Asymmetric Cryptography
Provable Security

IML, Luminy, France
November 9th, 2004

David Pointcheval
CNRS-ENS

Provable Security

- Need of Computational Assumptions
- Provable Security
- Security Notions

Asymmetric Cryptography

Alice ⟵ secrecy / authenticity ⟶ Bob

Diffie-Hellman 1976

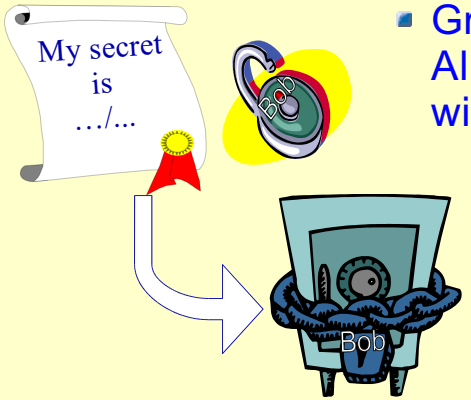
Asymmetric Encryption:
Bob owns two "keys"

- A public key (encryption k_e)
  - so that anybody can encrypt a message for him

  ⇒ known by everybody (including Alice)
- A private key (decryption k_d)
  - to help him to decrypt

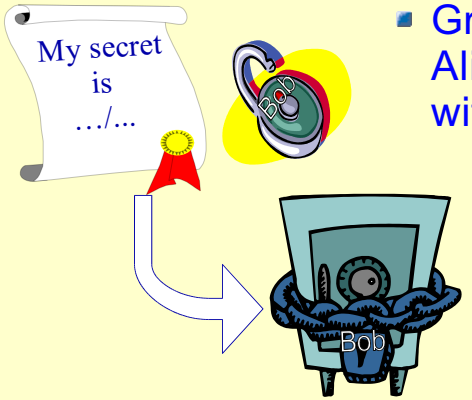
  ⇒ known by Bob only

Encryption / decryption attack



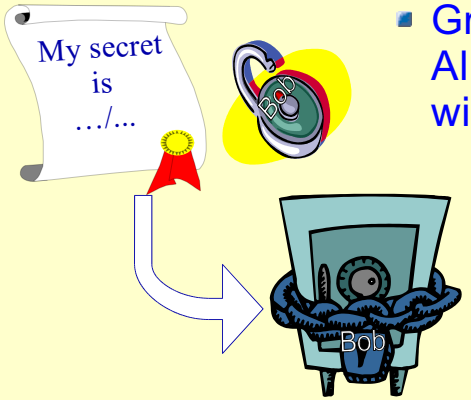
- Granted Bob's public key, Alice can lock the safe, with the message inside (*encrypt the message*)

Encryption / decryption attack



- Granted Bob's public key, Alice can lock the safe, with the message inside (*encrypt the message*)
- Alice sends the safe to Bob no one can unlock it (*impossible to break*)

Encryption / decryption attack



- Granted Bob's public key, Alice can lock the safe, with the message inside (*encrypt the message*)
- Excepted Bob, granted his private key (*Bob can decrypt*)
- Alice sends the safe to Bob no one can unlock it (*impossible to break*)

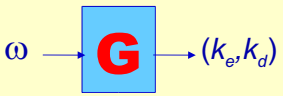
Provable Security

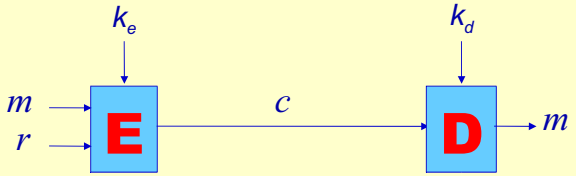
- Need of Computational Assumptions
- Provable Security
- Security Notions

Encryption Scheme

3 algorithms:

- **G** - key generation
- **E** - encryption
- **D** - decryption





Conditional Secrecy

The ciphertext comes from $c = \mathbf{E}_{k_e}(m; r)$

- The encryption key k_e is public
- A unique m satisfies the relation (with possibly several r)

At least exhaustive search on m and r can lead to m, maybe a better attack!

⇒ unconditional secrecy impossible

Algorithmic assumptions

Integer Factoring and RSA

One-Way Function

- Multiplication/Factorization:
  - $p, q \rightarrow n = p.q$ easy (quadratic)
  - $n = p.q \rightarrow p, q$ difficult (super-polynomial)

Integer Factoring and RSA

One-Way Function

- Multiplication/Factorization:
  - $p, q \rightarrow n = p.q$ easy (quadratic)
  - $n = p.q \rightarrow p, q$ difficult (super-polynomial)
- RSA Function, from $\mathbf{Z}_n$ in $\mathbf{Z}_n$ (with $n=pq$)

  for a fixed exponent e Rivest-Shamir-Adleman 1978
  - $x \rightarrow x^e \bmod n$ easy (cubic)
  - $y=x^e \bmod n \rightarrow x$ difficult (without p or q) RSA Problem

    $x = y^d \bmod n$ where $d = e^{-1} \bmod \varphi(n)$

Integer Factoring and RSA

One-Way Function

- Multiplication/Factorization:
  - $p, q \rightarrow n = p.q$ easy (quadratic)
  - $n = p.q \rightarrow p, q$ difficult (super-polynomial)
- RSA Function, from $\mathbf{Z}_n$ in $\mathbf{Z}_n$ (with $n=pq$)

  for a fixed exponent e Rivest-Shamir-Adleman 1978
  - $x \rightarrow x^e \bmod n$ easy (cubic)
  - $y=x^e \bmod n \rightarrow x$ difficult (without p or q)

    $x = y^d \bmod n$ where $d = e^{-1} \bmod \varphi(n)$

encryption

Integer Factoring and RSA

- Multiplication/Factorization:
  - $p, q \rightarrow n = p.q$ easy (quadratic)
  - $n = p.q \rightarrow p, q$ difficult (super-polynomial)

One-Way Function

- RSA Function, from $\mathbf{Z}_n$ in $\mathbf{Z}_n$ (with $n=pq$)

  for a fixed exponent e Rivest-Shamir-Adleman 1978
  - $x \rightarrow x^e \bmod n$ easy (cubic)
  - $y=x^e \bmod n \rightarrow x$ difficult (without p or q)

    $x = y^d \bmod n$ where $d = e^{-1} \bmod \varphi(n)$

difficult to break

Integer Factoring and RSA

- Multiplication/Factorization:
  - $p, q \rightarrow n = p.q$ easy (quadratic)
  - $n = p.q \rightarrow p, q$ difficult (super-polynomial)

One-Way Function

- RSA Function, from $\mathbf{Z}_n$ in $\mathbf{Z}_n$ (with $n=pq$)

  for a fixed exponent e Rivest-Shamir-Adleman 1978
  - $x \rightarrow x^e \bmod n$ easy (cubic)
  - $y=x^e \bmod n \rightarrow x$ difficult (without p or q)

    $x = y^d \bmod n$ where $d = e^{-1} \bmod \varphi(n)$

trapdoor key decryption

The RSA Problems

Let $n=pq$ where p and q are large primes

- The RSA problem: for a fixed exponent e

$$\mathrm{Succ}^{\mathrm{rsa}}_{n,e}(\mathcal{A}) = \Pr_{y \in \mathbf{Z}_n^*}\left[y = x^e \bmod n \,\middle|\, \mathcal{A}(y) = x \right]$$

- The Flexible RSA problem:

$$\mathrm{Succ}^{\text{fl-rsa}}_{n}(\mathcal{A}) = \Pr_{y \in \mathbf{Z}_n^*}\left[y = x^e \bmod n \,\middle|\, \mathcal{A}(y) = (x,e) \right]$$

with the restriction for e to be prime

The Discrete Logarithm

- Let $\mathbf{G} = (\langle g \rangle, \times)$ be any finite cyclic group
- For any $y \in \mathbf{G}$, one defines

$$\mathrm{Log}_g(y) = \min\{x \geq 0 \mid y = g^x\}$$

One-way function

- $x \rightarrow y = g^x$ easy (cubic)
- $y = g^x \rightarrow x$ difficult (super-polynomial)

$$\mathrm{Succ}^{\mathrm{dl}}_{g}(\mathcal{A}) = \Pr_{x \in \mathbf{Z}_q}\left[\mathcal{A}(y) = x \,\middle|\, y = g^x \right]$$

Any Trapdoor …?

- The Discrete Logarithm is difficult and no information could help!
- The Diffie-Hellman Problem (1976):
  - Given $A=g^a$ and $B=g^b$
  - Compute DH$(A,B) = C=g^{ab}$

Clearly CDH ≤ DL: with $a = \mathrm{Log}_g A$, $C = B^a$

$$\mathrm{Succ}_g^{\mathrm{cdh}}(\mathcal{A}) = \Pr_{a,b\in\mathbf{Z}_q}\left[\mathcal{A}(A,B) = C \mid A = g^a, B = g^b, C = g^{ab}\right]$$

Complexity Estimates

Estimates for integer factoring Lenstra-Verheul 2000

Modulus (bits)	Mips-Year ($\log_2$)	Operations (en $\log_2$)
512	13	58
1024	35	80
2048	66	111
4096	104	149
8192	156	201

Can be used for RSA too
Lower-bounds for DL in $\mathbf{Z}_p^*$

Provable Security

- Need of Computational Assumptions
- Provable Security
- Security Notions

Algorithmic Assumptions *necessary*

- $n=pq$: **public modulus**
- e : **public exponent**
- $d=e^{-1} \bmod \varphi(n)$: **private**

RSA Encryption

- $\mathbf{E}(m) = m^e \bmod n$
- $\mathbf{D}(c) = c^d \bmod n$

If the RSA problem is easy, secrecy is not satisfied: anybody may recover m from c

Algorithmic Assumptions *sufficient?*

Security proofs give the guarantee that the assumption is **enough** for secrecy:

- if an adversary can break the secrecy
- one can break the assumption

⇒ "reductionist" proof

Proof by Reduction

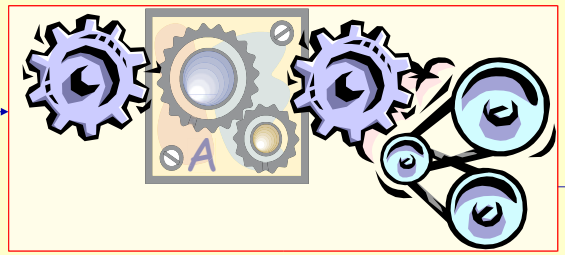
Reduction of a problem **P** to an attack *Atk*:

- Let *A* be an adversary that breaks the scheme
- Then *A* can be used to solve **P**

Proof by Reduction

Reduction of a problem **P** to an attack *Atk*:

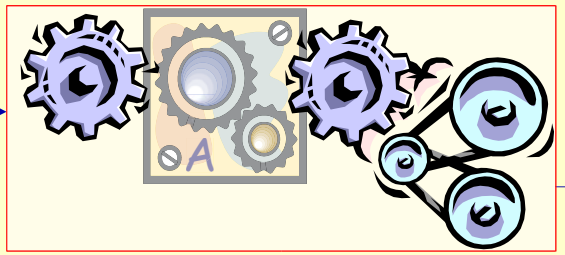
- Let *A* be an adversary that breaks the scheme
- Then *A* can be used to solve **P**

Instance **I** of **P** →  → Solution of **I**

Proof by Reduction

Reduction of a problem **P** to an attack *Atk*:

- Let *A* be an adversary that breaks the scheme
- Then *A* can be used to solve **P**

Instance **I** of **P** →  → Solution of **I**

P intractable ⇒ scheme unbreakable

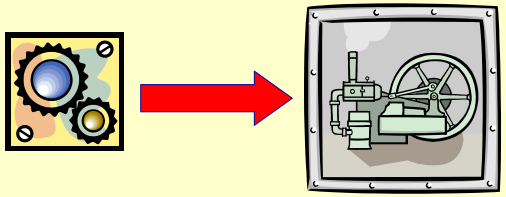
Provably Secure Scheme

To prove the security of a cryptographic scheme, one has to make precise

- the algorithmic assumptions
  - some have been presented
- the security notions to be guaranteed
  - depend on the scheme
- a reduction:
  an adversary can help
  to break the assumption

Practical Security

Adversary within t

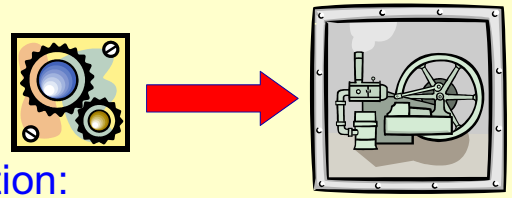


Algorithm against **P** within $t' = T(t)$

- Complexity theory: T polynomial
- Exact Security: T explicit
- Practical Security: T small (linear)

Complexity Theory

Adversary within t



Algorithm against **P** within $t' = T(t)$

- Assumption:
  - **P** is hard = no polynomial algorithm
- Reduction:
  - polynomial = T is a polynomial
- Security result:
  - no polynomial adversary

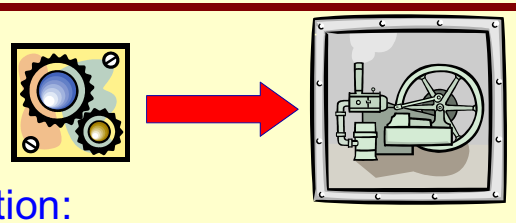
  ⇒ no attack for parameters **large enough**

Complexity Theory: Results

General results (under polynomial reductions, and against polynomial time adversaries):

- One-way functions are enough
  for secure signatures
- Trap-door one-way permutations are enough
  for secure encryption

Exact Security

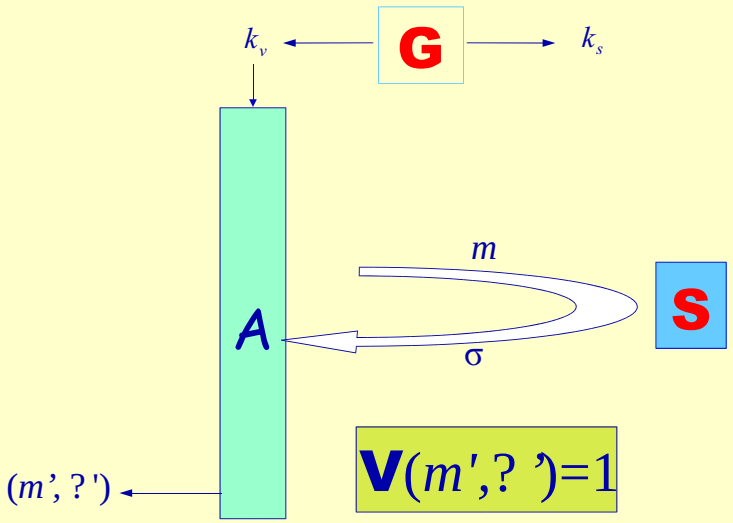


- Assumption:
  - Solving **P** requires N operations (or time τ)
- Reduction:
  - Exact cost for T,
    in t, and some other parameters
- Security result:
  - no adversary within time t such that $T(t) \leq \tau$

Provable Security

- Need of Computational Assumptions
- Provable Security
- Security Notions

Signature: EF-CMA



Exact Security: FDH

- Signature FDH (Bellare-Rogaway 1996):

$$\mathrm{Succ}^{ef-cma}(t) \leq (q_H + q_{\mathbf{S}} + 1) \times \mathrm{Succ}^{ow}_f(t + (q_H + q_{\mathbf{S}})T_f)$$

- Security bound: 2^{75}
  - and 2^{55} hash queries and 2^{30} signing queries
- Break the scheme within t, invert f within time
  $t' \leq (q_H + q_{\mathbf{S}} + 1)\,(t + (q_H + q_{\mathbf{S}})\,T_f) \leq 2^{110}\,T_f$
  - RSA: 1024 bits → 2^{130} (NFS: 2^{80}) ✘
    2048 bits → 2^{132} (NFS: 2^{111}) ✘
    4096 bits → 2^{134} (NFS: 2^{149}) ✔

Practical Security: FDH

- Signature FDH (Coron 2000):

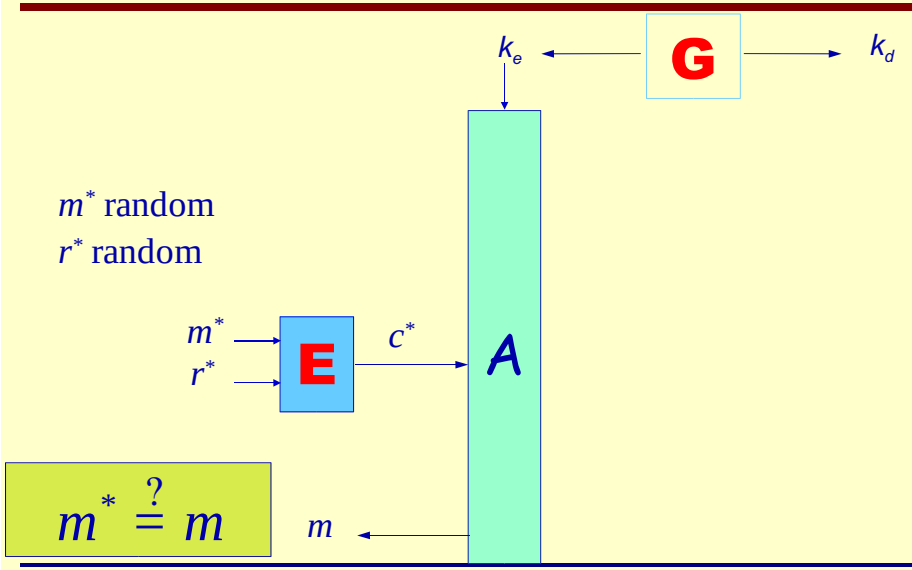
$$\mathrm{Succ}^{ef-cma}(t) \le \frac{q_{\mathbf{s}}+1}{e} \times \mathrm{Succ}_f^{ow}(t+(q_H+q_{\mathbf{s}}+1)T_f)$$

- Security bound: 2^{75}
  - and 2^{55} hash queries and 2^{30} signing queries
- Break the scheme within t, invert f within time

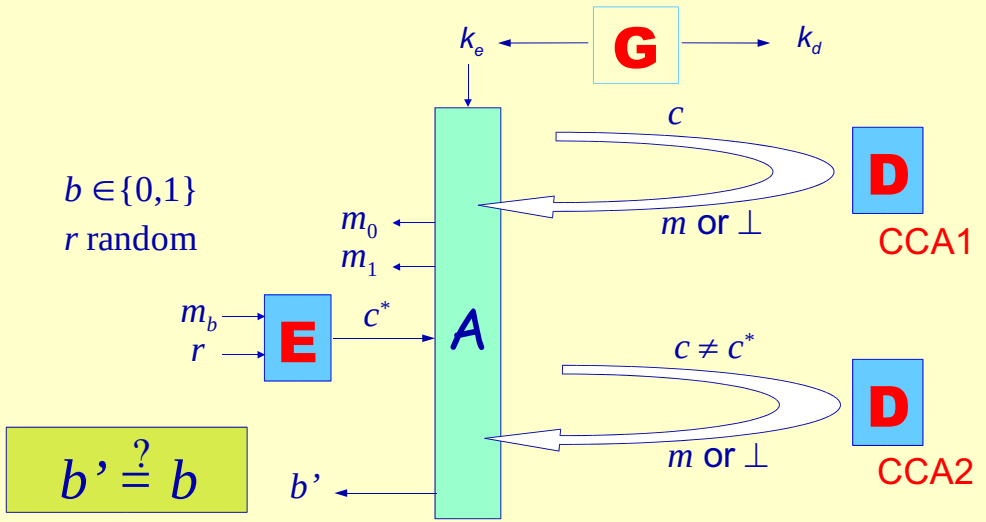
$$t' \le (q_{\mathbf{s}}+1)\,(t+(q_H+q_{\mathbf{s}}+1)\,T_f)/e \le 2^{30}\,t + 2^{85}\,T_f$$

  - RSA: 1024 bits → 2^{105} (NFS: 2^{80}) ✘
    2048 bits → 2^{107} (NFS: 2^{111}) ✔
    4096 bits → 2^{109} (NFS: 2^{149}) ✔

Encryption: One-Wayness



Encryption: IND-CCA2



Practical Security: Encryption

- Security bound: 2^{75}
  - and 2^{55} hash queries
- RSA-OAEP
  - 1024 bits → 2^{143} (NFS: 2^{80}) ✘
  - 2048 bits → 2^{146} (NFS: 2^{111}) ✘
  - 4096 bits → 2^{149} (NFS: 2^{149}) ✔
- RSA-REACT: $t' \approx 2t$
  - 1024 bits → 2^{76} (NFS: 2^{80}) ✔