CertiCrypt

Esame di Sicurezza e Crittografia

Matteo Acerbi

2013-12-18

Outline

Section 1

Introduction

Who, where, web

People and places

- Gilles Barthe (IMDEA Software Institute, Spain)
- Benjamin Grégoire (INRIA Sophia-Antipolis Méditerranée, France)
- Federico Olmedo (IMDEA Software Institute, Spain)
- Santiago Zanella-Béguelin (Microsoft Research, UK)

Former members

- Daniel Hedin (Chalmers University of Technology, Sweden)
- Sylvain Heraud (Prove & Run, France)

Link

- `https://certicrypt.gforge.inria.fr/`

What is CertiCrypt?

A Coq library with

- Cryptographic proofs organised as sequences of games [12]
- *Deep* embedding of an imperative language with probabilistic assignment (pWhile)
- Probabilistic semantics based on ALEA library [2]
- *Shallow* embedding of a *relational* probabilistic program logic (pRHL)
- Program analysis tactics
- Special-purpose program logics
- ...

Why am I talking about this?

- **Sicurezza e Crittografia**
- Probabilistic programs/games in Type Theory
  - Syntax
  - Semantics
  - Reasoning
- Large and complex Coq library
  - Reusability of components
  - Maintainability

What is Coq?

An implementation of Type Theory

- Calculus of Inductive Constructions
  - higher-order logic
  - dependently-typed programming language
- tactic-based Interactive Theorem Prover

Similar systems

- Matita
- Dependent types: Agda, Idris, ATS
- ITPs: Isabelle, HOL-Light, ACL2

Why a proof assistant?

- Easy to make mistakes in cryptographic proofs
- (game-based) cryptographic proofs
  = (probabilistic) program verification
- Coq excels at mechanised program verification
  - CompCert
  - Ynot, Bedrock
  - Software Foundations
  - ...

Features

- Faithful and rigorous encoding of games
- Both asymptotic and exact security
- Independent verifiability
- Mechanised reasoning
  - reuse of software verification technology
  - automation

Complexity proofs by reduction

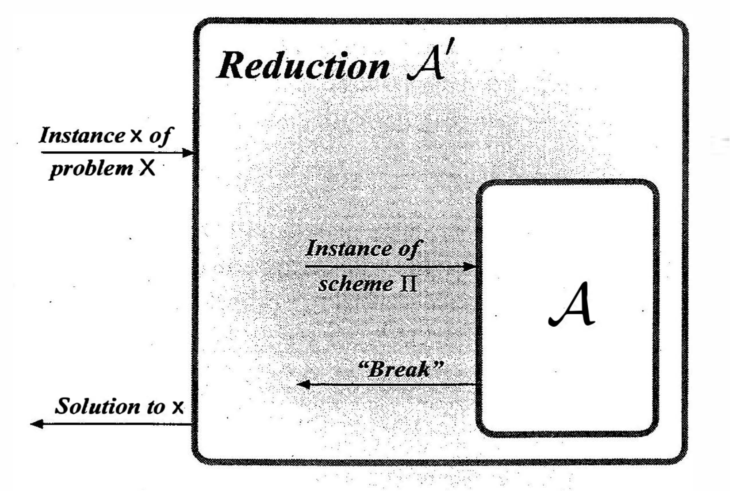

Figure : from [10]

Negligible sequence of real numbers

$\nu : \mathbb{N} \to \mathbb{R}$ is said negligible if it decreases faster than the inverse of any polynomial

$$\forall c \in \mathbb{N}. \exists n_c \in \mathbb{N}. \forall n \in \mathbb{N}. n \geq n_c \Rightarrow |\nu(n)| \leq n^{-c}$$

Game-based security definition

Indistinguishability under chosen-plaintext attacks

$$\begin{array}{l}\textbf{Game } \mathsf{IND\text{-}CPA}: \\ (pk, sk) \leftarrow \mathcal{KG}(\eta); \\ (m_0, m_1) \leftarrow \mathcal{A}_1(pk); \\ b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathcal{E}(pk, m_b); \\ \bar{b} \leftarrow \mathcal{A}_2(c)\end{array}$$

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CPA}} = \left| \Pr\left[\mathsf{IND\text{-}CPA} : b = \bar{b}\right] - \frac{1}{2} \right|$$

IND-CPA security $\triangleq$ the advantage of any PPT adversary is a negligible function of η

Security proofs as sequences of games

- Many cryptographic proofs can be organized as sequences of games
- Advantages
  - clarity
  - easier formalisation
  - easier automation
- Three types of transitions, for the following goals
  - 1, 2: $|Pr[G_i : \phi] - Pr[G_{i+1} : \psi]|$ negligible
  - $3 : Pr[G_i : \phi] = Pr[G_{i+1} : \psi]$
- Shoup 2004 [12]

Shoup: 1 - Transitions based on indistinguishability

- Assumption P_1, P_2 are *indistinguishable* distributions, i.e.

$$|P_1 - P_2| \text{ is negligible}$$

- We want to prove that $|Pr[G_i : \phi] - Pr[G_{i+1} : \psi]|$ is negligible
- We look for an algorithm D which
  - when given input drawn from P_1 outputs 1 with probability $Pr[G_i : \phi]$
  - when given input drawn from P_2 outputs 1 with probability $Pr[G_{i+1} : \psi]$
- If we find such an algorithm, the assumption implies that $|Pr[G_i : \phi] - Pr[G_{i+1} : \psi]|$ is negligible

Shoup: 2 - Transitions based on failure events

- Assumption G_i and G_{i+1} proceed identically unless a certain *failure event* F occurs, i.e. $(G_i : \phi) \wedge \neg F \Leftrightarrow (G_{i+1} : \psi) \wedge \neg F$
- A difference lemma entails: $|Pr[G_i : \phi] - Pr[G_{i+1} : \psi]| \leq Pr[F]$
- To prove that $|Pr[G_i : \phi] - Pr[G_{i+1} : \psi]|$ is negligible it is sufficient to prove that the failure event F has negligible probability to occur

Shoup: 3 - Bridging steps

- Assumption Subsequent games may differ only syntactically
- We must prove that games G_i and G_{i+1} have equivalent semantics with respect to events ϕ, ψ
- We can conclude that $Pr[G_i : \phi] = Pr[G_{i+1} : \psi]$

Section 2

Probabilistic reasoning in ALEA

ALEA

Coq library

- Christine Paulin-Mohring
- David Baelde, Pierre Courtieu
- `https://www.lri.fr/~paulin/ALEA/`

Most relevant to CertiCrypt

- Unit interval $[0, 1]$
- Distribution monad
- Lifting of predicates and relations

Unit interval $[0,1]$: (some) axioms

ω-CPO

$\leq$	:	$[0,1]^2 \to \star$	reflexive, transitive
$=$	:	$[0,1]^2 \to \star$	$\forall x, y.\ x \leq y \to y \leq x \to x = y$
sup	:	$(\mathbb{N} \to^m [0,1]) \to [0,1]$	$\forall f, n.\ f(n) \leq sup(f)$
0	:	$[0,1]$	$\forall x.\ 0 \leq x$

Operations

Addition: $(x, y) \mapsto \min(x + y, 1)$, where $+$ denotes addition over reals;

Inversion: $x \mapsto 1 - x$, where $-$ denotes subtraction over reals;

Multiplication: $(x, y) \mapsto x \times y$, where $\times$ denotes multiplication over reals;

Division: $(x, y \neq 0) \mapsto \min(x/y, 1)$, where $/$ denotes division over reals; moreover, if $y = 0$, for convenience division is defined to be 0.

Distribution monad

$$\mu : \mathcal{D}(A) \triangleq (\text{weight} : A \to [0,1]) \to^{m} [0,1]$$

$$\begin{array}{ll}
\textit{Monotonicity}: & f \leq g \Longrightarrow \mu\ f \leq \mu\ g; \\
\textit{Compatibility with inverse}: & \mu\ (\mathbb{1} - f) \leq 1 - \mu\ f; \\
\textit{Additive linearity}: & f \leq \mathbb{1} - g \Longrightarrow \mu\ (f + g) = \mu\ f + \mu\ g; \\
\textit{Multiplicative linearity}: & \mu\ (k \times f) = k \times \mu\ f; \\
\textit{Continuity}: & \text{if } f : \mathbb{N} \to (A \to [0,1]) \text{ is monotonic, then } \mu\ (\sup\ f) \leq \\
& \sup\ (\mu \circ f)
\end{array}$$

$$\begin{array}{lll}
\mathsf{unit} : A \to \mathcal{D}(A) & \stackrel{\text{def}}{=} & \lambda x.\ \lambda f.\ f\ x \\
\mathsf{bind} : \mathcal{D}(A) \to (A \to \mathcal{D}(B)) \to \mathcal{D}(B) & \stackrel{\text{def}}{=} & \lambda \mu.\ \lambda F.\ \lambda f.\ \mu\ (\lambda x.\ (F\ x)\ f)
\end{array}$$

- Cont R A $=$ $(A \to R) \to R$
- Ramsey, Pfeffer (2002) [11], Giry (1982) [9]

Lifting predicates and relations

$$\mathsf{range}\ P\ \mu \stackrel{\text{def}}{=} \forall f.\ (\forall x.\ P\ x \Longrightarrow f\ x = 0) \Longrightarrow \mu\ f = 0$$

lifting of a relation $R \subseteq A \times B$ to μ_1 and μ_2

$$\mu_1\, \mathcal{L}(R)\, \mu_2 \stackrel{\text{def}}{=} \exists \mu : \mathcal{D}(A \times B).\ \pi_1(\mu) = \mu_1\ \wedge\ \pi_2(\mu) = \mu_2\ \wedge\ \mathsf{range}\ R\ \mu$$

$$\pi_1(\mu) \stackrel{\text{def}}{=} \mathsf{bind}\ \mu\ (\mathsf{unit} \circ \mathsf{fst}) \qquad \pi_2(\mu) \stackrel{\text{def}}{=} \mathsf{bind}\ \mu\ (\mathsf{unit} \circ \mathsf{snd})$$

Section 3

Games as programs

pWhile syntax

$$
\begin{array}{rcll}
\mathcal{I} & ::= & \mathcal{V} \leftarrow \mathcal{E} & \text{deterministic assignment} \\
& | & \mathcal{V} \overset{\$}{\leftarrow} \mathcal{DE} & \text{probabilistic assignment} \\
& | & \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ \mathcal{C} & \text{conditional} \\
& | & \mathsf{while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{C} & \text{while loop} \\
& | & \mathcal{V} \leftarrow \mathcal{P}(\mathcal{E}, \ldots, \mathcal{E}) & \text{procedure call} \\
& | & \mathsf{assert}\ \mathcal{E} & \text{runtime assertion} \\
\mathcal{C} & ::= & \mathsf{skip} & \text{nop} \\
& | & \mathcal{I};\ \mathcal{C} & \text{sequence}
\end{array}
$$

$$
\begin{array}{r}
\text{command } c \in \mathcal{C} \\
\text{environment } E : \forall l\ t.\ \mathcal{P}_{(l,t)} \to \mathsf{decl}_{(l,t)} \\
\text{procedure } p \in \mathcal{P}_{(l,t)} \\
\mathsf{decl}_{(l,t)} \stackrel{\text{def}}{=} \{\mathsf{args} : {\mathcal{V}_l}^*;\ \mathsf{body} : \mathcal{C};\ \mathsf{re} : \mathcal{E}_t\}
\end{array}
$$

pWhile datatype

$$
\begin{array}{l}
\textbf{Inductive } \mathcal{I} : \mathsf{Type} := \\
\mid \mathsf{Assign} : \forall t.\ \mathcal{V}_t \to \mathcal{E}_t \to \mathcal{I} \\
\mid \mathsf{Rand} \quad : \forall t.\ \mathcal{V}_t \to \mathcal{D}\mathcal{E}_t \to \mathcal{I} \\
\mid \mathsf{Cond} \quad : \mathcal{E}_{\mathbb{B}} \to \mathcal{C} \to \mathcal{C} \to \mathcal{I} \\
\mid \mathsf{While} \quad : \mathcal{E}_{\mathbb{B}} \to \mathcal{C} \to \mathcal{I} \\
\mid \mathsf{Call} \quad\ : \forall l\ t.\ \mathcal{P}_{(l,t)} \to \ \mathcal{V}_t \to {\mathcal{E}_l}^* \to \mathcal{I} \\
\textbf{where } \mathcal{C} := \mathcal{I}^*
\end{array}
$$

Denotational semantics

$$[\![-]\!] : \mathcal{C} \to \mathcal{M} \to \mathcal{D}(\mathcal{M})$$

$$\begin{array}{lcl}
[\![\mathsf{skip}]\!]\ m & = & \mathsf{unit}\ m \\
[\![i;\ c]\!]\ m & = & \mathsf{bind}\ ([\![i]\!]\ m)\ [\![c]\!] \\
[\![x \leftarrow e]\!]\ m & = & \mathsf{unit}\ m\,\{[\![e]\!]_{\mathcal{E}}\ m/x\} \\
[\![x \xleftarrow{\$} d]\!]\ m & = & \mathsf{bind}\ ([\![d]\!]_{\mathcal{DE}}\ m)\ (\lambda v.\ \mathsf{unit}\ m\,\{v/x\}) \\
[\![x \leftarrow p(\vec{e})]\!]\ m & = & \mathsf{bind}\ ([\![p.\mathsf{body}]\!]\ (\varnothing\,\{[\![\vec{e}]\!]_{\mathcal{E}}\ m/p.\mathsf{args}\}\,, m.\mathsf{glob}) \\
 & & \qquad (\lambda m'.\ (m.\mathsf{loc}, m'.\mathsf{glob})\,\{[\![p.\mathsf{re}]\!]_{\mathcal{E}}\ m'/x\}) \\
[\![\mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2]\!]\ m & = & \begin{cases} [\![c_1]\!]\ m & \text{if } [\![e]\!]_{\mathcal{E}}\ m = \mathsf{true} \\ [\![c_2]\!]\ m & \text{if } [\![e]\!]_{\mathcal{E}}\ m = \mathsf{false} \end{cases} \\
[\![\mathsf{while}\ e\ \mathsf{do}\ c]\!]\ m & = & \lambda f.\sup\ (\lambda n.\ [\![[\mathsf{while}\ e\ \mathsf{do}\ c]_n]\!]\ m\ f) \\
 & & \text{where} \\
 & & \quad \begin{array}{lcl} [\mathsf{while}\ e\ \mathsf{do}\ c]_0 & = & \mathsf{assert}\ \neg e \\ [\mathsf{while}\ e\ \mathsf{do}\ c]_{n+1} & = & \mathsf{if}\ e\ \mathsf{then}\ c;\ [\mathsf{while}\ e\ \mathsf{do}\ c]_n \end{array}
\end{array}$$

Denotational semantics of pWHILE programs

PPT programs

Program termination

$$\mathsf{lossless}(c) \stackrel{\text{def}}{=} \forall m.\ \Pr[c, m : \mathsf{true}] = 1$$

Polynomially bounded distribution

A family of distributions $\mu_\eta : \mathcal{D}(\mathcal{M} \times \mathbb{N})$ is bounded by polynomials p, q if, for any η and any pair (m, n) with non-zero probability in μ_η, m's size is bounded by $p(\eta)$ and n is bounded by $q(\eta)$.

$$\mathsf{bounded}(p, q, \mu) \stackrel{\text{def}}{=} \forall \eta.\ \mathsf{range}\ (\lambda(m, n).\ \forall x \in \mathcal{V}.\ |m(x)| \leq p(\eta) \land n \leq q(\eta))\ \mu_\eta$$

(Strict) Probabilistic Polynomial-Time program c

- lossless(c)
- $\exists F, G$ *polynomial transformers* such that

$$\mu_\eta\ (p, q)\text{-bounded} \Rightarrow (\mathtt{bind}\ \mu_\eta\ [\![c]\!])\ (F(p), q + G(p))\text{-bounded}$$

Section 4

pRHL

Shallow embedding

- There is no inductive family corresponding to pRHL rules
- Rules are generic theorems validated by the probabilistic semantics
- No need to prove soundness

pRHL judgment

- c_1 and c_2 are equivalent with respect to pre-condition Ψ and post-condition Φ iff

$$\models c_1 \sim c_2 : \Psi \Rightarrow \Phi \stackrel{\text{def}}{=} \forall m_1\ m_2.\ m_1\ \Psi\ m_2 \Longrightarrow (\llbracket c_1 \rrbracket\ m_1)\,\mathcal{L}(\Phi)\,(\llbracket c_2 \rrbracket\ m_2)$$

- c_1, c_2 are semantically equivalent ($\models c_1 \equiv c_2$) if they are equivalent w.r.t **equality on memories** as pre- and post-condition.

(Some) pRHL derived rules [1/2]

$$\models \mathsf{skip} \sim \mathsf{skip} : \Phi \Rightarrow \Phi \ [\text{Skip}] \qquad \frac{\models c_1 \sim c_2 : \Psi \Rightarrow \Theta \qquad \models c_1' \sim c_2' : \Theta \Rightarrow \Phi}{\models c_1; c_1' \sim c_2; c_2' : \Psi \Rightarrow \Phi}[\text{Seq}]$$

$$\frac{m_1 \ \Psi \ m_2 = (m_1 \{[\![e_1]\!] m_1 / x_1\}) \ \Phi \ (m_2 \{[\![e_2]\!] m_2 / x_2\})}{\models x_1 \leftarrow e_1 \sim x_2 \leftarrow e_2 : \Psi \Rightarrow \Phi}[\text{Assn}]$$

$$\frac{m_1 \ \Psi \ m_2 \Longrightarrow ([\![d_1]\!] \ m_1) \, \mathcal{L}(\Theta) \, ([\![d_2]\!] \ m_2) \quad \text{where } v_1 \ \Theta \ v_2 = (m_1 \{v_1 / x_1\}) \ \Phi \ (m_2 \{v_2 / x_2\})}{\models x_1 \overset{\$}{\leftarrow} d_1 \sim x_2 \overset{\$}{\leftarrow} d_2 : \Psi \Rightarrow \Phi}[\text{Rnd}]$$

$$\frac{\begin{array}{c} m_1 \ \Psi \ m_2 \Longrightarrow [\![e_1]\!] \ m_1 = [\![e_2]\!] \ m_2 \\ \models c_1 \sim c_2 : \Psi \wedge e_1\langle 1\rangle \Rightarrow \Phi \qquad \models c_1' \sim c_2' : \Psi \wedge \neg e_1\langle 1\rangle \Rightarrow \Phi \end{array}}{\models \mathsf{if}\ e_1\ \mathsf{then}\ c_1\ \mathsf{else}\ c_1' \sim \mathsf{if}\ e_2\ \mathsf{then}\ c_2\ \mathsf{else}\ c_2' : \Psi \Rightarrow \Phi}[\text{Cond}]$$

(Some) pRHL derived rules [2/2]

$$\frac{m_1 \ \Phi \ m_2 \Longrightarrow [\![e_1]\!] \ m_1 = [\![e_2]\!] \ m_2 \qquad \models c_1 \sim c_2 : \Phi \wedge e_1\langle 1\rangle \Rightarrow \Phi}{\models \mathsf{while}\ e_1\ \mathsf{do}\ c_1 \sim \mathsf{while}\ e_2\ \mathsf{do}\ c_2 : \Phi \Rightarrow \Phi \wedge \neg e_1\langle 1\rangle}[\text{While}]$$

$$\frac{\Psi \subseteq \Psi' \quad \models c_1 \sim c_2 : \Psi' \Rightarrow \Phi' \quad \Phi' \subseteq \Phi}{\models c_1 \sim c_2 : \Psi \Rightarrow \Phi}[\text{Sub}] \qquad \frac{\models c_1 \sim c_2 : \Psi \Rightarrow \Phi \quad \mathsf{SYM}(\Psi) \quad \mathsf{SYM}(\Phi)}{\models c_2 \sim c_1 : \Psi \Rightarrow \Phi}[\text{Sym}]$$

$$\models c \equiv c \ [\text{Refl}] \qquad \frac{\models c_1 \sim c_2 : \Psi \Rightarrow \Phi \quad \models c_2 \sim c_3 : \Psi \Rightarrow \Phi \quad \mathsf{PER}(\Psi) \quad \mathsf{PER}(\Phi)}{\models c_1 \sim c_3 : \Psi \Rightarrow \Phi}[\text{Trans}]$$

$$\frac{\models c_1 \sim c_2 : \Psi \wedge \Psi' \Rightarrow \Phi \quad \models c_1 \sim c_2 : \Psi \wedge \neg\Psi' \Rightarrow \Phi}{\models c_1 \sim c_2 : \Psi \Rightarrow \Phi}[\text{Case}]$$

Observational equivalence

Defined in terms of a relational Hoare judgment where pre- and post-conditions are restricted to equality over a subset of program variables.
More formally, observational equivalence of programs c_1, c_2 w.r.t an input set of variables I and an output set of variables O is defined as follows.

$$\models c_1 \simeq^I_O c_2 \ \triangleq \models c_1 \sim c_2 \ : =_I \implies =_O$$

Section 5

Proof methods

Bridging steps

- Certified program transformations
  - tactics `eqobs`, `swap`, `deadcode`
  - certified optimisation function implementing dataflow analysis
    - tactic `ep`
- Algebraic equivalences
  - e.g.: in a cyclic multiplicative group multiplying by a uniformly sampled element acts as a one-time pad
- Interprocedural code motion
  - eager/lazy sampling
  - a logic for swapping (reordering) statements

Proof methods for failure events

- Difference lemma (generalisation)
  - $|Pr[G_1 : A] - Pr[G_2 : B]| \leq max(Pr[G_1 : F], Pr[G_2 : F])$
- A logic for bounding the probability of events

Section 6

Example - ElGamal

Public-key encryption scheme

Key generation: Given a security parameter η, the key generation algorithm $\mathcal{KG}(\eta)$ returns a public/secret key pair (pk, sk);

Encryption: Given a public key pk and a plaintext m, the encryption algorithm $\mathcal{E}(pk, m)$ computes a ciphertext corresponding to the encryption of m under pk;

Decryption: Given a secret key sk and a ciphertext c, the decryption algorithm $\mathcal{D}(sk, c)$ returns either the plaintext corresponding to the decryption of c, if it is a valid ciphertext, or a distinguished value $\perp$ otherwise.

Correctness

$$\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$$

Game-based security definition

Indistinguishability under chosen-plaintext attacks

$$
\begin{array}{l}
\textbf{Game } \mathsf{IND\text{-}CPA} : \\
\ (pk, sk) \leftarrow \mathcal{KG}(\eta); \\
\ (m_0, m_1) \leftarrow \mathcal{A}_1(pk); \\
\ b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathcal{E}(pk, m_b); \\
\ \bar{b} \leftarrow \mathcal{A}_2(c)
\end{array}
$$

$$
\mathbf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CPA}} = \left| \Pr\left[\mathsf{IND\text{-}CPA} : b = \bar{b}\right] - \frac{1}{2} \right|
$$

IND-CPA security $\triangleq$ the advantage of any PPT adversary is a negligible function of η

Groups

Group $(\mathcal{G} : \star, \times : \mathcal{G} \to \mathcal{G} \to \mathcal{G})$

Identity	$\exists e.\ \forall g.\ e \times g = g = g \times e$
Inverse	$\forall g.\ \exists h.\ g \times h = g = h \times g$
Associativity	$\forall g_1, g_2, g_3.\ \ (g_1 \times g_2) \times g_3 = g_1 \times (g_2 \times g_3)$

Exponentiation

$$\begin{array}{lcl} g^0 & = & e \\ g^{1+m} & = & g \times g^m \end{array}$$

Finite/Cyclic Groups

Finite Groups

- $\mathcal{G}$ finite
- Order $m = |\mathcal{G}|$

- Theorem For any $\mathcal{G}$ finite group, $m \in \mathcal{G}$, choosing g randomly from G and setting $g' := m \times g$ gives the same distribution for g' as choosing a random g' from $\mathcal{G}$
  - Essence of *one-time pad* encryption ($\mathcal{G} = \{0,1\}^l, \square$))

Cyclic groups

- Every element g generates a (finite) subgroup $\{g^i\}_{i \in \mathbb{N}} \mathcal{G}$
- If an element $g \in \mathcal{G}$ generates $\mathcal{G}$, this is called cyclic, and every element $h \in \mathcal{G}$ is equal to g^x from some $x \in \{0, \ldots, m-1\}$
- If x is uniformly distributed also g^x is uniformly distributed

DDH assumption

Games and advantage of the adversary (distinguisher)

$$\mathbb{Z}_n = \{i \mid 0 \leq i < n\}$$

$$\begin{array}{ll} \textbf{Game } \mathsf{DDH}_0 : & x, y \xleftarrow{\$} \mathbb{Z}_q; \quad\ \ d \leftarrow \mathcal{B}(g^x, g^y, g^{xy}) \\ \textbf{Game } \mathsf{DDH}_1 : & x, y, z \xleftarrow{\$} \mathbb{Z}_q;\ d \leftarrow \mathcal{B}(g^x, g^y, g^z) \end{array}$$

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DDH}} \stackrel{\text{def}}{=} |\Pr[\mathsf{DDH}_0 : d = 1] - \Pr[\mathsf{DDH}_1 : d = 1]|$$

DDH assumption

For any PPT $\mathcal{B}$, $Adv_{\mathcal{B}}^{DDH}$ is a negligible function of the security parameter

ElGamal [7]

$$
\begin{array}{lll}
\mathcal{KG}(\eta) & \stackrel{\text{def}}{=} & x \xleftarrow{\$} \mathbb{Z}_q;\ \mathsf{return}\ (g^x, x) \\
\mathcal{E}(\alpha, m) & \stackrel{\text{def}}{=} & y \xleftarrow{\$} \mathbb{Z}_q;\ \mathsf{return}\ (g^y, \alpha^y \times m) \\
\mathcal{D}(x, (\beta, \zeta)) & \stackrel{\text{def}}{=} & \mathsf{return}\ (\zeta \times \beta^{-x})
\end{array}
$$

$\mathcal{D}(x, \mathcal{E}(g^x, m)) = m$

$$
\begin{array}{rcl}
\alpha & = & g^x \\
\beta & = & g^y \\
\zeta & = & \alpha^y \times m \\
\zeta \times \beta^{-x} & = & \alpha^y \times m \times (g^y)^{-x} = (g^x)^y \times m \times g^{-xy} = g^{xy} \times m \times g^{-xy} = m
\end{array}
$$

Derivation

Game IND-CPA :
$(\alpha, x) \leftarrow \mathcal{KG}(\eta);$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\alpha);$
$b \xleftarrow{\$} \{0,1\};$
$(\beta, \zeta) \leftarrow \mathcal{E}(\alpha, m_b);$
$\overline{b} \leftarrow \mathcal{A}_2(\beta, \zeta);$
$d \leftarrow b = \overline{b}$

(1)
```
inline KG
inline E
ep
deadcode
swap
eqobs_in
```

$\simeq_{\{d\}}$

Game G_1 :
$x, y \xleftarrow{\$} \mathbb{Z}_q;$
$(m_0, m_1) \leftarrow \mathcal{A}_1(g^x);$
$b \xleftarrow{\$} \{0,1\};$
$\zeta \leftarrow g^{xy} \times m_b;$
$\overline{b} \leftarrow \mathcal{A}_2(g^y, \zeta);$
$d \leftarrow b = \overline{b}$

(2)
```
inline B
ep
deadcode
eqobs_in
```

$\simeq_{\{d\}}$

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

Adversary $\mathcal{B}(\alpha, \beta, \gamma)$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(\alpha);$
$b \xleftarrow{\$} \{0,1\};$
$b' \leftarrow \mathcal{A}_2(\beta, \gamma \times m_b);$
return $b = b'$

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

$\simeq_{\{d\}}$

(5)
```
inline B
ep
deadcode
swap
eqobs_in
```

Game G_2 :
$x, y \xleftarrow{\$} \mathbb{Z}_q;$
$(m_0, m_1) \leftarrow \mathcal{A}_1(g^x);$
$b \xleftarrow{\$} \{0,1\};$
$z \xleftarrow{\$} \mathbb{Z}_q; \zeta \leftarrow g^z \times m_b;$
$\overline{b} \leftarrow \mathcal{A}_2(g^y, \zeta);$
$d \leftarrow b = \overline{b}$

$\simeq_{\{d\}}$

(4)
```
swap
eqobs_hd 4
eqobs_tl 2
apply otp
```

Game G_3 :
$x, y \xleftarrow{\$} \mathbb{Z}_q;$
$(m_0, m_1) \leftarrow \mathcal{A}_1(g^x);$
$z \xleftarrow{\$} \mathbb{Z}_q;\ \zeta \leftarrow g^z;$
$\overline{b} \leftarrow \mathcal{A}_2(g^y, \zeta);$
$b \xleftarrow{\$} \{0,1\};$
$d \leftarrow b = \overline{b}$

ElGamal semantic security

Equations

$$\begin{array}{lclclcl} Pr[\text{IND-CPA} : b = \bar{b}] & = & Pr[G_1 : b = \bar{b}] & = & Pr[\text{DDH}_0 : d] & & \\ Pr[\text{DDH}_1 : d] & = & Pr[G_2 : d] & = & Pr[G_3 : d] & = & \frac{1}{2} \end{array}$$

$$\begin{array}{lcl} Adv_{\mathcal{B}}^{\text{DDH}} & = & |Pr[DDH_0 : d] - Pr[DDH_1 : d]| \\ & = & |Pr[\text{IND-CPA} : b = \bar{b}] - \dfrac{1}{2}| \\ & = & Adv_{\mathcal{A}}^{\text{IND-CPA}} \end{array}$$

Section 7

Conclusions

Other examples

- Hashed ElGamal [7]
- Full Domain Hash Signature scheme [13]
- Zero-knowledge proofs (Σ-protocols) [5]
- IND-CCA for OAEP (padding method/pubkey scheme) [6]
- Boneh-Franklin Identity-Based Encryption (e.g. OTR) [4]
- Differential privacy [8] [3]

Related projects

EasyCrypt

- (non-certified) SMT-based verification/inference of pRHL derivations
- used to generate CertiCrypt proofs

ZKCrypt [1]

- cryptographic compiler for zero-knowledge protocols
- backends
  - Java
  - C
- EasyCrypt integration

Similar projects

- CryptoVerif
  - Blanchet et al. (INRIA)
- rF*
  - Barthe (IMDEA)
  - Fournet, Grégoire, Swamy, Zanella-Béguelin (Microsoft Research)
  - Strub (MSR-INRIA & IMDEA)
  - Relational Hoare logic for a higher-order stateful probabilistic language
- `crypto-agda`
  - Nicolas Pouillard
  - `https://github.com/crypto-agda`

Thanks for your attention

Section 8

Backup

Complexity proofs by reduction (deterministic case)

Polytime reduction

A polynomial time reduction from a problem A to a problem B is an algorithm that solves problem A using a polynomial number of calls to a subroutine for problem B, and executes in polynomial time outside of those subroutine calls.

Complexity proofs by reduction (probabilistic case)

Negligible sequence

$\nu : \mathbb{N} \to \mathbb{R}$ is said negligible if it decreases faster than the inverse of any polynomial: $\forall c \in \mathbb{N}. \exists n_c \in \mathbb{N}. \forall n \in \mathbb{N}. n \geq n_c \Rightarrow |\nu(n)| \leq n^{-c}$

Probabilistic Polytime (PPT) reduction

a is a PPT reduction from problem A to problem B if for any algorithm b solving B with success probability p_b

- a can be defined using a polynomial number of calls to b (for every execution path)
- the overhead of algorithm a with respect to algorithm b is polynomially bounded
- a solves A with success probability p_a
- $|p_a - p_b|$ is a negligible function of the input (in cryptography, the security parameter)

Describing adversaries

$$I \vdash \mathsf{skip} : I \qquad \frac{I \vdash i : I' \quad I' \vdash c : O}{I \vdash i; c : O} \qquad \frac{\mathsf{writable}(x) \quad \mathsf{fv}(e) \subseteq I}{I \vdash x \leftarrow e : I \cup \{x\}} \qquad \frac{\mathsf{writable}(x) \quad \mathsf{fv}(d) \subseteq I}{I \vdash x \xleftarrow{\$} d : I \cup \{x\}}$$

$$\frac{\mathsf{fv}(e) \subseteq I \quad I \vdash c_1 : O_1 \quad I \vdash c_2 : O_2}{I \vdash \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 : O_1 \cap O_2} \qquad \frac{\mathsf{fv}(e) \subseteq I \quad I \vdash c : I}{I \vdash \mathsf{while}\ e\ \mathsf{do}\ c : I}$$

$$\frac{\mathsf{fv}(\vec{e}) \subseteq I \quad \mathsf{writable}(x) \quad p \in \mathcal{O}}{I \vdash x \leftarrow p(\vec{e}) : I \cup \{x\}} \qquad \frac{\mathsf{fv}(\vec{e}) \subseteq I \quad \mathsf{writable}(x) \quad \vdash_{\mathrm{wf}} \mathcal{B}}{I \vdash x \leftarrow \mathcal{B}(\vec{e}) : I \cup \{x\}}$$

$$\frac{\mathcal{RW} \cup \mathcal{R} \cup \mathcal{A}.\mathsf{args} \vdash \mathcal{A}.\mathsf{body} : O \quad \mathsf{fv}(\mathcal{A}.\mathsf{re}) \subseteq O}{\vdash_{\mathrm{wf}} \mathcal{A}}$$

$$\mathsf{writable}(x) \stackrel{\mathrm{def}}{=} \mathsf{local}(x) \vee x \in \mathcal{RW}$$

well-formedness of an adversary against interface $(\mathcal{O}, \mathcal{RW}, \mathcal{R})$

Section 9

Example - PRP/PRF switching lemma

Random Oracle Model methodology

- Random oracle $\triangleq$ a player H which
  - responds to every *unique query* with a truly random response chosen uniformly from its output domain
  - responds in the same way everytime it receives the same query
- Security proofs in the Random Oracle Model use oracles when implementations use hash functions
- Oracles are made available to all players
- Proofs show that a system is secure by showing that an attacker would require impossible behaviour from the oracle, or solve efficiently other problems believed to be harder

PRP/PRF Switching Lemma

Game G_{RP} :
$\boldsymbol{L} \leftarrow \mathsf{nil};\ b \leftarrow \mathcal{A}()$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathsf{dom}(\boldsymbol{L})$ then
$\quad y \xleftarrow{\$} \{0,1\}^{\ell} \setminus \mathsf{ran}(\boldsymbol{L});$
$\quad \boldsymbol{L} \leftarrow (x,y) :: \boldsymbol{L}$
return $\boldsymbol{L}[x]$

Game G_{RF} :
$\boldsymbol{L} \leftarrow \mathsf{nil};\ b \leftarrow \mathcal{A}()$

Oracle $\mathcal{O}(x)$:
if $x \notin \mathsf{dom}(\boldsymbol{L})$ then
$\quad y \xleftarrow{\$} \{0,1\}^{\ell};$
$\quad \boldsymbol{L} \leftarrow (x,y) :: \boldsymbol{L}$
return $\boldsymbol{L}[x]$

Lemma 8.6 PRP/PRF switching lemma. *Suppose $\mathcal{A}$ makes at most $q >$ queries to oracle $\mathcal{O}$. Then,*

$$|\Pr[\mathsf{G}_{\mathsf{RP}} : b = 1] - \Pr[\mathsf{G}_{\mathsf{RF}} : b = 1]| \leq \frac{q(q-1)}{2^{\ell+1}}$$

Failure Event / Eager Sampling

Game $\mathsf{G}_{\mathsf{RF}}^{\mathbf{bad}}$:
$\boldsymbol{L} \leftarrow \mathsf{nil};\ b \leftarrow \mathcal{A}()$

Oracle $\mathcal{O}(x)$:
```
if x ∉ dom(L) then
  y ←$ {0,1}^ℓ;
  if y ∈ ran(L) then
    bad ← true
  L ← (x, y) :: L
return L[x]
```

Game $\mathsf{G}_{\mathsf{RF}}^{\mathsf{eager}}$:
$\boldsymbol{L} \leftarrow \mathsf{nil};\ S;\ b \leftarrow \mathcal{A}()$

Oracle $\mathcal{O}(x)$:
```
if x ∉ dom(L) then
  if 0 < |Y| then
    y ← hd(Y);
    Y ← tl(Y)
  else y ←$ {0,1}^ℓ
  L ← (x, y) :: L
return L[x]
```

$$S \stackrel{\text{def}}{=} \boldsymbol{Y} \leftarrow \mathsf{nil};\ \mathsf{while}\ |\boldsymbol{Y}| < q\ \mathsf{do}\ \left(y \xleftarrow{\$} \{0,1\}^{\ell};\ \boldsymbol{Y} \leftarrow y :: \boldsymbol{Y}\right)$$

Figure : Games used in the proofs of the PRP/PRF Switching Lemma

Proof based on Eager Sampling [1/2]

- Assumption The adversary makes exactly q distinct queries
- By difference lemma

$$|Pr[G_{RP} : b = 1] - Pr[G_{RF} : b = 1]| \leq Pr[G_{RF}^{bad} : bad]$$

- $bad \Rightarrow col(L)$, hence

$$Pr[G_{RF}^{bad} : bad] \leq Pr[G_{RF} : col(L)]$$

- We apply *eager sampling* (S contains the anticipated random assignments) and we notice that S does not modify L

$$Pr[G_{RF} : col(L)] = Pr[G_{RF}; S : col(L)] = Pr[G_{RF}^{eager} : col(L)]$$

Proof based on Eager Sampling [2/2]

- Via pRHL, having a collision in the range of L at the end of G_{RF}^{eager} is the same as having a collision in Y right after S
- If there are no collisions in a memory m, by induction on $(q - |Y|)$ we can compute the probability of sampling a colliding value in the remaining loop iterations:

$$Pr[S, m : \exists i, j \in \mathbb{N}. i < j < q \wedge Y[i] = Y[j]] = \sum_{i=|Y|}^{q-1} \frac{i}{2^l}$$

- We can conclude that

$$\begin{aligned} P[G_{RF}^{eager} : col(L)] &\leq Pr[S, m\{nil/Y\} : \exists i, j \in \mathbb{N}. i < j < q \wedge Y[i] = Y[j]] \\ &= \frac{q(q-1)}{2^{l+1}} \end{aligned}$$

Section 10

References

Section 11

References

- José Bacelar Almeida et al. "Full Proof Cryptography: Verifiable Compilation of Efficient Zero-Knowledge Protocols". In: *19th ACM Conference on Computer and Communications Security*. ACM, 2012, pp. 488–500. URL: http://dx.doi.org/10.1145/2382196.2382249.
- Philippe Audebaud and Christine Paulin-Mohring. "Proofs of Randomized Algorithms in Coq". In: *Science of Computer Programming* 74.8 (2009). Extended version of [**audebaud06mpc**], pp. 568–589. URL: http://dx.doi.org/10.1016/j.scico.2007.09.002.
- Gilles Barthe and Federico Olmedo. "Beyond Differential Privacy: Composition Theorems and Relational Logic for f-divergences between Probabilistic Programs". In: *39th International Colloquium on Automata, Languages, and Programming*. Lecture Notes in Computer Science. To appear. Springer, 2013.
- Gilles Barthe, Federico Olmedo, and Santiago Zanella-Béguelin. "Verifiable Security of Boneh-Franklin Identity-Based Encryption". In: *5th International Conference on Provable Security -- ProvSec 2011*.

Vol. 6980. Lecture Notes in Computer Science. Springer, 2011, pp. 68–83. URL: http://dx.doi.org/10.1007/978-3-642-24316-5_7.

Gilles Barthe et al. "A Machine-Checked Formalization of Sigma-Protocols". In: *23rd IEEE Computer Security Foundations Symposium, CSF 2010*. IEEE Computer Society, 2010, pp. 246–260. URL: http://dx.doi.org/10.1109/CSF.2010.24.

Gilles Barthe et al. "Beyond Provable Security. Verifiable IND-CCA Security of OAEP". In: *Topics in Cryptology -- CT-RSA 2011*. Vol. 6558. Lecture Notes in Computer Science. Springer, 2011, pp. 180–196. URL: http://dx.doi.org/10.1007/978-3-642-19074-2_13.

Gilles Barthe et al. "Formal Certification of ElGamal encryption. A gentle introduction to CertiCrypt". In: *5th International Workshop on Formal Aspects in Security and Trust, FAST 2008*. Vol. 5491. Lectures Notes in Computer Science. Springer, 2009, pp. 1–19. URL: http://dx.doi.org/10.1007/978-3-642-01465-9_1.

Gilles Barthe et al. “Probabilistic Relational Reasoning for Differential Privacy”. In: *39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012*. ACM, 2012, pp. 97–110. URL: http://dx.doi.org/10.1145/2103656.2103670.

Michèle Giry. “A categorical approach to probability theory”. In: *Categorical Aspects of Topology and Analysis*. Ed. by B. Banaschewski. Vol. 915. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1982, pp. 68–85. ISBN: 978-3-540-11211-2. DOI: 10.1007/BFb0092872. URL: http://dx.doi.org/10.1007/BFb0092872.

Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. ISBN: 1584885513.

Norman Ramsey and Avi Pfeffer. “Stochastic lambda calculus and monads of probability distributions”. In: *POPL*. Ed. by John Launchbury and John C. Mitchell. ACM, 2002, pp. 154–165. ISBN: 1-58113-450-9.

- Victor Shoup. *Sequences of games: a tool for taming complexity in security proofs.* Cryptology ePrint Archive, Report 2004/332. http://eprint.iacr.org/. 2004.
- Santiago Zanella-Béguelin et al. "Formally Certifying the Security of Digital Signature Schemes". In: *30th IEEE Symposium on Security and Privacy, S&P 2009.* IEEE Computer Society, 2009, pp. 237–250. URL: http://dx.doi.org/10.1109/SP.2009.17.