



Faculty of Sciences,
Technology
and Communication

Information Security Basics

Course Overview and Organization

Alex Biryukov
Jean-Sebastien Coron
Johann Großschädl
Peter Y. A. Ryan

Course Organization

- This course is divided into **3 parts**:
 - Symmetric cryptography (A. Biryukov + J. Großschädl)
 - Public-key cryptography (J.-S. Coron)
 - Security protocols (P. Y. A. Ryan)
- Homework
 - Separate homework **for each part**
 - Homework is individual assignment, **no collaboration!**
 - **Correction of homework** in a revision lecture in December
- Grading
 - Homework (for all three parts together): **40 %**
 - Final written exam (in January): **60%**

Information Security Profile

- This course introduces the Information Security Profile
- After successful completion of this profile a student will be able to:
 - Explain the **principles and techniques** of information security
 - Identify multiple **applications** for information security
 - Apply the most important **standards and methods** related to the management of information security in a given context
 - Know the **state-of-the-art** and thereby pursue **life-long learning** of new developments in information security

Semester 2

- **Required courses**

- Algorithms for Numbers and Public-Key Cryptography
- Principles of Security Engineering
- Symmetric Key Cryptography and Security of Communications

- **Recommended courses**

- Software Vulnerabilities: Exploitation and Mitigation
- Formal Methods (advised for Security Protocols)

Semester 3

- **Required courses**
 - Cryptocurrencies and the Cryptographic Blockchain
 - Security Modelling
 - Security Protocols (taking Formal Methods before this course is advised)
- **Recommended courses**
 - Fault and Intrusion Tolerance
 - Management of Information Security
 - Open Network Security

Information Security Basics

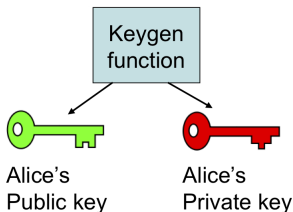
Part 1: Introduction to public-key cryptography

Jean-Sébastien Coron

University of Luxembourg

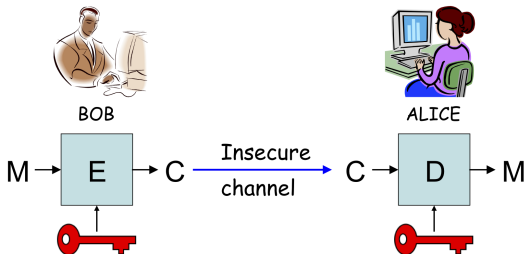
Public-key cryptography

- 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.



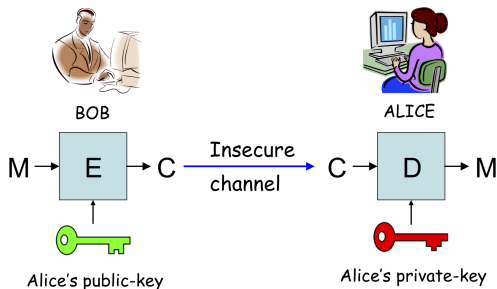
Key distribution issue

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



Public-key encryption

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



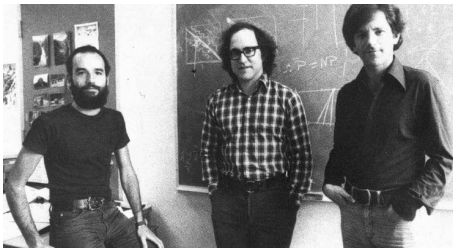
Analogy: the mailbox

- Bob wants to send a letter to Alice
 - Bob obtains Alice's adress
 - Bob puts his letter in Alice's mailbox
 - Alice opens her mailbox and read Bob's letter.
- Properties of the mailbox
 - Anybody can put a letter in the mailbox
 - Only Alice can open her mailbox



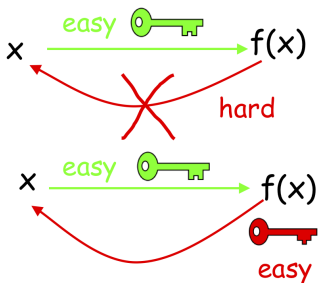
The RSA algorithm

- 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.

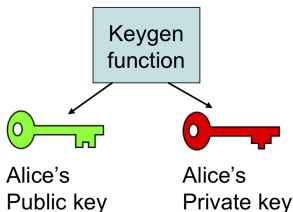
- Key generation:

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

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

- Decryption with private-key d
 - Given a ciphertext c , to recover m , compute:

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

- 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.

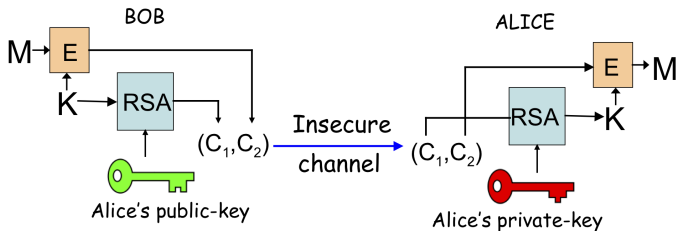
Implementation of RSA

- 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

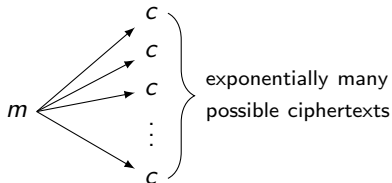


Security of RSA

- 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 (2020): a 829-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

Elementary attacks

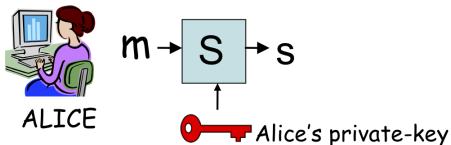
- Textbook RSA encryption: dictionary attack
 - If only two possible messages m_0 and m_1 , then only $c_0 = (m_0)^e \bmod N$ and $c_1 = (m_1)^e \bmod N$.
 - \Rightarrow encryption must be probabilistic.



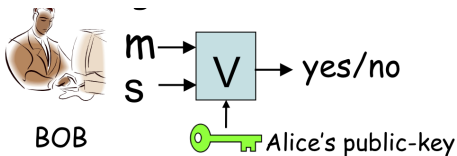
- Example: PKCS#1 v1.5 (1993)
 - $\mu(m) = 0002\|r\|00\|m$
 - $c = \mu(m)^e \bmod N$
 - Still insufficient
(Bleichenbacher's attack, 1998)

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





- A digital signature provides:
 - Authenticity: only Alice can produce a signature of a message valid under her public-key.
 - Integrity: the signed message cannot be modified.
 - Non-repudiation: Alice cannot later claim that she did not sign the message

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 $d \cdot e = 1 \pmod{\phi(N)}$
- To sign a message m , the signer computes :
 - $s = m^d \pmod{N}$
 - Only the signer can sign the message.
- To verify the signature, one checks that:
 - $m = s^e \pmod{N}$
 - Anybody can verify the signature

Hash-and-sign paradigm

- There are many attacks on basic RSA signatures:
 - Existential forgery: $r^e = m \pmod{N}$
 - Chosen-message attack: $(m_1 \cdot m_2)^d = m_1^d \cdot m_2^d \pmod{N}$
- To prevent from these attacks, one usually uses a hash function. The message is first hashed, then padded.

$$\begin{aligned} m &\longrightarrow H(m) \longrightarrow 1001 \dots 0101 \| H(m) \\ &\quad \downarrow \\ \sigma &= (1001 \dots 0101 \| H(m))^d \pmod{N} \end{aligned}$$

- Example: PKCS#1 v1.5 (1993)

$$\mu(m) = 0001 \text{ FF} \dots \text{FF}00 \| c_{\text{SHA}} \| \text{SHA}(m)$$

- The signature is then
$$\sigma = \mu(m)^d \pmod{N}$$

Other signature schemes

- Digital Signature Algorithm (DSA) (1991)
 - Digital Signature Standard (DSS) proposed by NIST, specified in FIPS 186.
 - Variant of Schnorr and ElGamal signature schemes
 - Security based on the hardness of discrete logarithm problem.
 - Public-key: $y = g^x \bmod p$
 - Signature: (r, s) , where $r = (g^k \bmod p) \bmod q$ and $s = k^{-1}(H(m) + x \cdot r) \bmod p$, where $k \xleftarrow{\$} \mathbb{Z}_q$
- ECDSA: a variant of DSA for elliptic-curves
 - Shorter public-key than DSA (160 bits instead of 1024 bits)
 - Used in Bitcoin to ensure that funds can only be spent by their rightful owners.

Diffie-Hellman key-exchange protocol

- Public parameters: g and p



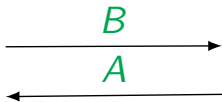
Bob

$$B = g^b [p]$$



Alice

$$A = g^a [p]$$



$$K_B = A^b = (g^a)^b = g^{ab} [p]$$

$$K_A = B^a = (g^b)^a = g^{ba} [p]$$

$$K_B = K_A$$

Security of Diffie-Hellman

- Based on the hardness of the discrete-log problem:
 - Given $A = g^a \pmod{p}$, find a
 - No efficient algorithm for large prime p .
- No authentication
 - Vulnerable to the man in the middle attack

Diffie-Hellman: man in the middle attack



Bob

$$B = g^b [p]$$



Alice

$$A = g^a [p]$$

B

A

$$K_B = A^b = g^{ab} [p]$$

$$K_A = B^a = g^{ba} [p]$$

$$K_B = K_A$$

Diffie-Hellman: man in the middle attack



Bob

$$B = g^b [p]$$



Eve



Alice

$$A = g^a [p]$$

$$K_B = A^b = g^{ab} [p]$$

$$K_B = K_A$$

$$K_A = B^a = g^{ba} [p]$$

Diffie-Hellman: man in the middle attack



Bob

$$B = g^b [p]$$

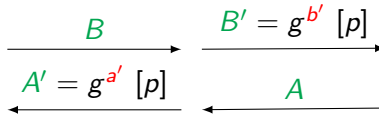


Eve



Alice

$$A = g^a [p]$$



$$K'_B = (A')^b = g^{a'b} [p]$$

$$K'_A = (B')^a = g^{b'a} [p]$$

$$K'_B = B^{a'} [p]$$

$$K'_A = A^{b'} [p]$$

Security of Diffie-Hellman

- Based on the hardness of the discrete-log problem:
 - Given $A = g^a \pmod{p}$, find a
 - No efficient algorithm for large prime p .
- No authentication
 - Vulnerable to the man in the middle attack
- Authenticated key exchange
 - Using a PKI. Alice and Bob can sign A and B
 - Password-authenticated key-exchange IEEE P1363.2

Lessons from the past

- Cryptography is a permanent race between construction and attacks
 - but somehow this has changed with modern cryptography and security proofs.
- Security should rely on the secrecy of the key and not of the algorithm
 - Open algorithms enables open scrutiny.
- Note: installation of Sage
 - Install Sage <https://www.sagemath.org>
 - Run a Jupyter notebook
 - \$ `sage -n jupyter`

Installation of Sage

- Install Sage <https://www.sagemath.org>
- Run a Jupyter notebook
\$ sage -n jupyter