Reasoning about Probabilistic Computations

Applications to Cryptography and Privacy

Gilles Barthe

IMDEA Software Institute, Madrid, Spain

Based on joint work with: Benjamin Grégoire, Santiago Zanella Béguelin, Sylvain Heraud, Boris Köpf, César Kunz, Yassine Lakhnech, Federico Olmedo

Formal verification and cryptography

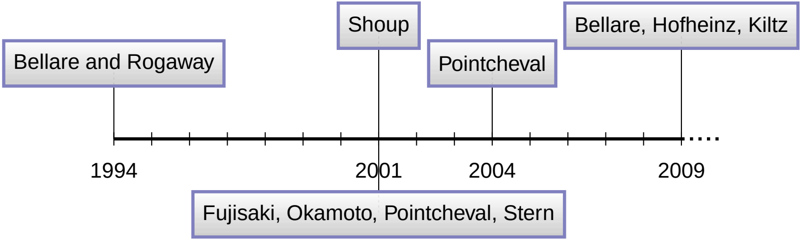
- Symbolic methods: analysis of logical flaws in protocols
- Computational soundness of symbolic models w.r.t. computational models
- Program verification: prove implementations are secure relative to adversarial model

These works assume perfect cryptography.

What's wrong with cryptographic proofs?

- *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
- *Security proofs in cryptography may be organized as sequences of games [...] this can be a useful tool in taming the complexity of security proofs that might otherwise become so messy, complicated, and subtle as to be nearly impossible to verify*
  V. Shoup, 2004

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 Filled gaps and marginally improved bound in 2004 proof

A plausible solution

- *I advocate creating an automated tool to help us [...] 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

This talk

Develop program verification methods for

- Provable Security (Goldwasser and Micali'84)
- Differential Privacy (Dwork'06)

This talk

Develop and verify program verification methods for

- Provable Security (Goldwasser and Micali'84)
- Differential Privacy (Dwork'06)

This talk

Develop and verify program verification methods for

- Provable Security (Goldwasser and Micali'84)
- Differential Privacy (Dwork'06)

CertiCrypt

Build and check exact provable security proofs in Coq

- Security goals, properties and hypotheses are explicit
- All proof steps are conducted in a unified formalism
- Proofs are independently verifiable

This talk

Develop and verify program verification methods for

- Provable Security (Goldwasser and Micali'84)
- Differential Privacy (Dwork'06)

CertiCrypt

Build and check exact provable security proofs in Coq

- Security goals, properties and hypotheses are explicit
- All proof steps are conducted in a unified formalism
- Proofs are independently verifiable

EasyCrypt

Automation with SMT solvers + generation of CertiCrypt proofs

(Deductive) program verification

- Art of proving that programs are correct
- Foundations: program logic (Hoare'69) and weakest precondition calculus (Floyd'67)
- Major advances in:
  - language coverage
    (functions, objects, concurrency, heap. . .)
  - automation
    (decision procedures, SMT solvers, invariant generation. . .)
  - proof engineering
    (intermediate languages. . .)

Hoare logic

- Judgments are of the form $\models c : P \Rightarrow Q$ (typically P and Q are f.o. formulae over program variables)
- A judgment $\models c : P \Rightarrow Q$ is valid iff for all states s and s', if such that $c, s \Downarrow s'$ and s satisfies P then s' satisfies Q.

Selected rules

$$\frac{}{\models x \leftarrow e : Q\{x := e\} \Rightarrow Q} \qquad \frac{\models c_1 : P \Rightarrow Q \quad \models c_2 : Q \Rightarrow R}{\models c_1; c_2 : P \Rightarrow R}$$

$$\frac{\models c_1 : P \wedge e = \mathrm{tt} \Rightarrow Q \quad \models c_2 : P \wedge e = \mathrm{ff} \Rightarrow Q}{\models \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q}$$

$$\frac{\models c : I \wedge e = \mathrm{tt} \Rightarrow I \quad P \Rightarrow I \quad I \wedge e = \mathrm{ff} \Rightarrow Q}{\models \text{while } e \text{ do } c : P \Rightarrow Q}$$

Verification condition generation

- Generate a set of verification conditions from annotated command and postcondition
- If all VCs are valid and $P \Rightarrow \mathrm{wp}(c, Q)$ then $\models C : P \Rightarrow Q$

Selected rules

$$
\begin{aligned}
\mathrm{wp}(x \leftarrow e, Q) &= Q\{x := e\} \\
\mathrm{wp}(c_1; c_2, R) &= \mathrm{wp}(c_1, \mathrm{wp}(c_2, R)) \\
\mathrm{wp}(\text{if } e \text{ then } c_1 \text{ else } c_2, Q) &= e{=}\mathrm{tt} \Rightarrow \mathrm{wp}(c_1, Q) \wedge e{=}\mathrm{ff} \Rightarrow \mathrm{wp}(c_2, Q) \\
\mathrm{wp}(\text{while } e \text{ do } cI, Q) &= I
\end{aligned}
$$

The while rule generates two proof obligations

$$
I \wedge e = \mathrm{tt} \Rightarrow \mathrm{wp}(c, I) \qquad I \wedge e = \mathrm{ff} \Rightarrow Q
$$

Beyond safety properties

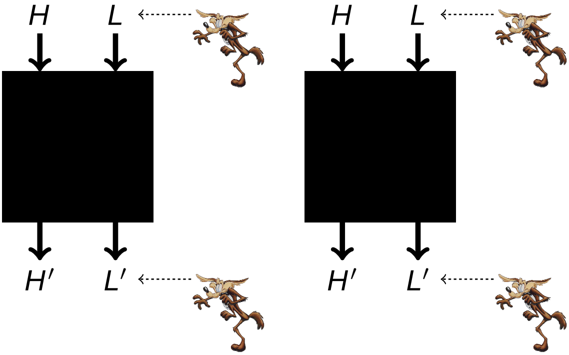
- Non-interference:

  "Low-security behavior of the program is not affected by any high-security data." Goguen & Meseguer 1982

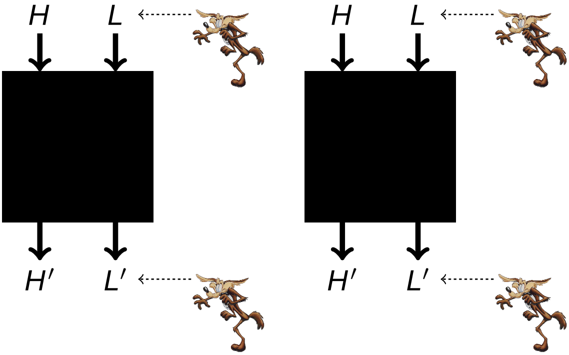
- An instance of:
  - a 2-safety property (Terauchi and Aiken'05),
  - an hyper-safety property (Clarkson and Schneider'06).

  Other 2-safety properties include continuity and determinacy

Beyond safety properties



Beyond safety properties



If $L\langle 1\rangle = L\langle 2\rangle$ then $L'\langle 1\rangle = L'\langle 2\rangle$. Or,

$$\models c \sim c : L\langle 1\rangle = L\langle 2\rangle \Rightarrow L'\langle 1\rangle = L'\langle 2\rangle$$

Relational judgments

- Judgments are of the form $\models c_1 \sim c_2 : P \Rightarrow Q$ (typically P and Q are f.o. formulae over tagged program variables of c_1 and c_2)
- A judgment $\models c_1 \sim c_2 : P \Rightarrow Q$ is valid iff for all states s_1, s_1', s_2, s_2', if $c_1, s_1 \Downarrow s_1'$ and $c_2, s_2 \Downarrow s_2'$ and (s_1, s_2) satisfies P then (s_1', s_2') satisfies Q.
- May require co-termination.

Verification methods

- Embedding into Hoare logic:
  - Self-composition (B, D'Argenio and Rezk'04)
  - Cross-products (Zaks and Pnueli'08)
- Relational Hoare Logic (Benton'04)

Embedding relational reasoning into Hoare logic

Construct $c_1 \times c_2$ s.t. (P' and Q' are renamings of P and Q)

$$\models c_1 \times c_2 : P' \Rightarrow Q' \quad \Rightarrow \quad \models c_1 \sim c_2 : P \Rightarrow Q$$

Self-composition

Set $c_1 \times c_2 = c_1 ; c_2$.

- + General (arbitrary programs) and (relatively) complete
- - Impractical

Cross-products

Define $c_1 \times c_2$ recursively, e.g.

$$\begin{aligned}\text{if } e \text{ then } c_1 \text{ else } c_2 \times \text{if } e' \text{ then } c'_1 \text{ else } c'_2 = \\ \text{if } e'' \text{ then } c_1 \times c'_1 \text{ else } c_2 \times c'_2\end{aligned}$$

- + Practical
- - Requires programs to be structurally equivalent

Relational Hoare Logic

Selected rules

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

$$\frac{\models c_1 \sim c_1' : P \Rightarrow Q \quad \models c_2 \sim c_2' : Q \Rightarrow R}{\models c_1; c_2 \sim c_1'; c_2' : P \Rightarrow R}$$

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

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

Provable security

A mathematical approach to correctness

Theorem

IF the security assumptions hold
THEN the scheme is secure against adversary $\mathcal{A}$

- Security assumptions are explicit and (ideally) standard
- Goal is clearly stated and (ideally) standard
- Adversarial model is well-defined and (usually) standard
- Proof is rigorous

Typical exact security statement

FOR ALL adversary $\mathcal{A}$ that can break the scheme
THERE EXISTS an adversary $\mathcal{B}$ that can break some security assumption with *little extra effort*

$$\Pr[\mathcal{A} \text{ breaks scheme in time } t] \\ \leq \Pr[\mathcal{B} \text{ breaks assumption in time } t + \Delta] + \epsilon$$

where Δ and ϵ depend on the number of oracles queries by $\mathcal{A}$

Proofs are constructive: $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine

The game-playing methodology

Game G_0^η :

$\ldots$

$\ldots \leftarrow \mathcal{A}(\ldots);$

$\ldots$

$\Pr_{\mathsf{G}_0^\eta}[A_0]$

The game-playing methodology

Game G_0^η :
$\ldots$
$\ldots \leftarrow \mathcal{A}(\ldots);$
$\ldots$

$\Pr_{\mathsf{G}_0^\eta}[A_0]$

Game G_n^η :
$\ldots$
$\ldots \leftarrow \mathcal{B}(\ldots);$
$\ldots$

$\Pr_{\mathsf{G}_n^\eta}[A_n]$

The game-playing methodology

Game G_0^η :	**Game** G_1^η :		**Game** G_n^η :
$\ldots$	$\ldots$		$\ldots$
$\ldots \leftarrow \mathcal{A}(\ldots);$	$\ldots$	$\ldots$	$\ldots \leftarrow \mathcal{B}(\ldots);$
$\ldots$	$\ldots$		$\ldots$

$$\Pr_{G_0^\eta}[A_0] \quad \leq \quad h_1(\Pr_{G_1^\eta}[A_1]) \quad \leq \ \ldots \ \leq \quad h_n(\Pr_{G_n^\eta}[A_n])$$

The game-playing methodology

Game G_0^η :	**Game** G_1^η :		**Game** G_n^η :
$\dots$	$\dots$		$\dots$
$\dots \leftarrow \mathcal{A}(\dots);$	$\dots$	$\dots$	$\dots \leftarrow \mathcal{B}(\dots);$
$\dots$	$\dots$		$\dots$

$$\Pr_{G_0^\eta}[A_0] \quad \leq \quad h_1(\Pr_{G_1^\eta}[A_1]) \quad \leq \dots \leq \quad h_n(\Pr_{G_n^\eta}[A_n])$$

Code-based approach

- Game = Probabilistic program
- Games have a formal semantics
- Correctness of transitions can be expressed formally

pWHILE: a probabilistic programming language

$$
\begin{array}{lll}
\mathcal{C} ::= & \text{skip} & \text{nop} \\
\mid & \text{assert } \mathcal{E} & \text{assertion} \\
\mid & \mathcal{C};\ \mathcal{C} & \text{sequence} \\
\mid & \mathcal{V} \leftarrow \mathcal{E} & \text{assignment} \\
\mid & \mathcal{V} \xleftarrow{\$} \mathcal{D} & \text{random sampling} \\
\mid & \text{if } \mathcal{E} \text{ then } \mathcal{C} \text{ else } \mathcal{C} & \text{conditional} \\
\mid & \text{while } \mathcal{E} \text{ do } \mathcal{C} & \text{while loop} \\
\mid & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{procedure call} \\
\mid & \mathcal{A} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{adversary}
\end{array}
$$

Semantics

$$[\![\cdot]\!] : \mathcal{C} \to \mathcal{M} \to \mathcal{D}_{\mathcal{M}} \text{ where } \mathcal{D}_A = (A \to [0,1]) \to [0,1]$$

Cost:

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

pRHL: a Relational Hoare Logic for pWHILE

$$\models c_1 \sim c_2 : P \Rightarrow Q \quad \text{iff} \quad \forall m_1\, m_2,\ P\ m_1\ m_2 \rightarrow \mathrm{lift}\ Q\ [\![c_1]\!]_{m_1}\ [\![c_2]\!]_{m_2}$$

(where P and Q are relations on memories)

Lifting of a relation

$$\begin{array}{l} \mathrm{lift}\ R\ (d_1 : \mathcal{D}_A)\ (d_2 : \mathcal{D}_B) := \\ \exists (d : \mathcal{D}_{A*B}),\ \pi_1(d) = d_1 \wedge \pi_2(d) = d_2\ \wedge \mathrm{range}\ R\ d \end{array}$$

- Can be interpreted as a max-flow problem
- If R is an e.r., then $\mathrm{lift}\ R\ d_1\ d_2$ iff $d_1[c] = d_2[c]$ for all $[c]$
- (Probabilistic) non-interference still expressed as

$$\models c_1 \sim c_2 : L\langle 1\rangle = L\langle 2\rangle \Rightarrow L'\langle 1\rangle = L'\langle 2\rangle$$

From pRHL to probabilities

Assume

$$\models c_1 \sim c_2 : P \Rightarrow Q$$

For all memories m_1 and m_2 such that

$$P\ m_1\ m_2$$

and events A and B such that

$$Q \Rightarrow A\langle 1\rangle \Leftrightarrow B\langle 2\rangle$$

we have

$$[\![c_1]\!]_{m_1}\ 1_A = [\![c_2]\!]_{m_2}\ 1_B$$

Proof rules

Two-sided rules, one-sided rules, program transformations

Selected rules

$$\frac{\models c_1 \sim c_2 : P \Rightarrow Q \quad P' \Rightarrow P \quad Q \Rightarrow Q'}{\models c_1 \sim c_2 : P' \Rightarrow Q'}$$

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

$$\frac{\models c_1 \sim c_1' : P \Rightarrow Q \quad \models c_2 \sim c_2' : Q \Rightarrow R}{\models c_1; c_2 \sim c_1'; c_2' : P \Rightarrow R}$$

$$\frac{\begin{array}{l} \models c_1 \sim c_1' : P \wedge e\langle 1\rangle = \mathrm{tt} \Rightarrow Q \\ \models c_2 \sim c_2' : P \wedge e\langle 1\rangle = \mathrm{ff} \Rightarrow Q \\ P \quad \Rightarrow \quad e\langle 1\rangle \quad \Leftrightarrow \quad e'\langle 2\rangle \end{array}}{\models \text{if } e \text{ then } c_1 \text{ else } c_2 \sim \text{if } e' \text{ then } c_1' \text{ else } c_2' : P \Rightarrow Q}$$

Random assignments

Let A be a finite set and let $f, g : A \to B$. Define

- $d = x \xleftarrow{\$} A; y \leftarrow f\ x$
- $d' = x \xleftarrow{\$} A; y \leftarrow g\ x$

Then $d = d'$ iff there exists $h : A \overset{1-1}{\to} A$ such that $g = f \circ h$

pRHL rule for random assignments

$$\frac{f \text{ is 1-1 and } Q' \overset{\text{def}}{=} \forall v, Q\{x\langle 1\rangle := f\ v, x\langle 2\rangle := v\}}{\models x \xleftarrow{\$} A \sim x \xleftarrow{\$} A : Q' \Rightarrow Q}$$

Optimistic sampling:

$$\models x \xleftarrow{\$} \{0,1\}^k; y \leftarrow x \oplus z \sim y \xleftarrow{\$} \{0,1\}^k; x \leftarrow y \oplus z$$
$$z\langle 1\rangle = z\langle 2\rangle \Rightarrow (x\langle 1\rangle = x\langle 2\rangle \wedge y\langle 1\rangle = y\langle 2\rangle \wedge z\langle 1\rangle = z\langle 2\rangle)$$

Adversaries

- Can read part of the memory
- Can perform arbitrary (PPT) computations
- Can call oracles (bounded number of calls for each oracle, but order and parameters of calls are arbitrary)
- Communicate with each other via shared variables

Must establish an invariant for each adversary call

Failure events

Fundamental Lemma

Assume c_1 and c_2 are absolutely terminating and behave identically unless a failure event "bad" fires. For every event A

$$|[\![c_1]\!]_m\ 1_A - [\![c_2]\!]_m\ 1_A| \leq [\![c_1]\!]_m\ 1_{\mathrm{bad}}$$

Assume $\models c_1 \sim c_2 : P \Rightarrow Q$ and c_1, c_2 terminate absolutely. If

$$Q \Rightarrow [(A \land \neg F)\langle 1\rangle \Leftrightarrow (B \land \neg G)\langle 2\rangle] \land [F\langle 1\rangle \Leftrightarrow G\langle 2\rangle]$$

then for all memories m_1 and m_2,

$$P\ m_1\ m_2 \quad \Longrightarrow \quad |[\![c_1]\!]_{m_1}\ 1_A - [\![c_2]\!]_{m_2}\ 1_B| \leq [\![c_1]\!]_{m_1}\ 1_F$$

Automation in EasyCrypt

VC generation

- Random assignment:
  - make programs in static single random assignments
  - perform random samplings eagerly (assuming termination)
  - give bijection (most of time identity works)
- Oracles: use self-composition
- Main game: use one-sided rules except for:
  - oracle calls: use inlining
  - adversary calls: infer invariants, use two-sided rules

Proofs

- SMT solvers and theorem provers
- Coq tactics for reasoning about rings and fields
- Tailored program to compute probabilities

Switching lemma

Game G_{RP} :
$L \leftarrow [\,];\ b \leftarrow \mathcal{A}()$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathrm{dom}(L)$ then
  $y \xleftarrow{\$} T \setminus \mathrm{ran}(L)$;
  $L \leftarrow (x, y) :: L$
return $L(x)$

Game G_{RF} :
$L \leftarrow [\,];\ b \leftarrow \mathcal{A}()$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathrm{dom}(L)$ then
  $y \xleftarrow{\$} T$;
  $L \leftarrow (x, y) :: L$
return $L(x)$

Suppose that $\mathcal{A}$ makes at most q queries to its oracle. Then

$$\left|\Pr_{G_{\mathrm{RP}}}[b] - \Pr_{G_{\mathrm{RF}}}[b]\right| \leq \frac{q(q-1)}{2\,(\#T)}$$

- First introduced by Impagliazzo and Rudich in 1989
- Proof fixed by Bellare and Rogaway (2006) and Shoup (2004)

Hashed ElGamal

$$
\begin{array}{lcl}
(x,\alpha) \leftarrow \mathcal{KG}(\,) & \stackrel{\mathrm{def}}{=} & x \stackrel{\$}{\leftarrow} \mathbb{Z}_q;\ \text{return } (x, g^x) \\
(\beta, v) \leftarrow \mathsf{Enc}(\alpha, m) & \stackrel{\mathrm{def}}{=} & y \stackrel{\$}{\leftarrow} \mathbb{Z}_q;\ h \leftarrow H(\alpha^y);\ \text{return } (g^y, h \oplus m) \\
m \leftarrow \mathsf{Dec}(x, \beta, v) & \stackrel{\mathrm{def}}{=} & h \leftarrow H(\beta^x);\ \text{return } h \oplus v
\end{array}
$$

where H is a random oracle:

$$
H(R) \stackrel{\mathrm{def}}{=} \begin{array}{l}
\text{if } R \notin L \text{ then } r \stackrel{\$}{\leftarrow} \{0,1\}^k;\ L \leftarrow (R, r) :: L \\
\text{else } r \leftarrow L[R] \\
\text{return } r
\end{array}
$$

Functional Correctness

$$
\mathsf{Dec}(x, g^y, H(g^{xy}) \oplus m) = H((g^y)^x) \oplus H(g^{xy}) \oplus m = m
$$

Semantic security: code-based definition

IND-CPA game

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

For all well-formed adversary $(\mathcal{A}, \mathcal{A}')$,

$$|\mathrm{Pr}_{\mathrm{IND-CPA}}[b = b'] - \frac{1}{2}| \leq \epsilon$$

where ϵ is negligible

Semantic Security of Hashed ElGamal

Hashed ElGamal is IND-CPA secure under the List CDH assumption.

List CDH assumption

$$\boxed{\begin{array}{l}\textbf{Game}\ \mathsf{LCDH}:\\ x, y \xleftarrow{\$} \mathbb{Z}_q;\\ L \leftarrow \mathcal{C}(g^x, g^y)\end{array}}$$

Let

$$\epsilon_{\mathsf{LCDH}} \stackrel{\text{def}}{=} \Pr_{\mathsf{LCDH}}[g^{xy} \in L]$$

For every PPT adversary $\mathcal{C}$, ϵ_{LCDH} is negligible

Transition from IND-CPA to G_1

Game $\mathsf{IND-CPA}$:	**Oracle** $H(\lambda)$:
$L \leftarrow [\,]$; $(x, \alpha) \leftarrow \mathcal{KG}(\,)$; $(m_0, m_1) \leftarrow \mathcal{A}(\alpha)$; $b \xleftarrow{\$} \{0,1\}$; $(\beta, v) \leftarrow \mathsf{Enc}(\alpha, m_b)$; $b' \leftarrow \mathcal{A}'(\alpha, \beta, v)$	if $\lambda \notin \mathsf{dom}(L)$ then $\quad h \xleftarrow{\$} \{0,1\}^\ell$; $\quad L \leftarrow (\lambda, h) :: L$ else $h \leftarrow L(\lambda)$ return h

Game G_1 :	**Oracle** $H(\lambda)$:
$L \leftarrow [\,]; x, y \xleftarrow{\$} \mathbb{Z}_q$; $(m_0, m_1) \leftarrow \mathcal{A}(g^x)$; $b \xleftarrow{\$} \{0,1\}$; $h \leftarrow H(g^{xy})$; $v \leftarrow h \oplus m_b$; $b' \leftarrow \mathcal{A}'(g^x, g^y, v)$	if $\lambda \notin \mathsf{dom}(L)$ then $\quad h \xleftarrow{\$} \{0,1\}^\ell$; $\quad L \leftarrow (\lambda, h) :: L$ else $h \leftarrow L(\lambda)$ return h

By inlining $\mathcal{KG}$ and Enc:

$$\Pr_{\text{IND-CPA}}\left[b = b'\right] = \Pr_{G_1}\left[b = b'\right]$$

Transition from G_1 to G_2

Game G_1 :	**Oracle** $H(\lambda)$:
$L \leftarrow [\,]; x, y \xleftarrow{\$} \mathbb{Z}_q;$	if $\lambda \notin \mathrm{dom}(L)$ then
$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$	$\quad h \xleftarrow{\$} \{0,1\}^\ell;$
$b \xleftarrow{\$} \{0,1\};$	$\quad L \leftarrow (\lambda, h) :: L$
$h \leftarrow H(g^{xy});$	else $h \leftarrow L(\lambda)$
$v \leftarrow h \oplus m_b;$	return h
$b' \leftarrow \mathcal{A}'(g^x, g^y, v)$	

Game G_2 :	**Oracle** $H(\lambda)$:
$L \leftarrow [\,]; x, y \xleftarrow{\$} \mathbb{Z}_q;$	if $\lambda \notin \mathrm{dom}(L)$ then
$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$	$\quad h \xleftarrow{\$} \{0,1\}^\ell;$
$b \xleftarrow{\$} \{0,1\};$	$\quad L \leftarrow (\lambda, h) :: L$
$h \xleftarrow{\$} \{0,1\}^\ell;$	else $h \leftarrow L(\lambda)$
$v \leftarrow h \oplus m_b;$	return h
$b' \leftarrow \mathcal{A}'(g^x, g^y, v)$	

By the *Fundamental Lemma*:

$$\left|\Pr_{G_1}\left[b = b'\right] - \Pr_{G_2}\left[b = b'\right]\right| \leq \Pr_{G_2}[g^{xy} \in L_{\mathcal{A}}]$$

Formally, we prove

$$\models G_2 \sim G_2 : \mathrm{true} \Rightarrow \Phi$$

where $\Phi = (g^{xy} \in \mathrm{dom}(L_{\mathcal{A}}))\langle 1\rangle \iff (g^{xy} \in \mathrm{dom}(L_{\mathcal{A}}))\langle 2\rangle$
$\quad \wedge (g^{xy} \in \mathrm{dom}(L_{\mathcal{A}})\langle 2\rangle \Rightarrow (b = b')\langle 1\rangle \Leftrightarrow (b = b')\langle 2\rangle$

Transition from G_2 to G_3

Game G_2 :	**Oracle** $H(\lambda)$:	**Game** G_3 :	**Oracle** $H(\lambda)$:
$L \leftarrow [\,]; x, y \xleftarrow{\$} \mathbb{Z}_q;$	if $\lambda \notin \mathrm{dom}(L)$ then	$L \leftarrow [\,]; x, y \xleftarrow{\$} \mathbb{Z}_q;$	if $\lambda \notin \mathrm{dom}(L)$ then
$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$	$h \xleftarrow{\$} \{0,1\}^\ell;$	$(m_0, m_1) \leftarrow \mathcal{A}(g^x);$	$h \xleftarrow{\$} \{0,1\}^\ell;$
$b \xleftarrow{\$} \{0,1\};$	$L \leftarrow (\lambda, h) :: L$	$b \xleftarrow{\$} \{0,1\};$	$L \leftarrow (\lambda, h) :: L$
$h \xleftarrow{\$} \{0,1\}^\ell;$	else $h \leftarrow L(\lambda)$	$h \xleftarrow{\$} \{0,1\}^\ell;$	else $h \leftarrow L(\lambda)$
$v \leftarrow h \oplus m_b;$	return h	$v \leftarrow h;$	return h
$b' \leftarrow \mathcal{A}'(g^x, g^y, v)$		$b' \leftarrow \mathcal{A}'(g^x, g^y, v)$	

$$\Pr_{G_2}[g^{xy} \in L_{\mathcal{A}}] = \Pr_{G_3}[g^{xy} \in L_{\mathcal{A}}]$$

$$\Pr_{G_2}\left[b = b'\right] = \Pr_{G_3}\left[b = b'\right] = \frac{1}{2}$$

Transition G_3 to G_{LCDH}

Game G_3 :	**Oracle** $H(\lambda)$:
$L \leftarrow []; x, y \xleftarrow{\$} \mathbb{Z}_q;$ $(m_0, m_1) \leftarrow \mathcal{A}(g^x);$ $b \xleftarrow{\$} \{0,1\};$ $h \xleftarrow{\$} \{0,1\}^\ell;$ $v \leftarrow h;$ $b' \leftarrow \mathcal{A}'(g^x, g^y, v)$	if $\lambda \notin \mathrm{dom}(L)$ then $h \xleftarrow{\$} \{0,1\}^\ell;$ $L \leftarrow (\lambda, h) :: L$ else $h \leftarrow L(\lambda)$ return h

Game LCDH :	**Oracle** $H(\lambda)$:
$x, y \xleftarrow{\$} \mathbb{Z}_q;$ $L' \leftarrow \mathcal{C}(g^x, g^y)$ **Adversary** $\mathcal{C}(\alpha, \beta)$: $L \leftarrow [];$ $(m_0, m_1) \leftarrow \mathcal{A}(\alpha);$ $v \xleftarrow{\$} \{0,1\}^\ell;$ $b' \leftarrow \mathcal{A}'(\alpha, \beta, v)$ return $\mathrm{dom}(L)$	if $\lambda \notin \mathrm{dom}(L)$ then $h \xleftarrow{\$} \{0,1\}^\ell;$ $L \leftarrow (\lambda, h) :: L$ else $h \leftarrow L(\lambda)$ return h

$$\Pr_{G_3}[g^{xy} \in \mathrm{dom}(L)] = \Pr_{\mathrm{LCDH}}\left[g^{xy} \in L'\right]$$

Summarizing

$$\begin{aligned}
|\Pr_{\text{IND-CPA}}[b = b'] - \tfrac{1}{2}| &= |\Pr_{G_1}[b = b'] - \Pr_{G_2}[b = b']| \\
&\leq \Pr_{G_2}[g^{xy} \in \mathrm{dom}(L_{\mathcal{A}})] \\
&= \Pr_{G_3}[g^{xy} \in \mathrm{dom}(L_{\mathcal{A}})] \\
&= \Pr_{\text{LCDH}}[g^{xy} \in \mathrm{dom}(L)] \\
&= \epsilon_{\text{LCDH}}
\end{aligned}$$

OAEP padding scheme

$$
Enc(M) \stackrel{\text{def}}{=} \quad \begin{array}{l} R \xleftarrow{\$} \{0,1\}^p; \\ S \leftarrow G(R) \oplus (M \| 0^{k_1}); \\ T \leftarrow H(S) \oplus R; \\ Y \leftarrow f(S \| T); \\ \text{return } Y \end{array}
$$

- $f : \{0,1\}^k \to \{0,1\}^k$ is a partial one-way function
- G and H are random oracles

$$
G(R) \stackrel{\text{def}}{=} \quad \begin{array}{l} \text{if } R \notin L \text{ then } r \xleftarrow{\$} \{0,1\}^k;\ L \leftarrow (R, r) :: L \\ \text{else } r \leftarrow L[R] \\ \text{return } r \end{array}
$$

Security against chosen ciphertext attacks

Game $G_{\text{IND-CCA2}}$:
$L_{\text{Dec}} \leftarrow [\,]$;
$(pk, sk) \leftarrow \mathcal{KG}(\eta)$;
$(m_0, m_1) \leftarrow \mathcal{A}_1(pk)$;
$b \xleftarrow{\$} \{0,1\}$;
$c^* \leftarrow \text{Enc}(m_b)$;
$c_{\text{def}} \leftarrow \text{true}$;
$\overline{b} \leftarrow \mathcal{A}_2(pk, c^*)$

Oracle $\text{Dec}(c)$:
$L_{\text{Dec}} \leftarrow (c_{\text{def}}, c) :: L_{\text{Dec}}$;
$\ldots$

$$\forall \mathcal{A}, \text{WF}(\mathcal{A}) \wedge \text{range}\ ((\text{true}, c^*) \notin L_{\text{Dec}})\ [\![G_{\text{IND-CCA2}}]\!] \implies$$
$$\left|\Pr_{G_{\text{IND-CCA2}}}\left[\overline{b} = b\right] - \tfrac{1}{2}\right| \leq \ldots$$

Restrictions on oracle calls

- Add counter for adversary calls; check number of calls as postcondition
- Check validity of queries as postcondition

Exact IND-CCA security of OAEP

Decryption oracle

Oracle $\mathrm{Dec}(c)$:
$L_{\mathrm{Dec}} \leftarrow (c_{\mathrm{def}}, c) :: L_{\mathrm{Dec}}$;
$(s, t) \leftarrow f^{-1}(sk, c)$;
$h \leftarrow H(s);\ r \leftarrow t \oplus h;\ g \leftarrow G(r)$;
if $[s \oplus g]_{k_1} = 0^{k_1}$ then
  return $[s \oplus g]^n$
else return $\perp$

Security statement

$$|Pr_{\mathbf{Game}}[b = b'] - \frac{1}{2}| \leq Pr_{I,f} + \frac{3q_D q_G + q_D^2 + 4q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}}$$

$Pr_{I,f}$ is the probability of an adversary I to partially invert f on a random element and q_X is the maximal number of queries to oracle X

More examples

- Encryption: Cramer-Shoup, IBE
- Signature: FDH, BLS
- Zero knowledge protocols
- Hash functions: Icart's construction, SHA3 finalists

The need for richer logics

Failure events cannot always be captured by a failure event: statistical zero knowledge, encodings

- Solution: make logics quantitative!
- Nice side-effect: applicable to differential privacy

Approximate Relational Hoare Logic

α-distance between distributions:

$$\Delta_\alpha(d_1, d_2) = \max_A(\max(d_1\ 1_A - \alpha\ (d_2\ 1_A), d_2\ 1_A - \alpha\ (d_1\ 1_A)))$$

Approximate lifting of a relation

$$\begin{aligned}
&\mathrm{lift}_{\alpha,\delta}\ R\ (d_1 : \mathcal{D}_A)\ (d_2 : \mathcal{D}_B) := \exists (d : \mathcal{D}_{A*B}),\\
&\ \pi_1(d) \leq d_1 \wedge \Delta_\alpha(\pi_1(d), d_1) \leq \delta\\
&\wedge \pi_2(d) \leq d_2 \wedge \Delta_\alpha(\pi_2(d), d_2) \leq \delta\\
&\wedge \mathrm{range}\ R\ d
\end{aligned}$$

Case $\alpha = 1$ coincides with Segala and Turrini (2007), Desharnais, Laviolette and Tracol (2008)

Assume $\models c_1 \sim_{\alpha,\delta} c_2 : P \Rightarrow =$. For all memories m_1 and m_2

$$P\ m_1\ m_2 \rightarrow \Delta_\alpha([\![c_1]\!]_{m_1}, [\![c_2]\!]_{m_2}) \leq \delta$$

Proof rules

Selected rules

$$\frac{\models c_1 \sim_{\alpha,\delta} c_2 : P \Rightarrow Q \quad P' \Rightarrow P \quad Q \Rightarrow Q'}{\models c_1 \sim_{\alpha,\delta} c_2 : P' \Rightarrow Q'}$$

$$\frac{}{\models x \leftarrow e \sim_{1,0} x \leftarrow e' : Q\{x\langle 1\rangle := e\langle 1\rangle, x\langle 2\rangle := e'\langle 2\rangle\} \Rightarrow Q}$$

$$\frac{\models c_1 \sim_{\alpha_1,\delta_1} c_1' : P \Rightarrow Q \quad \models c_2 \sim_{\alpha_2,\delta_2} c_2' : Q \Rightarrow R}{\models c_1; c_2 \sim_{\alpha_1\alpha_2,\delta_1+\delta_2} c_1'; c_2' : P \Rightarrow R}$$

$$\frac{\begin{array}{c} \models c_1 \sim_{\alpha,\delta} c_1' : P \wedge e\langle 1\rangle = \mathrm{tt} \Rightarrow Q \\ \models c_2 \sim_{\alpha,\delta} c_2' : P \wedge e\langle 1\rangle = \mathrm{ff} \Rightarrow Q \\ P \Rightarrow e\langle 1\rangle \Leftrightarrow e'\langle 2\rangle \end{array}}{\models \text{if } e \text{ then } c_1 \text{ else } c_2 \sim_{\alpha,\delta} \text{if } e' \text{ then } c_1' \text{ else } c_2' : P \Rightarrow Q}$$

Differential privacy (Dwork'06)

- Bound distance of output distributions corresponding to nearby secret inputs
- Protect individual bits (when inputs are bitstrings)

A randomized algorithm c satisfies (ϵ, δ)-differential privacy w.r.t. P iff for all memories m_1 and m_2 such that $P\ m_1\ m_2$:

- for all memories m_1 and m_2 such that $P\ m_1\ m_2$, and for every event E,

$$(\llbracket c_1 \rrbracket_{m_1} 1_E) \leq e^{\epsilon}\ (\llbracket c_2 \rrbracket_{m_2} 1_E) + \delta$$

- for all memories m_1 and m_2 such that $P\ m_1\ m_2$, $\Delta_{e^{\epsilon}}(\llbracket c_1 \rrbracket_{m_1}, \llbracket c_2 \rrbracket_{m_2}) \leq \delta$
- $\models c \sim_{e^{\epsilon},\delta} c : P \Rightarrow =$

Mechanisms for differentially private computations

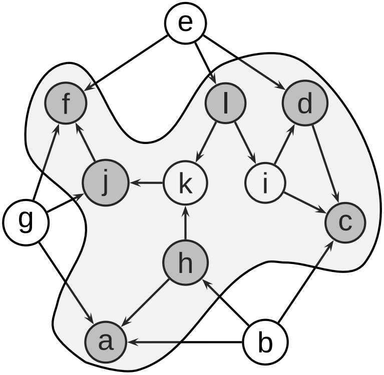
Differential Privacy from Output Pertubation (Dwork'06)

- (Real-valued) function f is k-sensitive iff for all a and a' such that $d(a, a') \leq 1$, we have $|f\ a - f\ a'| \leq k$
- Laplacian $\mathcal{L}(r, \sigma)$ with mean r and scale factor σ: prob. of x prop. to $\exp\left(-\frac{|x-r|}{\sigma}\right)$
- If f is k-sensitive, then $\lambda a.\ \mathcal{L}(f(a), k/\epsilon)$ is ϵ-DP

$$\frac{m_1\ \Psi\ m_2 \implies |[\![r]\!]\ m_1 - [\![r]\!]\ m_2| \leq k \qquad \exp(\epsilon) \leq \alpha}{\vDash \Psi \sim_{\alpha,0} x \xleftarrow{\$} \mathcal{L}(r, \frac{k}{\epsilon}) : y \xleftarrow{\$} \mathcal{L}(r, \frac{k}{\epsilon}) \Rightarrow x\langle 1\rangle = y\langle 2\rangle}$$

- Exponential mechanisms (Talwar and McSherry'07)
- Composition theorems

Differentially Private Vertex Cover (Gupta et al'10)




```
function VertexCover(V, E, ε)
    n ← |V|; π ← [ ]; i ← 0;
    while i < n do
        v ←$ choose(V, E, ε, n, i);
        π ← v :: π;
        V ← V \ {v}; E ← E \ ({v} × V);
        i ← i + 1
    end
```

where

$$\mathrm{choose}(V, E, \epsilon, n, i) = \frac{d_E(v) + w_i}{\sum_{x \in V} d_E(x) + w_i}$$

and

$$w_i = \frac{4}{\epsilon}\sqrt{\frac{n}{n-i}}$$

Formal statement

$$\models \textsc{VertexCover}(V, E, \epsilon) \sim_{e^\epsilon, 0} \textsc{VertexCover}(V, E, \epsilon) :$$
$$\Psi \pi\langle 1\rangle = \pi\langle 2\rangle$$

where

$$\Psi \stackrel{\text{def}}{=} V\langle 1\rangle = V\langle 2\rangle \wedge E\langle 2\rangle = E\langle 1\rangle \cup \{(t, u)\}$$

Proof intuition

Case analysis on the vertex v chosen in the iteration i

- v is not one of t, u and neither t nor u are in π
- v is one of t, u
- either t or u is already in π

One case

v is not one of t, u and neither t nor u are in π.

$$\frac{\Pr[v\langle 1\rangle = x]}{\Pr[v\langle 2\rangle = x]} = \frac{(d_{E\langle 1\rangle}(x) + w_i)\sum_{y\in V}(d_{E\langle 2\rangle}(y) + w_i)}{(d_{E\langle 2\rangle}(x) + w_i)\sum_{y\in V}(d_{E\langle 1\rangle}(y) + w_i)}$$

$$= \frac{(d_{E\langle 1\rangle}(x) + w_i)(2|E\langle 1\rangle| + (n-i)w_i + 2)}{(d_{E\langle 1\rangle}(x) + w_i)(2|E\langle 1\rangle| + (n-i)w_i)}$$

$$\le 1 + \frac{2}{(n-i)w_i} \le \exp\left(\frac{2}{(n-i)w_i}\right)$$

$$\frac{\Pr[v\langle 2\rangle = x]}{\Pr[v\langle 1\rangle = x]} \le 1$$

A general rule for while loops

$$I \Rightarrow (b_1\langle 1\rangle \equiv b_2\langle 2\rangle \wedge P_1\langle 1\rangle \equiv P_2\langle 2\rangle \wedge i\langle 1\rangle = i\langle 2\rangle)$$

$$\forall m_1\, m_2.\ m_1\ I\ m_2 \Rightarrow [\![\text{while } b_1 \text{ do } c_1]\!]_{m_1} = [\![[\text{while } b_1 \text{ do } c_1]_n]\!]_{m_1}$$

$$\begin{array}{l}\models c_1;\ \text{assert } (\neg P_1) \sim_{\alpha_1(j),0} c_2;\ \text{assert } (\neg P_2) : \\ \qquad I \wedge b_1\langle 1\rangle \wedge i\langle 1\rangle = j \ \wedge \neg P_1\langle 1\rangle \Rightarrow I \wedge i\langle 1\rangle = j + 1\end{array}$$

$$\begin{array}{l}\models c_1;\ \text{assert } (P_1) \sim_{\alpha_2,0} c_2;\ \text{assert } (P_2) : \\ \qquad I \wedge b_1\langle 1\rangle \wedge i\langle 1\rangle = j \wedge \neg P_1\langle 1\rangle \Rightarrow I \wedge i\langle 1\rangle = j + 1\end{array}$$

$$\frac{\begin{array}{c}\models c_1 \sim_{1,0} c_2 : \\ I \wedge b_1\langle 1\rangle \wedge i\langle 1\rangle = j \wedge P_1\langle 1\rangle \Rightarrow I \wedge i\langle 1\rangle = j + 1 \wedge P_1\langle 1\rangle\end{array}}{\begin{array}{c}\models \text{while } b_1 \text{ do } c_1 \sim_{(\prod_{i=a}^{a+n} \alpha_1(i))\times\alpha_2,0} \text{while } b_2 \text{ do } c_2 : \\ I \wedge i\langle 1\rangle = a \Rightarrow I \wedge \neg b_1\langle 1\rangle\end{array}}$$

Conclusion

- Crypto proofs can be formalized within reasonable time
- Fun topic, lots of new and interesting problems

Further work:

- Develop theory: decidability, probabilities as effects
- Improve tool: invariant generation, automation, modularity
- More examples: multi-party computation, computational DP, protocols
- Synthesis/automated transformation
- Other application domains: continuous distributions
- Implementations: F#, C