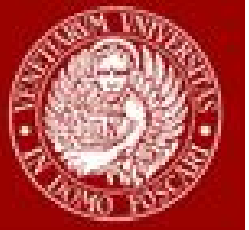


Secure Communication Primitives

Riccardo Focardi

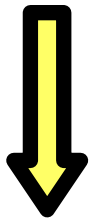
in collaborazione con Michele Bugliesi

Formal Methods for Security Research Group
(A. Bossi, M. Bugliesi, A. Cortesi, R. Focardi, S. Rossi,
M. Centenaro, D. Macedonio, P. Modesti)

Terzo Workshop Annuale del Dipartimento di Informatica
giovedì, 17 aprile 2008

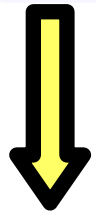
Secure Communication: How?

Msg



Encrypt with K_{AB}

$C = \{ Msg \}_{K_{AB}}$



Decrypt with K_{AB}

Msg

Secure Communication: which properties?

Alice

$\{ Msg \}_{K_{AB}}$

Bob

If K_{AB} is only known by Alice and Bob:

- Secrecy: only Alice and Bob can read Msg
- Integrity: only Alice and Bob can modify Msg

No authentication ...

Alice

Bob

$\{ Msg \}_{K_{AB}}$

$\{ Msg \}_{K_{AB}}$

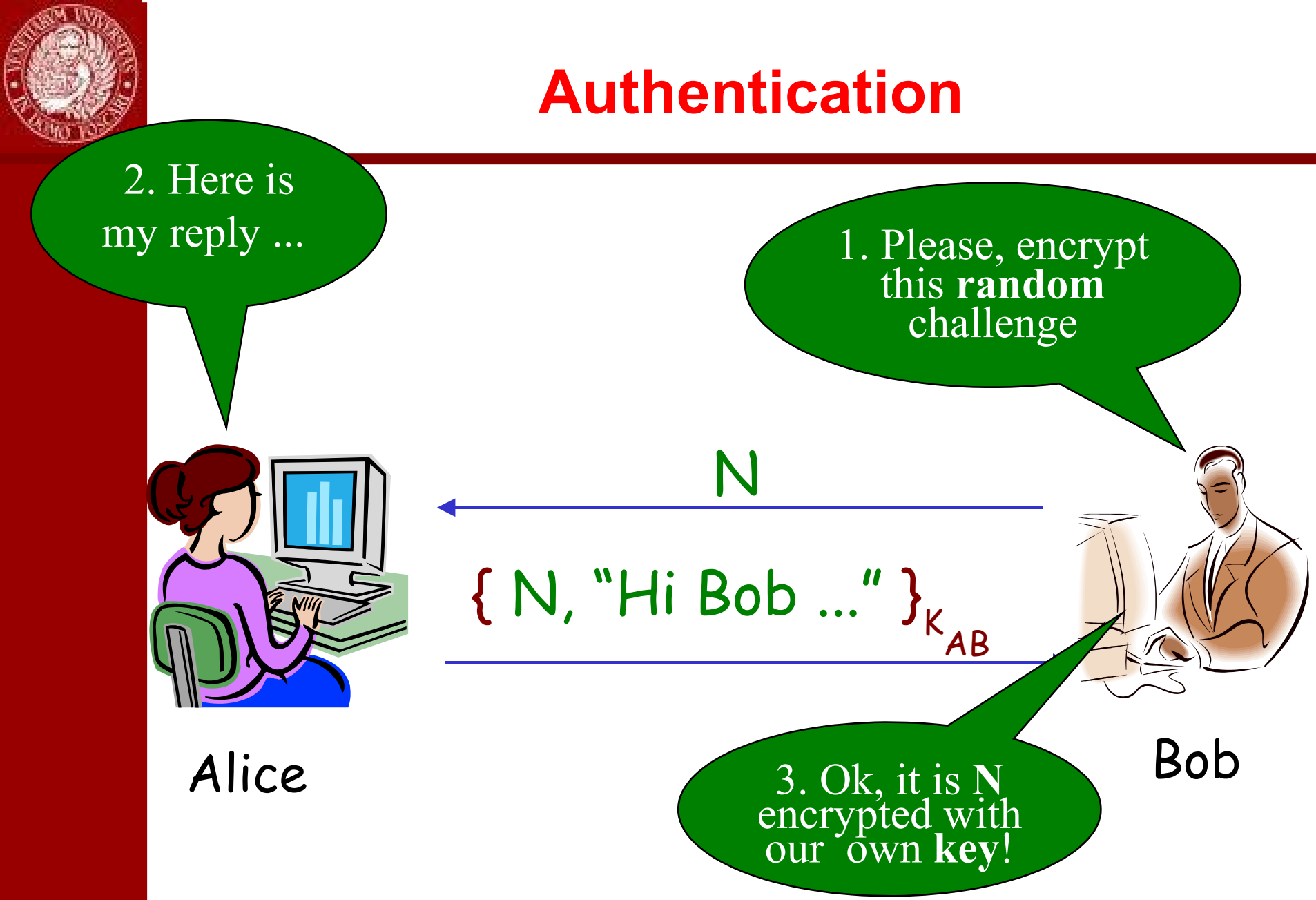
Intruder

$\{ Msg \}_{K_{AB}}$

.....

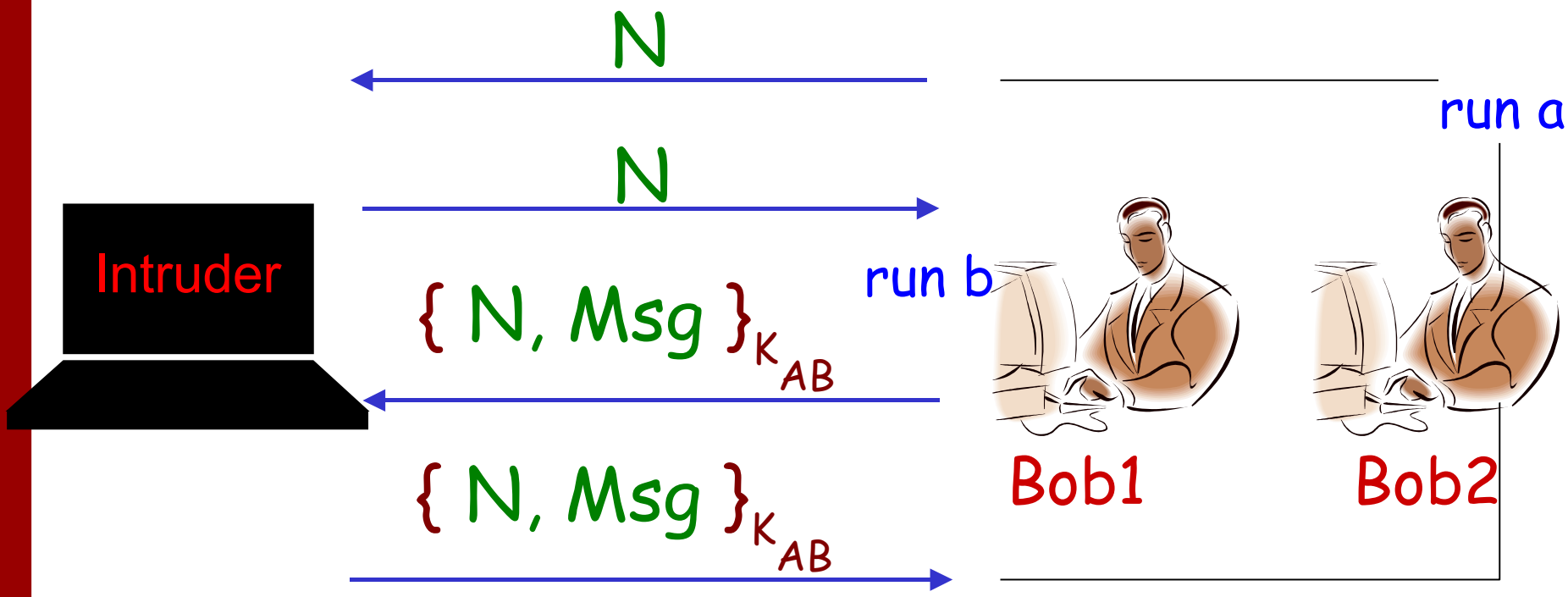
... messages can be replayed!

Authentication

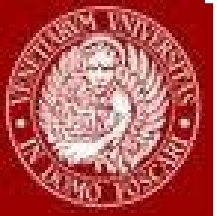


"Nonce" N enables Bob to discover possible replays

A standard "reflection attack"

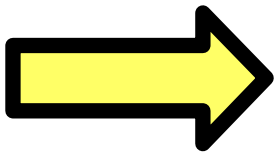


- Bob authenticates Msg from Alice
- "Direction" should be explicit as in { A, N, Msg }$_{K_{AB}}$
- Many protocol variants, e.g., public-key based, ...

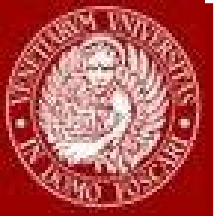


Lesson learned ...

- Many different security properties
  - many more: non-repudiation, fairness, anonymity, ...
- Many possible implementations
  - shared key, public key, specific crypto-techniques, ...
- Subtle attacks even on simple protocols
  - Message format is relevant
- Protocol composition may be non-trivial

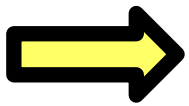


*How can we write **secure** distributed programs?*

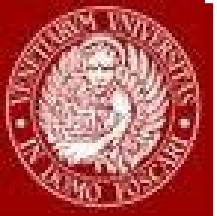


How Secure is a Program?

- Formal Verification (Model Checking)
  - a lot of work done, and attacks discovered, **but**:
    - *finite number of sessions and participants*
    - *state space explosion exponential in the number of participants*
- Language-Based (Static) Analysis
  - types / control-flow / abstract-interpretation :
    - *unbounded number of sessions and participants!*
    - *quite complicate even on simple protocols*



NEW: Primitives which are secure by construction

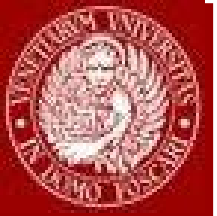


Secure primitives

- As standard message-passing but
  - explicit security properties and identities
  - no “visible” cryptography
    - *the compiler will use the right protocols for us*

A Sends B (Msg_1,...,Msg_n[,Secret][,Intgr][,Auth]...)

B Receives A (x1, ..., xn[,Secret][,Intgr][,Auth]...)



An example: secure session

Alice	**Bob**
New C	New D
A Sends B (C, [Secret,] Auth)	B Rcv A (y, [Secret,] Auth)
A Rcv B (x, [Secret,] Auth)	A Sends B (D, [Secret,] Auth)
<SESSION>	<SESSION>
C Sends x (MSG, Secret)	D Rcv y (z, Secret)
....	...

Compiled in different ways depending on the “Secret” Flag

Without flag "secret"

PK_C, SK_C

Alice

N

$\{ B, PK_C, N, N' \}_{Sign_A}$

$\{ A, PK_D, N', N \}_{Sign_B}$

PK_D, SK_D

Bob

$\{ MSG1 \}_{PK_D}$

$\{ MSG2 \}_{PK_C}$

With flag “secret”

C

D

N

$\{ B, \{C, A\}_{PK_B}, N, N' \}_{Sign_A}$

$\{ A, \{D, B\}_{PK_A}, N', N \}_{Sign_B}$

Alice

Bob

$Ks = h(C,D)$

$Ks = h(C,D)$

$\{ MSG1 \}_{Ks}$

$\{ MSG2 \}_{Ks}$

Conclusion

- Secure primitives might help writing secure distributed applications / web-services / ....
- Theory developed in [Bugliesi-Focardi, *IEEE Computer Security Foundation Symposium 2008*]
- To do's:
  - Theorems for secure implementation
  - Security types
  - Real languages (OCAML, F#, Paolo Modesti)
  - new properties (e.g. e-voting, Matteo Zanioli)
  - language integration (next talk, Matteo Centenaro)
  - ...