Logical Foundations for Security Protocol Analysis

Patrick Lincoln John Mitchell
Mark Mitchell Andre Scedrov

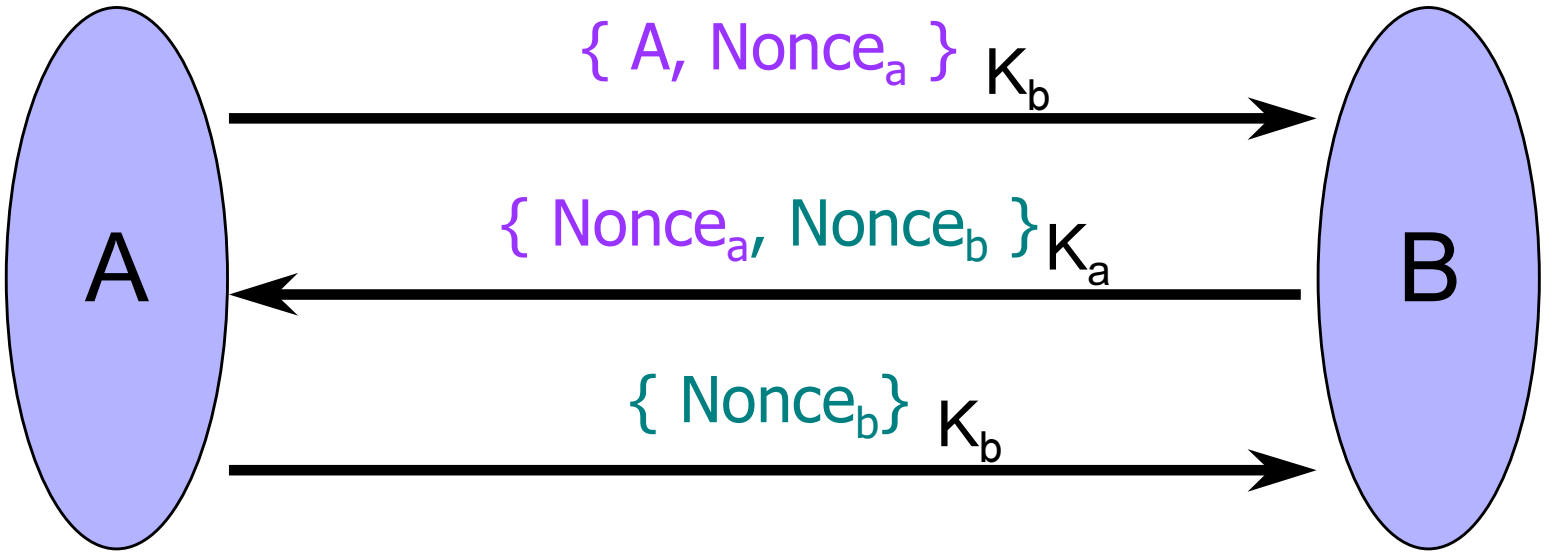
Correctness vs Security

- Program or System Correctness
  - Program satisfies specification
    - For reasonable input, get reasonable output
- Program or System Security
  - Program resists attack
    - For unreasonable input, output is not completely disastrous
- Main difference
  - Active interference from environment

Main Scientific Problem

- How powerful is the adversary?
  - Simple replay of previous messages
  - Decompose, reassemble and resend
  - Statistical analysis of network traffic
  - Timing attacks
- No absolute notion of security
  - Weak adversary: any correct system is secure
  - Strong adversary: nothing is secure
    - If I can read your mind, you have no secrets

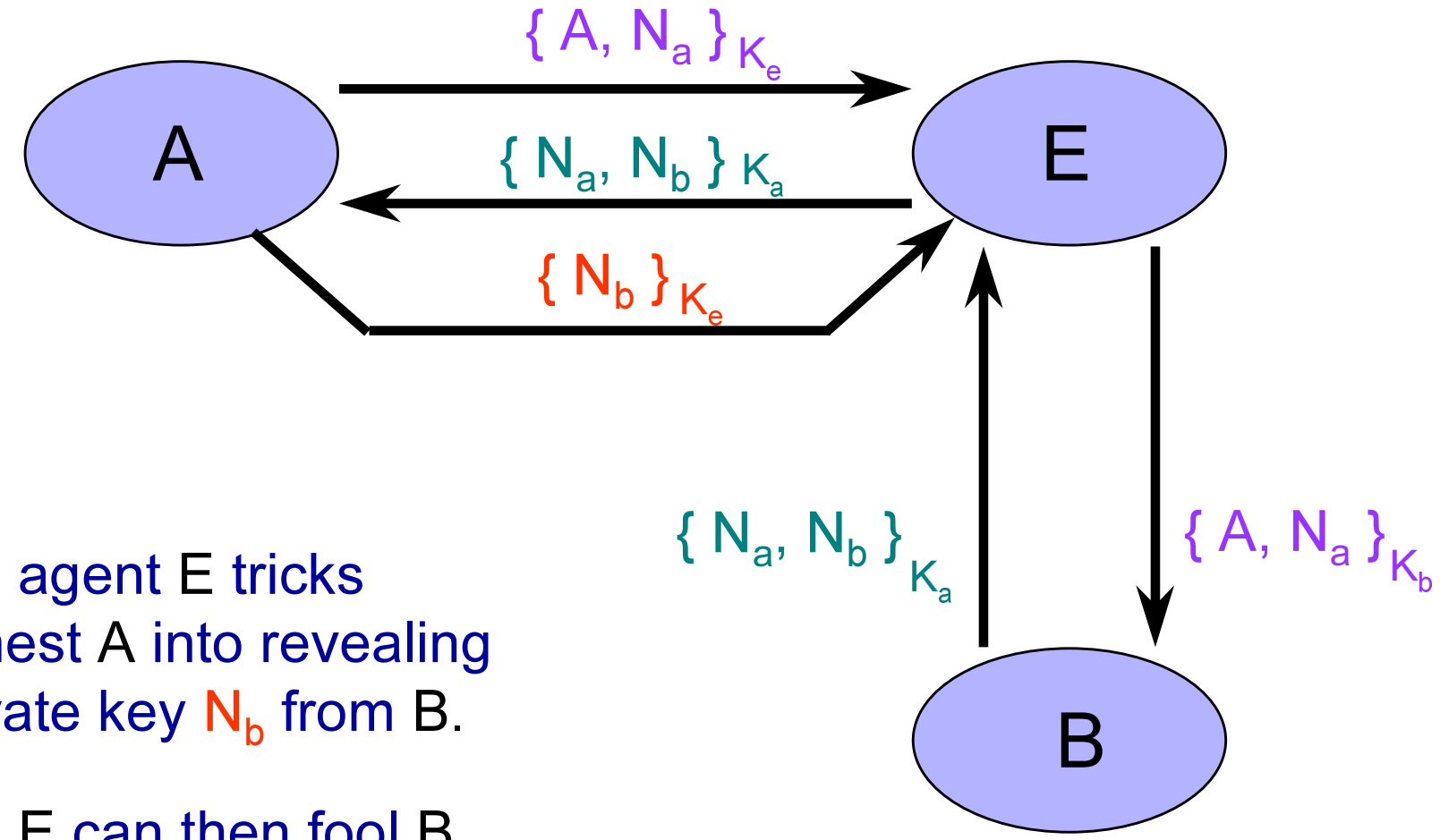
Needham-Schroeder Key Exchange



Result: A and B share two private numbers not known to any observer without K_a^{-1}, K_b^{-1}

Anomaly in Needham-Schroeder

[Lowe]



Evil agent E tricks honest A into revealing private key N_b from B.

Evil E can then fool B.

Analyzing Security Protocols

- Think long and hard
- BAN and other belief logics
- Specialized tools using proof search
- Exhaustive state-enumeration tools
  - Model checking using CSP, Murφ, ...
- New directions
  - Abadi-Gordon Spi-calculus
  - Probabilistic poly-time framework

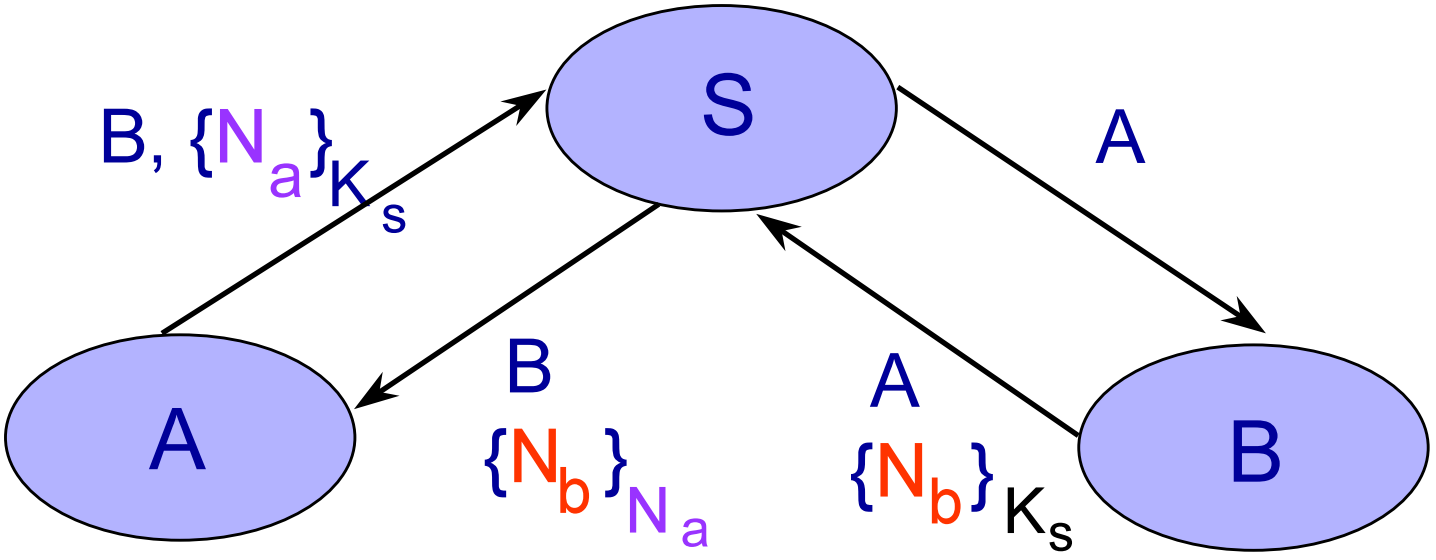
Prior state of the art

- Formal protocol analysis uses Dolev-Yao model
  - Adversary is nondeterministic process
  - Adversary can
    - Block network traffic
    - Read any message, decompose into parts
    - Decrypt if key is known to adversary
    - Insert new message from data it has observed
  - Adversary *cannot*
    - Gain partial knowledge
    - Guess part of a key
    - Perform statistical tests, …

Power and limitations

- Can find some attacks
  - Needham-Schroeder by exhaustive search
- Other attacks are outside model
  - Interaction between protocol and encryption
- Some protocols cannot be modeled
  - Probabilistic protocols
  - Steps that require specific properties of encryption
- Possible to prove erroneous protocol correct

Example: TMN Cell Phone Protocol



- Replay attack if N_b not fresh
  - Server rejects N_b and requests different number from B
- RSA Encryption: encrypt(k,msg) = msg^k mod N
  - Replay $\{N_b\}_{Ks} * \{i\}_{Ks} = N_b^{Ks} * i^{Ks} = (N_b * i)^{Ks}$ and divide later

Recent Language Approach [AG97]

- Write protocol in process calculus
- Express security using observational equivalence
  - Standard relation from programming language theory

    $P \approx Q$ iff for all contexts C[ ], same observations about C[P] and C[Q]
  - Context (environment) represents adversary
- Use proof rules for $\approx$ to prove security
  - Protocol is secure if no adversary can distinguish it from some idealized version of the protocol

Our Framework

Probabilistic Poly-time Analysis

- Adopt spi-calculus approach, add probability
- Probabilistic polynomial-time process calculus
  - Protocols use probabilistic primitives
    - Key generation, nonce, probabilistic encryption, ...
  - Adversary may be probabilistic
  - Modal type system guarantees complexity bounds
- Express protocol and specification in calculus
- Study security using observational equivalence
  - Use probabilistic form of process equivalence

Technical Challenges

- Language for prob. poly-time functions
  - Extend Hofmann language with rand
- Replace nondeterminism with probability
  - Otherwise adversary is too strong ...
- Define probabilistic equivalence
  - Related to poly-time statistical tests ...
- Develop specification by equivalence
  - Several examples carried out
- Proof systems for probabilistic equivalence
  - Goal for the future

Example protocol in process calc

- "Notation found in the literature"

  $A \rightarrow B: \quad \{ m \}_K$

  $B \rightarrow A: \quad \{ m+1 \}_K$

- Process calculus with cryptographic primitives

output on port AB

```
let k = new_key(n) in
      let m = pick_a_number(n) in  AB ⟨encrypt(k,m)⟩
  |   AB(x). BA ⟨encrypt(k, decrypt(k,x)+1)⟩
end
```

not m

This form makes assumptions and response explicit

How we specify secrecy

- Original protocol P

  $A \rightarrow B: \quad \{ m \}_K$

  $B \rightarrow A: \quad \{ m+1 \}_K$

- "Obviously" secret protocol Q (zero knowledge)

  $A \rightarrow B: \quad \{ random_number \}_K$

  $B \rightarrow A: \quad \{ random_number \}_K$

- Basic idea:

  $P \approx Q$ implies P preserves secrecy

  If not, then some context can obtain some information from the original protocol

Nondeterminism is traditional, but ...

- Nondeterminism is a useful idealization
  - Classical ∃ disguised as a computational primitive
  - Expresses extreme "good luck" or "bad luck"
    - Nondeterministic algorithm for traveling salesman
      - "Guess" a path and check that it is correct
    - Nondeterministic semantics for parallel composition
      - Treat any possible interleaving as significantly possible
      - Appropriate for "worst case" correctness
- Not an intrinsic property of system itself

Nondeterminism breaks encryption

- Alice encrypts message and sends to Bob

  $A \rightarrow B: \quad \{ msg \}_K$

- Adversary uses nondeterministic parallelism

  Process E_0 $\quad E\langle 0\rangle \mid E\langle 0\rangle \mid \ldots \mid E\langle 0\rangle$

  Process E_1 $\quad E\langle 1\rangle \mid E\langle 1\rangle \mid \ldots \mid E\langle 1\rangle$

  Process E $\quad E\langle b_1\rangle.E\langle b_2\rangle...E\langle b_n\rangle.$ decrypt($b_1b_2...b_n$, msg)

In reality, adversary has $\leq 2^{-n}$ chance to guess n-bit key

Solution: probabilistic scheduler

- Define operational semantics
  - Probabilistic steps let x = M in P $\rightarrow_r$ [v/x]P
  - Nondeterministic choice between parallel processes

- Each run requires probabilistic scheduler
  - Chooses step from "nondeterministic" alternatives
  - Scheduler runs in probabilistic polynomial time
  - Quantify over schedulers to get universal properties

Similar ideas in literature on Markov decision diagrams

Toward probabilistic equivalence

- Background: poly-time statistical tests
  - Standard notion from cryptography
  - Define *crypto. strong pseudo-random sequence*
- Main ideas
  - Pseudo-random generator family $G = \{G_n\}_{n>0}$
  - Test generator G_n in time poly(n)
    - Compare Test(G_k(random(n)) to Test(random(n^k))
    - Generator "secure" if results within 1/poly(n)

Observing Probabilistic Process

- Observations
  - Compare $|\text{Prob}[P \rightarrow \text{“yes”}] - \text{Prob}[Q \rightarrow \text{“yes”}]| < \varepsilon$
  - How small ε is small ?
    - Less than 1/2, 1/4, … ? (not equiv relation for fixed ε)
    - Vanishingly small ?
    - How fast should $\varepsilon \rightarrow 0$? As a function of what?
- Cryptographic protocols
  - Use encryption keys of a certain length
    - Protocol is family $\{ P_n \}_{n>0}$ indexed by key length
  - Increasing key length $\Rightarrow$ increasing security

Probabilistic Observational Equiv

- Processes P, Q are ε-indistinguishable

  $P \approx_\varepsilon Q$ if $\forall$ contexts C[ ]. $\forall$ observations v.

  $|Prob[C[P] \rightarrow v] - Prob[C[Q] \rightarrow v]| < \varepsilon$

- Asymptotically within f

  Process, context families $\{ P_n \}_{n>0}$ $\{ Q_n \}_{n>0}$ $\{ C_n \}_{n>0}$

  $P \approx_f Q$ if $\forall$ contexts C[ ]. $\forall$ obs v. $\exists n_0$. $\forall$ $n > n_0$.

  $| Prob[C_n[P_n] \rightarrow v] - Prob[C_n[Q_n] \rightarrow v] | < f(n)$

- Asymptotically polynomially indistinguishable

  $P \approx Q$ if $P \approx_f Q$ for every polynomial $f(n) = 1/p(n)$

Final def'n gives robust equivalence relation

Basic example

- Sequence generated from random seed

  P_n: let b = n^k-*bit sequence generated from* n *random bits*

  in $\overline{\text{PUBLIC}}\ \langle b \rangle$ end

- Truly random sequence

  Q_n: let b = *sequence of* n^k *random bits*

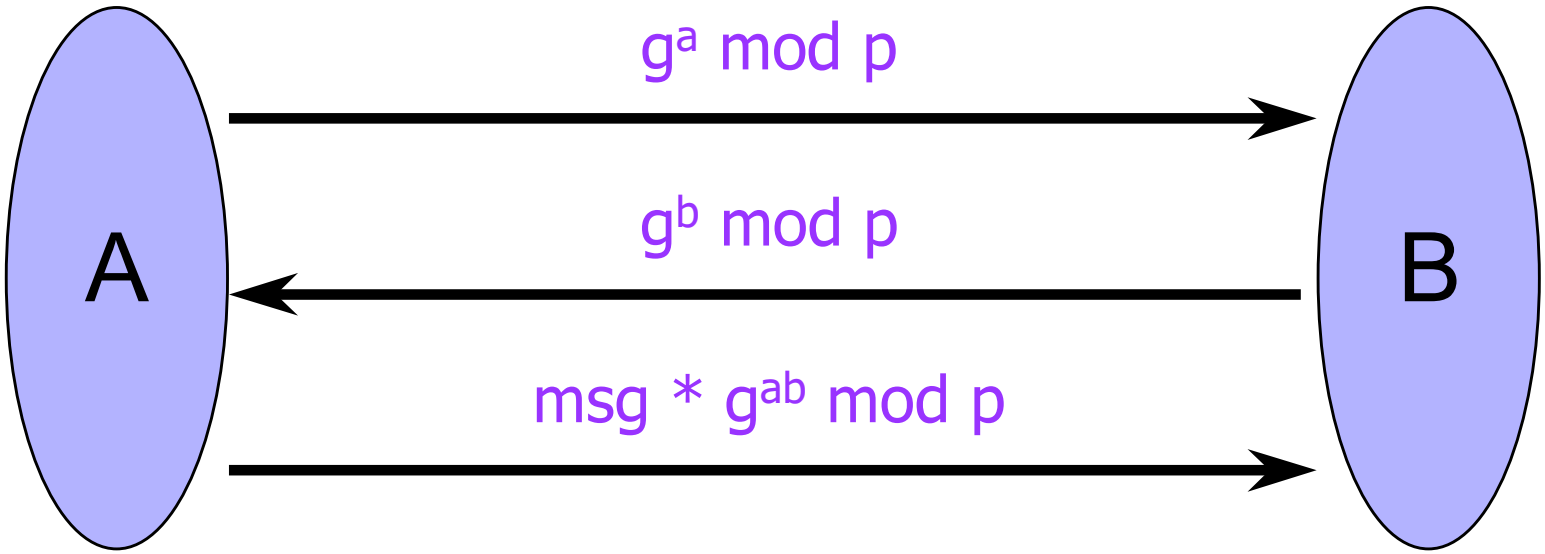
  in $\overline{\text{PUBLIC}}\ \langle b \rangle$ end

- P is crypto strong pseudo-random generator

  $P \approx Q$

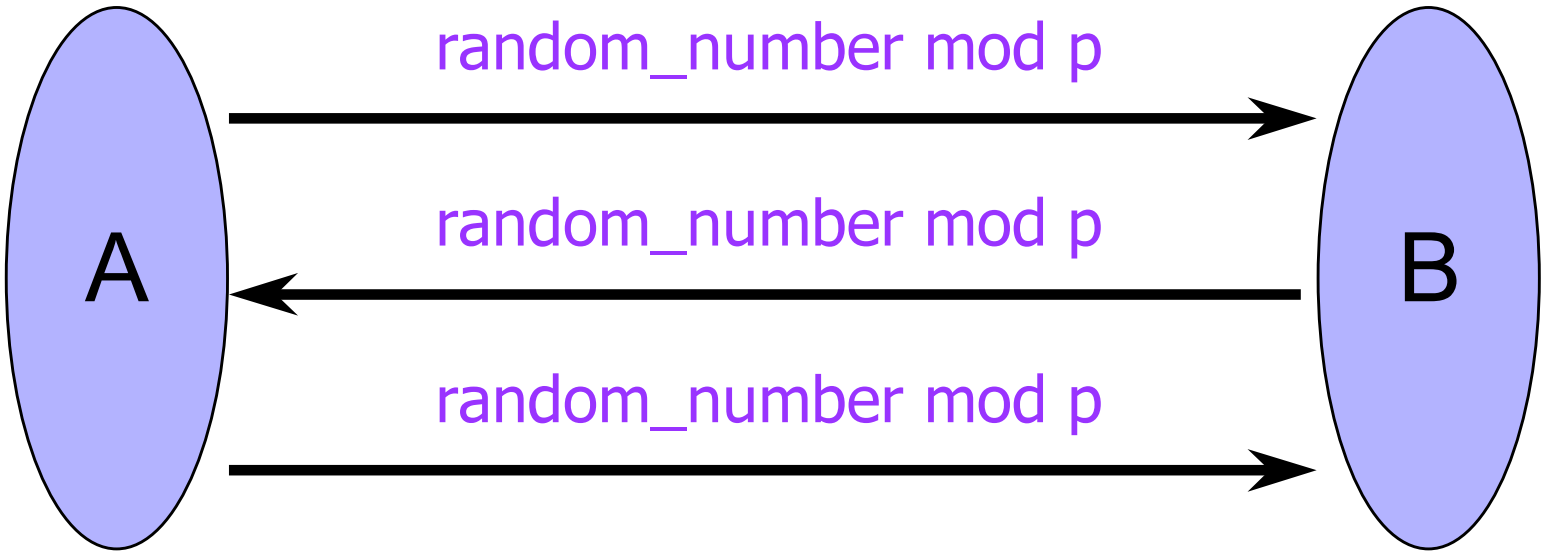
Protocol P

[Diffie, Hellman, ElGamal]



- Prime p and generator g of Z_p are public
- Passive eavesdropper has small chance at msg

Specification Q



- Network traffic should look like 3 random numbers

Analysis

- Prove P ≈ Q ?
  - Prove difficulty of computing discrete logarithm ?
- Better: reduction from a discrete log problem
  - Strategy to distinguish P from Q with prob > 1/poly
    ⇒ win Diffie-Hellman game with prob >1/poly
- Decision-Diffie-Hellman problem
  - Given two triples: $\langle x, y, z \rangle$ $\langle g^u, g^v, g^{uv} \rangle$
  - Decide which is which (u,v,x,y,z chosen randomly)

Note: this is for passive eavesdropper only

ElGamal Analysis: So what?

- Characterize security by number-theoretic game
  - Decision Diffie-Hellman appears in literature
  - Previously studied, believed hard
- Remove doubt about protocol, up to common cryptographic assumptions
  - Simplified example since this protocol can be subverted by replacing g_a by g_c

Current state of project

- Better foundations for protocol analysis ?
  - Determine crypto requirements of protocols !
- Probabilistic ptime language
  - Extended Hofmann language with rand
- Pi-calculus-like process framework
  - replaced nondeterminism with rand
  - equivalence based on ptime statistical tests
- Specifications of secrecy, authenticity
- Simple examples
- Work in progress...