Introduction to Cryptography

François Koeune – Olivier Pereira

Slide 02

Computational Security

Let $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ be a scheme s.t. $|\mathcal{K}| < |\mathcal{M}|$
Then Π is not a perfectly secret encryption scheme.

$\Rightarrow$ Somehow, we want *imperfect* encryption

What can we expect from imperfect security?

- Replace *impossibility* by *infeasibility* (what?)
- Practical scheme emulating perfect scheme for practical purposes

Computational Security

Emulating perfectness? (informally, for encryption)

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$, and adversary $\mathcal{A}$
Define the following experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}{}'$:

1. $\mathcal{A}$ not unbounded outputs $m_0, m_1 \in \mathcal{M}$
2. Choose $k \leftarrow \mathrm{Gen}$ and $b \leftarrow \{0, 1\}$, and send $c = \mathrm{Enc}_k(m_b)$ to $\mathcal{A}$
3. $\mathcal{A}(c)$ outputs b'
4. Define $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}{}' := 1$ iff $b = b'$

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}{}' = 1] = \frac{1}{2} + \text{negligible term}$$

Computational Security: Concrete Bounds

Two relaxations:

1. $\mathcal{A}$ does not have unbounded power
   - Suppose $\mathcal{A}$ makes a brute force search
     - 10^9 keys/sec
     - 10^6 computers
     - 10^9 seconds (≈ 30 years)

     $\Rightarrow$ makes $\approx 2^{80}$ steps
2. $\mathcal{A}$ can have a small success probability
   - Say, 2^{-48}
     - Event with Pr $\approx 2^{-30}/sec$ occurs 1/100 years
     - Proba. to win the Belgian Lotto: $\approx 2^{-22}$

$\Rightarrow$ Suggests that $|\mathcal{K}| = 2^{128}$ would be very nice!

Computational Security: Concrete Bounds

Two relaxations:

1. $\mathcal{A}$ does not have unbounded power
2. $\mathcal{A}$ can have small success probability

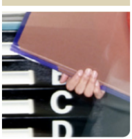
Proposed solution:

- design schemes that are (t, ϵ)-secure

What does it say?

- Concrete bounds: we know what we can hold
- Contests can be organized, ...

Computational Security: Concrete Bounds



Elliptic curves over F_2m

Curve	Field size (in bits)	Estimated number of machine days	Prize (US$)	Status
ECC2-79	79	352	Handbook of Applied Cryptography & Maple V software	SOLVED December 1997
ECC2-89	89	11278	Handbook of Applied Cryptography & Maple V software	SOLVED February 1998
ECC2K-95	97	8637	$ 5,000	SOLVED May 1998
ECC2-97	97	180448	$ 5,000	SOLVED September 1999
ECC2K-108	109	1.3×10^6	$ 10,000	SOLVED April 2000
ECC2-109	109	2.1×10^7	$ 10,000	SOLVED April 2004
ECC2K-130	131	2.7×10^9	$ 20,000	
ECC2-131	131	6.6×10^{10}	$ 20,000	
ECC2-163	163	2.9×10^{15}	$ 30,000	

See: `https://www.certicom.com/content/certicom/en/the-certicom-ecc-challenge.html`

Computational Security: Asymptotic Bounds

Concrete security bounds:

- Suppose $\langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ is (t, ϵ)-secure.
  What about $5t$?
- How does (t, ϵ) change with key size?
  Can encryption cost become $>$ cryptanalysis cost?

Another solution:

- Consider (t, ϵ) as functions of a security parameter n (e.g., the length of the key)
- Define a class of "feasible" algorithms
  Number of steps as a $f(n)$
- Show: $\forall$ feasible strategy of $\mathcal{A}$, $\Pr[\mathcal{A}$ wins$] \rightarrow 0$ fast when n increases

Computational Security: Asymptotic Bounds

Asymptotic security:

Π is secure if

- every "feasible" adversary strategy
- succeeds with "negligible" probability

as a function of the security parameter

An increase of computational power is bad news for $\mathcal{A}$

Feasible Strategy / Efficient Algorithm

Let n be the length of the numbers we use...

Simple computational problems:

- Addition: $\mathcal{O}(n)$
- Multiplication: $\mathcal{O}(n^2)$
- (Modular) Exponentiation: $\mathcal{O}(n^3)$
- Search in ordered list: $\mathcal{O}(\log(n))$

Computationally hard problems:

- Integer factorization: $2^{\mathcal{O}(\sqrt{n \cdot \log(n)})}$
- Exhaustive key search: 2^n

(The complexities listed here reflect classical algorithms, not the best known algorithms, which have more complex expressions.)

Feasible Strategy / Efficient Algorithm

Observations suggest:

Polynomial-time algorithms are efficient

- $\mathcal{A}$ is efficient if $\exists$ polynomial p s.t. $\mathcal{A}(x)$ takes $\leq p(|x|)$ steps

"Arbitrary" choice!?

- Various computing architectures can make complexity change by a polynomial factor
- Algorithmic improvements typically change complexity by a polynomial factor
- But is $|x|^{1000}$ less efficient than $e^{\frac{|x|}{1000}}$?
- What tells asymptotic security about $|x| = 128$?

For discussion see, e.g., `http://www.cs.princeton.edu/theory/complexity/`

Feasible Strategy / Efficient Algorithm

Observations suggest:

Polynomial-time algorithms are efficient

- $\mathcal{A}$ is efficient if $\exists$ polynomial p s.t.
  $\mathcal{A}(x)$ takes $\leq p(|x|)$ steps

Does it capture what we want?

- Cryptography is about randomness
  e.g., choosing a key, ...
- If honest parties can use randomness, so should $\mathcal{A}$!
- Is a probabilistic $\mathcal{A}$ more powerful?
- Where do we find randomness?

Feasible Strategy / Efficient Algorithm

Efficient $\Leftrightarrow$ *Probabilistic polynomial-time* (PPT)

- $\mathcal{A}$ is efficient if $\exists$ polynomial p s.t.
  $\mathcal{A}(x)$ takes $\leq p(|x|)$ steps
- Steps include tossing a coin

Negligible Functions

What should be a "negligible" success probability?

- $1/p(n)$ is not small for PPT $\mathcal{A}$:
  Polynomial repetition of $\mathcal{A}$ is still PPT

$\Rightarrow$ "negligible" $\Leftrightarrow$ "decreases faster than any inverse poly."

f is *negligible* iff:
$\forall$ positive polynomial p, $\exists N$ such that $\forall n \geq N$:

$$f(n) \leq \frac{1}{p(n)}$$

Examples: 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log(n)}$

Asymptotic security

Why using such an asymptotic treatment?

- Tells us how primitives behave
- PPT algos and negl. functions are convenient:
  - PPT algo running PPT algos is still PPT
  - f negl and p poly $\Rightarrow p \cdot f$ negl
- PPT/negl capture our experience
- PPT/negl are robust to model changes

This will be adopted in the rest of this course

Encryption

What would be encryption in a computational world?

A triple $\langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ of PPT algos: (Introduce n)

- Gen probabilistically selects $k \leftarrow \mathrm{Gen}(1^n)$
  $1^n := 111\cdots1$ (n times) n is the security parameter
- Enc provides $c \leftarrow \mathrm{Enc}_k(m)$
- Dec provides $m := \mathrm{Dec}_k(c)$

Correctness: "for any security parameter..."

- $\forall n$, $k \leftarrow \mathrm{Gen}(1^n)$, and $\forall m$: $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$

Security for Encryption

What would this become in a computational world?

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$, and adversary $\mathcal{A}$, define the experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. $\mathcal{A}(1^n)$ outputs m_0, m_1 of identical lengths
2. Pick $k \leftarrow \mathrm{Gen}(1^n)$, $b \leftarrow \{0, 1\}$, and send $c \leftarrow \mathrm{Enc}_k(m_b)$ to $\mathcal{A}$
3. $\mathcal{A}(c)$ outputs b'
4. Define $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) := 1$ iff $b = b'$

Security of Encryption

$\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ has *indistinguishable encryptions in the presence of eavesdroppers* if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ :

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \epsilon(n)$$

Efficiency and Security depend on the security parameter

Building Secure Encryption Schemes

Emulating a one-time pad!

$\mathrm{Gen}(1^n)$: pick $k \leftarrow \{0,1\}^{l(n)}$, set $\mathcal{M} = \{0,1\}^{l(n)}$

$\rightarrow$ We want $|k| < |m| = l(n)$, for any $m \in \mathcal{M}$

$\rightarrow$ Expand $k \in \{0,1\}^n$ to $G(k) \in \{0,1\}^{l(n)}$

$\mathrm{Enc}_k(m)$: compute and return $c = m \oplus k$

$\rightarrow$ Instead, compute $c = m \oplus G(k)$

$\rightarrow$ Enough if $G(k)$ is *indistinguishable* of random

$\mathrm{Dec}_k(c)$: compute and return $m = c \oplus k$

$\rightarrow$ Instead, compute $c \oplus G(k)$

G is called a *pseudorandom generator*

Pseudorandomness

A deterministic poly-time algorithm G is a *pseudorandom generator* (or simply a PRG) only if, for any n,

- $\forall s \in \{0,1\}^n$, $G(s) \in \{0,1\}^{l(n)}$ and $l(n) > n$
- $\forall$ PPT D, $\exists$ negligible function ϵ s.t.

$$\Big| \Pr[D(r) = 1] - \Pr[D(G(s)) = 1] \Big| \leq \epsilon(n)$$

where $r \leftarrow \{0,1\}^{l(n)}$ and $s \leftarrow \{0,1\}^n$

String s is called the *seed*
Function l is called the *expansion factor* of G

Pseudorandomness

Pseudorandom generators:

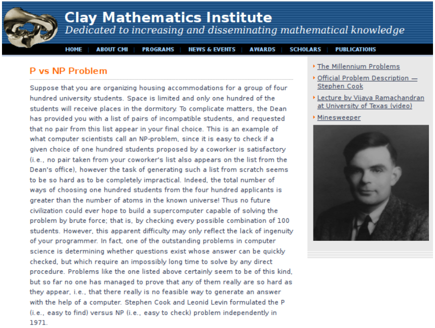
- Can only be computationally secure: try $l(n) = 2 \cdot n$
- n should be large enough to avoid brute-force

Theorem: [Håstad, Impagliazzo, Levin, Luby, 1999]
Pseudorandom generators exist iff one-way functions exist

Existence of one-way functions $\Rightarrow$ $\mathbf{P} \neq \mathbf{NP}$

Pseudorandomness

Proving the existence of a pseudorandom generator is worth $1.000.000!



Clay Mathematics Institute

Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

P vs NP Problem

Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

- The Millennium Problems
- Official Problem Description — Stephen Cook
- Lecture by Vijaya Ramachandran at University of Texas (video)
- Minesweeper

See: `http://www.claymath.org/millennium-problems/p-vs-np-problem/`

Pseudorandomness

Assumption:

Pseudorandom generators exist

Building Secure Encryption Schemes

Our intuition leads us to $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$:

$\mathrm{Gen}(1^n)$: pick $k \leftarrow \{0,1\}^n$, set $\mathcal{M} = \{0,1\}^{l(n)}$

$\mathrm{Enc}_k(m)$: compute and return $c = m \oplus G(k)$

$\mathrm{Dec}_k(c)$: compute and return $m = c \oplus G(k)$

What about security?

Definition $\Rightarrow$ Assumption $\Rightarrow$ Reduction

Building Encryption Schemes

THEOREM

If G is a pseudorandom generator, then Π is a (fixed length) encryption scheme that has indistinguishable encryption in the presence of an eavesdropper.

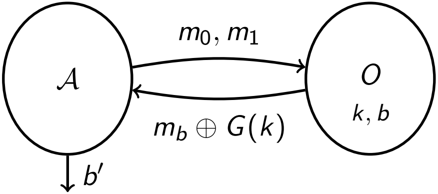
PROOF.

- $\forall$ PPT $\mathcal{A}$ for Π with η advantage,
- $\exists$ PPT D for G with η advantage.

Therefore, η must be negligible. $\square$

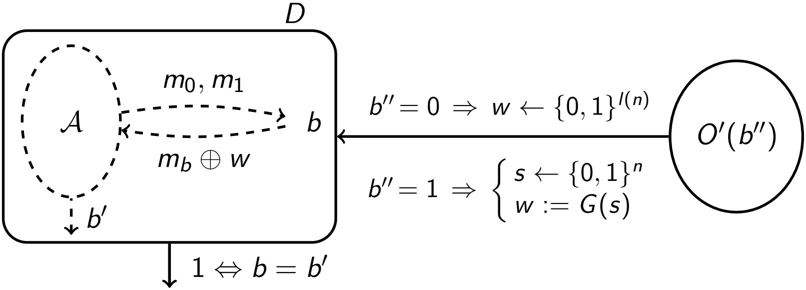
Reduction

Suppose $\Pr[\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \eta(n)$



$\Rightarrow \mathcal{A}$ is s.t. $\Pr[b = b'] = \frac{1}{2} + \eta(n)$

Reduction



Observe:

- If $b'' = 0$, $\Pr[D \text{ outputs } 1] = \frac{1}{2}$: one-time pad
- If $b'' = 1$, $\Pr[D \text{ outputs } 1] = \frac{1}{2} + \eta(n)$: $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$
- D distinguishes with advantage $\eta(n)$
- If $\mathcal{A}$ is PPT, then D is PPT too

Building Encryption Schemes

THEOREM

If G is a pseudorandom generator, then Π is a (fixed length) encryption scheme that has indistinguishable encryption in the presence of an eavesdropper.

Observations:

- Encryption with keys shorter than messages!
- Proof is *conditional*, by reduction
- Proof holds for computationally bounded adversaries, and leaves some probability of attack

Variable-length encryption

Π is restricted to messages with $|m| \leq l(|k|)$

- We need G with variable-length output
- Use G repeatedly until output is long enough (see KL, Fig. 7.1)
- Preserves security if G is used polynomially many times

Conclusion

- Computational security opens doors for cryptography with shorter keys

  We can encrypt m of length $p(|k|)$!

- What about encrypting multiple messages? (same key)

  Even the one-time pad becomes insecure!