Public-Key Cryptography

Definition

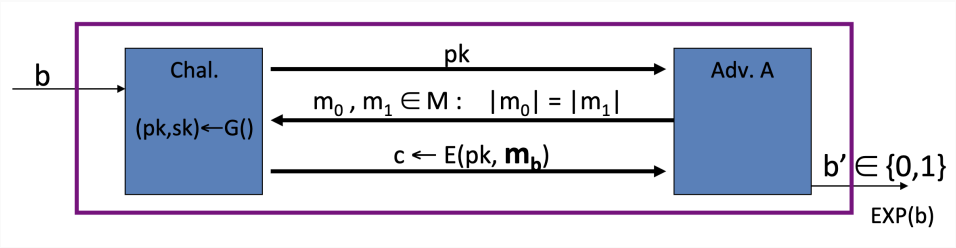
Definition: a public-key encryption scheme consists of three algorithms: G, E, D such that:

- G(): a randomized algorithm that outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
- E(pk, m): a randomized algorithm that takes as input the public key pk and a message m, and outputs a ciphertext c.
- D(sk, c): a deterministic algorithm that takes as input the secret key sk and a ciphertext c, and outputs a message m.

Correctness: $\forall(\mathsf{sk}, \mathsf{pk})$ output by G: $D(\mathsf{sk}, E(\mathsf{pk}, m)) = m$.

Security against Passive Attacks

For $b \in \{0,1\}$, define experiments EXP(0) and EXP(1) as:

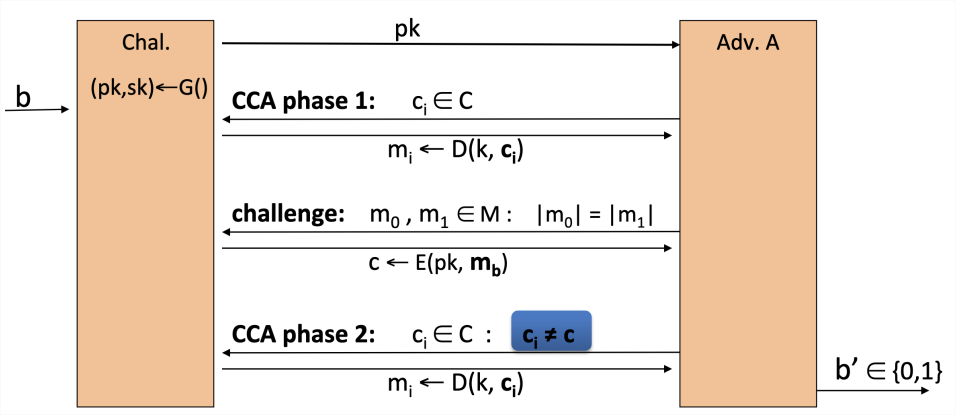


Def: (G, E, D) is `semantically secure` (i.e., IND-CPA) if for all efficient adversaries $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}(n) = |\Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1]| \leq \mathrm{negl}(n)$.

Corollary: IND-CPA public-key encryption must be `randomized`.

Security against Active Attacks

For $b \in \{0,1\}$, define experiments EXP(0) and EXP(1) as:



Security against Active Attacks (cont.)

Def (G, E, D) is `IND-CCA secure` if for all efficient adversaries $\mathcal{A}$,
$\mathsf{Adv}_{\mathcal{A}}(n) = |\Pr[\mathsf{EXP}(0) = 1] - \Pr[\mathsf{EXP}(1) = 1]| \leq \mathsf{negl}(n)$.

Theorem: IND-CCA security implies IND-CPA security. But IND-CPA security `does not` imply IND-CCA security.

IND-CCA security is the standard security requirements for modern public-key cryptosystems.

Textbook RSA Encryption

- G():
  - choose random prime numbers p, q. Set $N = pq$.
  - choose integers e, d such that $e \cdot d = 1 \mod \varphi(N)$.
  - output $\mathsf{pk} = (N, e)$ and $\mathsf{sk} = (N, d)$.
- $\mathsf{E}(\mathsf{pk}, m)$: return m^e in $\mathbb{Z}_N$.
- $\mathsf{D}(\mathsf{sk}, c)$: return c^d in $\mathbb{Z}_N$.

Correctness: For a ciphertext $c = m^e$, we have
$c^d = (m^e)^d = m^{ed} = m^{k \cdot \varphi(N)+1} = (m^{k \cdot \varphi(N)}) \cdot m = m$.

Key Length: For security, p, q are usually of size 2048 or 3072 bits. e is usually $65537 = 2^{16} + 1$. See keylength.com.

Note: This is a textbook RSA scheme, and it is totally `insecure` (not even semantically secure) and many attacks exist.

Hard Problems for RSA

- Know (p, q), compute $p \cdot q$ is easy.
- Know $n = p \cdot q$, find p and q is hard.

Best algorithm for factorization: $\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$.

RSA function Compute the e-th root:

- $x \to x^e \mod n$ is easy
- $y = x^e \to x \mod n$ is hard.

With the trapdoor $d = e^{-1} \mod \varphi(n)$, $y^d = x^{ed} = x \mod n$.

Theorem. RSA se réduit à la factorisation !

Textbook RSA Encryption

Textbook RSA encryption is `deterministic`.

Question. Is textbook RSA encryption IND-CPA secure?

A simple attack. Let $k \in \{0,1\}^{64}$ be a random session-key Alice and Bob want to exchange, and Alice is going to use Bob's public key N, e to encrypt k. The attacker sees $c = k^e \mod N$.

Expected security: 2^{64} (as long as k is hidden). We will show an attack with complexity less than 2^{64}.

If $k = k_1 \cdot k_2$ where $k_1, k_2 < 2^{34}$ (with probability $\approx 20\%$), then $c/k_1^e = k_2^e \mod N$.

Perform meet-in-the-attack:

- Build a table: $c/1^e, c/2^e, \ldots, c/(2^{34})^e$. Take 2^{34} operations.
- For each $k_2 \in [0, \ldots, 2^{34} - 1]$, test if k_2 is in the table. Take 2^{34} operations.
- Output the matching k_1, k_2.

In total, the attack times $2^{35} \ll 2^{64}$.

Chosen-ciphertext attack on textbook RSA

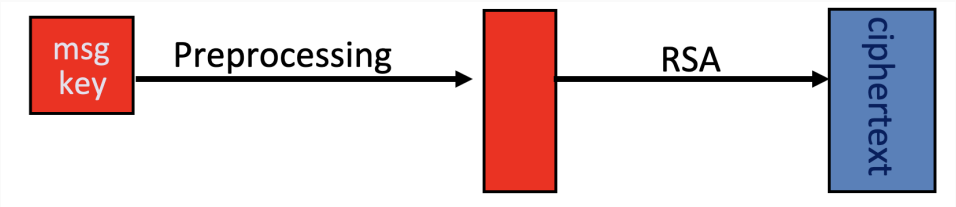
- Let (e, N) an RSA public key and d the private key. Let $c = m^e$ a ciphertext of a unknown message m.
- Given c, prove that if we can ask for a (single) decryption of $c' \neq c$, then we can recover m.

Solution

RSA in Practice

Never use textbook RSA.

RSA in practice:



Main questions:

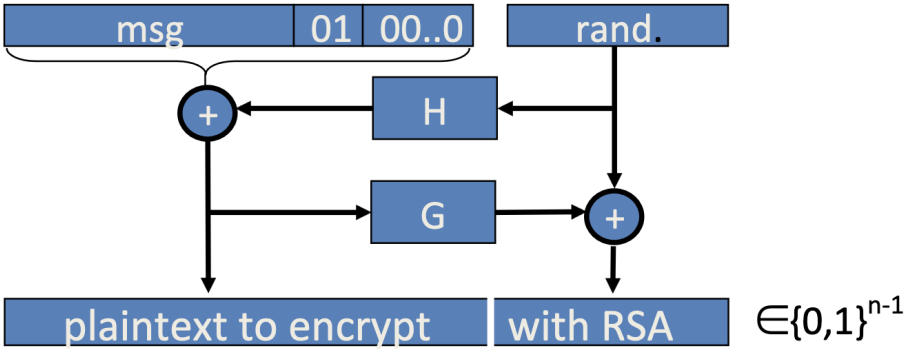
- How should the preprocessing be done?
- Security of resulting system?

RSA in Practice: PKCS1

History

- 1977: RSA invented.
- March 1991: version 1.1 - 1.3 (private).
- June 1994: version 1.4 (public).
- November 1993: version 1.5. Attacked by Bleichenbacher in 1998.
- September 1998: version 2.0 - OAEP [BR94].
- 2001: proof of security for OAEP [FOPS01].

RSA-PKCS1 v2: OAEP



Theorem [FOPS01]. RSA-OAEP is CCA secure when H,G are random oracles.

In practice, H, G are usually SHA-256.

Attacking RSA implementations

- Side-channel attacks:
  - Timing attacks: time to compute c^d can expose d.
  - Power attacks: the power consumption of a smartcard while it is computing c^d can expose d.
  - Faults attacks: a computer error during c^d can expose d.
- RSA Key Generation

How NOT to improve RSA's performance

RSA encryption and decryption performance are decided by the number of exponentiation operations.

- To speed up RSA decryption use small private key d.
  - Wiener's attack: if $d < N^{0.292}$ then RSA is insecure.
- To speed up RSA encryption use small public key e.
  - Broadcast attack: e.g., $e = 3$.

Question.

- Why is everyone so obsessed with multiplying two primes for RSA. Why not just use one?
- Using one prime factor was definitely a bad idea so I'll try using over 30 instead. What can be wrong?
  - The Elliptic Curve Factorization Method: the fastest way to factor a known composite integer if one of the factors is relatively small.

Conclusion. Implementing RSA is hard. Do not implement it yourself. Use Tink.

Broadcast Attack with e = 3

Let $n_1 = 3 * 5, n_2 = 7 * 11, n_3 = 13 * 17$. Let m be a message and we know that encryption of m with same $e = 3$ under different modulus are:

- $m^3 = 2 \mod n_1$
- $m^3 = 9 \mod n_2$
- $m^3 = 15 \mod n_3$

Find m.

Solution.

Key Management Problem

Problem: how to exchange messages between n users in the symmetric-key setting?

Each user has to store n keys. Storing mutual secret keys is difficult.

Question can we generate shared keys over public channel?

Starting point of cryptography:

- Diffie-Hellman - 1976
- RSA - 1977
- More advanced public-key primitives.

Question design a key-exchange protocol against passive attackers using public-key encryption.

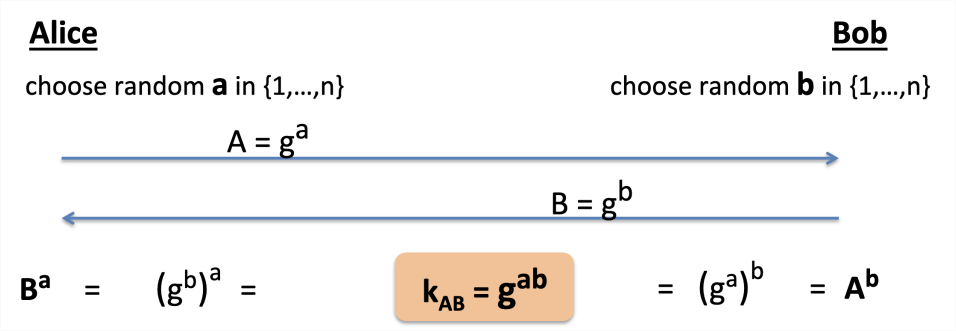
Diffie-Hellman Key Exchange

Goal: Alice and Bob want shared secret, unknown to eavesdropper.

Security: eavesdropping only (no tampering).

Diffie-Hellman key-exchange protocol.

Fix a finite cyclic group $G = \mathbb{Z}_p^*$ of order n with a generator g.



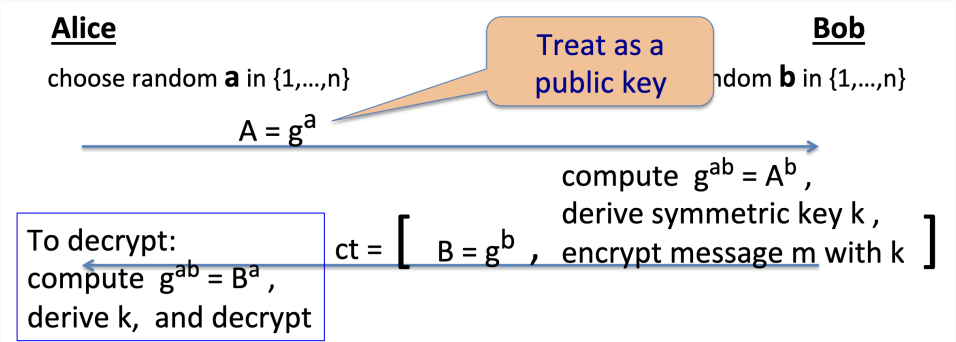
Diffie-Hellman - Hard Problems

Let $\mathbb{G}$ be a multiplicative cyclic group with generator g.

- Computational Diffie-Hellman (CDH): Given g, $A = g^a, B = g^b$. Compute $C = CDH(A, B) = g^{ab}$.
- Decision Diffie-Hellman (DDH): Given $g, A = g^a, B = g^b$ and $C = g^c$ in $\mathbb{G}$. Decide if $C = g^{ab}$.
- If CDH is easy, Diffie-Hellman is broken: the attacker can re-compute the shared key.
- If DDH is easy, Diffie-Hellman is broken: the attacker can distinguish a random key and a DH key.

ElGamal Encryption

Can we convert Diffie-Hellman protocol into an encryption scheme?



ElGamal Encryption (cont.)

- G():
  - Choose a finite cyclic group $G = \mathbb{Z}_p^*$ of order n with a generator g.
  - Choose a random element $s \in G$. Output $\mathsf{pk} = g^s, \mathsf{sk} = s$.
- E(pk, m): Choose a random element $e \in G$, return $c = (g^r, \mathsf{pk}^r \cdot m)$.
- D(sk, c): Parse $c = (c_1, c_2)$. Return c_2 / c_1^{sk}.

Correctness: We have $\frac{c_2}{c_1^{\mathsf{sk}}} = \frac{(g^s)^r \cdot m}{(g^r)^s} = \frac{g^{sr} \cdot m}{g^{rs}} = m$.

Key Length: See keylength.com.

Chosen-ciphertext attack on ElGamal

- Let g^s an ElGamal public key and s the private key. Let $c = (c_1 = g^r, c_2 = g^{rs} \cdot m)$ a ciphertext of a unknown message m.
- Given c, prove that if we can ask for a (single) decryption of $c' \neq c$, then we can recover m.

Solution

More attacks on ElGamal

If we encrypt two different messages with the same randomness in ElGamal encryption, what happens?

- Let $(c_1, d_1 = g^r, y^r \cdot m_1)$ an encryption of m_1 with randomness r.
- Let $(c_2, d_2 = g^r, y^r \cdot m_2)$ an encryption of m_2 with randomness r.

Can we compute m_1/m_2?

Hard Problems for ElGamal

Definition. Discrete Logarithm Problem (DLOG)

- Let $\mathbb{G}$ a multiplicative group, and $g \in \mathbb{G}$, given $(g, y \in \langle g \rangle)$, compute $\log_g(y) = x$ where $g^x = y$.

Theorem. ElGamal se réduit au logarithme discret !

Hybrid Encryption

- Symmetric-key is fast, but requires pre-shared key.
- Asymmetric-key is slow, but do not requires pre-shared key.

Can we achieve the best of both worlds?

Answer: yes.

The Hybrid Encryption primitive combines the efficiency of symmetric encryption with the convenience of public key (asymmetric) cryptography. Anyone can encrypt data using the public key, but only users with the private key can decrypt the data.

- For Hybrid Encryption, the sender generates a fresh symmetric key to encrypt the plaintext of each message to produce a ciphertext. That symmetric key is encapsulated with the recipient's public key.
- For Hybrid Decryption, the symmetric key is decapsulated by the recipient and then used to decrypt the ciphertext to recover the original plaintext.

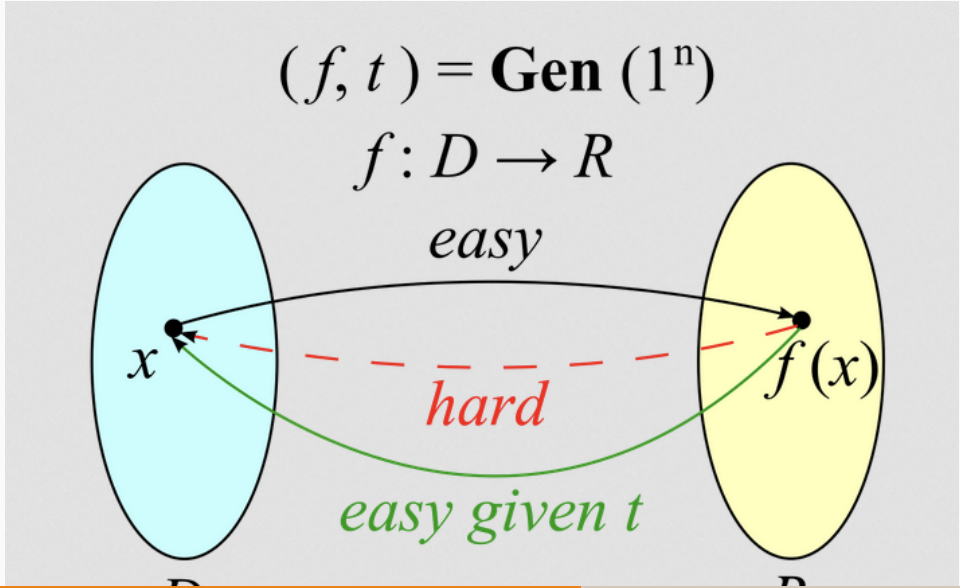
Hybrid Encryption

Let PKE, SE be a public-key encryption and a symmetric-key encryption, respectively.

Hybrid Encryption:

- G():
  - Generate and output a key pair (pk, sk) using PKE.
- E(pk, m):
  - Generate a random key k for SE.
  - Encrypt m using SE with key k.
  - Encrypt k using PKE with pk.
  - Output two ciphertexts.
- D(sk, c):
  - Decrypt the second ciphertext to obtain a secret key k using PKE with sk.
  - Decrypt the first ciphertext to obtain the message m using SE with decrypted k.

Trapdoor Function



Hard Problems

- Let $\mathbb{G}$ a multiplicative cyclic group with generator g. Recall DLOG, CDH, DDH

Theorem. DLOG $<$ CDH $<$ DDH.

Proof

- DLOG $<$ CDH: Given $g, A = g^a, B = g^b$. Solve DLOG$(B) = b$. Compute $A^b = g^{ab}$.
- CDH $<$ DDH: Given $g, A = g^a, B = g^b, C = g^c$. Solve CDH$(A, B) = g^{ab}$. Compare C and g^{ab}.

Security Models

To analyze the security of a cryptosystem, we need:

- Specify the goal of the attacker.
- Specify the means of the attacker: computational power, access to extra information, etc
- Examine what the chances are for an attacker to achieve its goal with the specified means
  - Success probability
  - Security reduction to a hard problem
- Conclude with the choice of parameters.

The adversary

- As smart as possible
- Can do whatever he wants ...
- ... as long as it satisfies the constraints (e.g., his computational power):
  - We do not consider algorithm running in 2^{60} years.
  - Can only perform at most 2^{60} operations.

Recall: the adversary is modeled as a Turing machine:

- has access to a source of randomness
- can only run in polynomial time in the size of the secret key.

The adversary

The adversary $\mathcal{A}$ is a probabilistic and polynomial-time Turing machine.

- For security, we require the security holds for all adversaries $\mathcal{A}$.
- For cryptanalysis, we only need to show there exists an adversary $\mathcal{A}$.

Proof of security

Principle. Prove by reduction.

- Step 1: `Define an assumption:` a hard problem P is a problem with no polynomial-time solver. Example: DLOG, CDH, DDH, RSA.
- Step 2: `Reduction:` if $\mathcal{A}$ is an adversary breaking the cryptosystem, we can use $\mathcal{A}$ to construct a polynomial time adversary $\mathcal{B}$ for hard problem P.
- Step 3: `Conclude:` There exists no such $\mathcal{A}$.

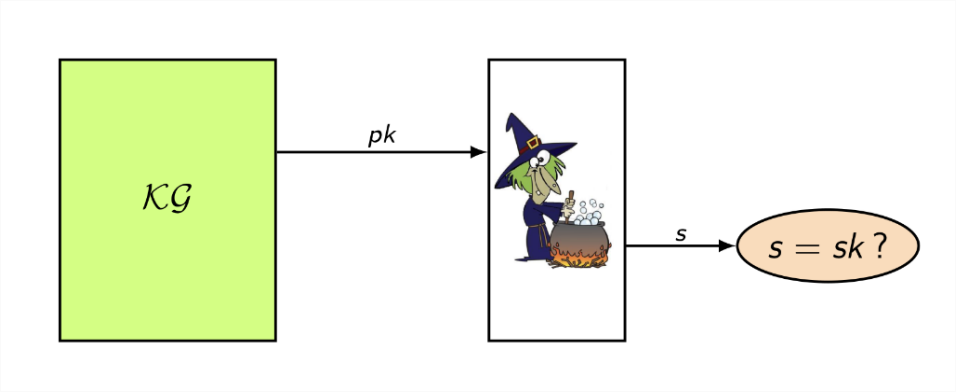
To define security for encryption:

- What are the goals of the adversary?
- What are the means of the adversary?

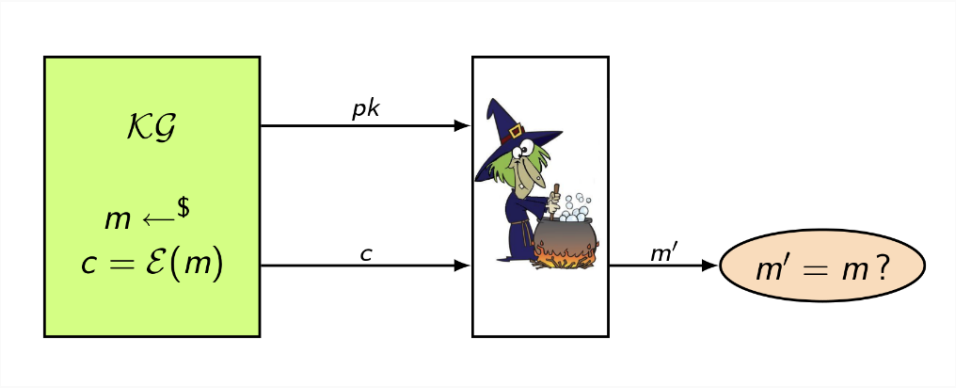
Attacker's Goals

- `BRK - Break the system`: find the secret key.
- `OW - One-wayness`: recover the "challenge" plaintext.
- `IND - Indistinguishability`: distinguish encryption of two different messages.
- `NM - Non-malleability`: modify a ciphertext to obtain another ciphertext so that the plaintext is related to the original plaintext.
  Theorem BRK $<$ OW $<$ IND $<$ NM.

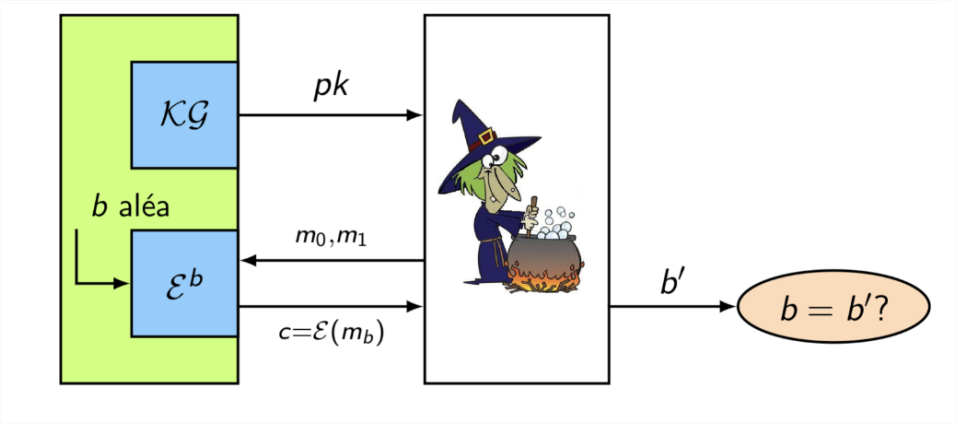
Break the system



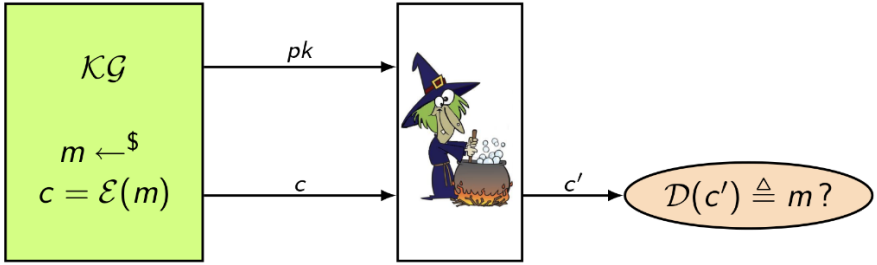
One-wayness



Indistinguishability



Non-Malleability



Attacker's Means

Attacker's means are defined by giving the adversary access to two oracles:

- Encryption oracle: the adversary can ask for any encryption of his choice.
- Decryption oracle: the adversary can ask for any decryption of his choice.
- `CPA - Chosen plaintext attacks`: the adversary has access to the encryption oracle only.
- `CCA1 - Non-adaptive chosen ciphertext attacks`: the adversary has access to the encryption oracle, and the decryption oracle before seeing the "challenge".
- `CCA2 - Adaptive chosen ciphertext attacks`: the adversary has access to both oracles.

Theorem CPA $<$ CCA1 $<$ CCA2.

Model of security

Security notion = goal + means

E.g., OW-CPA: one-wayness against chosen-plaintext attacks.

Recall IND-CPA, IND-CCA2.

Theorem IND-CCA2 is the standard security notion for encryption.

Security Analysis for RSA

- `BRK-CPA`: finding the secret key: reduce to factorization problem.
- `OW-CPA`: find the e-th root m of $c = m^e$: reduce to the RSA problem.

Textbook RSA is IND-CPA/IND-CCA insecure.

Security Analysis for ElGamal

- `BRK-CPA`: reduce to DLOG
- `OW-CPA`: reduce to CDH
- `IND-CPA`: reduce to DDH

ElGamal encryption is OW-CCA insecure.

Homomorphic Encryption

Definition Homomorphic encryption is a form of encryption with an additional evaluation capability for computing over encrypted data without access to the secret key. The result of such a computation remains encrypted.

Example:

- Multiplicative Homomorphic Encryption: $\mathrm{E}(m_1) \times \mathrm{E}(m_2) = \mathrm{E}(m_1 \times m_2)$
- Additive Homomorphic Encryption: $\mathrm{E}(m_1) + \mathrm{E}(m_2) = \mathrm{E}(m_1 + m_2)$
- Fully homomorphic encryption: allow both multiplications and additions.

Multiplicative Homomorphic Encryption

Theorem RSA encryption and ElGamal encryption are multiplicative homomorphic encryption.

Question Prove the above theorem.

Solution

- For RSA, let $c_1 = m_1^e$ and $c_2 = m_2^e$. What is $c_1 \times c_2$? What is the encryption of $m_1 \times m_2$?
- For ElGamal, let $(c_1, d_1) = (g^{r_1}, g^{r_1 s} \cdot m_1)$ and $(c_2, d_2) = (g^{r_2}, g^{r_2 s} \cdot m_2)$. How to obtain encryption of $m_1 \times m_2$?

Additive Homomorphic Encryption - Paillier's Cryptosystem

- G(): same as RSA, pk $= N$ where $N = pq$ and sk $= \varphi(N) = (p-1)(q-1)$.
- E(m): to encrypt a message $0 \leq m < N$, choose a random number $0 < r < N$ and output $c = (1+N)^m \cdot r^N \mod N^2$.
- D(c): output $m = \frac{c^{\varphi(N)}-1}{N} \cdot \varphi(N)^{-1} \mod N^2$.

How decryption works

$$\begin{aligned} c^{\varphi(N)} &= ((N+1)^m \cdot r^N)^{\varphi(N)} \mod N^2 \\ &= (N+1)^{m\varphi(N)} \cdot r^{N\varphi(N)} \mod N^2 \end{aligned}$$

- First, we have $r^{N\varphi(N)} = 1 \mod N^2$. Why? (hint: what is the order of the group modulo N^2?)

Additive Homomorphic Encryption - Paillier's Cryptosystem (cont.)

How decryption works

- Secondly, using the Binomial expansion:

$$(1+x)^n = \binom{n}{0}x^0 + \binom{n}{1}x^1 + \cdots \binom{n}{n}x^n,$$

so $(1+N)^m = 1 + m \cdot N \mod N^2$.
Then $(1+N)^{m\varphi(N)} = (1+m\cdot N)^{\varphi(N)} = (1 + m \cdot N \cdot \varphi(N)) \mod N^2$.

Theorem Paillier's cryptosystem is additive homomorphic encryption.