The RSA cryptosystem

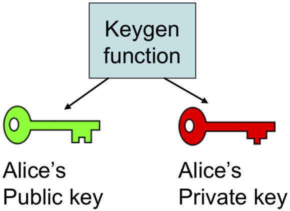
Part 1: encryption and signature

Jean-Sébastien Coron

University of Luxembourg

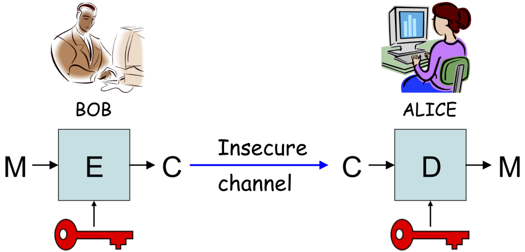
Public-key cryptography

- Invented by Diffie and Hellman in 1976. Revolutionized the field.
- Each user now has two keys
  - A public key
  - A private key
  - Should be hard to compute the private key from the public key.
- Enables:
  - Asymmetric encryption
  - Digital signatures
  - Key exchange, identification, and many other protocols.

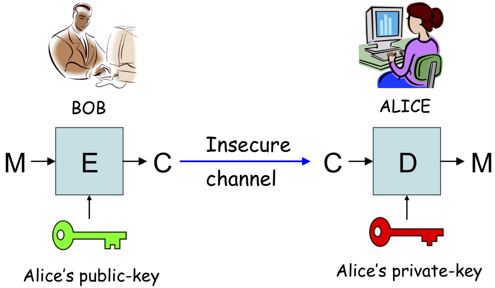


Key distribution issue

- Symmetric cryptography
  - Problem: how to initially distribute the key to establish a secure channel ?

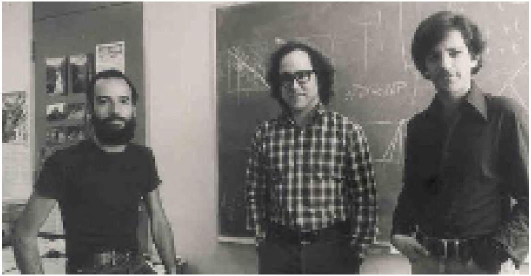


- Public-key encryption (or asymmetric encryption)
  - Solves the key distribution issue



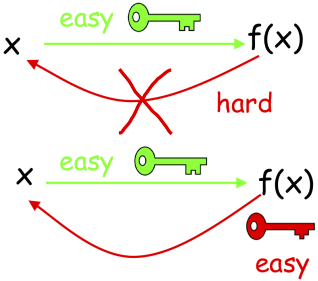
The RSA algorithm

- The RSA algorithm is the most widely-used public-key encryption algorithm
  - Invented in 1977 by Rivest, Shamir and Adleman.
  - Implements a trapdoor one-way permutation
  - Used for encryption and signature.
  - Widely used in electronic commerce protocols (SSL), secure email, and many other applications.



Trapdoor one-way permutation

- Trapdoor one-way permutation
  - Computing $f(x)$ from x is easy
  - Computing x from $f(x)$ is hard without the trapdoor
- Public-key encryption
  - Anybody can compute the encryption $c = f(m)$ of the message m
  - One can recover m from the ciphertext c only with the trapdoor



- Key generation:
  - Generate two large distinct primes p and q of same bit-size $k/2$, where k is a parameter.
  - Compute $n = p \cdot q$ and $\phi = (p-1)(q-1)$.
  - Select a random integer e, $1 < e < \phi$ such that $\gcd(e, \phi) = 1$
  - Compute the unique integer d such that

    $$e \cdot d \equiv 1 \pmod{\phi}$$

    using the extended Euclidean algorithm.
  - The public key is (n, e).
  - The private key is d.

RSA encryption

- Encryption
  - Given a message $m \in [0, n-1]$ and the recipent's public-key (n, e), compute the ciphertext:

$$c = m^e \mod n$$

- Decryption
  - Given a ciphertext c, to recover m, compute:

$$m = c^d \mod n$$

- Message encoding
  - The message m is viewed as an integer between 0 and $n-1$
  - One can always interpret a bit-string of length less than $\lfloor \log_2 n \rfloor$ as such a number.

Reminder: Fermat's little theorem

- Theorem
  - For any prime p and any integer $a \neq 0 \mod p$, we have $a^{p-1} \equiv 1 \mod p$. Moreover, for any integer a, we have $a^p \equiv a \mod p$.
- Proof
  - Follows from Euler's theorem and $\phi(p) = p - 1$.

Proof that decryption works

- We must show that $m^{ed} = m \bmod n$.
- Since $e \cdot d \equiv 1 \mod \phi$, there is an integer k such that $e \cdot d = 1 + k \cdot \phi = 1 + k \cdot (p-1) \cdot (q-1)$. Therefore we must show that:

$$m^{1+k\cdot(p-1)\cdot(q-1)} \equiv m \pmod{n}$$

- If $m \neq 0 \mod p$, then by Fermat's little theorem $m^{p-1} \equiv 1 \pmod{p}$, which gives :

$$m^{1+k\cdot(p-1)\cdot(q-1)} \equiv m \pmod{p}$$

  - This is also true if $m \equiv 0 \pmod{p}$.
  - This gives $m^{ed} \equiv m \pmod{p}$ for all m.
  - Similarly, $m^{ed} \equiv m \pmod{q}$ for all m.
  - By the Chinese Remainder Theorem, if $p \neq q$, then $m^{ed} \equiv m \pmod{n}$

Decrypting with CRT

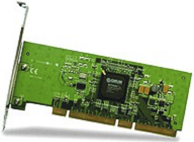
- Given the factors p and q of $n = p \cdot q$, instead of computing $m = c^d \bmod n$, compute:
  - $m_p = c^{d_p} \bmod p$, where $d_p = d \bmod (p-1)$
  - $m_q = c^{d_q} \bmod q$, where $d_q = d \bmod (q-1)$
  - Using CRT, find m such that $m \equiv m_p \pmod{p}$ and $m \equiv m_q \pmod{q}$:

$$m = \left(m_p \cdot (q^{-1} \bmod p) \cdot q + m_q \cdot (p^{-1} \bmod q) \cdot p\right) \bmod n$$

- Since exponentiation is cubic, this is roughly 4 times faster.

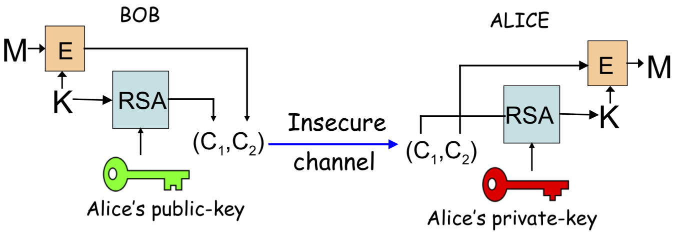
Implementation of RSA

- Required: computing with large integers
  - more than 1024 bits.
- In software
  - big integer library: GMP, NTL
- In hardware
  - Cryptoprocessor for smart-card
  - Hardware accelerator for PC.



Speed of RSA

- RSA much slower than AES and other secret key algorithms.
- To encrypt long messages
  - encrypt a symmetric key K with RSA
  - and encrypt the long message with K



Security of RSA

- The security of RSA is based on the hardness of factoring.
  - Given $n = p \cdot q$, it should be difficult to recover p and q.
  - No efficient algorithm is known to do that. Best algorithms have sub-exponential complexity.
  - Factoring record: a 768-bit RSA modulus n.
  - In practice, one uses at least 1024-bit RSA moduli.
- However, there are many other lines of attacks.
  - Attacks against textbook RSA encryption
  - Low private / public exponent attacks
  - Implementation attacks: timing attacks, power attacks and fault attacks

Factoring attack

- Factoring large integers
  - Best factoring algorithm: Number Field Sieve
  - Sub-exponential complexity

$$\exp\left((c + \circ(1))\, n^{1/3} \log^{2/3} n\right)$$

    for n-bit integer.
  - Current factoring record: 768-bit RSA modulus.
- Use at least 1024-bit RSA moduli
  - 2048-bit for long-term security.

Factoring vs breaking RSA

- Breaking RSA:
  - Given (N, e) and y, find x such that $y = x^e \mod N$
- Open problem
  - Is breaking RSA equivalent to factoring ?
- Knowing d is equivalent to factoring
  - Probabilistic algorithm (RSA, 1978)
  - Deterministic algorithm (A. May 2004, J.S. Coron and A. May 2007)

Elementary attacks

- Textbook RSA encryption: dictionary attack
  - If only two possible messages m_0 and m_1, then only $c_0 = (m_0)^e \mod N$ and $c_1 = (m_1)^e \mod N$.
  - $\Rightarrow$ encryption must be probabilistic.
- PKCS#1 v1.5
  - $\mu(m) = 0002 \| r \| 00 \| m$
  - $c = \mu(m)^e \mod N$
  - Still insufficient (Bleichenbacher's attack, 1998)

Chosen ciphertext attack against textbook RSA

- Chosen-ciphertext attack:
  - Given ciphertext c to be decrypted
  - Generate a random r
  - Ask for the decryption of the random looking ciphertext $c' = c \cdot r^e \pmod{n}$
  - One gets $m' = (c')^d = c^d \cdot (r^e)^d = c^d \cdot r = m \cdot r \pmod{n}$
  - This enables to compute $m = m'/r \pmod{n}$
- Conclusion: do not use textbook RSA encryption !

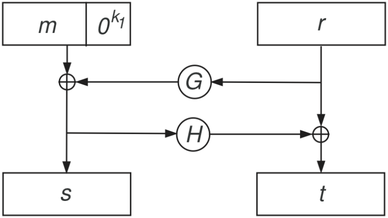
Proofs for encryption schemes

- Security notion for encryption.
  - From a ciphertext c, an attacker should not be able to derive any information from the corresponding plaintext m.
  - Even if the attacker can obtain the decryption of any ciphertext, c excepted.
  - This is called indistinguishability against a chosen ciphertext attack (IND-CCA2).
- Security proof for encryption
  - Prove that if an attacker can distinguish between the encryption of two plaintexts,
    then it can be used to break RSA.

IND-CCA2 security

- The attack scenario:
  - The adversary $\mathcal{A}$ receives the public key pk
  - $\mathcal{A}$ makes decryption queries for any ciphertexts y.
  - $\mathcal{A}$ chooses two messages M_0 and M_1 of identical length, and receives the encryption c of M_b for a random b.
  - $\mathcal{A}$ continues to make decryption queries. The only restriction is that the adversary can not obtain the decryption of c.
  - $\mathcal{A}$ outputs a bit b', representing its "guess" of b.
- IND-CCA2 security:
  - An encryption scheme is said to be IND-CCA2 secure if for any polynomial-time bounded $\mathcal{A}$, the advantage $\mathrm{Adv}(\mathcal{A}) = |2 \cdot \Pr[b' = b] - 1|$ is a negligible function of the security parameter.

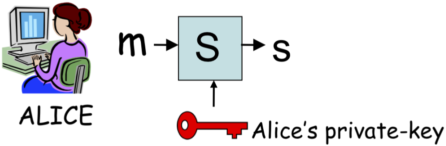
- OAEP (Bellare and Rogaway, E'94)
  - IND-CCA2, assuming that RSA is hard to invert.
  - PKCS #1 v2.1



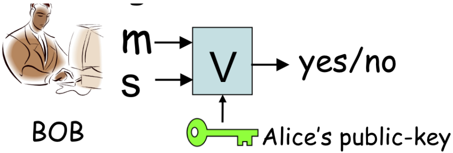
$$c = (s \| t)^e \mod N$$

Digital signatures

- A digital signature σ is a bit string that depends on the message m and the user's public-key pk
  - Only Alice can sign a message m using her private-key sk



  - Anybody can verify Alice's signature of the message m given her public-key pk



The RSA signature scheme

- Key generation :
  - Public modulus: $N = p \cdot q$ where p and q are large primes.
  - Public exponent : e
  - Private exponent: d, such that $d \cdot e = 1 \mod \phi(N)$
- To sign a message m, the signer computes :
  - $s = m^d \mod N$
  - Only the signer can sign the message.
- To verify the signature, one checks that:
  - $m = s^e \mod N$
  - Anybody can verify the signature

Hash-and-sign paradigm

- There are many attacks on basic RSA signatures:
  - Existential forgery: $r^e = m \mod N$
  - Chosen-message attack: $(m_1 \cdot m_2)^d = m_1^d \cdot m_2^d \mod N$
- To prevent from these attacks, one usually uses a hash function. The message is first hashed, then padded.
  - $m \longrightarrow H(m) \longrightarrow \texttt{1001}\ldots\texttt{0101}\|H(m)$
  - Example: PKCS#1 v1.5:
    $\mu(m) = \texttt{0001 FF....FF00}\|c_{\mathsf{SHA}}\|\mathsf{SHA}(m)$
  - The signature is then $\sigma = \mu(m)^d \mod N$

Conclusion

- The RSA cryptosystem
  - RSA encryption. Elementary attacks. IND-CCA2 security. OAEP
  - RSA signatures. Elementary attacks.
- Next lectures
  - More complex attacks. Coppersmith's theorem.
  - Security proofs for RSA signature schemes

Appendix

Probabilistic equivalence between knowing d and factoring

- We consider the particular case $N = pq$ with $p \equiv 3 \pmod 4$ and $q \equiv 3 \pmod 4$.
- Algorithm:
  - Write $u = e \cdot d - 1$. Therefore u is a multiple of $\phi(N) = (p-1) \cdot (q-1)$.
  - Write $u = 2^r \cdot t$ for odd t.
  - Generate a random $a \in \mathbb{Z}_N^*$
  - Compute $b \equiv a^t \pmod N$
  - Return $\gcd(b+1, N)$

Analysis

- We have $t = s \cdot \frac{p-1}{2} \cdot \frac{q-1}{2}$ for some odd s.
- Let $Q_p = \{x \in \mathbb{Z}_p^* \mid x^{(p-1)/2} \equiv 1 \pmod{p}\}$
  - Q_p is a subgroup of $\mathbb{Z}_p$ of order $(p-1)/2$
  - therefore $(a \bmod p) \in Q_p$ with probability $1/2$
  - Moreover:

$$\begin{aligned} a \in Q_p \quad &\Rightarrow \quad b \equiv 1 \pmod{p} \\ a \notin Q_p \quad &\Rightarrow \quad b \equiv -1 \pmod{p} \end{aligned}$$

- We obtain the factorization of N if $(a \in Q_p \land b \notin Q_q)$ or $(a \notin Q_p \land b \in Q_q)$
  - This happens with probability $1/2$