Cours Modèles pour la sécurité

Organisation des 4 séances :

1. **Cours 1 :** Modèles d'attaques - Sécurité inconditionnelle, prouvée, sémantique
   Chiffrement symétrique inconditionnellement sûrs
2. **Cours 2 :** Sécurité prouvée d'un chiffrement asymétrique : RSA
3. **Cours 3 :** Fonctions de hachage cryptographiquement sûres
   Générateur aléatoire cryptographiquement sûr et padding
4. **Cours 4 :** Protocoles à divulgation nulle de connaissance (zero-knowledge)
   Applications.

Ref : *Théorie des Codes : compression, cryptage, compression.*
JG Dumas, JL Roch, E Tannier, S Varrette. Dunod.
`http ://www-id.imag.fr/~jlroch/perso_html/COURS/ISI-3A/`

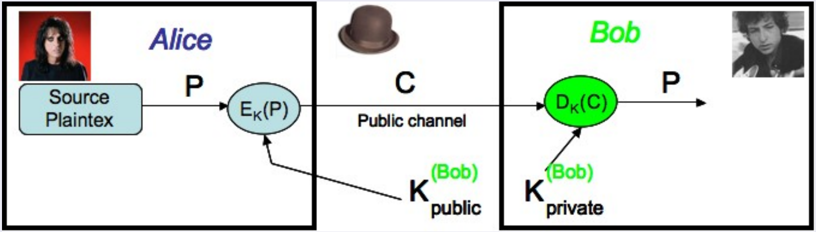
Outlines of lecture 2

Asymmetric protocols and provable security

- Part 1 : Asymmetric cryptography, one way function, complexity
- Part 2 : arithmetic complexity and lower bounds : exponentiation
- Part 3 : Provable security and polynomial time reduction : P, NP classes. Reduction. One-way function and NP class.
- Part 4 : RSA : the algorithm
- Part 5 : Provable security of RSA
- Part 6 : Attacks and importance of padding.

Asymmetric cryptography : not unconditionally secure

Model of an asymmetric cryptosystem



- 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)

In 2007, no proof is known of the existence of one-way function.

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.

Provable security is based on complexity :

- Remind about arithmetic complexity :
  exponentiation and discrete logarithm
- Remind P and NP classes :
  relation to one-way function
- Problems commonly used in provable security

Arithmetic complexity : an example

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

- Input : $x \in G$, $n \in \{0, \dots, |G| - 1\}$ 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

Notation : $\mathrm{LB}(n)$ = minimum number of multiplications to compute x^n.

Evaluation of $\mathrm{LB}(n)$

#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 : $\mathrm{LB}(n) \geq \log_2 n$

Proof : by recurrence [*dynamic programming*]

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