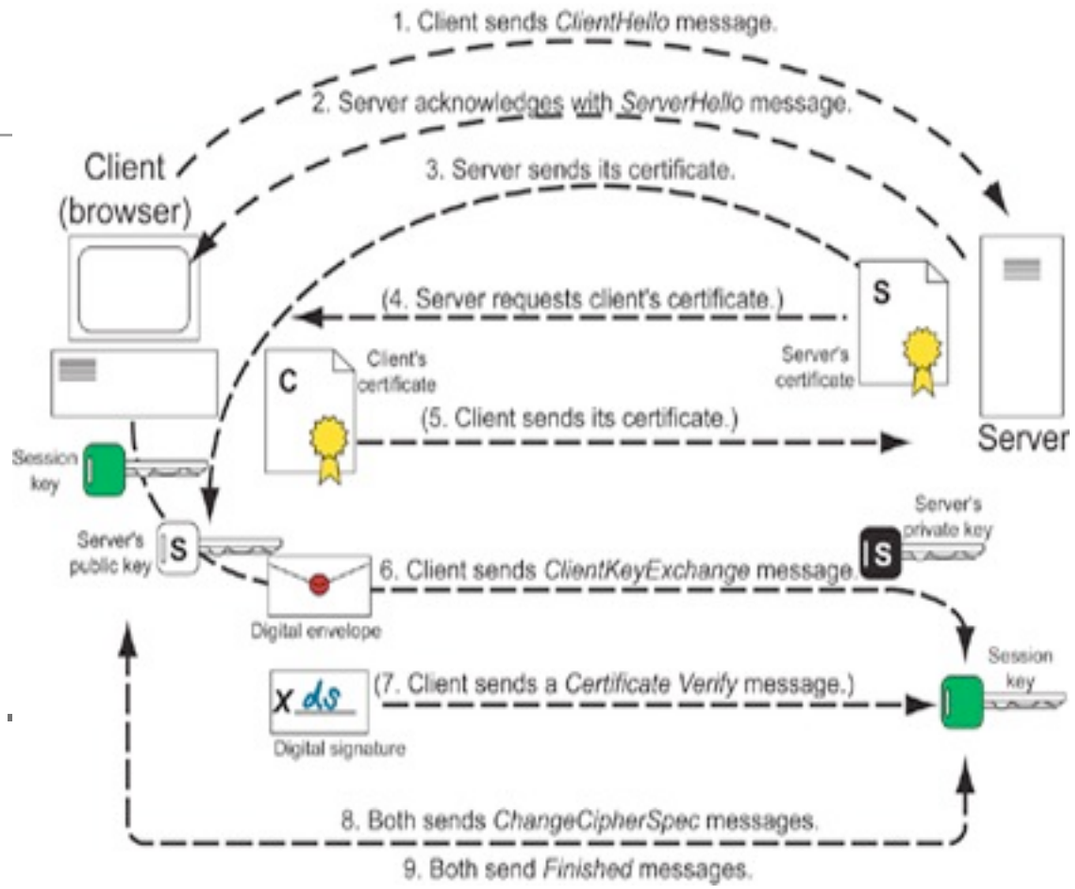
Cryptographic protocols

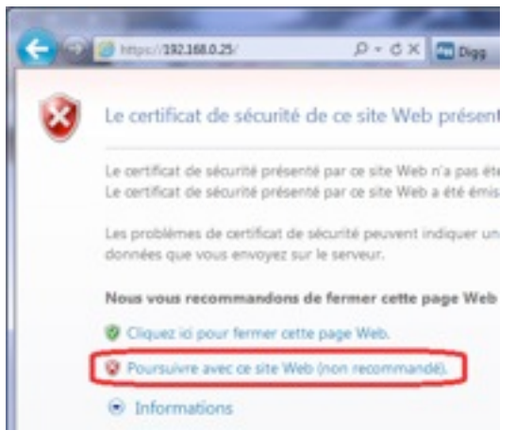
- Specifications of messages exchanged based on cryptography
  = **layer above crypto**

- Must satisfy security properties: confidentiality, authentification, anonymity...

- Can be broken
  even with perfect cryptography

- Our business: invent verification techniques for checking security properties of security protocols





Symbolic methods

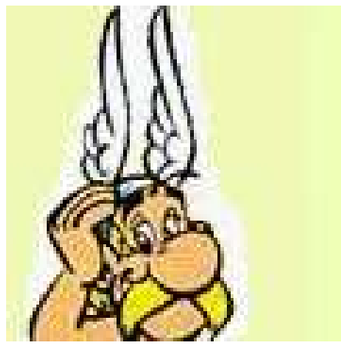
- Encode messages as first-order terms

- Encode protocols as
  Horn clauses theories / deduction systems / process algebras

$$
\begin{array}{lll}
A \to B: & \nu N_A, r_1. & \mathsf{enc}(<A, N_A>, pk(B), r_1) \\
B \to A: & \nu N_B, r_2. & \mathsf{enc}(<N_A, N_B>, pk(A), r_2) \\
A \to B: & \nu r_3. & \mathsf{enc}(N_B, pk(B), r_3)
\end{array}
$$

- Encode security property as
  Horn query (reachability) / process equivalence (e.g., voter anonymity)

The Needham-Schroeder protocol



$$
\begin{array}{lcll}
A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\
B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Questions

- Is N_b a shared secret between A and B?
- When B receives $\{N_b\}_{\mathrm{pub}(B)}$, does this message really originate from A ?

An attack was found 17 years after its publication! [Lowe'95]

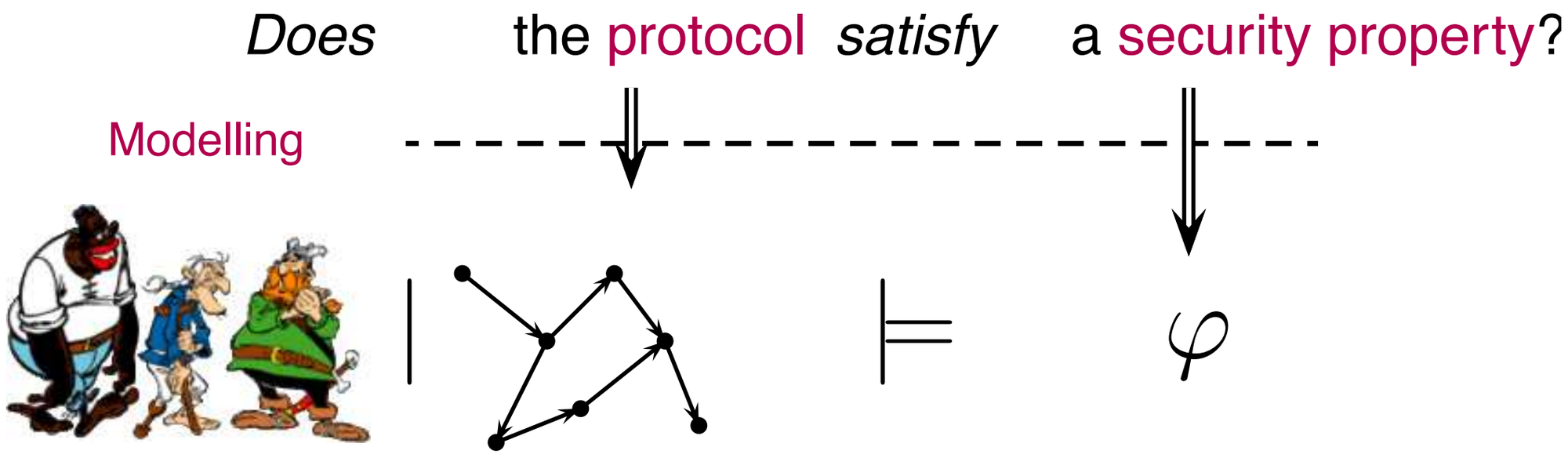
The Needham-Schroeder protocol

Answers

- Is N_b a shared secret between A and B?
  $\hookrightarrow$ No
- When B receives $\{N_b\}_{pub(B)}$, does this message really originate from A?
  $\hookrightarrow$ No

Remark : Crypto has not been broken
$\hookrightarrow$ Attack on the protocol logic.

Verification in an adversarial context



- protocol is executed in adversarial environment
- protocols are modelled in first-order logic or in process algebra (e.g., the applied pi calculus);
- attackers are any process which can be written in the applied pi calculus
- partial automation with H1 (home made) or ProVerif.

Needham-Schroeder in the pi-calculus

$$
\begin{aligned}
\mathit{INIT} \triangleq\ & \mathsf{in}(c, \mathit{xpkb}).\nu \mathit{na}.\\
& \mathsf{out}(c, \mathsf{aenc}(\langle \mathsf{pk}(\mathit{ska}), \mathit{na}\rangle, \mathit{xpkb})).\\
& \mathsf{in}(c, x).\\
& \mathsf{if}\ \mathsf{fst}(\mathsf{adec}(x, \mathit{ska})) = \mathit{na}\ \mathsf{then}\\
& \mathsf{let}\ \mathit{xnb} = \mathsf{snd}(\mathsf{adec}(x, \mathit{ska}))\ \mathsf{in}\\
& \mathsf{out}(c, \mathsf{aenc}(\mathit{xnb}, \mathit{xpkb})).0
\end{aligned}
$$

$$
\begin{aligned}
\mathit{RESP} \triangleq\ & \mathsf{in}(c, y)\\
& \mathsf{let}\ \mathit{ypka} = \mathsf{fst}(\mathsf{adec}(y, \mathit{skb}))\ \mathsf{in}\\
& \mathsf{let}\ \mathit{yna} = \mathsf{snd}(\mathsf{adec}(y, \mathit{skb}))\ \mathsf{in}\\
& \nu \mathit{nb}.\mathsf{out}(c, \mathsf{aenc}(\langle \mathit{yna}, \mathit{nb}\rangle, \mathit{ypka}))\\
& \mathsf{in}(c, z).\\
& \mathsf{if}\ \mathsf{adec}(z, \mathit{skb}) = \mathit{nb}\ \mathsf{then}\ P
\end{aligned}
$$

$$\mathit{NSPK} \triangleq \nu \mathit{ska}.\mathsf{out}(\mathsf{pk}(\mathit{ska})).!\mathit{INIT} \quad | \quad \nu \mathit{skB}.\mathsf{out}(\mathsf{pk}(\mathit{skb})).!\mathit{RESP}$$

Claim: Confidentiality ($P \not\vdash s$)

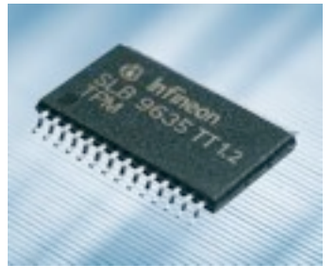
(wrong, here, of course!)

for all processes A we have that:

$$\text{if } P \mid A \rightarrow^* B \text{ then } B \not\equiv \mathsf{out}(c, s).B_1 \mid B_2$$

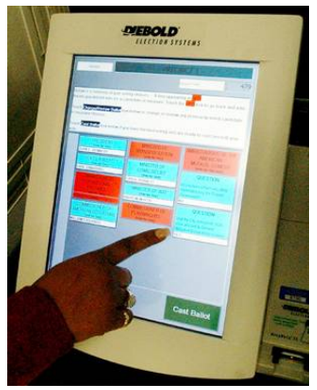
Some applications we have looked at

- Verification algorithms for the TPM
  (Trusted Platform Module)



- Attacks on privacy, European electronic passport
  (French version only)

- Verification algorithms for electronic voting protocols
  Voter anonymity, coercion resistance,
  individual/universal verifiability, eligibility, etc.



- Verification algorithms for
  security APIs
  (PKCS#11)

Outline

1. Verification of cryptographic protocols

2. **Focus I: verification of electronic voting protocols**

3. Focus II: computational models of security

4. Scientific project

Electronic voting

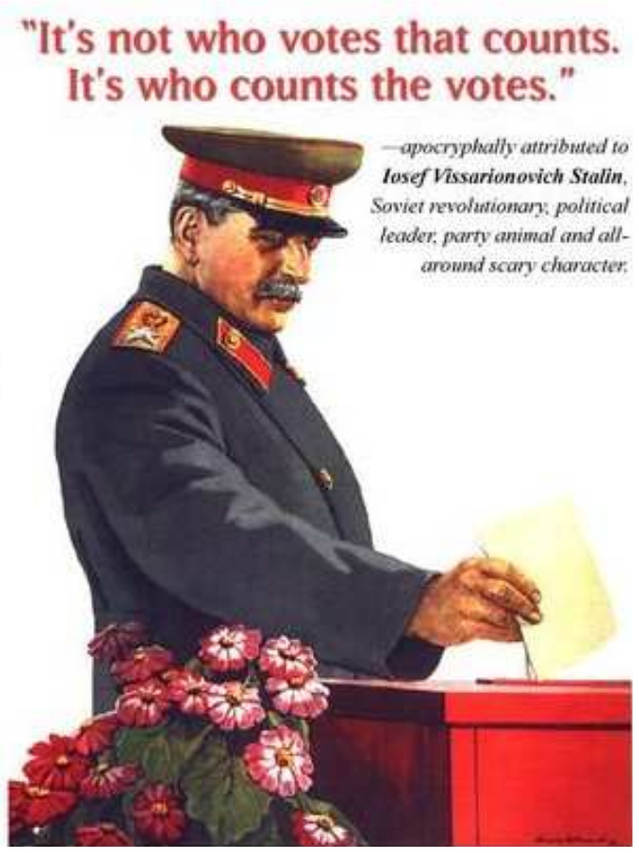
- Elections: cornerstone of democracy
  security-sensitive

- E-voting promises to be: convenient, efficient, secure
  for all kinds of elections (committees, national elections)

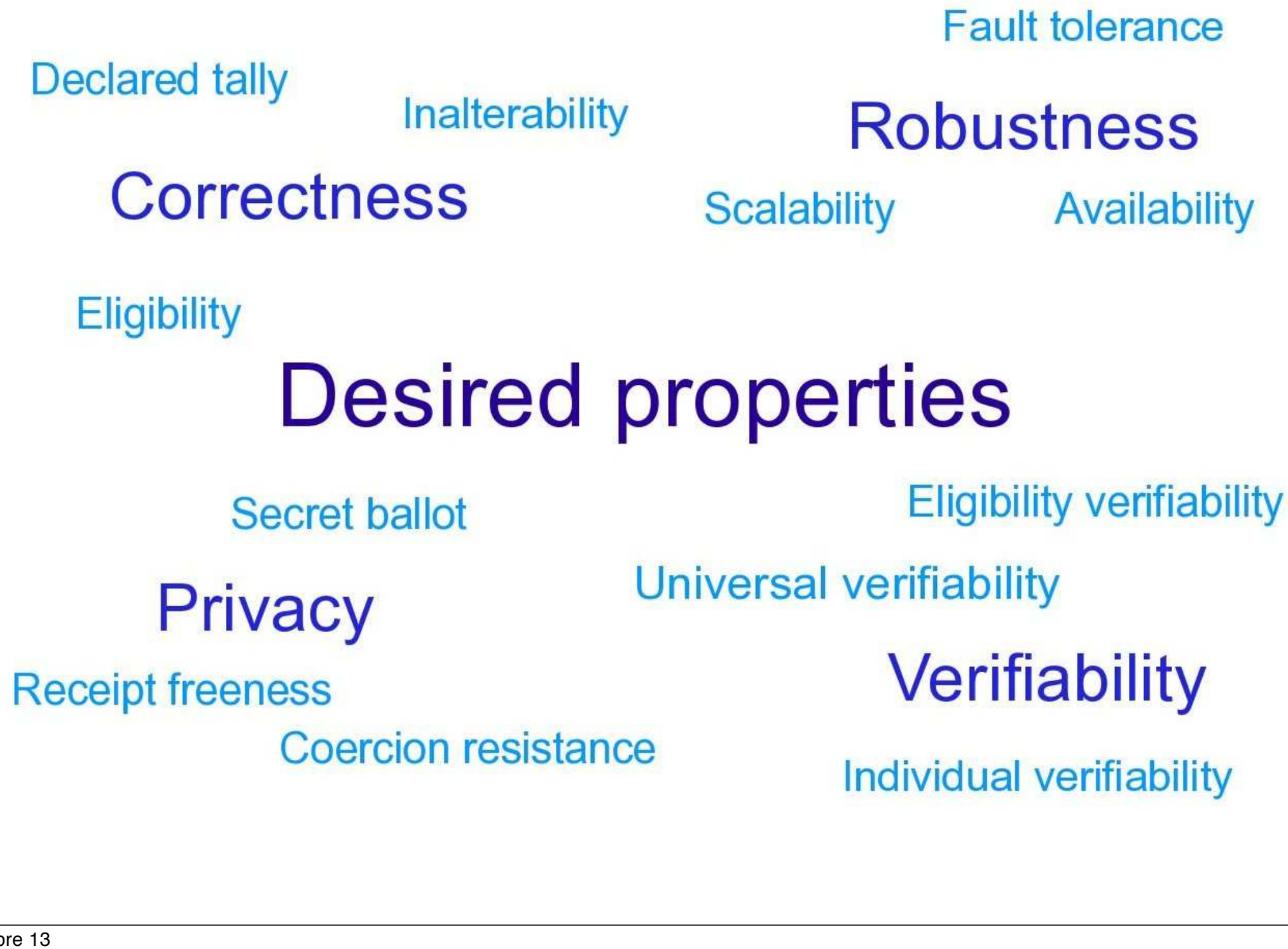
- Legally binding, e.g.:

  - Parliament elections: Switzerland, Estonia, 2011

  - French overseas parliament elections, 2013



A flurry of security properties to wish for



Verifying protocols in the applied pi-calculus

- Use equations to model crypto

- For each property, decide
  - who has to be protected
  - who is honest

- Encode honest parties as processes

- Encode security property as
  - reachability, or
  - observational equivalence
    (harder to check)

  e.g., privacy is:

$$Sys \approx Sys[A \rightleftarrows B]$$

```
dec (enc (M, pk(K)), K) = M
                [usual decryption]
enc (enc (M, K1), K2) = enc (enc (M, K2), K1)
                [modular exponentiation]
unblind (sign (blind (M, R), K), R) = sign (M, K)
                [blind signatures]
```

```
processV =
  new b; new c;
  let bcv = blind(commit(v,c),b) in
  out(ch, (sign(bcv, skv)));
  in(ch,m2);
  if getMess(m2,pka)=bcv then
  let scv = unblind(m2,b) in
  phase 1;
  out(ch, scv);
  in(ch,(l, =scv));
  phase 2;
  out(ch,(l,c)).
```

Property	*type*	*intuition*
Eligibility	reach.	only eligible voters taken into account
Fairness	reach.	without last phase, no votes published
Privacy	obs. eq.	“your vote is secret”
Receipt-freeness	obs. eq.	same, even if you cooperate with attacker

Results

Property	*Fujioka et al.*	*Okamoto*	*Lee et al.*
Vote-privacy trusted auth:	✓ none	✓ timeliness member	✓ administrator
Receipt-freeness trusted auth:	× n/a	✓ timeliness member	✓ admin. & collector
Coercion-resistance trusted auth:	× n/a	× n/a	✓ admin. & collector

- In 2010, our proofs were by hand + auxiliary lemmas done by ProVerif

- We now have tools for observational equivalence:
  AKISS (S. Ciobaca, based on Horn clauses)
  APTE (V. Cheval, deciding more cases)

Other research in security crypto protocols

- Intruder deduction under equational theories

```
dec (enc (M, pk(K)), K) = M
          [usual decryption]
enc (enc (M, K1), K2) = enc (enc (M, K2), K1)
          [modular exponentiation]
unblind (sign (blind (M, R), K), R) = sign (M, K)
          [blind signatures]
```

- From trace properties to observational equivalence properties

P_1 : $A \to B$: $\mathrm{enc}(s, \mathrm{pub}(B))$ P_2 : $A \to B$: $\mathrm{enc}(N_a, \mathrm{pub}(B))$
$B \to A$: N_a

Question: What about the secrecy of s?

- Composition results:
  are parallel compositions of secure protocols still secure? [No]

- Perhaps our most visible success: analysis of crypto APIs (PKCS#11)

Complete Insecurity [DelauneKremerSteel, CSF'08]

There is no combination of attributes that makes PKCS#11 both secure and usable.

[Steel et al., CCS'10]:

Outline

1. Verification of cryptographic protocols

2. Focus I: verification of electronic voting protocols

3. **Focus II: computational models of security**

4. Scientific project

Symbolic models vs. reality

- Remember that Needham-Schroeder was broken?

$$
\begin{array}{lcll}
A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\
B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Symbolic models vs. reality

- Remember that Needham-Schroeder was broken?

$$
\begin{array}{lcll}
A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\
B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Add *B* here (Lowe)

- Needham-Schroeder-Lowe is secure
  ... as proved by many systems

Symbolic models vs. reality

- Remember that Needham-Schroeder was broken?

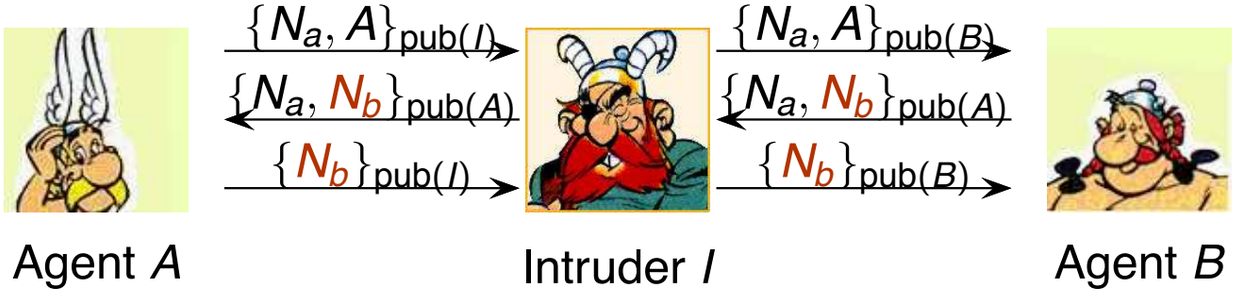
$$
\begin{array}{lcll}
A & \rightarrow & B: & \{A, N_a\}_{\mathrm{pub}(B)} \\
B & \rightarrow & A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A & \rightarrow & B: & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Add *B* here (Lowe)

- Needham-Schroeder-Lowe is secure ... as proved by many systems

- Now implement encryption by the secure ElGamal encryption scheme

- The implementation is not secure! ... Lowe's attack applies anew



- **Problem**: ElGamal is malleable, i.e. satisfies additional equations

Symbolic models vs. reality

- In crypto, one prefers computational models over symbolic models

Adversary is now a PP-time Turing machine

- Verification can be done using CryptoVerif, or EasyCrypt

- or we may bring the computational and symbolic models closer:
  - add **new equations** to the symbolic model
  - or replace crypto primitives by ones with **less equations**

- In any case, require a full abstraction result (a.k.a., computational soundness)

Soundness results

- Many results, starting from Abadi and Rogaway 2000.
  Key point: make crypto assumptions precise
  and as weak as possible

> **Theorem [HCL + Cortier 2008]**:
> Symbolic observational equivalence of processes implies computational indistinguishability,
> for IND-CPA + IND-CTXT encryption schemes.

- Under fairly general additional assumptions

- Still...

Beyond standard assumptions for soundness

- No key cycle / existence of a key hierarchy

- Encryption is plaintext-concealing, should it be:
  key-concealing?
  length-concealing?

- Keys are generated at random (honestly): dishonest keys?

- Static corruption: extendible to adaptive corruption?

- We have had a few good results,
  but the subtle pitfalls of computational security are **accumulating**
  Needed: a completely new approach

Outline

1. Verification of cryptographic protocols

2. Focus I: verification of electronic voting protocols

3. Focus II: computational models of security

4. Scientific project

Scientific project

logic-based
verification techniques for
security properties

- New, realistic, symbolic models of security
  The Comon-Lundh/Bana approach: axiomatize what adversary **cannot** do
  Find **minimal** security assumptions on primitives

- Equivalence properties and S. Delaune's VIP program:
  models (new privacy properties in routing, in vehicular networks)
  algorithms (trace/observational equiv., finer abstractions)
  combination (of decision procedures, of intruder theories)
  composition (when is P1 || P2 secure? When is !P secure?)

- New proof techniques:
  automated/mechanized **formal** proofs (e.g., in Coq; circuit security)
  probabilistic/non-deterministic systems (**full abstraction** results)

Recommendations from the 2009 evaluation committee... and what we did about them

- «A améliorer : collaborations industrielles, valorisation des outils logiciels»
  → Dassault Sys., Hispano-Suiza, Safran in CPP; EADS IW, Cassidian, Thalès
  → CryptoSense **startup**
  → not just industry! **DGA**, ANSSI

- «Recommandations : [...] objectifs ambitieux et pertinents [...] unité de l'axe de recherche et sa place au LSV moins évidentes qu'il y a quatre ans.»
  → SECSI is still applying **logic** to **security**: LSV never ceased to be our **home**

- « [...] intégrer ces approches, à l'occasion d'études de cas plus appliquées.»
  → e-voting, secure MANET routing, European e-passport, TPM, crypto APIs...

- « [...] techniques de vérification directement dans le modèle calculatoire.»
  → the Comon-Lundh/Bana approach,
  a novel **symbolic** framework for **computational** security