La cryptographie et la Sécurité Concrète

CEA - Saclay
26 Mars 2007

David Pointcheval
CNRS-ENS, Paris, France

Summary

- Introduction to Cryptography
- Computational Assumptions
- Provable Security
- Example: Signature
- Example: Encryption

Summary

- **Introduction to Cryptography**
- Computational Assumptions
- Provable Security
- Example: Signature
- Example: Encryption

Cryptography: 3 Goals

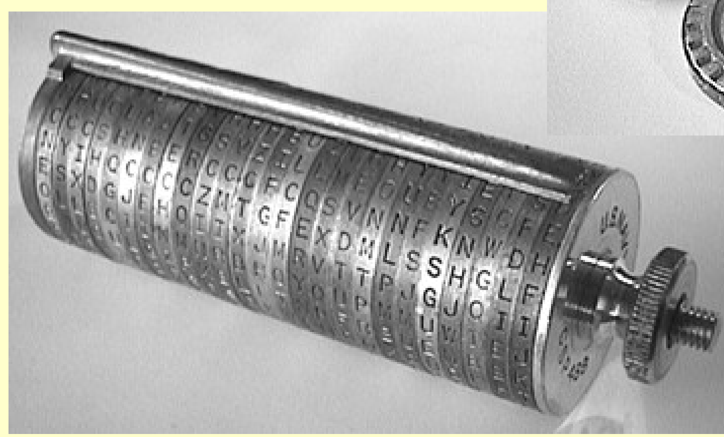
- Integrity:
  - Messages have not been altered
- Authenticity:
  - Message-sender relation
- Secrecy:
  - Message unknown to anybody else

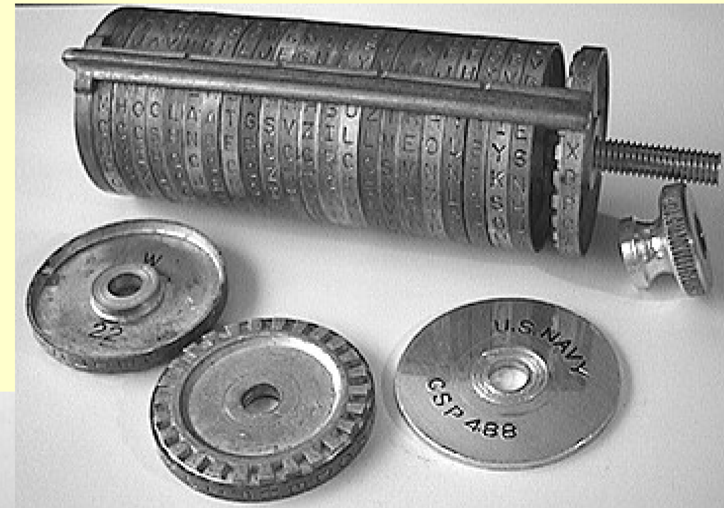
Cryptography: 3 Periods

- Ancient period: until 1918
- Technical period: from 1919 until 1975
- Paradoxical period : from 1976 until

Ancient Period

Substitutions and permutations





- Cipher disk
- Wheel cipher – M 94 (CSP 488)

Security = secrecy of the mechanisms

Technical Period

Cipher machines

Automatism
of permutations
and substitutions

but no proof
of better security!

- Enigma

Paradoxical Period

- Symmetric cryptography
- Asymmetric cryptography

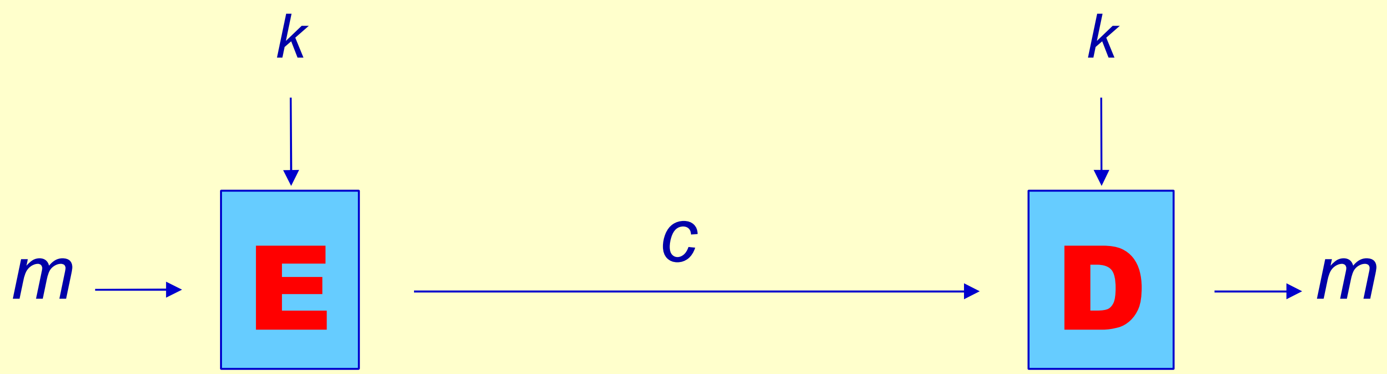
Security based on complexity assumptions

$$\mathcal{P} \neq \mathcal{NP}$$

Symmetric Encryption

One common secret

Encryption algorithm, **E** Decryption algorithm, **D**



Security = secrecy:
impossible to recover m from c only
(without the short secret k)

Asymmetric Cryptography

- Public parameter
- Short secret

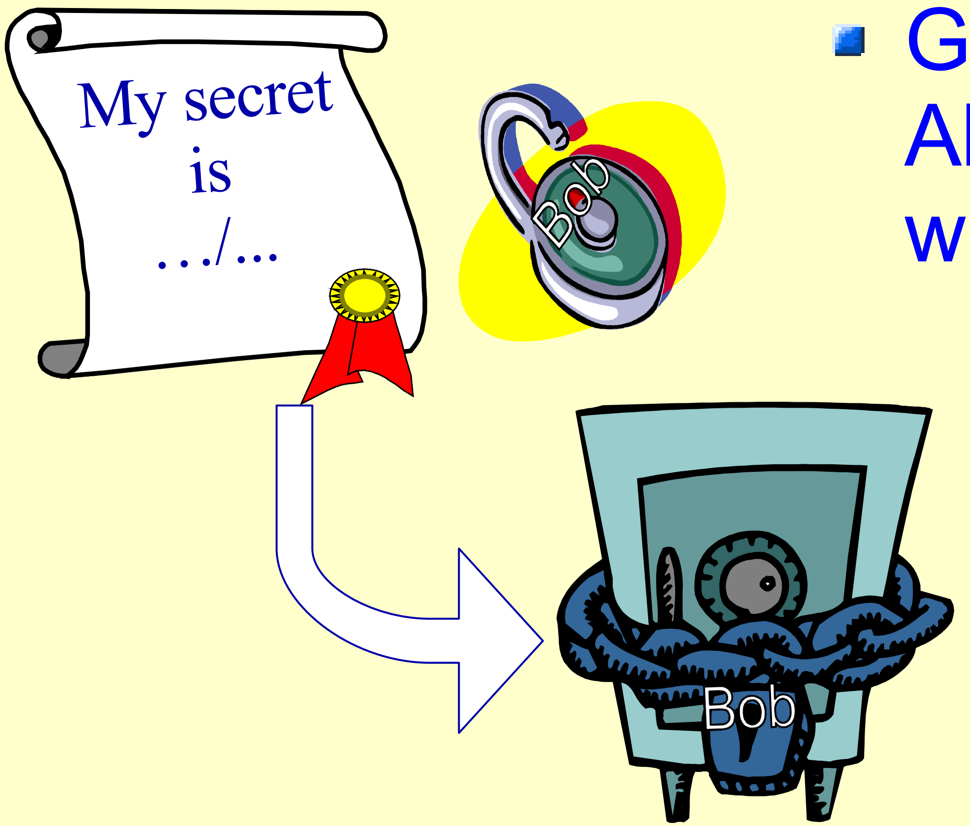
Alice ⟷ Bob (secrecy / authenticity)

Diffie-Hellman 1976

Asymmetric encryption:
Bob owns two “keys”

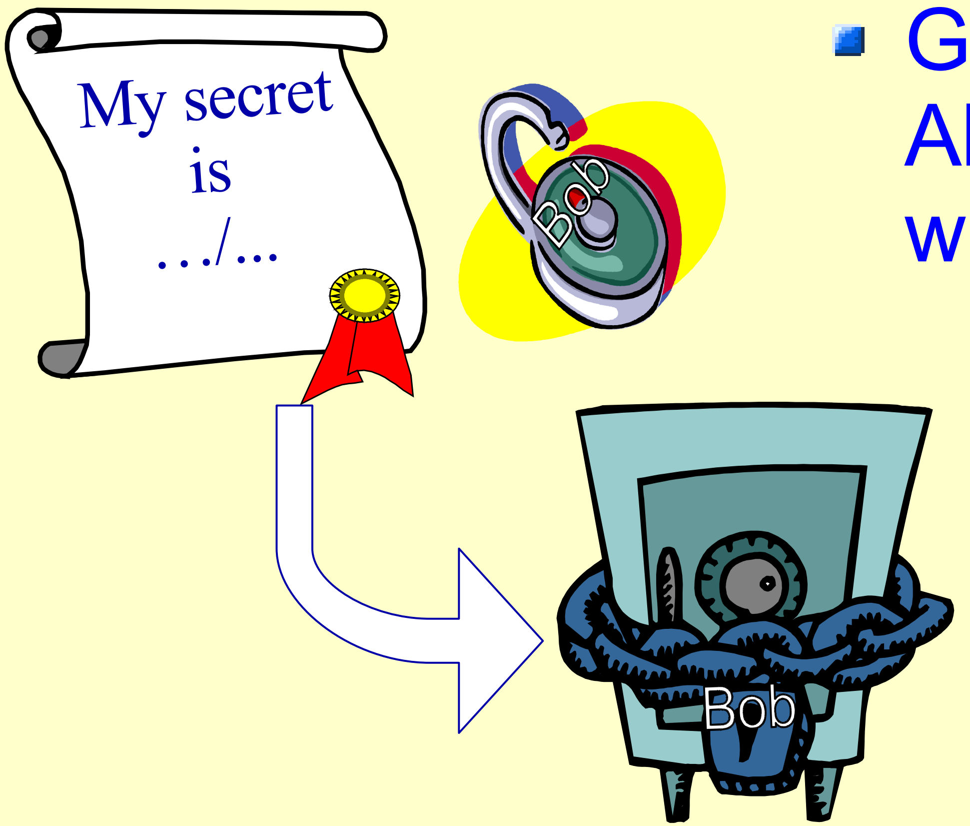
- A public key (encryption k_e) ⇒ known by everybody (including Alice)
  - so that anybody can encrypt a message for him
- A private key (decryption k_d) ⇒ known by Bob only
  - to help him to decrypt

Encryption / decryption attack



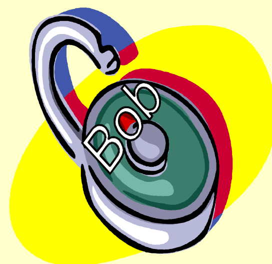
- Granted Bob's public key, Alice can lock the safe, with the message inside (*encrypt the message*)

Encryption / decryption attack

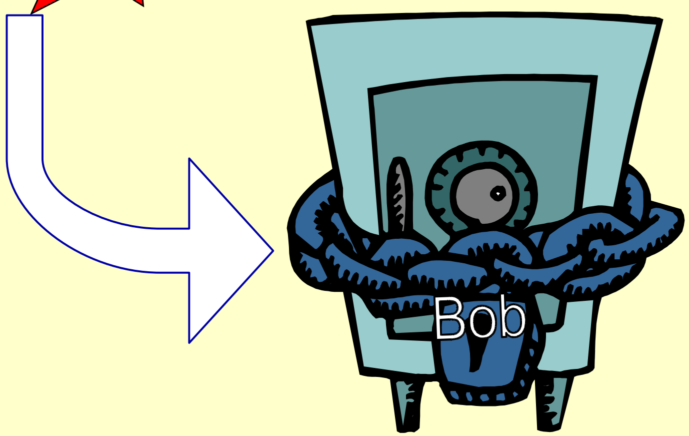


- Granted Bob's public key, Alice can lock the safe, with the message inside
  (*encrypt the message*)
- Alice sends the safe to Bob
  no one can unlock it
  (impossible to break)

Encryption / decryption attack



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



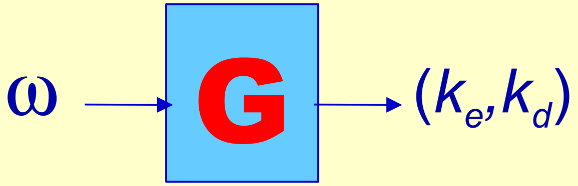
- Excepted Bob, granted his private key
  (Bob can decrypt)

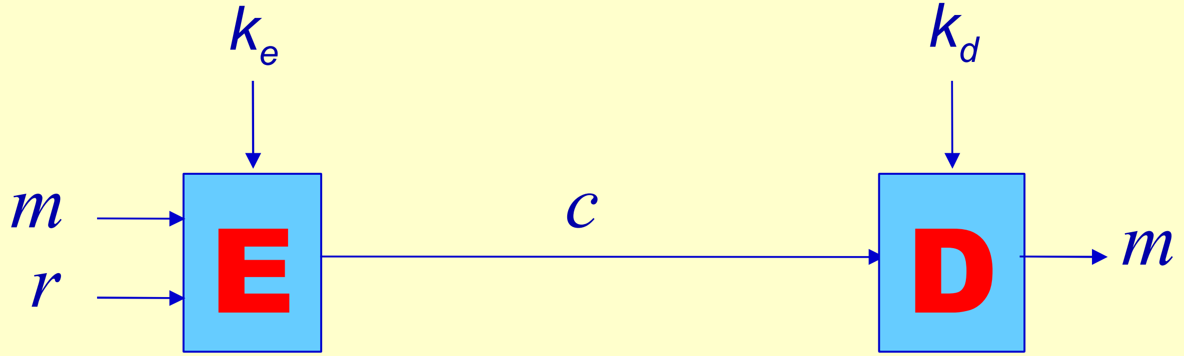
- Alice sends the safe to Bob no one can unlock it
  (impossible to break)

Asymmetric 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!

$\Rightarrow$ unconditional secrecy impossible

Algorithmic assumptions

Summary

- Introduction to Cryptography
- Computational Assumptions
- Provable Security
- Example: Signature
- Example: Encryption

Integer Factoring and RSA

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

One-Way Function

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 $\mathbb{Z}_n$ in $\mathbb{Z}_n$ (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 $d = e^{-1} \bmod \varphi(n)$

RSA Problem

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 $\mathbb{Z}_n$ in $\mathbb{Z}_n$ (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

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

hard
to break

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

trapdoor

key

decryption

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

Summary

- Introduction to Cryptography
- Computational Assumptions
- Provable Security
- Example: Signature
- Example: Encryption

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:

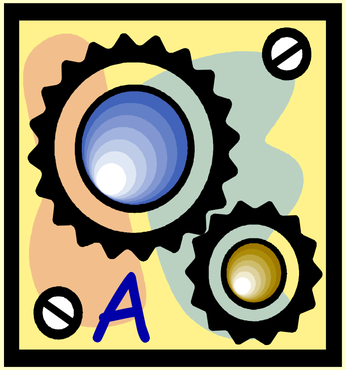
- if an adversary can break the secrecy
- one can break the assumption

$\Rightarrow$ “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**

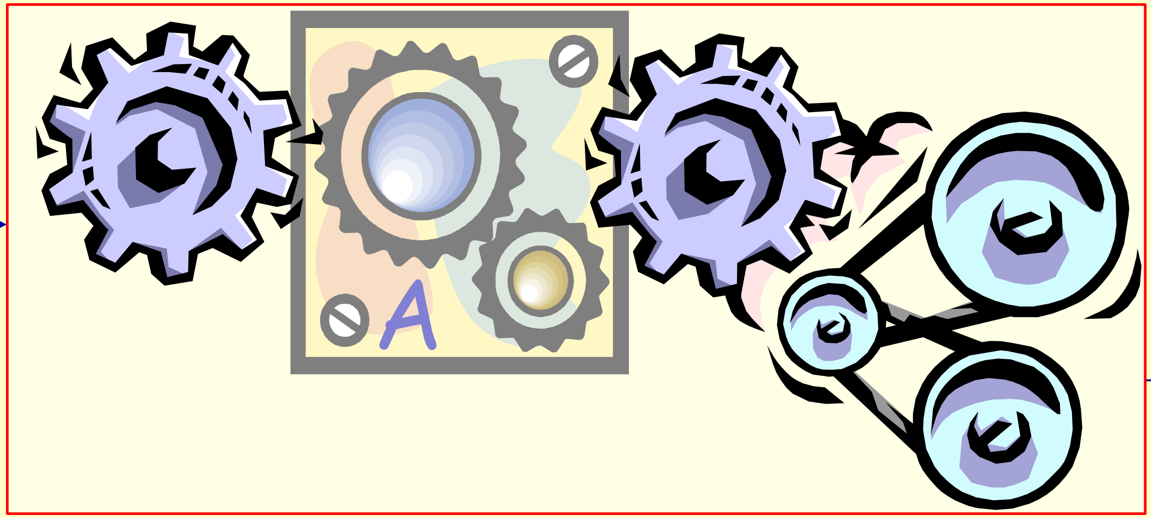


Proof by Reduction

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** →



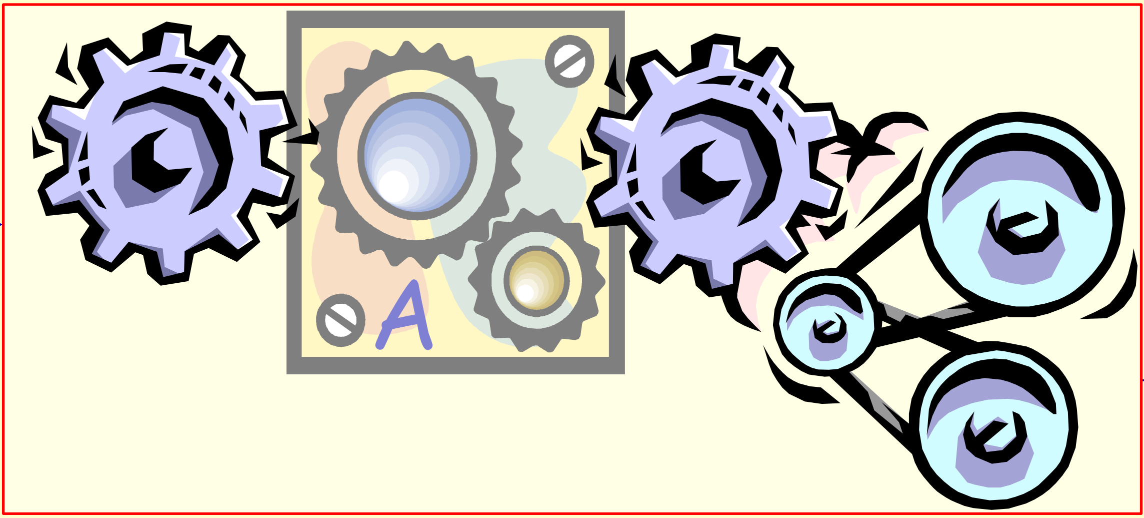
→ Solution of **I**

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

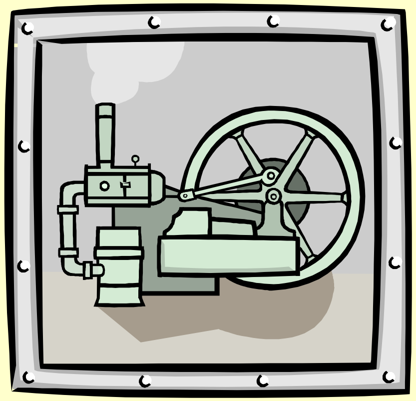
Provably Secure Scheme

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

- the algorithmic assumptions
  - such as the RSA intractability
- the security notions to be guaranteed
  - depend on the scheme (signature, encryption, etc)
- a reduction:
  an adversary can help
  to break the assumption

Practical Security

Adversary within t

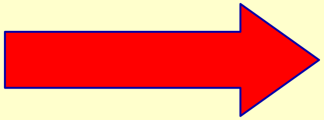


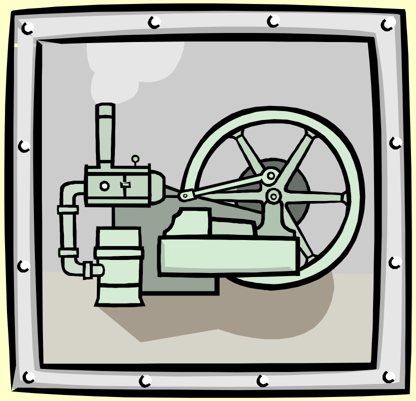
Algorithm against **P** within $t' = f(t)$

- Complexity theory: f polynomial
- Exact security: f explicit
- Practical security: f small (linear)

Complexity Theory

Adversary within t





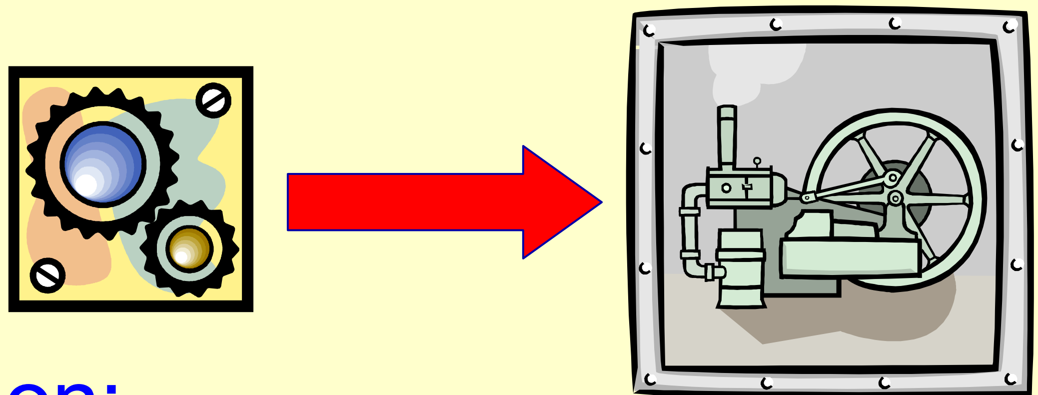
Algorithm against **P** within $t' = f(t)$

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

  ⇒ no attack for parameters **large enough**

Exact Security

Adversary within t



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

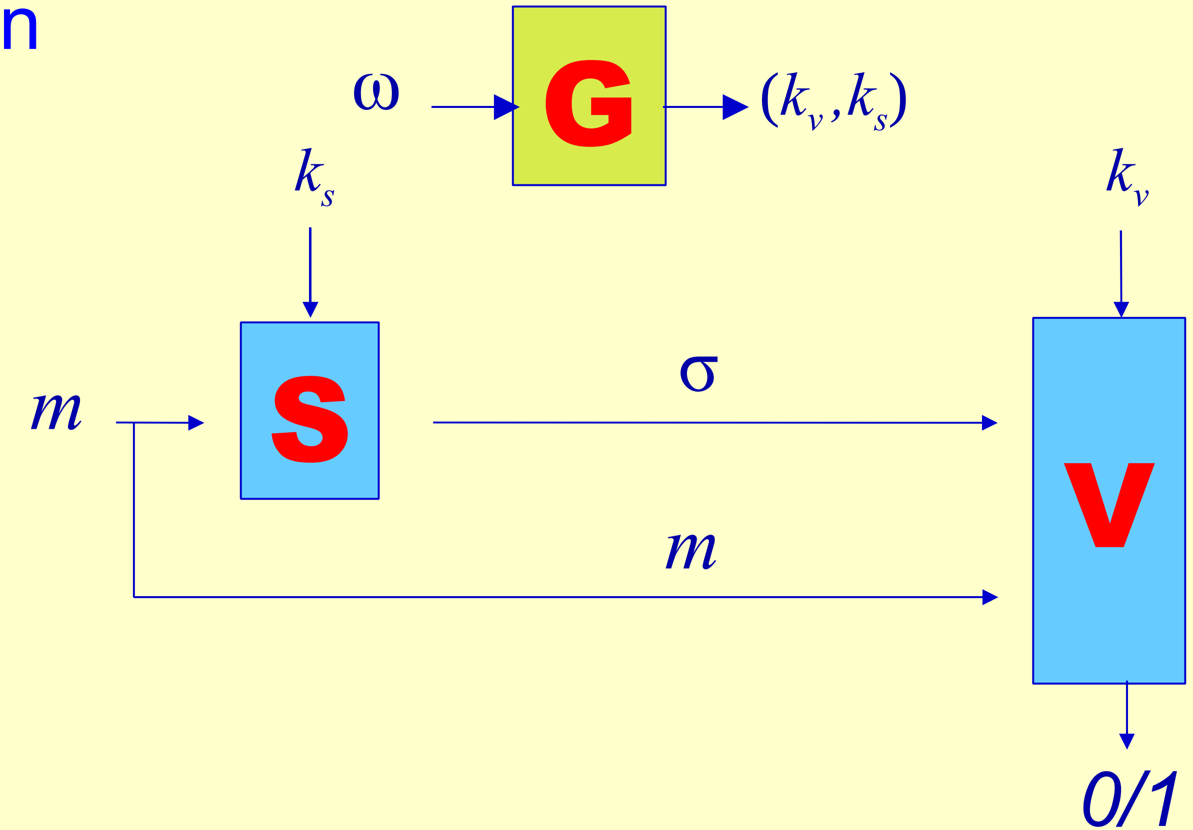
- Assumption:
  - Solving **P** requires N operations (or time τ)
- Reduction:
  - Exact cost for f,
    in t, and some other parameters
- Security result:
  - no adversary within time t such that $f(t) \leq \tau$

Summary

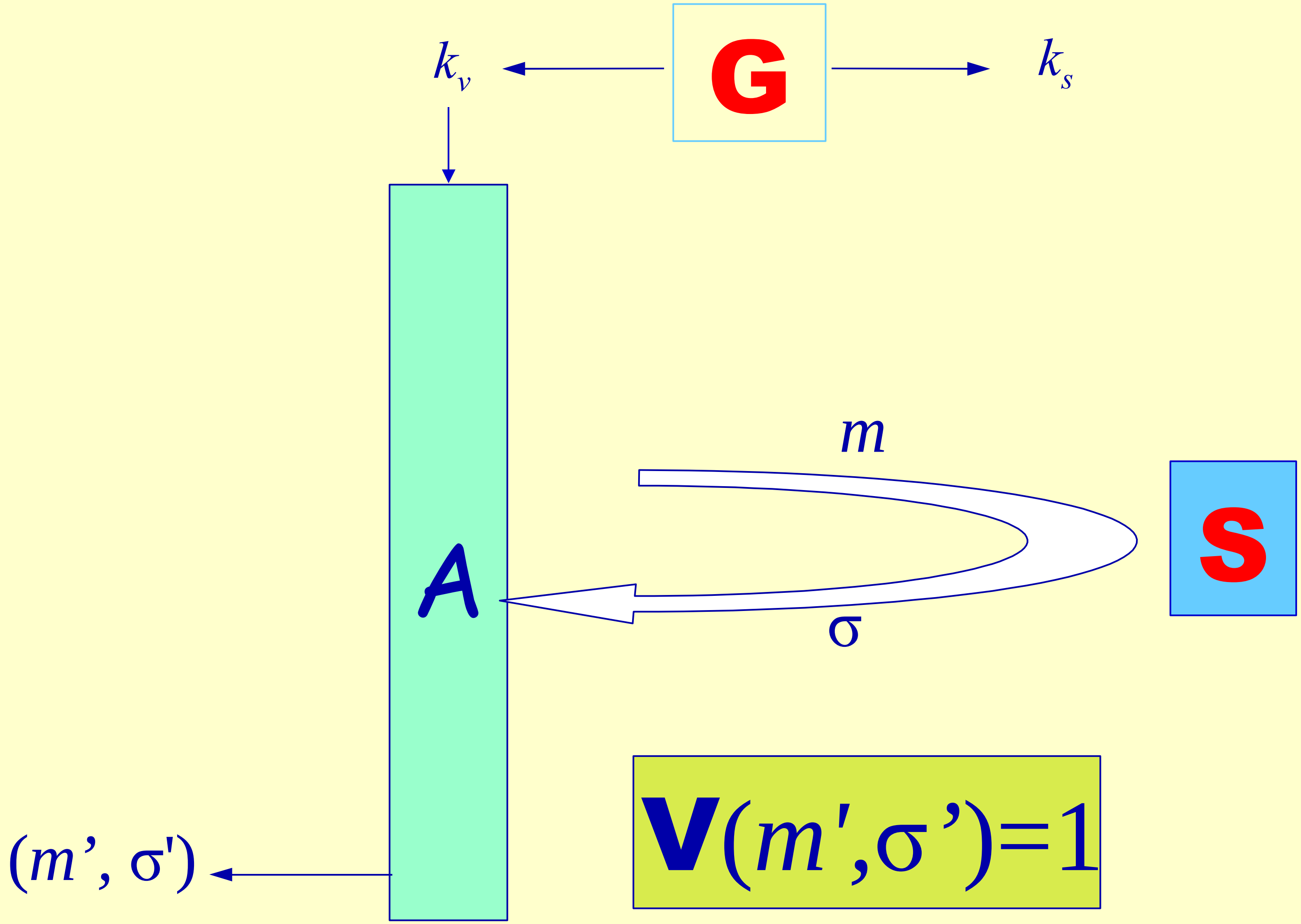
- Introduction to Cryptography
- Computational Assumptions
- Provable Security
- Example: Signature
- Example: Encryption

Signature

- A signature scheme $\mathcal{S}$ = (**G**,**S**,**V**) is defined by 3 algorithms:
  - **G** – key generation
  - **S** – signature
  - **V** – verification



Security: EF-CMA



RSA Signature

- $n = pq$, product of large primes
- e, relatively prime to $\varphi(n) = (p-1)(q-1)$
- n, e : **public** key
- $d = e^{-1} \bmod \varphi(n)$: **private** key

$$\sigma = \mathbf{S}(m) = (\ m\ )^d \bmod n$$

$$\mathbf{V}(m, \sigma) = [\ \sigma^e = \ m \ \bmod n\ ]$$

Existential Forgery = easy!

FDH-RSA Signature

- $n = pq$, product of large primes
- e, relatively prime to $\varphi(n) = (p-1)(q-1)$
- n, e : **public** key
- $d = e^{-1} \bmod \varphi(n)$: **private** key
- H : hash function onto $\mathbf{Z}_n$

$$\sigma = \mathbf{S}(m) = (\mathrm{H}(m))^d \bmod n$$

$$\mathbf{V}(m, \sigma) = [\ \sigma^e = \mathrm{H}(m) \bmod n\ ]$$

Existential Forgery = RSA Problem

FDH-RSA: Exact Security

- If one can forge a signature in expected time T, one can break the RSA problem in expected time $T' \leq (q_H + q_s + 1)\,(T + (q_H + q_s)\, T_{rsa})$
- Expected security level: 2^{75}
  - and 2^{55} hash queries and 2^{30} signing queries
- An efficient adversary leads to $T' \leq 2^{56}\,(t + 2^{55}\, T_{rsa}\,)$
- Contradiction: 1024 bits $\rightarrow 2^{131}$ (NFS: 2^{80}) ✘

  fixed exponent e 2048 bits $\rightarrow 2^{133}$ (NFS: 2^{111}) ✘

  T_{rsa} quadratic 4096 bits $\rightarrow 2^{135}$ (NFS: 2^{149}) ✔

RSA PKCS#1 Standard: RSA-PSS

- More intricate padding before applying the RSA function, proposed by Bellare-Rogaway – 1996

$$T' \leq T + (q_H + q_{\mathbf{s}})\, T_{\text{rsa}}$$

- Security bound: 2^{75}
  - and 2^{55} hash queries and 2^{30} signing queries

$$\Rightarrow T' \leq 2^{75} + 2^{56}\, T_{\text{rsa}}$$

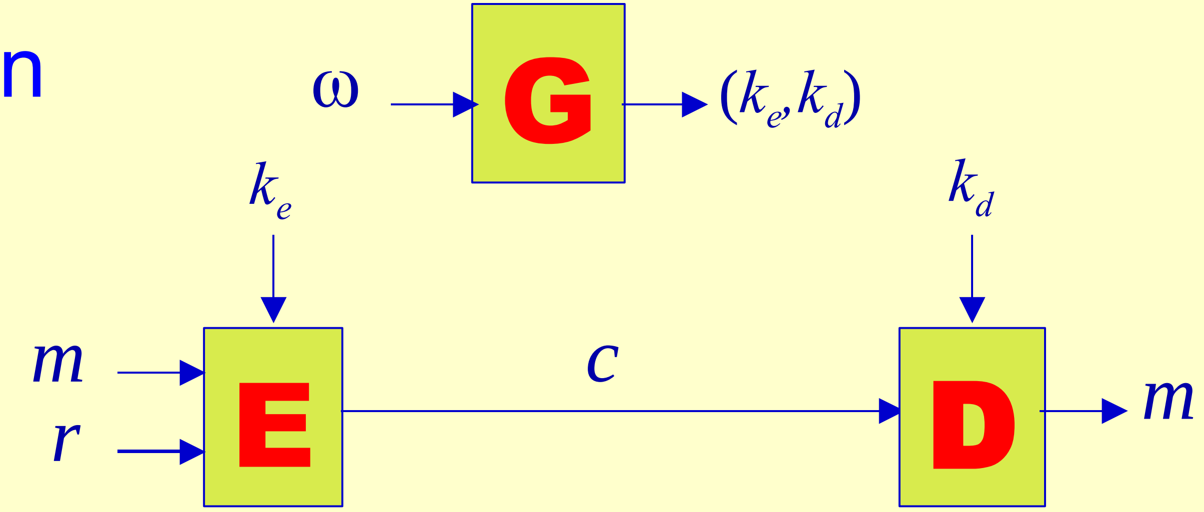
- Contradiction: 1024 bits → 2^{77} (NFS: 2^{80}) ✔
  2048 bits → 2^{79} (NFS: 2^{111}) ✔
  4096 bits → 2^{81} (NFS: 2^{149}) ✔

Summary

- Introduction to Cryptography
- Computational Assumptions
- Provable Security
- Example: Signature
- Example: Encryption

Asymmetric Encryption

- An asymmetric encryption scheme **π** = (**G**,**E**,**D**) is defined by 3 algorithms:
  - **G** – key generation
  - **E** – encryption
  - **D** – decryption

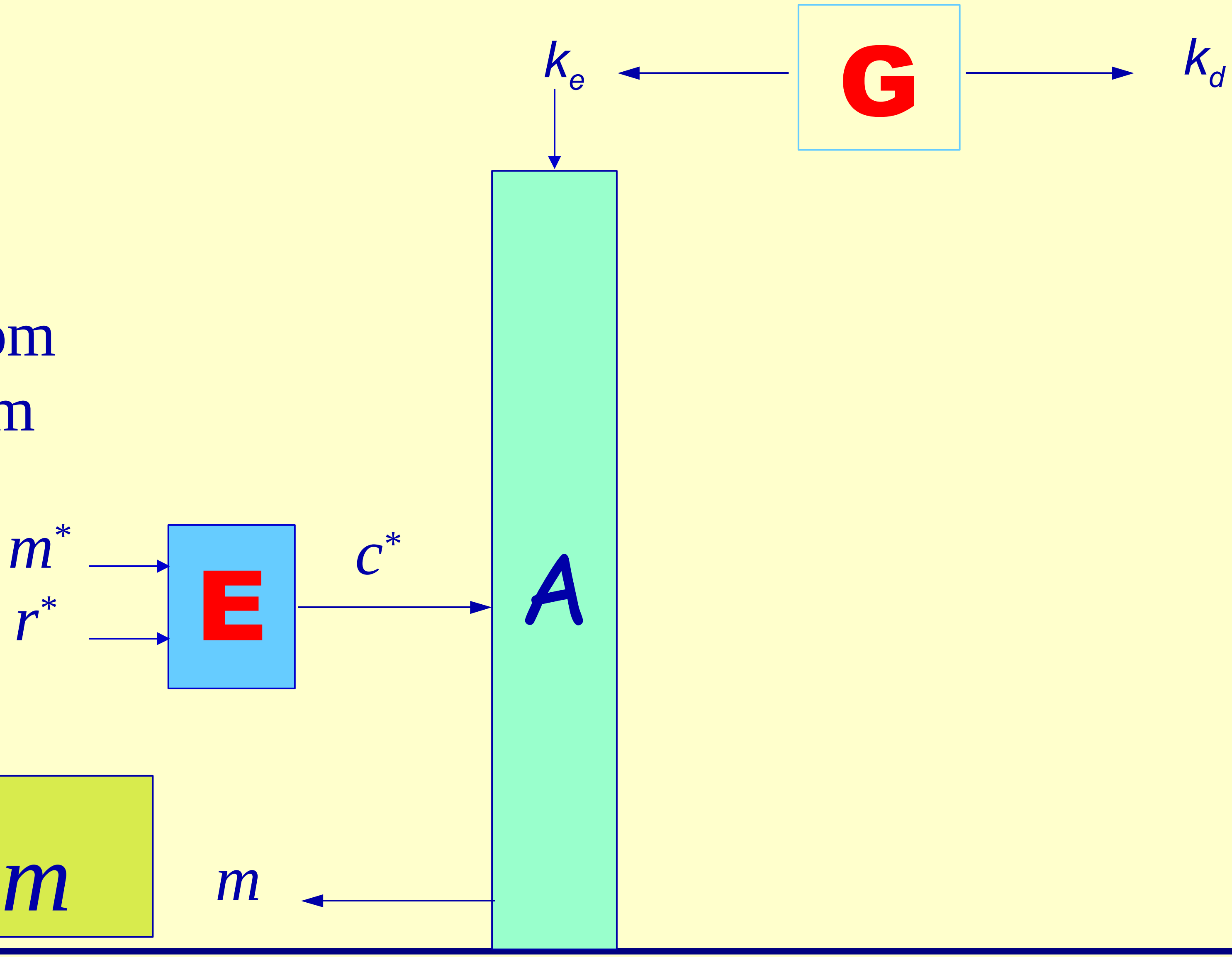


Security = secrecy : impossible
to recover m from public information
(i.e from c, but without k_d)

One-Wayness

m^* random

r^* random



$$m^* \stackrel{?}{=} m$$

Not Enough

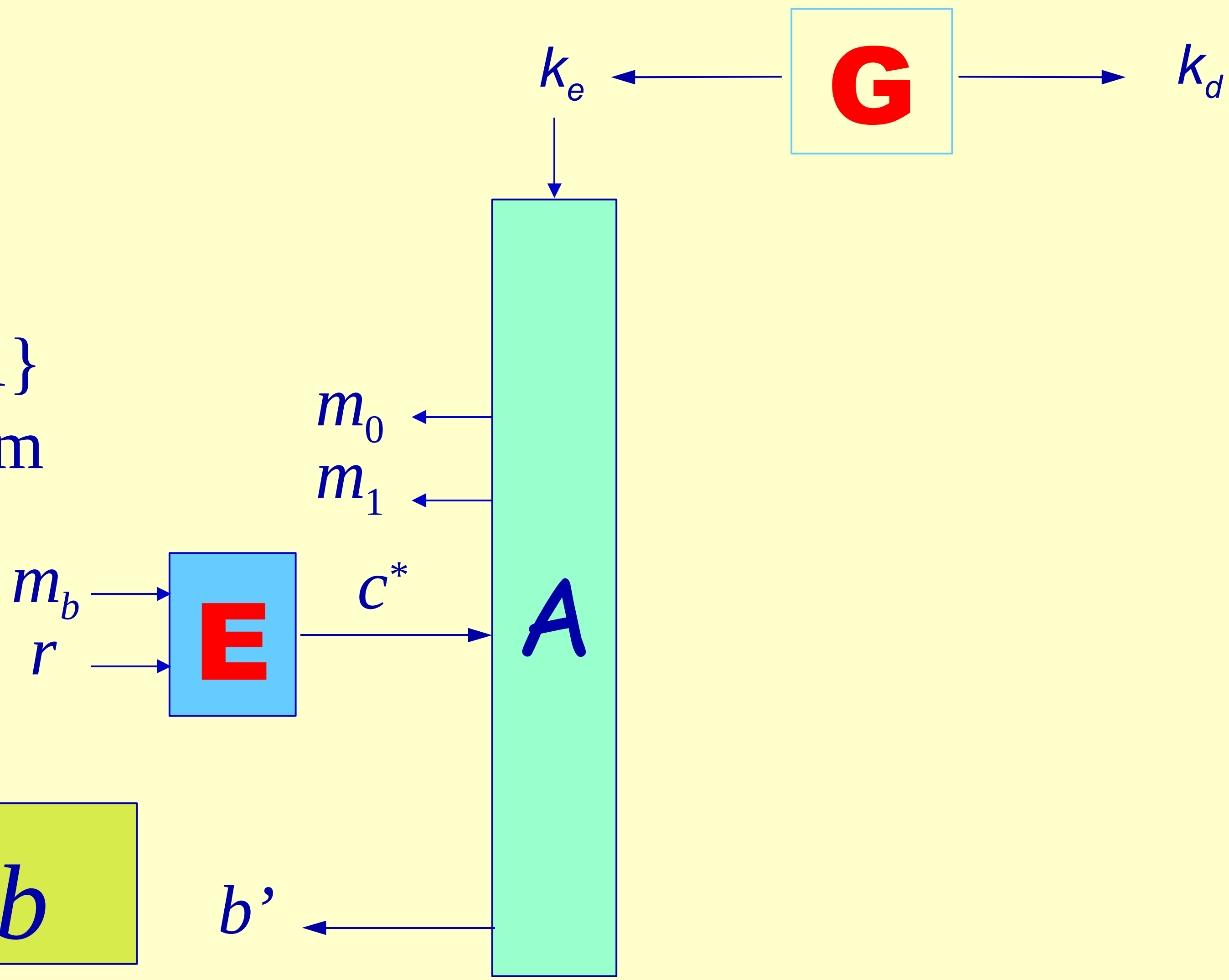
- One-Wayness (OW) :
  - without the private key, it is computationally impossible to recover the plaintext
  - but it does not exclude the possibility of recovering half of the plaintext!
- It is not enough if one already has some information about m:
  - “Subject: XXXXX”
  - “My answer is XXX” (XXX = Yes/No)

Even Worse

- Let g be a "one-way" function
- Let us define $f(x \| y) = x \| g(y)$
  - function f is also one-way
  - from $f(x \| y)$,
    one easily recovers most of the pre-image!

Semantic Security

$b \in \{0,1\}$
r random



$$b' \stackrel{?}{=} b$$

Basic Attacks

- Chosen-Plaintext Attacks (CPA)

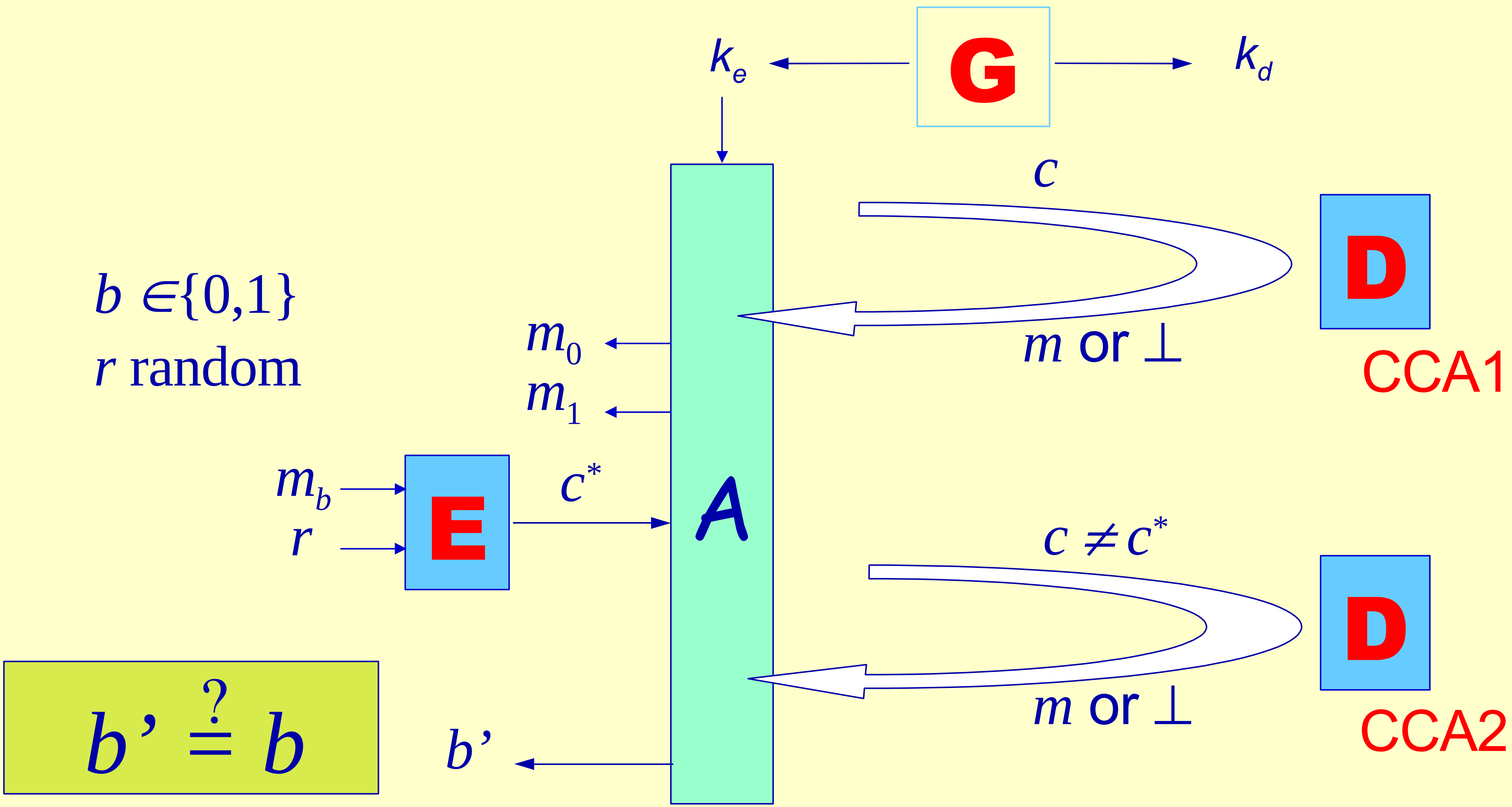
  In public-key cryptography setting,
  the adversary can encrypt any message
  of his choice, granted the public key

  $\Rightarrow$ the basic attack

Improved Attacks

- More information: oracle access
  - reaction attacks
    - oracle which answers, on c, whether the ciphertext c is valid or not
  - plaintext-checking attacks
    - oracle which answers, on a pair (m,c), whether the plaintext m is really encrypted in c or not (whether $m = \mathbf{D}(c)$)

IND-CCA2



k_e
G
k_d
c
D
m or $\perp$
CCA1
$b \in \{0,1\}$
r random
m_0
m_1
m_b
r
E
c^*
A
$c \neq c^*$
D
m or $\perp$
CCA2
$b' \stackrel{?}{=} b$
b'

Generic Construction Bellare-Rogaway '93

- Let f be a trapdoor one-way permutation then (with $G \to \{0,1\}^n$ and $H \to \{0,1\}^k$)
- **E**$(m;r) = f(r) \,\|\, m \oplus G(r) \,\|\, H(m,r)$
- **D**(a,b,c) :
  - $r = f^{-1}(a)$
  - $m = b \oplus G(r)$
  - $c = H(m,r)$?

Practical Security

$$\mathrm{Adv}^{ind}(\mathcal{A}) \leq 2q_D/2^k + q_H/2^n + \mathrm{Succ}^{ow}(t+(q_G+q_H)T_f)$$

- Security bound: 2^{75}
  - and 2^{55} hash queries and 2^{30} decryption queries
- Break the scheme within t, invert f within time $t' \leq t + (q_G + q_H)\, T_f \leq t + 2^{55}\, T_f$
  - RSA: 1024 bits → 2^{76} (NFS: 2^{80}) ✔
    2048 bits → 2^{78} (NFS: 2^{111}) ✔
    4096 bits → 2^{80} (NFS: 2^{149}) ✔

Conclusion

With provable security, one can prove that a cryptographic scheme actually achieves a specific security level

- Under well-defined computational assumptions
- In a precise (communication) security model
  - Side-channel attacks are not considered