Security Models: Proofs

Lecture 2

Jean-Louis Roch

Master-2 Security, Cryptology and Coding of Information Systems

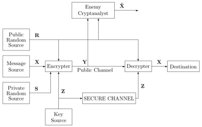
Grenoble University, France

ENSIMAG/INPG – UJF

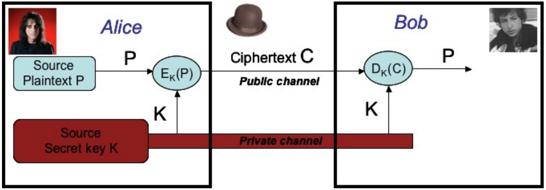
Outlines

- Lecture 1 : attacks ; security defs ; unconditional security and entropy.
- Lecture 2 :
  - Part 2 : Asymmetric protocols and provable security
    1. Asymmetric cryptography is not unconditionally secure
    2. Provable security : arithmetic complexity and reduction
    3. Complexity and lower bounds : exponentiation
    4. P, NP classes

Symmetric cryptosystem and unconditional security



General model



Simplified model

Definition : *Unconditional security* or *Perfect secrecy*

The symmetric cipher is unconditionally secure iff $H(P|C) = H(P)$

Shannon's theorem : necessary condition, lower bound on K

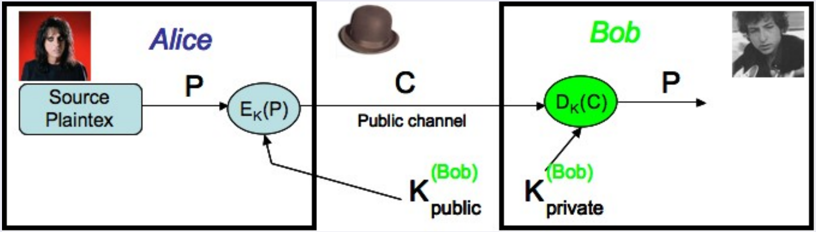
In any unconditionally secure cryptosystem : $H(K) \geq H(P)$.

Existence of unconditionally secure cryptosystem

In any group G, Vernam cipher (or One-Time-Pad) is unconditionally secure.

Asymmetric cryptography : not unconditionally secure

Model of asymmetric cryptography



- Let K_e= public key ; let K_d=secret key. The public key K_e is fixed and known ; then C gives all information about P :

$$H(P|C) = 0$$

$\Longrightarrow$ **asymmetric cryptography is not unconditionally secure.**

- Moreover, $D_{K_d} = E_{K_e}^{-1}$: then $H(K_d|K_e) = 0$.
- Shannon's information theory cannot characterize the security of an asymmetric cryptosystem $\hookrightarrow$ **complexity theory**

Asymmetric cipher and Provable security

Definition : one-way function

A bijection (i.e. one-to-one mapping) f is **one-way** iff

- (i) It is easy to compute $f(x)$ from x ;
- (ii) Computation of $x = f^{-1}(y)$ from $y = f(x)$ is intractable, i.e. requires too much operations, e.g. $10^{120} \simeq 2^{400}$

How to prove one-way ?

- (i) Analyze the arithmetic complexity of an algorithm that computes f.
- (ii) Provide a lower bound on the minimum arithmetic complexity to compute $x = f^{-1}(y)$ given y
  - very hard to obtain lower bounds in complexity theory
  - it is related both to the problem f^{-1} and the input y (i.e. x)

Provable security [*Contradiction proof, by reduction*] if computation of f^{-1} is not intractable, then a well-studied and presumed intractable problem could be solved.

Polynomial reductions

Definition of P-reduction : $\leq_P$

- Let A and B be two problems
- Def : **oracle** for B : OracleB(x) computes $B(x)$ in time $|x|$.
- Def : $A \leq_P B$ iff there exists an algorithm AlgoA that computes $A(x)$ in polynomial time
  i.e. in Time $\leq \alpha.|x|^k = |x|^{O(1)}$ using standard operations (DTM or RAM model) and oracles for B.

Example : Brown's reduction for RSA (with straight line program)

LE-RSA == RSA with low exponent e
if there is an efficient program that, given N, constructs a straight line program that efficiently solves LE-RSA with modulus N, (*i.e. constructs a polynomial that inverts the RSA encryption function*), then the program can also be used to efficiently factor N.
$\hookrightarrow$ This suggests that LE-RSA may very well be equivalent to factoring.

Arithmetic complexity : an example

Exponentiation in a group $(G, \otimes, e)$ with $m = |G|$ elements

- Input : $x \in G$, $n \in \{0, \ldots, |G| - 1\}$ an integer
- Output : $y \in G$ such that $y = x^n$;
- In practice : G is finite but has at least 10^{120} elements

Naive algorithm

- `y=e ; for (i=0 ; i < n ; i++) y=y ⊗ x ;`
- This algorithm does not work in practice : why ?

What's about this one ?

```
G power( G x, int n)
{
    return (n==0) ? e : x ⊗ power( x, n-1 ) ;
}
```

$\hookrightarrow$ **Can you do better ?**

Recursive binary exponentiation : $x^n = \left(x^{n/2}\right)^2 \otimes x^{n\%2}$

```
G power( G x, int n)
{
    if (n==0) { return e ; }
    elsif (n==1) { return x ; }
    else {  G tmp = power( x, n/2) ;
           tmp = tmp ⊗ tmp ;
           return (n%2 ==0) ? tmp : tmp ⊗ x ;
         }
}
```

Arithmetic complexity

$$\log_2 n \leq \#\text{multiplications} \leq 2 \log_2 n$$

E.g. : x^{15} : computed with 6 multiplications

$\hookrightarrow$ **Can you do better ?**

Lower bound for #multiplications to compute x^n

Let $g(n)$ = minimum number of multiplications to compute x^n.

Direct lower bound of $g(n)$

#multiplications	x^n
1	x^2
2	x^3, x^4
3	x^5, x^6, x^7, x^8
4	$x^9, x^{10}, x^{11}, x^{12}, x^{13}, x^{14}, x^{15}, x^{16}$

$\Longrightarrow$ recursive binary powering is not optimal (e.g. x^{15})

Theorem : $g(n) \geq \log_2 n$

Proof : by recurrence [*dynamic programming*]

- $g(2) = 1$;
- $g(n) = \min_{i=1,\ldots,n-1} \max(g(i); g(n-i)) + 1 \geq \log_2(n-1) + 1 \geq \log_2 n$

Outlines

- Lecture 2 : Asymmetric protocols and provable security
  - Asymmetric cryptography is not unconditionally secure
  - Provable security : arithmetic complexity and reduction
  - Complexity and lower bounds : exponentiation
  - **P, NP classes. Reduction.**

P and NP : defintions

1. Complexity classes : Two basic computing models :
   - Deterministc Turing Machine (DTM) (almost equivalent to RAM)
   - Non-Deterministic Turing Machine (NDTM) : **prefix N**
   - $X \subset NX \subset NX - SPACE = X - SPACE$
2. Decision problems and NP class
3. Equivalent definitions of NP :
   - $==$ set of decision problems that can be solved in polynomial-time on a Non-Deterministic TM (NDTM) $\hookrightarrow$ Non-determinstic Polynomial time
   - $==$ set of decision problems which YES output can be proved by a certification algorithm that runs in polynomial time on a Deterministic TM $\hookrightarrow$ Polynomial-time proofs
4. NP includes P.

   Examples : IsCompose $\in$ NP ; IsPrime $\in$ co-NP (indeed both are in P)

ref "The status of the P versus NP problem", Lance Fortnow, Communications of the ACM, 2009, Sept

1. P-reduction : $\leq_P$
   Theorem : if $(A \leq_P B)$ and $(B \in P)$ then $A \in P$.
   Contrapositive : if $(A \leq_P B)$ and $(A \notin P)$ then $(B \notin P)$.
2. Rem. NP is closed under Karp-P-reduction (but non known with general Turing P-reduction)
3. NP-hard, NP-complete
4. Examples of NP-complete problems : SAT (NP-complete), SubsetSum (NP-Complete, APX)
5. Examples of presumed intractable problems :
   Discrete logarithm (?), Integer Factorization (?)
6. One-way trapdoor function and NP-completeness
7. Building a one-way trapdoor function from an presumed untractable problem :
   Example : Merkle-Hellman