Introduction to Cryptology: confidentiality, integrity, authenticity

2015/12 — Yaoundé, Cameroun

Damien Robert

Équipe LFANT, Inria Bordeaux Sud-Ouest
Institut de Mathématiques de Bordeaux
Équipe MACISA, Laboratoire International de Recherche en Informatique et Mathématiques Appliquées

Cryptology

Cryptology = Cryptography + Cryptanalysis

Usage:

- SSL/TLS, ssh, gpg
- GSM, Wifi, Bluetooth
- Credit Card, Transport card, Passport

Remark

Cryptology $\subset$ Security

Public Canal

Alice communicates with Bob through a public canal. Eve does passive attacks on this canal (spying) and Charlie does active attacks.

Active attacks:

- Usurpation of identity;
- Altering data;
- Repudiation
- Replay, repetition
- Man in the middle
- Delay, Destruction

Primitives

- Confidentiality
- Integrity
- Authenticity

- Key generation
- Disponibility
- Non repudiation

Primitives

- Confidentiality Symmetric encryption, Asymmetric encryption
- Integrity Cryptographic hash functions
- Authenticity Signature, MAC

- Key generation Randomness
- Disponibility
- Non repudiation

Primitive

- Without key: hash, random generator
- With key
  - symmetric
    - MAC
    - Encryption: stream, block
  - asymmetric: number theory, algebraic geometry, codes, lattices…
    - signature
    - Encryption

Confidentiality, Authenticity

- Confidentiality: $E:\{0,1\}^n \to \{0,1\}^n$ a permutation; $D = E^{-1}$. Encryption: $m \mapsto c = E(m)$, Decryption: $c \mapsto m = D(c)$.
- Authenticity: $c = A(m)$. Alice sends (m, v). Bob receives (m', v'). Verification: $V(m', v') =$ OK, NOT OK.

Kerchoff's laws

- E and D needs to be secret;
- So no external validation of security possible;
- And transmitting the algorithmes in painful;
- Kerchoff: parametrizes the algorithms by a key K;
- $E:\{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}^n$ such that $E(\cdot, K)$ is a permutation for all K;
- E can be public, the only secret is the secret key K.

Channels of communications

- Public
- Authenticated
- Confidential
- Authenticated + Confidential

Goals

Using a authenticated and/or confidential channel, construct an authenticated and/or confidential channels inside a public channel.

The goal if of course to use the preexisting authenticated/confidential channel as little as possible, and do everything else in the authenticated/confidential channel constructed inside the public channel.

Transforming a public channel into a confidential/authenticated channel

- Authenticated $\Rightarrow$ Authenticated;
- Authenticated+Confidential $\Rightarrow$ Authenticated
- Authenticated $\Rightarrow$ Authenticated
- Authenticated+Confidential $\Rightarrow$ Confidential
- Authenticated $\Rightarrow$ Confidential
- Authenticated $\Rightarrow$ Authenticated+Confidential $\Rightarrow$ Authenticated+Confidential

Example 1: integrity

Alice sends m through the public channel, and $h(m)$ through the integrity channel, where $h : \{0,1\}^* \rightarrow \{0,1\}^{256}$ is a cryptographic hash function.

Security:

- Preimage resistance;
- Second-Preimage resistance;
- Collision resistance.

Example 2: authenticity (symmetric)

A secret key K is generated (randomisation) and sent to Alice and Bob through a authenticated and confidential channel. Alice sends $(m, \mathrm{MAC}(m, K))$ through the public channel. Bob verify via $\mathrm{VERIF}(m, K)$. The MAC is a hash function parametrized by K.

Security: from several couples (M, C)

- Can't retrieve K;
- Can't generate a new (M', C');
- Can't distinguish the distribution C from a uniform distribution.

Example 2: authenticity (asymmetric)

A couple (K_S, K_P) is generated by Alice and she sends the public key K_P to Bob via an authenticated channel. Alice sends $(m, \mathrm{SIGN}(m, K_S))$ through the public channel, and Bob verify via $\mathrm{VERIF}(m, K_P)$.

Security: from several couples (M, C) and K_P

- Can't retrieve K_S;
- Can't generate a new (M', C');
- Can't distinguish the distribution C from a uniform distribution.

Signature vs MAC:

- Public verification
- Non repudiation

Example 3: confidentiality (symmetric)

A secret key K is generated (randomisation) and sent to Alice and Bob through a authenticated and confidential channel. Alice sends $c = E(m, K)$ through the public channel, Bob decrypts via $m = D(c, K)$.

Security: from several ciphers C, several couples $(M, C = E(M, K))$ (chosen plain text attack) and several couples $(C, M = D(M, K))$ (chosen cipher text attack)

- Can't retrieve K;
- Can't find M' from a new C';
- Can't distinguish the distribution C from a uniform distribution.

Example 3: confidentiality (asymmetric)

A couple (K_S, K_P) is generated by Bob and he sends the public key K_P to Alice via an authenticated channel. Alice sends $c = E(m, K_P)$ through the public channel, and Bob decrypt via $m = D(c, K_S)$.

Security: from K_P, several ciphers C, several couples $(M, C = E(M, K))$ (chosen plain text attack) and several couples $(C, M = D(M, K))$ (chosen cipher text attack)

- Can't retrieve K_S;
- Can't find M' from a new C';
- Can't distinguish the distribution C from a uniform distribution.

Asymmetric vs symmetric

- N persons $\Rightarrow N$ keys rather than N^2;
- Does not need a confidential channel;
- Much slower.

$\Rightarrow$ Use an asymmetric cipher to send a symmetric secret key and switch to the symmetric channel to increase speed.

Security

- Only cipher known
- Chosen plain text: CPA, Adaptative chosen plain text: CPA2
- Chosen cipher text: CCA, Adaptative chosen cipher text: CCA2

$\Rightarrow$ Invert the function: OW (One Wayness)

$\Rightarrow$ Indistiguinbility: IND. Detect an encryption of 1 from an encryption of 0 with probability $> 0.5+\varepsilon$. IND = Semantic security.

Ultimate goal: IND-CCA2 cryptosystem.

Attacks:

- Black box or structural analysis;
- Side channels.

Security parameters

- 2^{40}: One minute on a standard computer;
- 2^{60}: One year on a standard computer;
- 2^{80}: One year with 10^6 cores at $5GHz$ = NSA?
- 2^{128}: security goal.

Remark

One may want to take higher security parameters 192 bits or 256 bits for very long term security and for protection against potential attacks. For instance quantum computers can divide by 2 the security of some problems (and completely kill others like factorisation or the discrete logarithm problem).

One way function

- Very important in cryptology, needs strong properties
- One way function $x \mapsto f(x)$ easy, but $y \mapsto f^{-1}(y)$ hard.

Example

Multiplication, exponentiation in $(\mathbb{Z}/p\mathbb{Z}^*, \times)$.

In asymmetric key cryptography, use of trapdoor one way function: a secret trap can allows to compute f^{-1}.

Example

If $N = pq$, $x \mapsto x^2$ is a trapdoor one way function.

Hash function

- Origin: hash table to speed up lookup;
- $h : \{0,1\}^* \longrightarrow \{0,1\}^n$;
- Very important in cryptology, needs strong properties:
- No preimage
- No second preimage
- No collision

Example

- A checksum is not an hash function
- $x \mapsto x^2 \mod pq$ is one way but not a cryptographic hash function.
- MD5 (Broken), SHA-1 (Almost broken), SHA-2, SHA-3

Collision

Let h be an hash function with n bits of output; $N = 2^n$.

- Given $y \in \{0,1\}^n$, finding y at random requires $\Theta(2^n)$ tries;
- What about a collision: $x_1 \neq x_2 \mid h(x_1) = h(x_2)$?
- After k tries, the probability of not finding a collision is

$$p(k) = (1 - 1/N)(1 - 2/N)\ldots(1 - k/N)$$

- So we have the inequalities (in fact it is an order of equivalence)

$$\log p(k) = \sum_{i=1}^{k} \log(1 - i/N) \leqslant -\sum_{i=1}^{k} i/N \leqslant -k^2/2N$$
$$p(k) \leqslant \exp(-k^2/2N)$$

- So if $k = \Theta(\sqrt{N})$, $p(k)$ is small and the probability of collision is high;
- To get 128 bits of security we need $n = 256$

Construction

- Compression function $f : \{0,1\}^{2n} \to \{0,1\}^n$;
- Merkle-Damgard (IV=Input Vector)

$$h(m_1, \ldots, m_k) = f(f(\ldots f(f(IV, m_1), m_2), \ldots, m_k)$$

- If m is not of length a multiple of n, pad n with its length

Theorem

If f is collision resistant, then h too.

Remark

- A small weakness in f can lead to a big weakness in h (MD5);
- From $h(m)$, we can compute $h(m \,||\, m_0)$ without knowing m.

Constructing f: mixing boolean operations, addition (non bit-linear), into multiple rounds with some magic constants.

Mise en gage

- Alice can publish $h(m)$ to prove later that she selected m;
- If m is small (YES/NO), necessity to pad with a random sequence. “The hash of my CB card is fe9e5fa6d82a071422a576064d0ed49a7266ccb49390037c255e6e7baa8d4535” is a bad idea. Better idea: publish the hash of (CB card + long random sequence of characters).

Example

- Head or tails by phone
- Zero-Knowledge via coloring graphs
- Stocking a password: pass, $h(\text{pass})$, $h(\text{login} \,||\, \text{pass})$, $h(\text{salt} \,||\, \text{pass})$.
  Question: What is the best method?
- Integrity of a software download (Exercice: why would a signature be better?)

MAC: Message authentification code

- $h_K(m)$ were K is a secret key between Alice and Bob to prove the identity of the emitter
- Security: Eve can not produce $(m', h_K(m'))$;
- $\mathrm{HMAC}(K, m) = H(K \oplus c_1 \,||\, H(K \oplus c_2) \,||\, m)$; proven secure
- $H(M \,||\, K)$ or $H(K \,||\, M)$ is a bad idea if h is constructed via Merkle-Damgard (MD5, SHA-1, SHA-2) but should be ok if h use a sponge function (SHA-3).
- h_K could also be a block cipher
- In fact a bloc cipher E can define a hash function via $h_i = E(h_{i-1}, m_i) \oplus h_{i-1}$ (Davies-Meyer) but we want a faster hash function.

Randomness generator

- Vital in a cryptosystem
- Sony: random = constant
- Debian ssh: random = date
- RSA key: a lot of common primes in public modulus

Statistic properties

- Standard randomness: good statistic properties
- Linear congruence: $x_{n+1} = a\,x_n + b \mod M$ very fast but some statistic bias
- Cryptographic randomness: needs much stronger properties
- Can't predict the next bit from the observed ones
- Broken for linear congruences

Seed

- True alea = non compressable (Kolmogorov)
- By definition an algorithm can't generate a true alea

$\Rightarrow$ Pseudorandom generator.

- Construction: a small seed (true alea) used by the pseudorandom generator
- Hash function = Compress state; PRNG = Expand state.
- s internal state: $s = f(s)$ (update internal state), $x = g(s)$ (output next random bit)

Remark

- Finite number of internal states $2^n \Rightarrow$ the PRNG will loop
- Birthday paradox: A "random" update of the seed loop in time $\sqrt{\text{internal states}}$
- Arithmetic PNRG can force a loop of 2^n
- Bad idea for cryptography

True alea

- We need a true alea to initialize the seed
- Use physical input: input/output, mouse mouvements, IP packets...

⇒ In linux, `/dev/random` collects the entropy and outputs a random sequence until the entropy is $0 \Rightarrow$ blocks waiting for new entropy

- `/dev/urandom` uses the entropy inside a PRNG to output a random sequence which never blocks
- Problem: in early boot, `urandom` may output a sequence while the seed had not enough entropy yet;
- In an idle machine not a lot of entropy; even worse for virtual machine without help from the container;
- Possible solution: with a good PRNG, we just need an initial seed of true 256 bits of entropy; keep the current state across reboots.

Entropy

- Quantity of information: if $p(x=1)=0.99$ and $p(x=0)=0.01$, observing $x=0$ is much more useful than $x=1$;
- Quantity of information: $QI(m=x)=\log_2(1/p(x))$
- The entropy is the average value of the QI:

$$e=-\sum p_i \log_2 p_i$$

- n bits of entropy $\approx$ information that needs n bits to be encoded.

Example

- $x \in \{0,1\ldots,15\}$ uniformly: 4 bits of entropy
- $p_A=0.5$, $p_B=0.25$, $p_C=0.25$. $e=1/2+2/4+2/4=3/2$. Encode A with 0, B with 10 and C with 11 $\Rightarrow$ 3/2 bits on average to encode the message.

Historical ciphers

- CESAR: translate letter in the alphabet by the same amount
- Alphabetical substitution
- VIGENERE: CESAR depending on the position of the letter:

  CRYPTOGRAPHYINGABON
  SECRETSECRETSECRETS
  UVAGXHYVCGLRARIRFHF

Statistical attacks

- The messages correspond to word in a language, they are not uniform;
- If the ciphers are still non uniform $\Rightarrow$ statistical attacks
- Alphabetical substitution: most frequent letters, vowels are linked with many other letters;
- Index of correlation: split the message into several lines. Probability that one letter is the same as the letter below (ie probability that two random letters are the same);
- Uniform messages: index of correlation is $1/26$;
- Far from the case in French: $10-15\%$;
- Vigenere: if we split the messages into blocks of length k and find an index of correlation similar to the French one then high probability than the length of the secret is a divisor of k and we are back to CESAR.

Enigma

- Keyboard Plug P (several disjoint transposition)
- Several rotors R_i (each rotor is a permutation): 3 then 4
- Reflector M: 13 couples for the 26 letters
  $A \longleftrightarrow D$, $B \longleftrightarrow M$, ...
- $E(m) = P^{-1} R_1^{-1} R_2^{-1} R_3^{-1} M R_3 R_2 R_1 P$
- After each output, R_1 makes a turn; if R_1 has made a full
- The secret state is given by the position of the plugboard and the initial position of the rotors.
- Feature: $E^2 = \mathrm{Id}$ so decryption use the same initial state as encryption;
- Security problem: for all letter x, $E(x) \neq x$. Big statistical drawback;
- Cryptanalysis of Enigma (Poland then England+USA). A bombe explores a lot of Enigma position, using statistical analysis to greatly speed up the process.

Enigma

- Keyboard Plug P (several disjoint transposition)
- Several rotors R_i (each rotor is a permutation): 3 then 4
- Reflector M: 13 couples for the 26 letters
  $A \longleftrightarrow D$, $B \longleftrightarrow M$, ...
- $E(m) = P^{-1} R_1^{-1} R_2^{-1} R_3^{-1} M R_3 R_2 R_1 P$
- After each output, R_1 makes a turn; if R_1 has made a full
- The secret state is given by the position of the plugboard and the initial position of the rotors.
- Feature: $E^2 = \mathrm{Id}$ so decryption use the same initial state as encryption;
- Security problem: for all letter x, $E(x) \neq x$. Big statistical drawback;
- Cryptanalysis of Enigma (Poland then England+USA). A bombe explores a lot of Enigma position, using statistical analysis to greatly speed up the process.

Modern attacks

- Linearity: solve big linear systems
- Algebraic attacks: solve big multivariate algebraic systems
- Differential attacks: let $E_k(m \oplus \Delta_m) = c \oplus \Delta_c$ and study the distribution of Δ_c.

Unconditional security

- Vernam cipher (One time pad) $c_i = m_i \oplus k_i$
- Shannon: unconditionally secure if $k = k_1 \ldots k_n$ is uniform random (Proof: distribution of $m \oplus$ uniform distribution = uniform distribution);
- Not convenient: key of same length as the message
- Reusing key (or part of the key) is catastrophic: if $c_1 = m_1 \oplus k$ and $c_2 = m_2 \oplus k$ then $c_1 \oplus c_2 = m_1 \oplus m_2$; this reveals a lot of information;
- Unconditonal security is too strong, we only care about computational security.

Stream cipher

- Simulate the One Time Pad by using a PRNG parametrized by a secret key k;
- Internal state: s_i. Update: $s_{i+1} = f(s_i, K)$. Output $x_i = g(s_i, K)$.
- Encryption/Decryption: $c_i = m_i \oplus x_i$.

Remark

Problems of synchronisation. Autosynchronising stream ciphers: use the last t ciphers as the state: $x_i = g(c_{i-t}, \dots, c_{i-1}, K)$. If there is an error of transmission, this corrupt the decryption for only t bits.

Linear Feedback Shift Register (LFSR)

A LFSR has L cells.

- Output x_0;
- Shift: $x_i = x_{i+1}$;
- Feedback: $x_{L-1} = x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k}$.

Definition

The retroaction polynomial is $P(x) = x^L + \sum x_{i_k} x^k$.

The LFSR is uniquely determined by its initial value and its retroaction polynomial.

Linear algebra

- The state of the vector $X = (x_0, \ldots, x_{L_1})$ in the LFSR is linear
- For instance if $P(x) = x^4 + x + 1$, then at step $i+1$, $X_{i+1} = MX_i$ where

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \in \mathbb{F}_2$$

- The characteristic polynomial of M is $P(x)$.
- The LFSR will loop when $M^k = \mathrm{Id}$
- This is the order of X in $\mathbb{F}_2[X]/P(x)$ (if P is the minimal polynomial).

Order of the LFSR

- If $P(x)$ is irreducible then $K = \mathbb{F}_2[X]/P(X) = \mathbb{F}_2(x)$ is a field of degree L. The order of x in K divides $2^L - 1$.
- If P is primitive the order is exactly $2^L - 1$;
- If $P = \prod P_i$ is a product of distinct irreducible polynomials then $K = \prod \mathbb{F}_2[X]/P_i(X)$ is a product of fields (CRT) and the period divides $\prod 2^{\deg P_i} - 1 < 2^L - 1$.
- The highest period is given by primitives polynomials.

Theorem

There is $\frac{\varphi(2^L-1)}{L}$ primitive polynomials of degree L in $F_2[x]$.
In particular if $2^L - 1$ is prime (a Mersenne prime) then there is $(2^L - 2)/L$ primitive polynomials of degree L and all irreducible polynomials are primitive.

Proof.

The splitting field of an irreducible polynomial of degree L is always $\mathbb{F}_{2^L}$ since the absolute Galois group is procyclic. There is $\varphi(2^L - 1)$ generators of the multiplicative group $\mathbb{F}_{2^L}$. The Galois group splits this group into $\varphi(2^L - 1)/L$ orbits (since the Frobenius is of order L), each orbit corresponds to a primitive polynomial. $\square$

Security

- An LFSR can have a high period;
- But the output is linear, from $2L$ terms one can recover its retroaction polynomial (Berlekamp Masse)

Proof.

There exists a fraction P_0/P_1 whose formal sum $\sum x_i X^i$ corresponds to the bit output by the LFSR. The Euclidean algorithm between $\sum_{i=0}^{2L-1} x_i X^i$ and X^{2L} recovers this fraction (as the continued fraction algorithm recovers the rational fraction p/q from its decimal development). $\square$

- In practice combine several LFSR with a non linear filter function
- A5/1 (GSM) combines 3 LFSR; but the filter function is weak ⇒ attacks if enough data is gathered.

Block cipher

- $c = E_K(m)$, $m = D_K(c)$ where m is a block of n bits and K is block of k bits;
- There is $(2^n)!$ bijections and 2^k possible keys; so we can have $k > m$.
- If the block is too small ($n = 8$) dictionary attacks;
- AES works with blocks of 128 bits but has three level of security: 128, 192 and 256 bits (which corresponds to 10, 12 and 14 rounds).
- Security: Observing (m, c) should reveal no information on K or allows to generate (m', c').
- Related keys: changing one bit of K should completely change the (m, c).

Feistel scheme

- Several rounds: $m = L_0 \,||\, R_0$.
- $L_{i+1} = R_i$, $R_{i+1} = L_i \oplus F_{K_i}(R_i)$ (K_i is a key derived from K for round i)
- This is always inversible, even if F_{K_i} is not injective!
- Decrypting: $R_i = L_{i+1}$, $L_i = R_{i+1} \oplus F_{K_i}(L_{i+1})$.
- Used by DES: Feistel scheme with 16 rounds.

DES

- Blocks of 64 bits, key of 56 bits;
- Good for the time (1976);
- Key size too low now.
- Triple DES used instead (now superseded by AES):
  $E_{K_1,K_2} = \mathrm{DES}_{K_1} \circ \mathrm{DES}_{K_2}^{-1} \circ \mathrm{DES}_{K_1}$;
- Key length of Triple DES is 112 bits.
- Some plain text attacks $\Rightarrow$ effective security of 80 bits.

Exercice

1. Why not simply use $E_{K_1,K_2} = \mathrm{DES}_{K_2} \circ \mathrm{DES}_{K_1}$?
2. Why the $\mathrm{DES}_{K_2}^{-1}$ in the middle?

AES

- Selection by NIST in 2001;
- Blocks of size 128, Keys of size 128, 192, 256;
- Several rounds (10, 12, 14) parametrized by subkeys;
- One round: 128 bits = 16 bytes, organized in a 4×4 square;

1. SubBytes: inversion in $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/x^8 + x^4 + x^3 + x + 1$
2. ShiftRows: rows are shifted (by a different value)
3. MixColumns: linear
4. AddRoundKey: XOR with the derived keys.

Electronic Code Book (ECB)

- $c_i = E_K(m_i)$;

Example (From Wikipedia)

Name + Salaries encrypted through ECB with blocks of 2 characters.
Jack salary is 105000€ by year, and the encrypted data is `Q92DFPVXC9IO`.
The other encrypted data are `TOAV6RFPY5VXC9`, `YPFGFPDFDFIO`, `Q9AXFPC9IOIO`, `ACED4TFPVXIOIO`, `UTJSDGFPRTAVIO`
What is the salary of Jane, Jack's boss?

Cipher-block chaining (CBC)

- Initialisation: $c_0 = IV$ (input/initialisation vector)
- $c_i = E_K(c_{i-1} \oplus m_i)$;
- $m_i = c_{i-1} \oplus D_K(c_i)$;
- Randomizing the IV $\Rightarrow$ same plain text to different cipher text.

Counter

- $I_0 = IV$
- $c_j = m_j \oplus E_K(I_j)$; $I_{j+1} = I_j + 1$,
- This is actually a stream cipher!

Output feedback (OFB)

- $I_0 = IV$
- $c_j = m_j \oplus I_j;\ I_{j+1} = E_K(I_j)$

MAC + Encryption

- A block cipher can also be used as a MAC: the last cipher block is the MAC (needs a good operational mode).
- MAC then Encrypt? (SSL); Encrypt and MAC? (SSH); Encrypt then MAC?
- Encrypt then MAC is secure; MAC then Encrypt has a lot of problem (decryption oracle); Encrypt and MAC has theoretical problems (For instance $\mathrm{MAC} = E \oplus m$) but no strong practical problems.

- Authenticity+Integrity: HMAC, Poly1305, Galois Message Authentication Code (GMAC);
- Confidentiality+Authenticity+Integrity: GCM (Galois Counter Mode), CCM (Counter Mode + CBC-Mac)
- Block ciphers: AES
- Stream ciphers: Salsa20 (and the variant Chacha20), also used in the BLAKE hash function, ESTREAM.

Authentification

Challenge / Answer

- Bob chooses a random r, computes $x = h(r)$ and sends the challenge $(x, E_{K_P}(r))$ to Alice;
- Alice decrypt to find r, checks that $x = h(r)$ and sends the answer r to Bob;
- Question: Why use a hash function here and not just send $E_{K_P}(r)$?

Signature

- Bob sends a random message r to Alice;
- Alice appends random noise to r and signs this.
- Question: Why the random noise?

Public Key Infrastructure

- Even with asymmetric cryptography, we still need an authenticated channel to transmit the public key K_P;
- Web of trust (decentralized): I trust the persons trusted by the persons I trust. Used by gpg.
- PKI (centralized): public key signed by an organism via a certificate. Verification done recursively until we find a root certificate. Used by TLS/SSL: 166 root certificates on my computer.
- Certificate for n persons: n certificates? 1 certificate using a binary hash tree: recursively if the node N has two children C_1, C_2 then $h(N) = h(C_1 \mid\mid C_2)$. We only need to verify the authenticity of the root node R; verification of a node N only uses the path between N and R $\Rightarrow O(\log n)$.

Protocols

- TLS Key exchange + Authentication algorithms: RSA, DHE-RSA, DHE-DSS, ECDH-ECDSA, ECDHE-ECDSA, ECDH-RSA, ECDHE-RSA
- TLS Ciphers: AES-CBC, AES-CCM, AES-GCM, Chacha20-Poly1305
- SSH Authentication: id_dsa, id_rsa, id_ecdsa, id_ed25519
- SSH Key exchange algorithms: curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group-exchange-sha1, diffie-hellman-group14-sha1, diffie-hellman-group1-sha1
- SSH Ciphers: aes128-ctr, aes192-ctr, aes256-ctr, arcfour256, arcfour128, aes128-gcm@openssh.com, aes256-gcm@openssh.com, chacha20-poly1305@openssh.com, aes128-cbc, 3des-cbc, blowfish-cbc, cast128-cbc, aes192-cbc, aes256-cbc, arcfour

Bibliography

Novels

- Dan Brown, Digital Fortress as an exercice to find the numerous technical mistakes about cryptography in the novel
- Neal Stephenson, Cryptonomicon, where the hero use the Solitaire encryption algorithm by Schneier which just needs a deck of card.

Historical

- David Kahn, The Codebreakers;
- Simon Singh, The Code Book (Histoire des Codes Secrets);
- Jacques Stern, La Science du Secret;

Reference

- Steven D. Galbraith, Mathematics of Public Key Cryptography.
- Jeffrey Hoffstein, Jill Pipher et Joseph H. Silverman, An Introduction to Mathematical Cryptography;
- Antoine Joux, Algorithmic Cryptanalysis;
- Alfred J. Menezes, Paul C. van Oorschot et Scott A. Vanstone, Handbook of Applied Cryptography http://www.cacr.math.uwaterloo.ca/hac/;
- Serge Vaudenay, A Classical Introduction to Cryptography;
- Bruce Schneier, Applied Cryptography;
- Douglas R. Stinson, Cryptography: Theory and Practice.