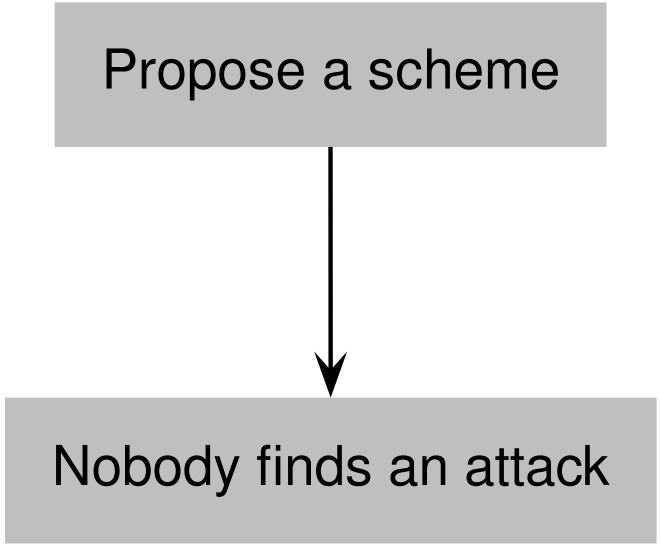
Computer-aided security proofs

Gilles Barthe

IMDEA Software Institute, Madrid, Spain

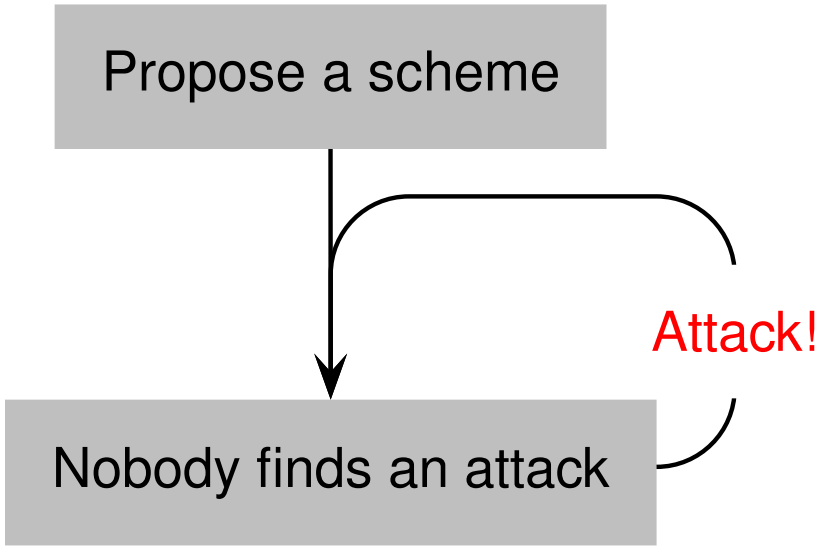
Based on joint work with: Benjamin Grégoire, Santiago Zanella Béguelin, César Kunz, Anne Pacalet, Federico Olmedo, Sylvain Heraud, Daniel Hedin, Yassine Lakhnech, Boris Köpf, Juan Manuel Crespo, Guido Genzone, Michael Backes, Matthias Berg, Malte Skoruppa, David Pointcheval, Bacelar Almeida, Manuel Barbosa, Endré Bangerter, Stephan Krenn, Martin Gagné. . .

A problem with cryptographic schemes



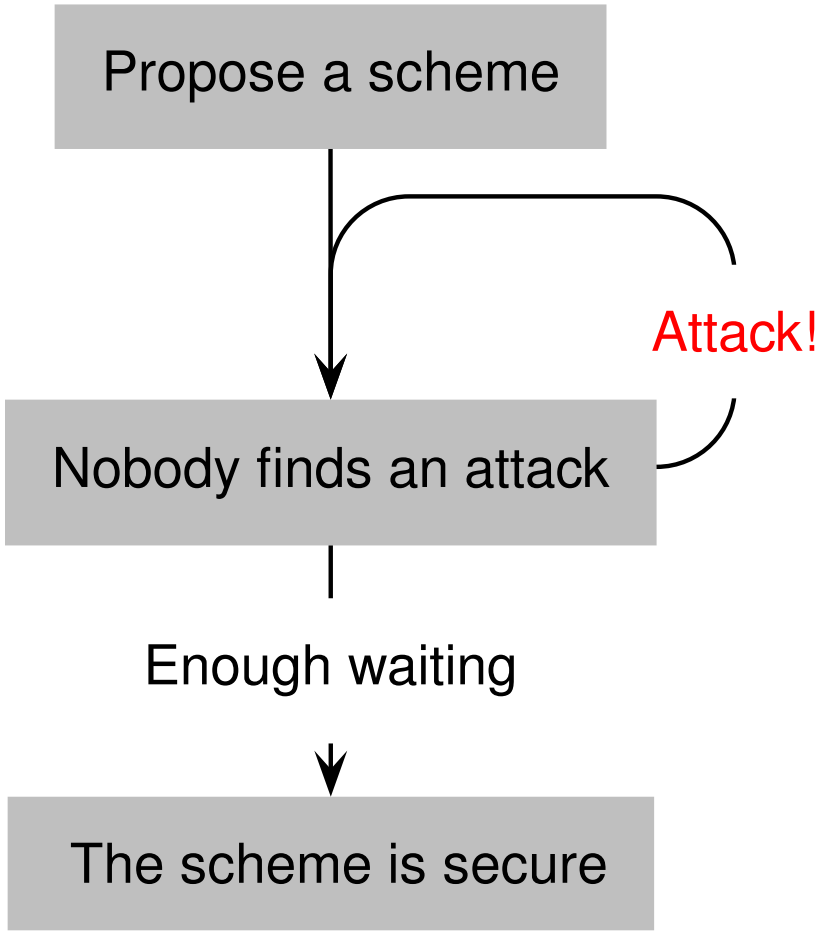
Cryptanalysis-driven security

A problem with cryptographic schemes



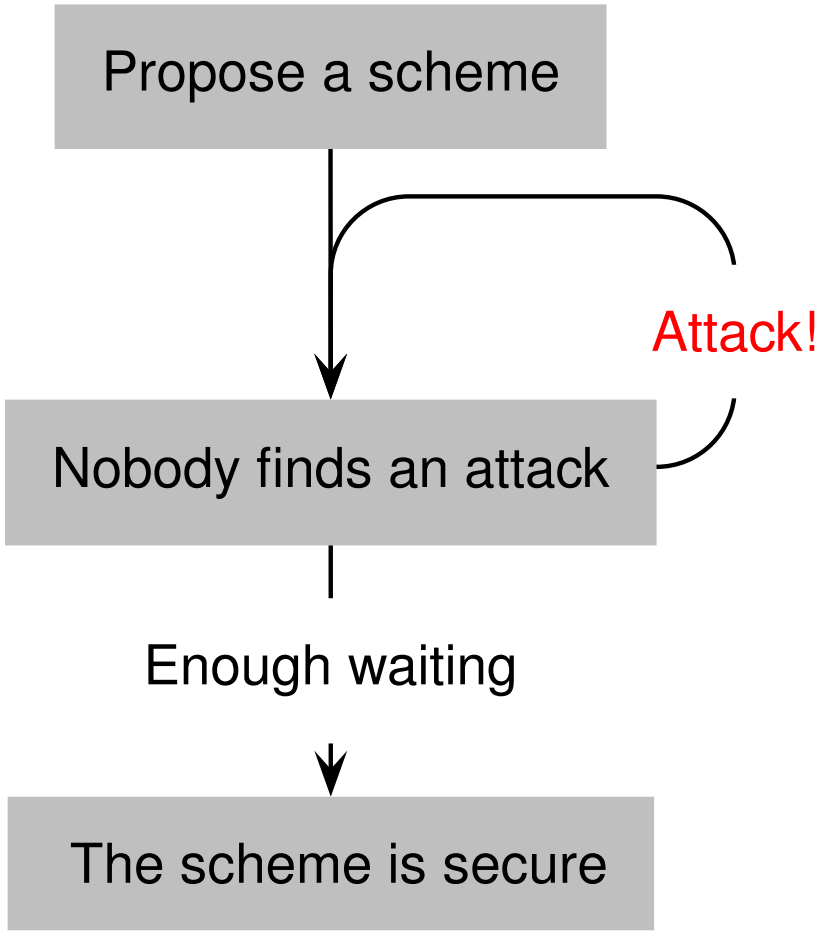
Cryptanalysis-driven security

A problem with cryptographic schemes



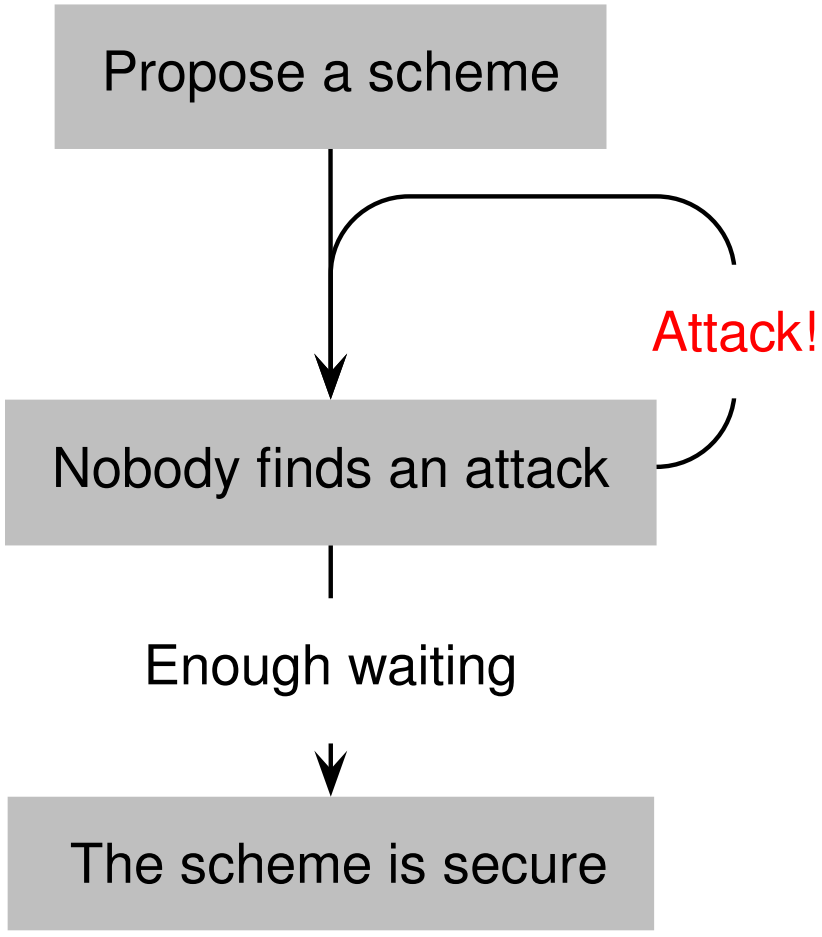
Cryptanalysis-driven security

A problem with cryptographic schemes



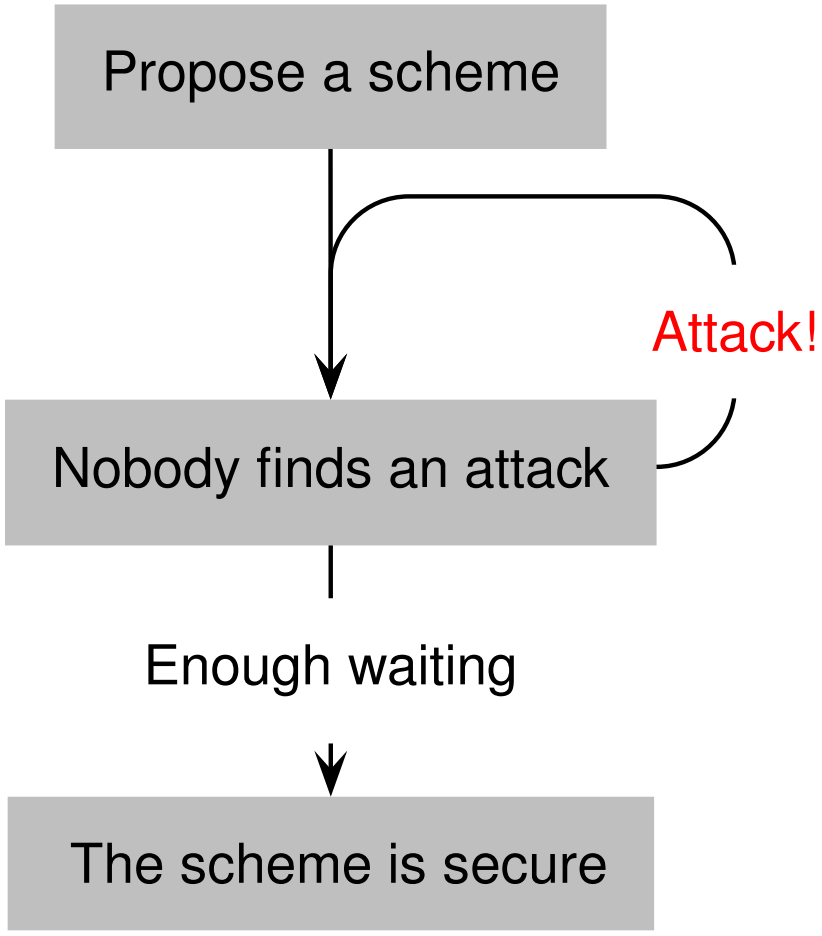
How much time is *enough*?

A problem with cryptographic schemes



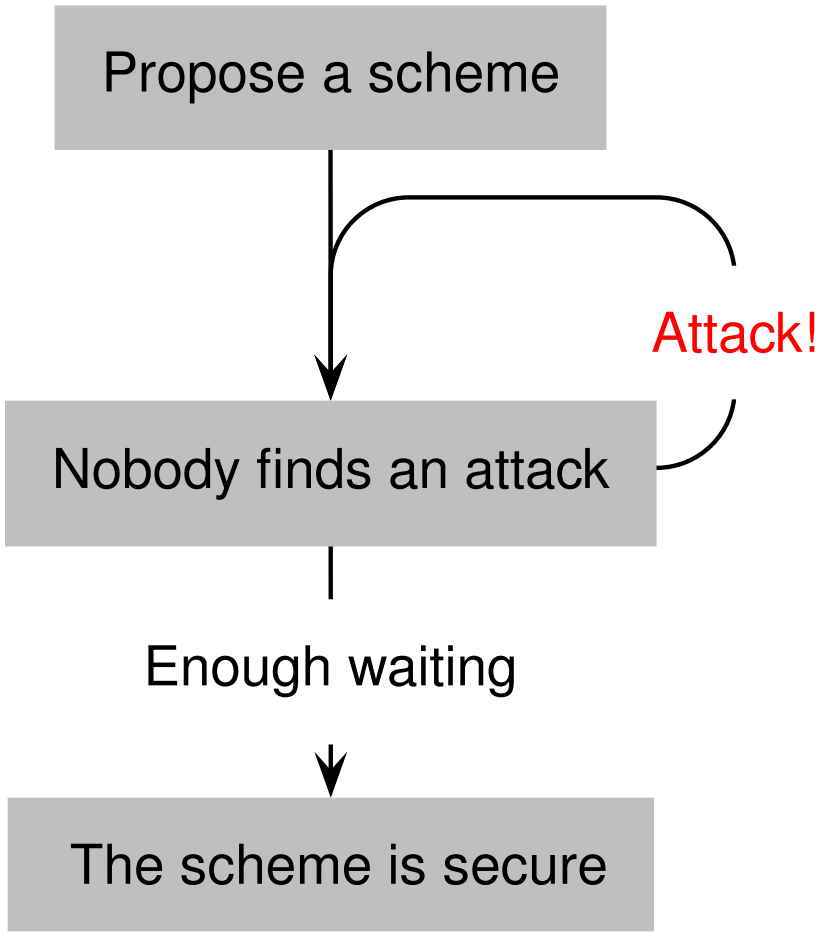
1 year? 2 years? 5 years? 10 years? 20 years?

A problem with cryptographic schemes



Can’t we do better?

A problem with cryptographic schemes

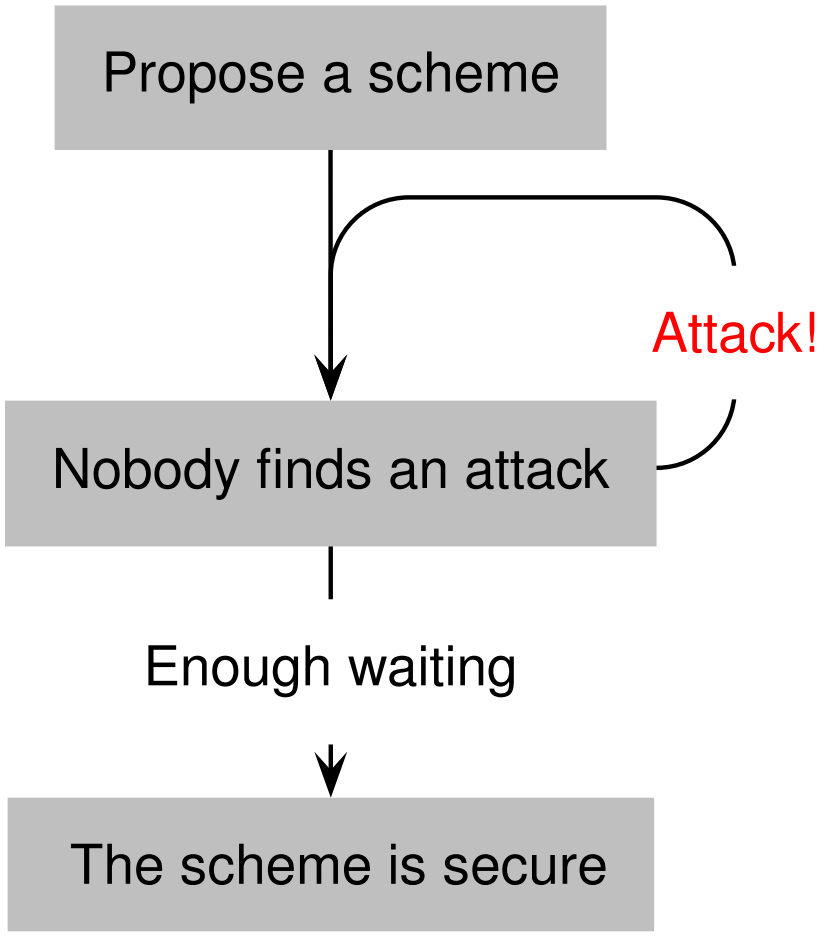


Prove a scheme secure

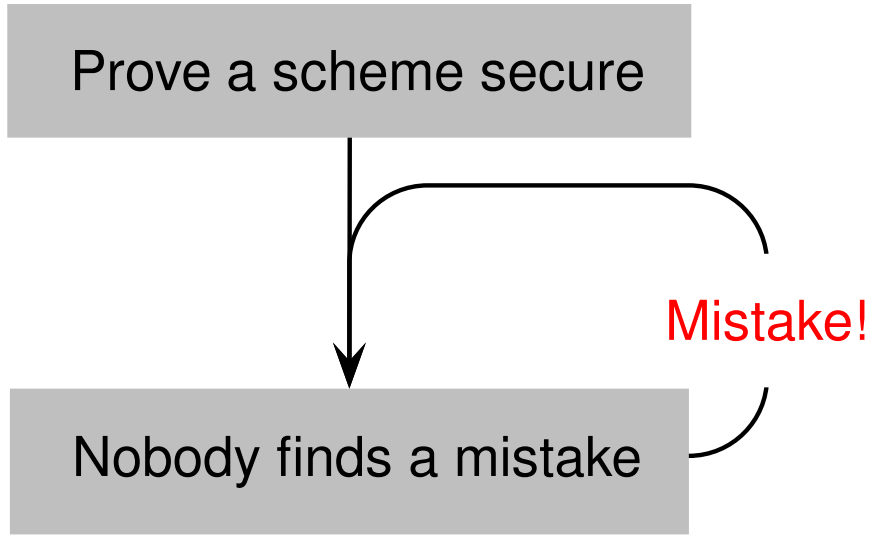
Cryptanalysis-driven security

Provable security

A problem with cryptographic proofs

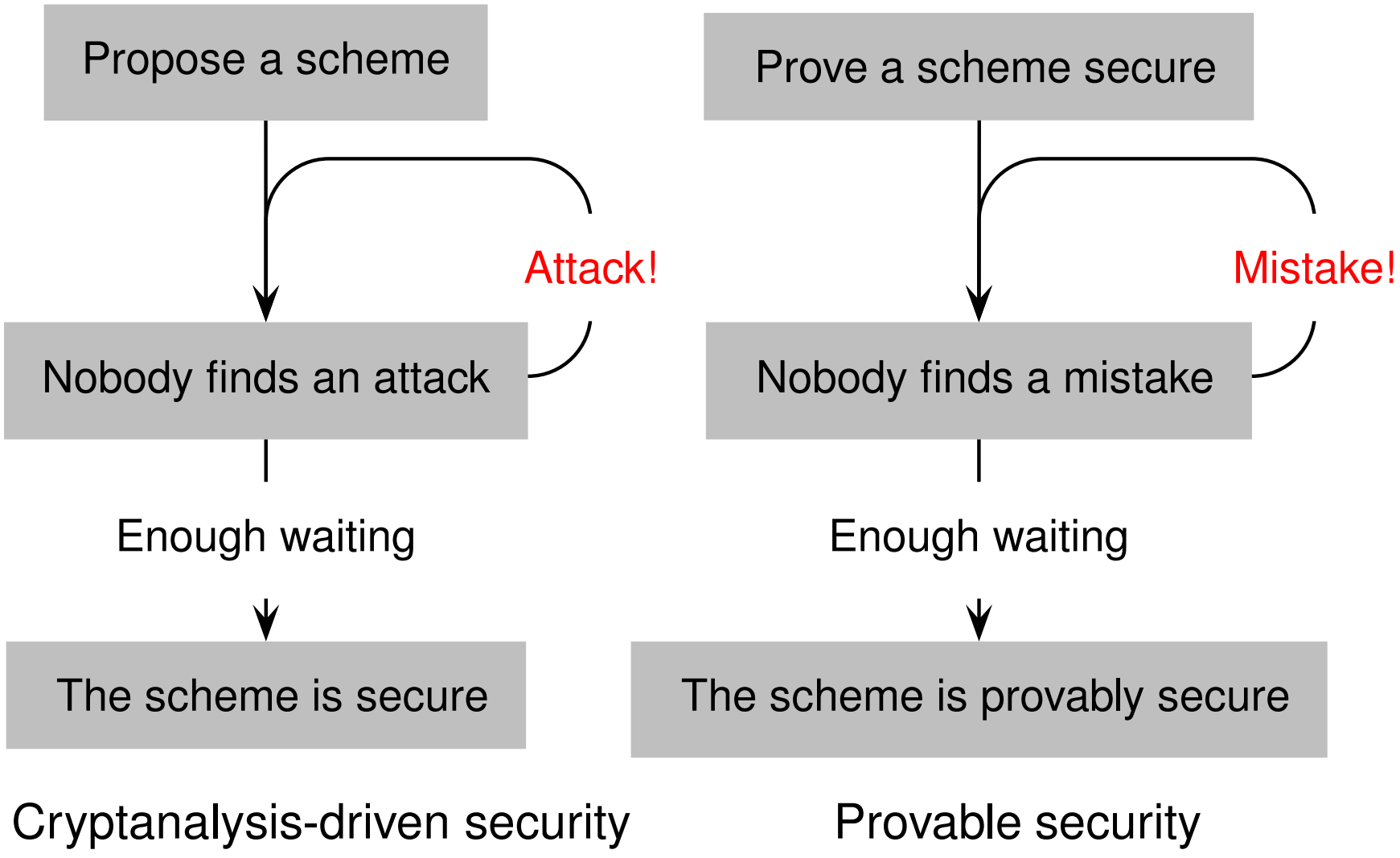


Cryptanalysis-driven security

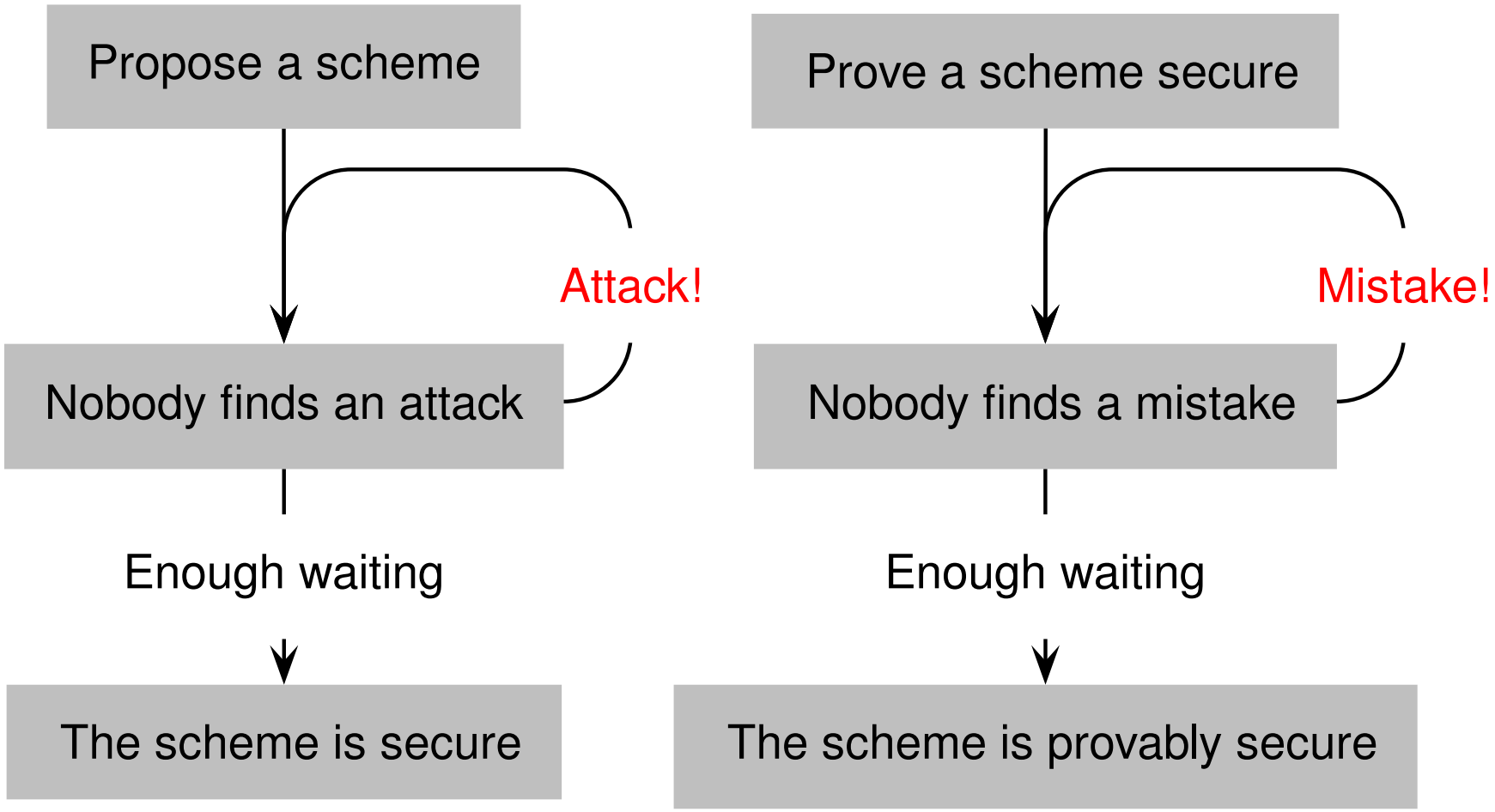


Provable security

A problem with cryptographic proofs

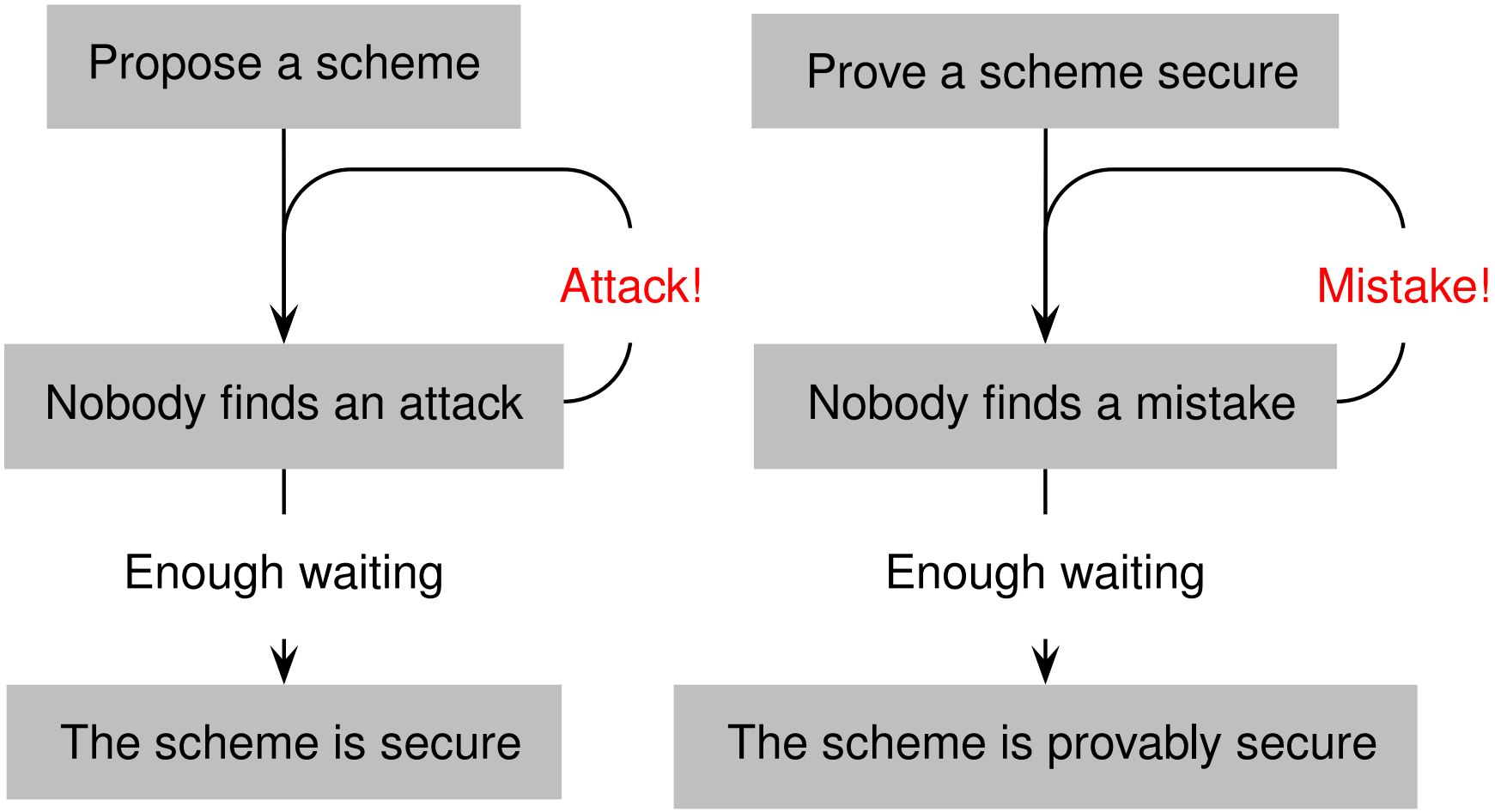


A problem with cryptographic proofs



How much time is *enough*?

A problem with cryptographic proofs



1 year? 2 years? 5 years? 10 years? 20 years?

A famous example: RSA-OAEP

Enc(m) :
$r \xleftarrow{\$} \{0,1\}^p$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
$y \leftarrow f(s \| t)$;
return y

Game IND-CCA2 :
$(sk, pk) \leftarrow \mathcal{KG}(\ )$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$\zeta \leftarrow \mathrm{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}'(pk, \zeta)$;
return $b = b'$

Advantage against IND-CCA2 security

$$\mathrm{Adv}_{\text{IND-CCA2}(\mathcal{A})} = \left| Pr_{\text{IND-CCA2}}[b = b'] - \tfrac{1}{2} \right|$$

A famous example: RSA-OAEP

$\mathrm{Enc}(m)$:
$r \xleftarrow{\$} \{0,1\}^p$;
$s \leftarrow G(r) \oplus (m \| 0^{k_1})$;
$t \leftarrow H(s) \oplus r$;
$y \leftarrow f(s \| t)$;
return y

Game IND-CCA2 :
$(sk, pk) \leftarrow \mathcal{KG}(\,)$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$\zeta \leftarrow \mathrm{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}'(pk, \zeta)$;
return $b = b'$

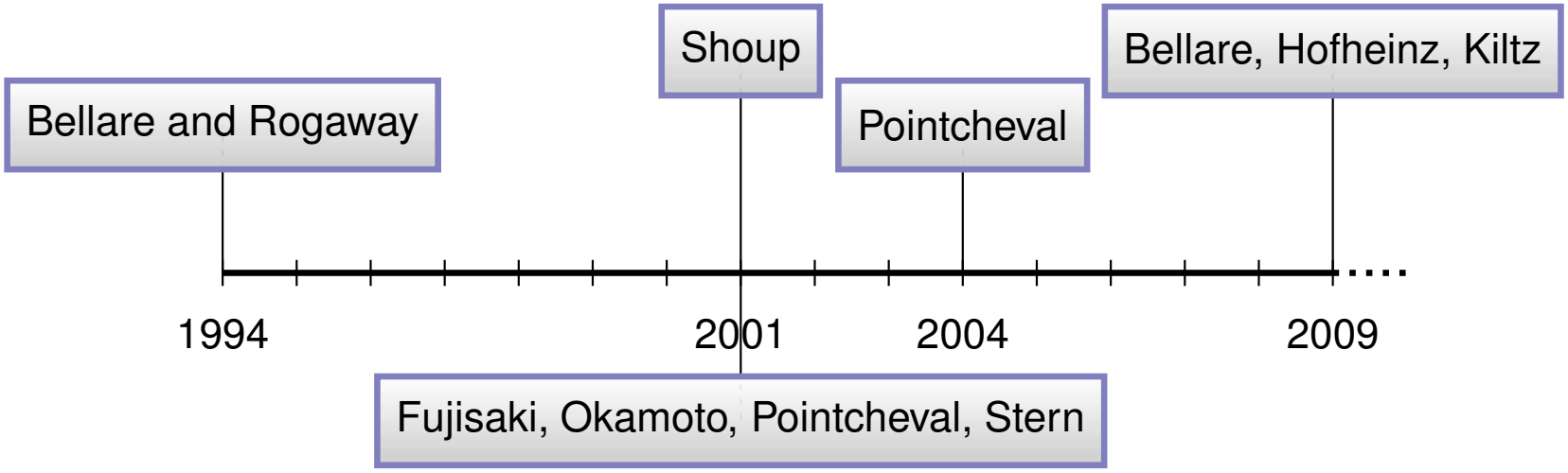
Exact IND-CCA2 security

$$\mathrm{Adv}_{\text{IND-CCA2}(\mathcal{A})} = \left| Pr_{\text{IND-CCA2}}[b = b'] - \tfrac{1}{2} \right|$$

$$\leq \mathbf{Succ}_f^{\mathrm{POW}}(\mathcal{I}) + \frac{3q_D q_G + q_D^2 + 4q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}}$$

where q_X upper bounds the number of queries to oracle X

A famous example: RSA-OAEP



1994 Purported proof of chosen-ciphertext security

2001 Proof establishes a weaker security notion, but desired security can be achieved

1. ...for a modified scheme, or
2. ...under stronger assumptions

2004 Filled gaps in Fujisaki et al. 2001 proof

2009 Security definition needs to be clarified

2010 Found gaps in 2004 proof

What's wrong with provable security?

- *In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor*. M. Bellare and P. Rogaway, 2004-2006
- *Do we have a problem with cryptographic proofs? Yes, we do [...] We generate more proofs than we carefully verify (and as a consequence some of our published proofs are incorrect)*. S. Halevi, 2005

$$\frac{\text{Large number of proofs + Proofs are increasingly complex}}{\text{Proofs are not carefully verified}}$$

The CertiCrypt/EasyCrypt project

- High-assurance cryptographic proofs
- Applying rigorously justified methods
- Machine-checked proofs, building upon off-the-shelf tools
- Accessible to cryptographers

The CertiCrypt/EasyCrypt project

- High-assurance cryptographic proofs
- Applying rigorously justified methods
- Machine-checked proofs, building upon off-the-shelf tools
- Accessible to cryptographers

Sources of inspiration

- *I advocate creating an automated tool to help [...] writing and checking [...] our proofs*
  Halevi, 2005
- *The possibility for tools [to help write and verify proofs] has always been one of our motivations, and one of the reasons why we focused on code-based games*
  Bellare and Rogaway, 2004-2006

The CertiCrypt/EasyCrypt project

- High-assurance cryptographic proofs
- Applying rigorously justified methods
- Machine-checked proofs, building upon off-the-shelf tools
- Accessible to cryptographers

Status

- Launched 2006
  - CertiCrypt: formally verified Coq libraries
  - EasyCrypt: SMT-based verification tool
- Many emblematic examples
  - encryption schemes, signatures, hash functions, key-exchange and zero-knowledge protocols
- Still working
  - Foundations: modularity, instantiation, program logics
  - Applications and tools: certifying crypto compilers...

The CertiCrypt/EasyCrypt project

- High-assurance cryptographic proofs
- Applying rigorously justified methods
- Machine-checked proofs, building upon off-the-shelf tools
- Accessible to cryptographers

Outline of talk

- Provable security and reductionist proofs
- Code-based cryptographic proofs
- Synthesis of encryption schemes

Provable security

(Goldwasser and Micali, 1984)

A mathematical, complexity-theoretic, approach to correctness

For all feasible adversary $\mathcal{A}$ against scheme **S** (wrt goal **G**), there exists a feasible adversary $\mathcal{B}$ against assumption **H** st

- the probability that $\mathcal{A}$ breaks scheme **S** is upper bounded as a function of the probability that $\mathcal{B}$ breaks assumption **H**
- the running time $t_{\mathcal{B}}$ of $\mathcal{B}$ is upper bounded as a function of the running time $t_{\mathcal{A}}$ of $\mathcal{A}$

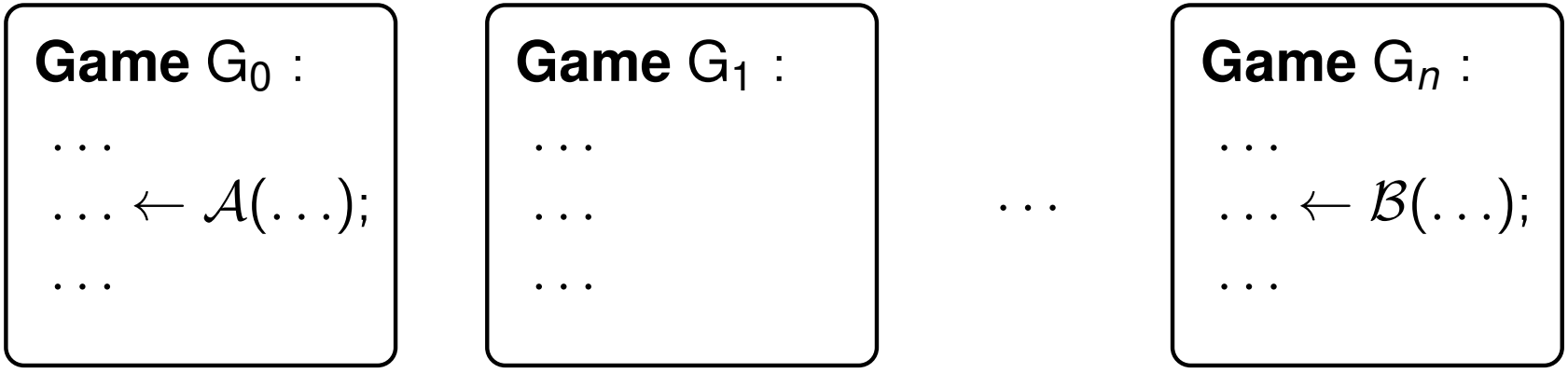
Ideally:

- winning probabilities and execution times are close
- if so, breaking **S** is equivalent to breaking **H**

Game-playing proofs

(Shoup 2004, Bellare and Rogaway 2004)

Organize proofs as sequences (or trees) of games



$$\Pr_{\mathsf{G}_0}[E_0] \leq h_1(\Pr_{\mathsf{G}_1}[E_1]) \leq \ldots \leq h_n(\Pr_{\mathsf{G}_n}[E_n])$$

Example: IND-CPA security of ElGamal

$$\begin{aligned}
\mathcal{KG}\ () &\stackrel{\text{def}}{=} x \xleftarrow{\$} \mathbb{Z}_q; \text{return } (x, g^x) \\
\text{Enc}(\alpha, m) &\stackrel{\text{def}}{=} y \xleftarrow{\$} \mathbb{Z}_q; \text{return } (g^y, \alpha^y \times m) \\
\text{Dec}(x, (\beta, \zeta)) &\stackrel{\text{def}}{=} \text{return } (\zeta \times \beta^{-x})
\end{aligned}$$

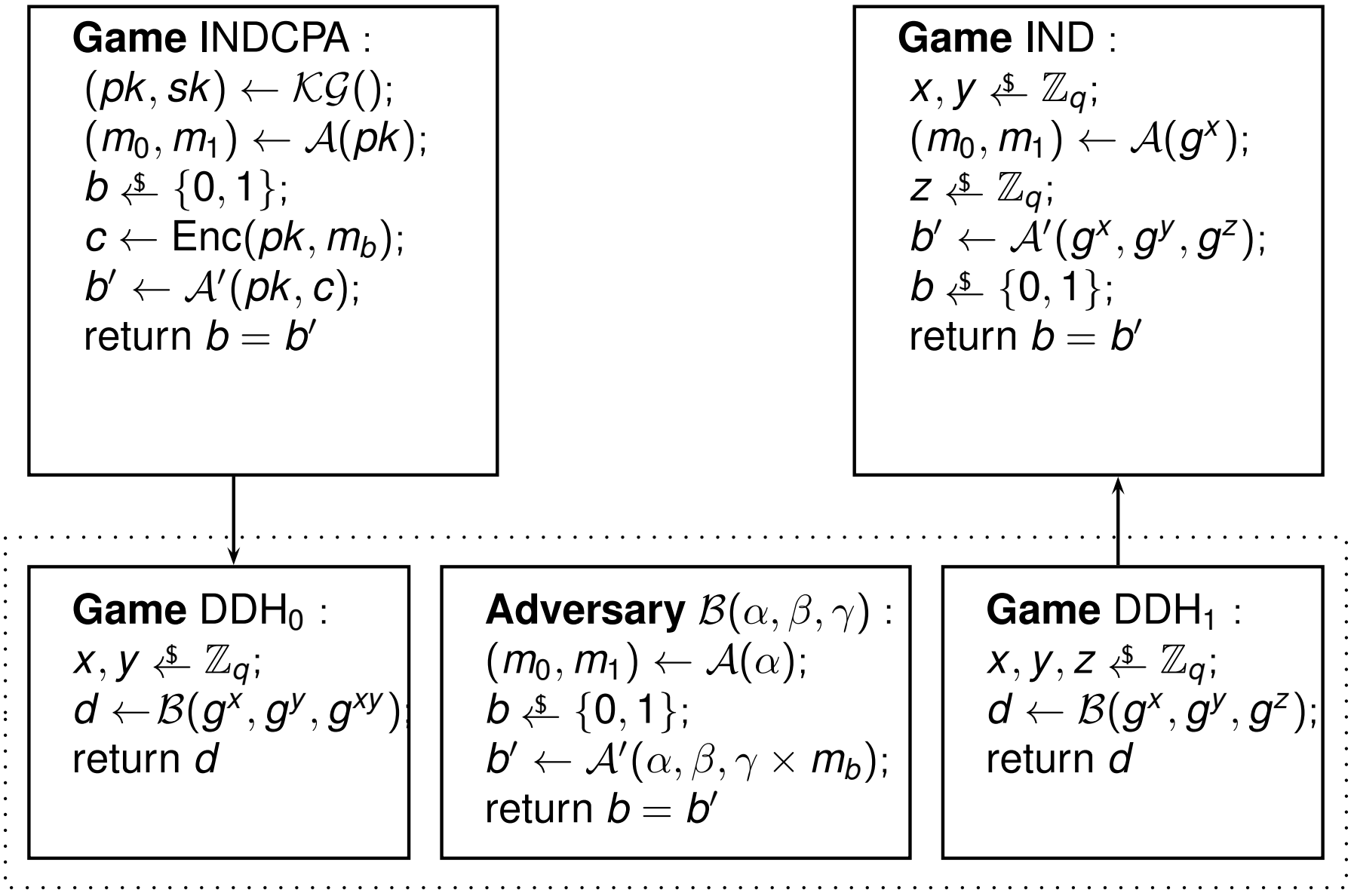
For every adversaries $(\mathcal{A}, \mathcal{A}')$, there exists an adversary $\mathcal{B}$ s.t.

$$\left| \Pr_{\text{IND-CPA}}[b - b'] - \frac{1}{2} \right| \leq \epsilon_{DDH}$$

where $\epsilon_{DDH} \stackrel{\text{def}}{=} |\Pr_{DDH_0}[b = 1] - \Pr_{DDH_1}[b = 1]|$

$$\begin{aligned}
DDH_0 &= x, y \xleftarrow{\$} \mathbb{Z}_q; \quad b \leftarrow \mathcal{B}(g^x, g^y, g^{xy}) \\
DDH_1 &= x, y, z \xleftarrow{\$} \mathbb{Z}_q; \quad b \leftarrow \mathcal{B}(g^x, g^y, g^z)
\end{aligned}$$

Game-playing proof



Reduction from INDCPA to DDH

Game INDCPA :
$(pk, sk) \leftarrow \mathcal{KG}()$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c \leftarrow \mathrm{Enc}(pk, m_b)$;
$b' \leftarrow \mathcal{A}'(pk, c)$;
$d \leftarrow b = b'$;
return d

Game DDH_0 :
$x, y \xleftarrow{\$} \mathbb{Z}_q$;
$d \leftarrow \mathcal{B}(g^x, g^y, g^{xy})$;
return d

$\mathcal{KG}\ () \stackrel{\mathrm{def}}{=}$
$x \xleftarrow{\$} \mathbb{Z}_q$; return (x, g^x)

$\mathrm{Enc}(\alpha, m) \stackrel{\mathrm{def}}{=}$
$y \xleftarrow{\$} \mathbb{Z}_q$;
return $(g^y, \alpha^y \times m)$

Adversary $\mathcal{B}(\alpha, \beta, \gamma)$:
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha)$;
$b \xleftarrow{\$} \{0,1\}$;
$b' \leftarrow \mathcal{A}'(\alpha, \beta, \gamma \times m_b)$;
return $b = b'$

Reduction from INDCPA to DDH

Game INDCPA :

$x \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$y \xleftarrow{\$} \mathbb{Z}_q$;
$\xi \leftarrow g^{xy} \times m_b$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, \xi)$;
return $b = b'$

Game DDH_0 :

$x, y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$b \xleftarrow{\$} \{0,1\}$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, g^{xy} \times m_b)$;
$d \leftarrow b = b'$;
return d

Reduction from DDH to IND

Game IND :
$x, y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$z \xleftarrow{\$} \mathbb{Z}_q$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, g^z)$;
$b \xleftarrow{\$} \{0, 1\}$;
return $b = b'$

Game DDH_1 :
$x, y, z \xleftarrow{\$} \mathbb{Z}_q$;
$d \leftarrow \mathcal{B}(g^x, g^y, g^z)$;
return d

Adversary $\mathcal{B}(\alpha, \beta, \gamma)$:
$(m_0, m_1) \leftarrow \mathcal{A}(\alpha)$;
$b \xleftarrow{\$} \{0, 1\}$;
$b' \leftarrow \mathcal{A}'(\beta, \gamma \times m_b)$;
return $b = b'$

Reduction from DDH to IND

Game IND :

$x, y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$z \xleftarrow{\$} \mathbb{Z}_q$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, g^z)$;
$b \xleftarrow{\$} \{0, 1\}$;
return $b = b'$

Game DDH_1 :

$x, y, z \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$b \xleftarrow{\$} \{0, 1\}$;
$b' \leftarrow \mathcal{A}'(g^y, g^y, g^z \times m_b)$;
$d \leftarrow b = b'$;
return d

Reduction from DDH to IND

Game IND :

$x, y \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$z \xleftarrow{\$} \mathbb{Z}_q$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, g^z)$;
$b \xleftarrow{\$} \{0, 1\}$;
return $b = b'$

Game DDH_1 :

$x, y, z \xleftarrow{\$} \mathbb{Z}_q$;
$(m_0, m_1) \leftarrow \mathcal{A}(g^x)$;
$b \xleftarrow{\$} \{0, 1\}$;
$b' \leftarrow \mathcal{A}'(g^x, g^y, g^z)$;
$d \leftarrow b = b'$;
return d

Code-based proofs

(Bellare and Rogaway, 2004)

Games as probabilistic programs in the pWHILE language

$$
\begin{array}{rcll}
\mathcal{C} & ::= & \mathcal{V} \leftarrow \mathcal{E} & \text{assignment} \\
 & | & \mathcal{V} \xleftarrow{\$} \mathcal{D} & \text{random sampling} \\
 & | & \mathcal{C};\ \mathcal{C} & \text{sequence} \\
 & | & \text{if } \mathcal{E} \text{ then } \mathcal{C} \text{ else } \mathcal{C} & \text{conditional} \\
 & | & \text{while } \mathcal{E} \text{ do } \mathcal{C} & \text{while loop} \\
 & | & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{procedure call}
\end{array}
$$

Rigorous notation for games. In our work:

- Rigorous justification of game-based proofs
- All reasoning is reduced to (instrumented) semantics

$$[\![\cdot]\!] : \mathcal{C} \to (\mathcal{M} \times \mathbb{N}) \to (\mathcal{M} \times \mathbb{N} \to [0,1]) \to [0,1]$$

where $\mathcal{D}_A = (A \to [0,1]) \to [0,1]$

Relational Hoare Logic

(Benton'04)

$\models c_1 \sim c_2 : P \Rightarrow Q$ is valid iff for all memories m_1, m_2, m_1', m_2',

$$c_1, m_1 \Downarrow m_1' \wedge c_2, m_2 \Downarrow m_2' \wedge (m_1, m_2) \models P \rightarrow (m_1', m_2') \models Q$$

Two-sided rules

$$\frac{}{\models x \leftarrow e \sim x' \leftarrow e' : Q\{x\langle 1\rangle := e\langle 1\rangle, x'\langle 2\rangle := e'\langle 2\rangle\} \Rightarrow Q}$$

$$\frac{\begin{array}{c} P \Rightarrow e\langle 1\rangle = e'\langle 2\rangle \\ \models c_1 \sim c_1' : P \wedge e\langle 1\rangle \Rightarrow Q \qquad \models c_2 \sim c_2' : P \wedge \neg e\langle 1\rangle \Rightarrow Q \end{array}}{\models \text{if } e \text{ then } c_1 \text{ else } c_2 \sim \text{if } e' \text{ then } c_1' \text{ else } c_2' : P \Rightarrow Q}$$

One-sided rules

$$\frac{}{\models x \leftarrow e \sim \text{nil} : Q\{x\langle 1\rangle := e\langle 1\rangle\} \Rightarrow Q}$$

$$\frac{\models c_1 \sim c : P \wedge e\langle 1\rangle \Rightarrow Q \qquad \models c_2 \sim c : P \wedge \neg e\langle 1\rangle \Rightarrow Q}{\models \text{if } e \text{ then } c_1 \text{ else } c_2 \sim c : P \Rightarrow Q}$$

pRHL: a Relational Hoare Logic for pWHILE

$\models c_1 \sim c_2 : P \Rightarrow Q$ is valid iff for all memories m_1 and m_2

$$(m_1, m_2) \models P \rightarrow (\llbracket c_1 \rrbracket_{m_1}, \llbracket c_2 \rrbracket_{m_2}) \models Q^\sharp$$

- pRHL is expressive:
  - validates many program transformations
  - subsumes probabilistic non-interference
  - captures many patterns of reasoning in crypto proofs
- Automation via relational weakest precondition calculus
- Probability claims follow from valid judgments, eg:
  - $\models c_1 \sim c_2 : P \Rightarrow A\langle 1 \rangle \rightarrow B\langle 2 \rangle$
  - $(m_1, m_2) \models P$

  implies $\Pr_{c_1, m_1}[A] \leq \Pr_{c_2, m_2}[B]$
- pRHL generalizes to approximate forms
  - statistical distance, differential privacy
  - f-divergences (relative entropy...)

Lifting

(Jonsson, Yi, Larsen, 2001, Deng and Du, 2011)

Let $\mu_1 \in \mathcal{D}(A)$ and $\mu_2 \in \mathcal{D}(B)$ and $Q \subseteq A \times B$.

Inductive definition

$Q^\sharp \subseteq \mathcal{D}(A) \times \mathcal{D}(B)$ is the smallest relation s.t.

- If $(a, b) \models Q$ then $(\delta_a, \delta_b) \models Q^\sharp$
- If $(\mu_i, \nu_i) \models Q^\sharp$ and $\sum_i p_i = 1$, then $(\sum_i p_i \mu_i, \sum_i p_i \nu_i) \models Q^\sharp$

Flow network definition

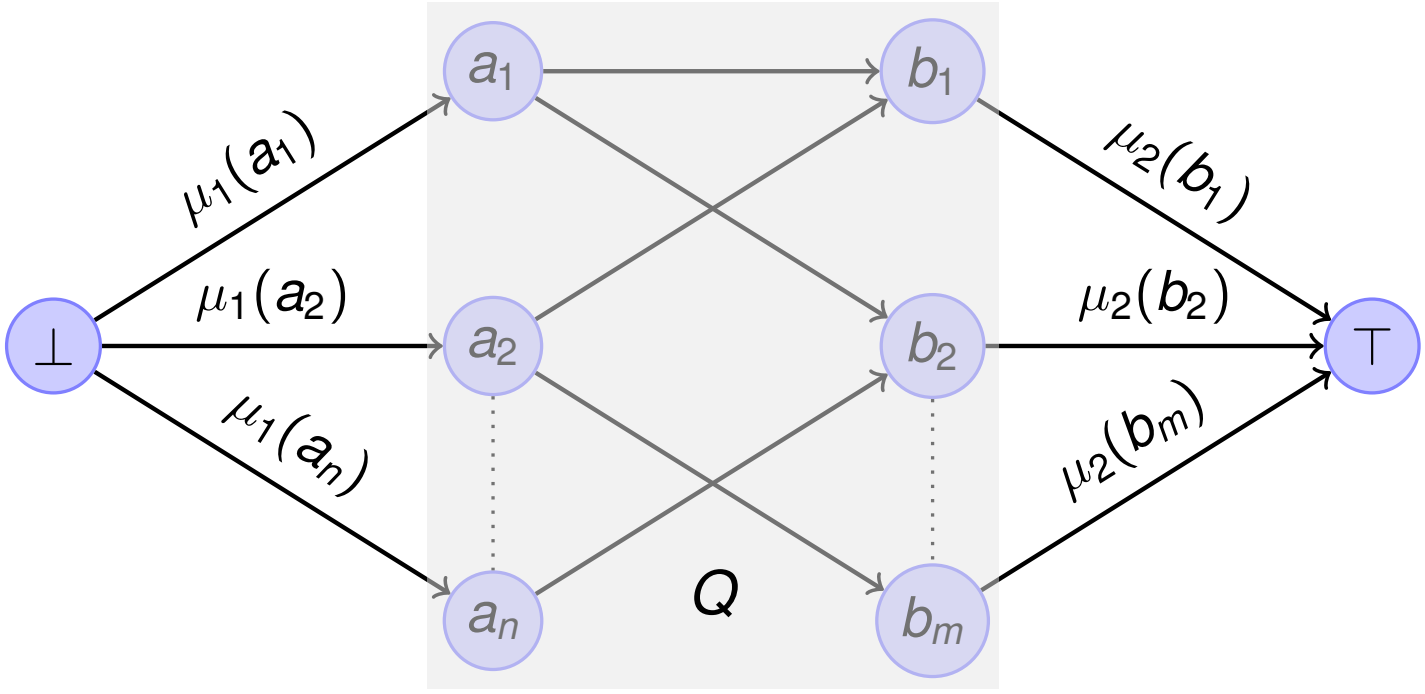
$(\mu_1, \mu_2) \models Q^\sharp$ iff there exists $\mu \in \mathcal{D}(\mathcal{A} \times B)$ s.t.

- For all $a \in A$, $\mu_1(a) = \sum_{b \in B} \mu(a, b) = \pi_1(\mu)(a)$
- For all $b \in B$, $\mu_2(b) = \sum_{a \in A} \mu(a, b) = \pi_2(\mu)(b)$
- For all $(a, b) \in A \times B$, if $\mu(a, b) > 0$ then $(a, b) \models Q$

Equivalently, the maximal flow in induced network is 1

Lifting as maximal flow

$(\mu_1, \mu_2) \vDash Q^\sharp$ iff the maximum flow in the following network is 1:

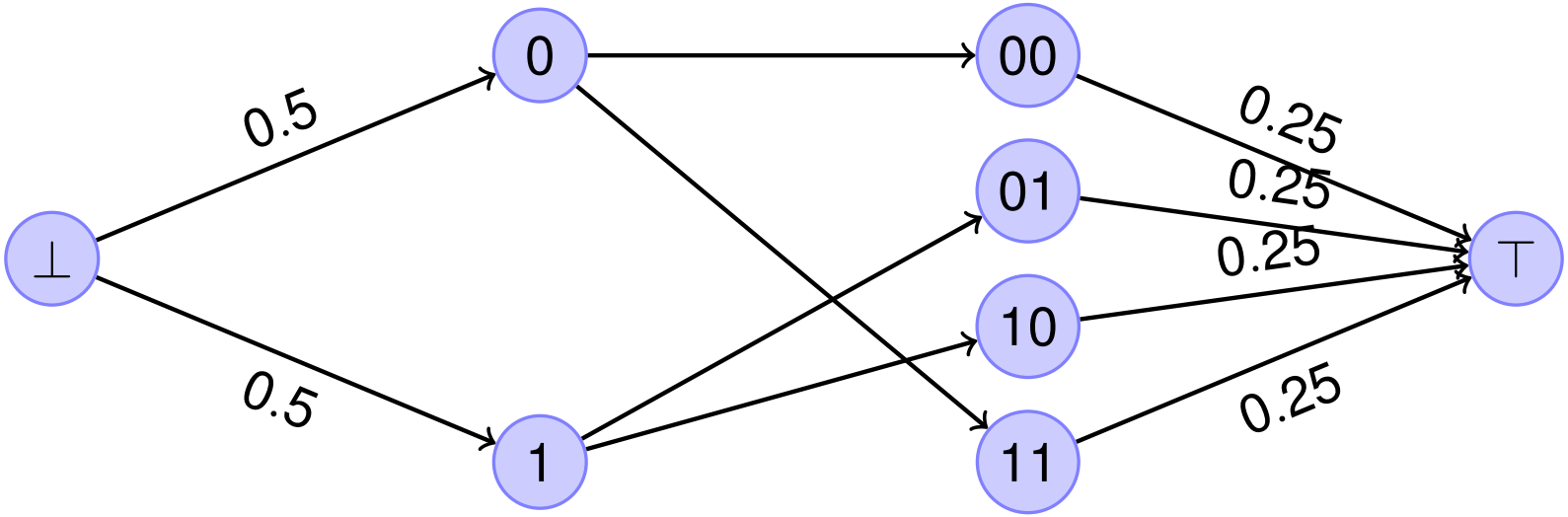


Lifting: example

$$c_1 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\} \qquad c_2 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\};\ y \xleftarrow{\$} \{0,1\}$$

- c_1 generates a distribution μ_1 over $\{0,1\}$
- c_2 generates a distribution μ_2 over $\{0,1\}^2$
- Consider $Q \stackrel{\text{def}}{=} x\langle 1\rangle = x\langle 2\rangle \oplus y\langle 2\rangle$

Q: Does $(\mu_1, \mu_2) \vDash Q$ hold?

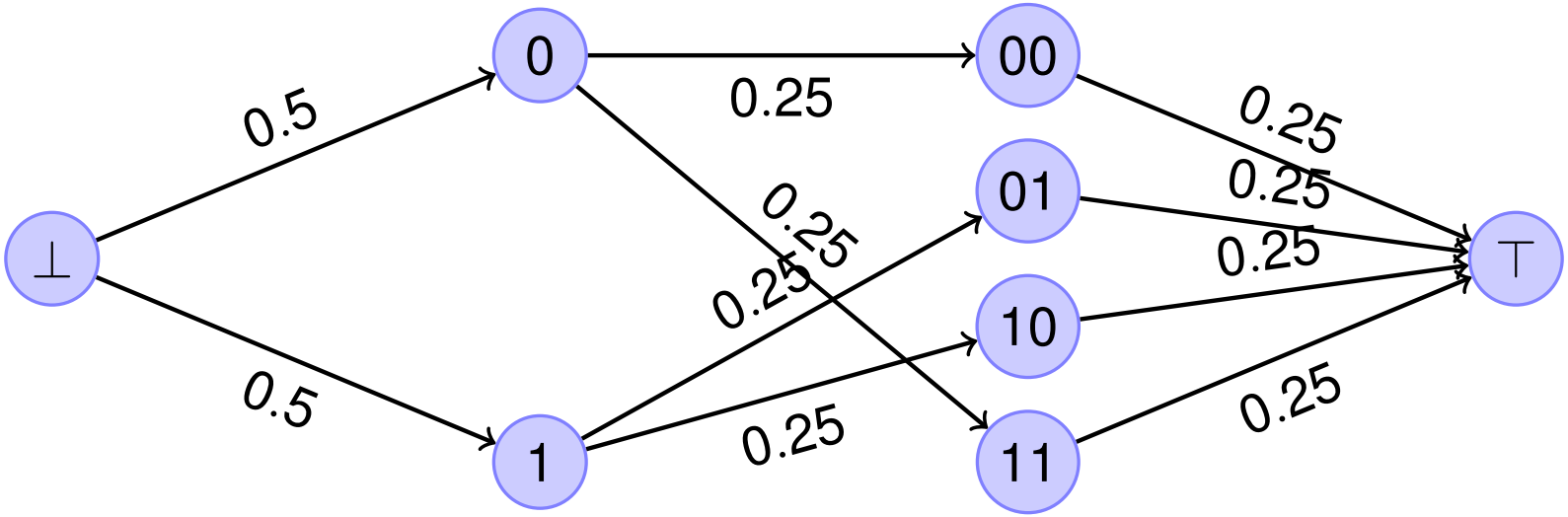


Lifting: example

$$c_1 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\} \qquad c_2 \stackrel{\text{def}}{=} x \xleftarrow{\$} \{0,1\};\ y \xleftarrow{\$} \{0,1\}$$

- c_1 generates a distribution μ_1 over $\{0,1\}$
- c_2 generates a distribution μ_2 over $\{0,1\}^2$
- Consider $Q \stackrel{\text{def}}{=} x\langle 1\rangle = x\langle 2\rangle \oplus y\langle 2\rangle$

Q: Does $(\mu_1, \mu_2) \vDash Q$ hold?
A: Yes, because we can construct a flow of value 1 in the corresponding network



pRH: rule for random assignments

$$\frac{}{\vDash x \xleftarrow{\$} A \sim x \xleftarrow{\$} A : (\forall v, Q\{x\langle 1\rangle := v, x\langle 2\rangle := f\ v\}) \Rightarrow Q}$$

where $f : A \to A$ is a 1-1 function.

Intuition: let $g_1, g_2 : A \to B$. The following are equivalent:

- $\vDash x \xleftarrow{\$} A; y \leftarrow g_1(x) \sim x \xleftarrow{\$} A; y \leftarrow g_2(x) : \top \Rightarrow y\langle 1\rangle{=}y\langle 2\rangle$
- for every $b \in B$, $\Pr_{x \xleftarrow{\$} A}[g_1(x) = b] = \Pr_{x \xleftarrow{\$} A}[g_2(x) = b]$
- for every $b \in B$, $\#g_1^{-1}(b) = \#g_2^{-1}(b)$
- there exists $f : A \overset{1-1}{\to} A$ s.t. for every $a \in A$, $g_1(a) = g_2(f(a))$

pRHL: rules for adversaries

Procedures:

- Inlining
- Functional/relational specs and modular reasoning

Adversaries:

- Abstract procedures w/ resource constraints
- Specifications must be relational
- Find Φ that is stable under sequence of oracle calls

$$\frac{\vDash x \leftarrow \mathcal{O}(\vec{y}) \sim x \leftarrow \mathcal{O}(\vec{y}) : \Phi \wedge =_{\vec{y}} \Rightarrow \Phi \wedge \mathsf{res}\langle 1\rangle = \mathsf{res}\langle 2\rangle}{\vDash x \leftarrow \mathcal{A}(\vec{e}) \sim x \leftarrow \mathcal{A}(\vec{e}) : \Phi \wedge =_{\vec{e}} \Rightarrow \Phi \wedge \mathsf{res}\langle 1\rangle = \mathsf{res}\langle 2\rangle}$$

- Possible to infer adversary specs

Other methods (1/2): failure events

Games G_1 and G_2 behave identically unless "bad" fires, e.g.

Game G_1 :

$\ldots$

bad $\leftarrow$ true; c_1

$\ldots$

Game G_2 :

$\ldots$

bad $\leftarrow$ true; c_2

$\ldots$

Fundamental Lemma

$$|\Pr_{G_1,m}[A] - \Pr_{G_2,m}[A]| \leq \Pr_{G_1,m}[\text{bad}]$$

- Formalization in pRHL
- Failure events need not be syntactic
- Hoare-like logic for computing probability of failure events

Other methods (2/2): eager/lazy sampling

Interprocedural code motion of random samplings:

- Eager sampling:
  - from oracle to main command
  - useful to compute probabilities
- Lazy sampling:
  - from main command to oracle
  - used to show that values are uniformly distributed and "independent from the adversary's view"

Logic for swapping statements

$$\models E,\ c; S \simeq E',\ S; c'$$

Examples (selection)

- Encryption: OAEP, Cramer-Shoup, Boneh-Franklin
- Signature: FDH, pFDH, BLS
- Hash functions: Merkle-Damgård, hash functions into elliptic curves
- Block ciphers: CBC, OCB
- AKE protocols
- ZK protocols: Σ-protocols, pre-images under special homomorphisms, AND/OR composition, interval proofs
- Differentially private computations: statistics, continual release, approximation algorithms, 2-party computation

Benefits and limitations

High assurance:

- Machine-checkable proofs

What does it take to trust a proof?

- The statement itself must be inspected
- You do not need to trust the proof, the sequence of games, etc.
- With CertiCrypt you do not need to trust the proof tools, only the library of probabilities and the semantics of pWHILE

- Back-end for cryptographic compilers: ZKCrypt...
- Allows avoiding elicitation

No panacea: same limitations as provable security

- Pseudocode, not implementation
- Side-channels mostly out of the model
- Hypotheses may be flawed; bounds may tell nothing

Synthesis

- Goal: generate automatically programs (or program snippets) from pre- and post-conditions or reference code
- Applications to loop-free programs and crypto protocols
- Our work: synthesis of public-key encryption schemes
  - Focus on one-way functions and random oracles
  - Many examples: BR93, REACT, OAEP, SAEP...

Expressions

$$
\begin{array}{rcll}
\mathcal{E} & ::= & m & \text{message} \\
 & | & \mathcal{R} & \text{uniform random variable} \\
 & | & \mathcal{E} \oplus \mathcal{E} & \text{exclusive or} \\
 & | & \mathcal{E} \parallel \mathcal{E} & \text{concatenation} \\
 & | & H_i(\mathcal{E}) & \text{random oracle} \\
 & | & f(\mathcal{E}) & \text{one-way}
\end{array}
$$

Method

- Generation applies symbolic filters for checking:
  - is encryption randomized? $f(m)$
  - can one extract randomness from ciphertext? $r \parallel f(m \oplus r)$
  - is there a decryption algorithm? $f(r)$
  - can one decrypt without the key? $m \parallel f(r)$
- Prove IND-CPA using specialized Hoare logic (CDELL'08)
- New check for plaintext awareness (not yet implemented)

Benchmarks

Instructions	3	4	5	6	7
Candidates	2	0	55	200	3323
Proved IND-CPA w/ CDELL'08	1	0	6	13	181

Ongoing work

- more filters
- strategies, extended CDELL'08
- IND-CCA check
- exact security
- other constructions (DDH...)

Minimal schemes

- BR: $c = f(r) \parallel (G(r) \oplus m)$ (provable with CDELL'08)
- ZAEP: $c = f(r \parallel G(r) \oplus m)$ (not provable with CDELL'08)

INDCCA Security of ZAEP

$$\left| \Pr_{\text{IND-CCA2}}[b = b'] - \frac{1}{2} \right| \leq \mathbf{Succ}_f^{\text{OW}}(\mathcal{I}) + \frac{q_{\text{Dec}}}{2^n}$$

provided there exists two efficient algorithms:

- CIE: given $f(r, s_1)$, $f(r, s_2)$ with $s_1 \neq s_2$, returns s_1, s_2 and r
- SIE: given $f(r, s)$ and r returns s

Applicable to RSA exponent 2 and 3

Conclusions

- Crypto proofs can be made rigorous using PL and verification techniques
- Crypto opens new PL and verification problems
- Synthesis of crypto primitives is possible!
- Many applications outside crypto to explore:
  - polynomial reductions
  - continuous distributions
  - approximation algorithms

Try it!

http://easycrypt.gforge.inria.fr