Cryptography – Exam Questions

Tim Herbstrith

2020

Contents

1 Cryptography principles / Basic model for secrecy / Cryptosystem for secrecy 2

2 Attacks on encryption algorithms 4

3 Examples of symmetric cryptosystems 5

4 Computational complexity 6

5 Three types of security 9

6 RSA cryptosystem 12

7 One-way functions 12

8 Hash functions 12

9 Discrete logarithm problem 12

10 ElGamal cryptosystem 12

11 Elliptic curves 12

12 Group structure on elliptic curves 13

13 Cayley-Bacharach's theorem 13

14 Associativity 13

15 Elliptic curves over finite fields 13

16 Diffie-Hellman key agreement protocol 13

17 Digital signature scheme 13

18 DSS with hashing 15

19 DSS and Public-key cryptosystems 16

20 ElGamal variant of DSS (Definition) 17

21 ElGamal and ECDSA variants of DSS 19

22 Digital currency 21

23 Bitcoin transactions 21

24 Bit generators 21

25 Distinguisher and next bit predictors 21

26 Error-correcting codes and expander graphs 21

27 Probabilistic pidgeonhole principle 21

28 Attacks on cryptosystems relying on structural weaknesses 22

29 Shanks algorithm 22

30 TODO-s: 22

All quotes are from Arzhantseva (2019).

1 Cryptography principles / Basic model for secrecy / Cryptosystem for secrecy

> Cryptography principles definitions, (non) examples. Basic cryptography concepts (primitive, protocol, cover time, etc.). Basic model for secrecy: (non)-examples. Cryptosystem for secrecy: definition, examples. Symmetric versus asymmetric cryptosystems.

1.1 Cryptography principles

- Confidentiality / secrecy:
  - limit access to information
- Data Integrity
  - data was not altered (intentionally or accidentally)
  - detection of alteration (not prevention)
- Data origin authentication / message authentication
  - confirms the origin of data with no temporal aspect to the **receiver**
  - not necessarily an immediate source / not when
- Entity authentication
  - a given entity is involved and currently active
  - e.g. log in at web service
- Non-Repudiation
  - a source of data cannot deny to a **third party** being at the origin

Data origin authentication ⇒ Data integrity

Non-Repudiation ⇒ Data origin authentication

Data origin authentication $\neq$ Entity authentication

Secrecy $\nRightarrow$ Data origin authentication

1.2 Different cryptographic concepts

- Cryptography = **toolkit**
- Cryptographic **primitive** = a basic tool in this toolkit
  - Examples: Encryption, hash function, MAC (message authentication code), digital signature, etc.
- Cryptographic **algorithm** = Cipher = a specification of a primitive
- Cryptographic **protocol** = a way to choose primitives and use them for a security goal
- Crypto**system** = implementation of primitives and the infrastructure

1.3 Basic model of a cryptosystem

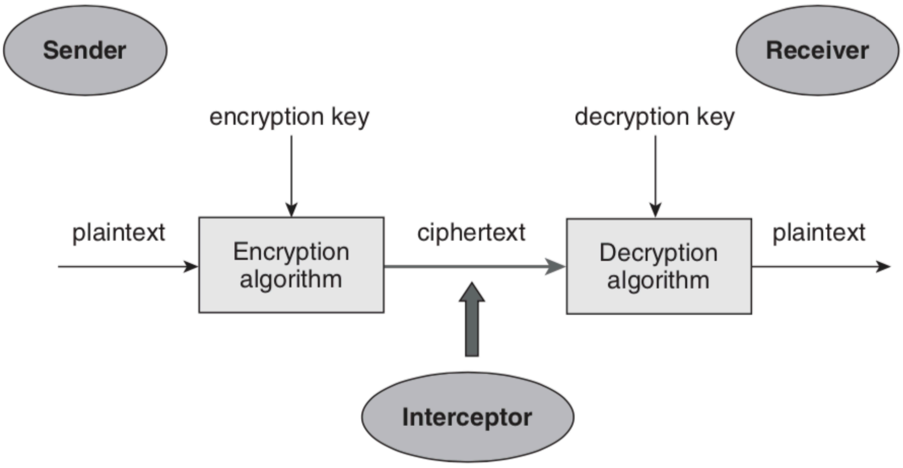

Figure 1: Basic model of a cryptosystem (Martin 2012)

Fig. 1 depicts a sender who wishes to transfer some data to a receiver in such a way that any party intercepting the transmitted data cannot determine the content. *The interceptor must not know the decryption key.*

Secrecy can be provided by (combination of):

1. Cryptography (via encryption)
2. Steganography (via information hiding)
3. Access control (via software or hardware)

1.4 Definition of Cryptosystem

Cryptosystem is a 5-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ satisfying:

- $\mathcal{P}$ is a finite set of possible **plaintexts**;
- $\mathcal{C}$ is a finite set of possible **ciphertexts**;

- $\mathcal{K}$, the keyspace, is a finite set of possible **keys**;
- $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ consists of **encryption functions** $E_k : \mathcal{P} \to \mathcal{C}$;
- $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ consists of **decryption functions** $D_k : \mathcal{C} \to \mathcal{P}$;
- For all $e \in \mathcal{K}$ there exists $d \in \mathcal{K}$ such that for all plaintexts $p \in \mathcal{P}$ we have:

$$D_d(E_e(p)) = p$$

The cryptosystem is

- **symmetric** if $e = d$ and
- **public-key** if d cannot be derived from e in a computationally feasible way

1.5 Cover time

Cover time = the time for which a plaintext must be kept secret.

2 Attacks on encryption algorithms

Main attacks on encryption algorithms. Passive versus active attacks. Keys: length, size. Brute-force attack: assumptions, estimates on key lengths.

2.1 Targets of attacks

- A practical method of determining the **decryption key** is found.
- A weakness in the encryption algorithm leads to a **plaintext**.

2.2 Passive vs active attacks

- The main type of **passive attack** is unauthorised access to data.
- An **active attack** involves either data being changed in some way, or a process being conducted on the data.

2.3 Key lengths and sizes

- **Length** of the key = number of bites it takes to represent the key
- **Size** of the keyspace = number of possible different decryption keys

2.4 Assumptions

- All keys from the keyspace are equally likely to be selected
- The correct decryption key is identified as soon as it is tested

2.5 Estimates on key length

If Size = $n = 2k$, then, on average, one needs $\sim 2k - 1$ attempts to find the correct decryption key:

$$\mathbb{E}[X] = \sum_{i=1}^{n} i\frac{1}{n} = \frac{n\,(n-1)}{2} = \frac{2^k + 1}{2} \sim 2^{k-1}$$

3 Examples of symmetric cryptosystems

Examples of symmetric cryptosystems: Caesar and Substitution ciphers. The letter frequency analysis. Monoalphabetic and polyalphabetic cyphers. Vigenère cipher. If the given key of a Vigenère Cipher has repeated letters, does it make it any easier to break?

3.1 Caeser Cipher

Replace each alphabet by another alphabet which is 'shifted' by some fixed number between 0 and 25. Key = 'secret shift number'. Length=1

Formally, we identify $\{\mathtt{A}, \mathtt{B}, \mathtt{C}, \ldots, \mathtt{Z}\} \cong \mathbb{Z}/26\mathbb{Z}$. Then $\mathcal{P} = \{(a_0, \ldots, a_n) \mid a_i \in \mathbb{Z}/26\mathbb{Z}, n \in \mathbb{N}\}$ and

$$E_k(a_1, \ldots, a_n) = (a_0 + k, \ldots, a_n + k)$$
$$D_k(c_1, \ldots, c_n) = (c_0 - k, \ldots, c_n - k).$$

(all operations are to be understood mod 26.)

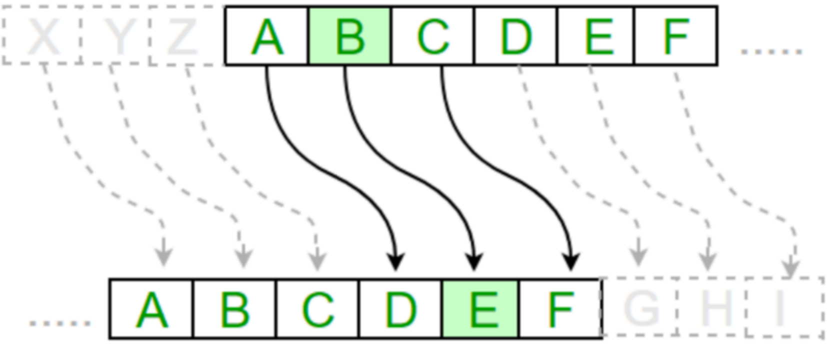

Figure 2: The Caeser Cipher shifts all letters by a fixed constant ("Caesar Cipher in Cryptography - Geeksforgeeks" 2020)

3.2 Simple Substitution Cipher

Replace each alphabet by another alphabet which is its random permutation. Key = a permutation of 26 letters. Length = 26

Formally, we identify $\{\mathtt{A}, \mathtt{B}, \mathtt{C}, \ldots, \mathtt{Z}\} \cong \mathbb{Z}/26\mathbb{Z}$. Then $\mathcal{P} = \{(a_0, \ldots, a_n) \mid a_i \in \mathbb{Z}/26\mathbb{Z}, n \in \mathbb{N}\}$ and

$$E_k(a_1, \ldots, a_n) = (\sigma(a_0), \ldots, \sigma(a_n))$$
$$D_k(c_1, \ldots, c_n) = (\sigma^{-1}(c_0), \ldots, \sigma^{-1}(c_n)),$$

where $\sigma \in Sym(26)$ is the encryption/decryption key.

Caeser Cipher is special case of Simple Substitution Cipher, where σ is a cyclic permutation.

The total key space has size 26!, which is greater than the number of stars in the universe.

Since every letter is encrypted by a unique letter, SSC is an example of a **monoalphabetic** cipher.

They are easily breakable by **letter frequency analysis**:

If the plaintext is known to be written in a specific language, we can compare letter frequencies of the language with the letter frequencies of the cipher text.

Thus, **large keyspace is necessary but not sufficient** for security!

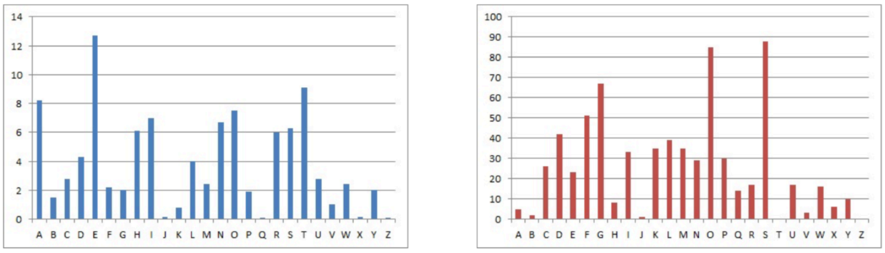

Figure 3: Comparing letter frequencies of known language with ciphertext ("Frequency Analysis: Breaking the Code - Crypto Corner" 2020)

3.3 Vigenère Cipher

Generate a key by repeating a given key until it matches the length of the plaintext. Replace each plaintext letter by another letter using a Caesar Cipher, whose key is the number associated to the corresponding letter of the generated key. Key = a string of letters.

Formally, we identify $\{\mathtt{A}, \mathtt{B}, \mathtt{C}, ..., \mathtt{Z}\} \cong \mathbb{Z}/26\mathbb{Z}$. If l is the length of the key, then $\mathcal{P} = \{(a_0, ..., a_n) \mid a_i \in \mathbb{Z}/26\mathbb{Z}, n \in \mathbb{N}\}$ and

$$E_k(a_1, ..., a_n) = (a_1 + k_{0 \mod l}, ..., a_n + k_{n \mod l})$$
$$D_k(c_1, ..., c_n) = (c_1 - k_{0 \mod l}, ..., c_n - k_{n \mod l}).$$

(all operations are to be understood $\mod 26$.)

Vigenère Cipher is an example of **polyalphabetic** ciphers (each given letter can be encrypted into l different letters).

The naive letter frequency analysis does not work, as the same letter is encrypted by different letters depending on its position.

If the key length is known, the Vigenère Cipher can be broken, breaking a sequence of Caesar Ciphers in strict rotation.

The key length can be guessed using simple analyses of the cipher text (Kasiski test, index of coincidence).

4 Computational complexity

Computational complexity of basic mathematical operations and of the exhaustive key search attack. Complexity classes of algorithms.

4.1 Turing Machines

4.1.1 Deterministic Turing machines

k-tape Turing machine is a triple $M = (A, Q, \tau)$ satisfying:

- A is a finite **alphabet** that the k tapes contain: $A = \{\square, \triangleright, 0, 1\}$;
- Q is a finite set of **states** that includes $q_{start}, q_{halt} \in Q$;
- $\tau : Q \times A^k \to Q \times A^{k-1} \times \{L, S, R\}^k$ is the **transition function** of M.

Tape is function $\mathbb{N} \to A$.

First tape is **input** tape (read-only), second to $k-1$-st tape are **work** tapes, k-th tape is **output** tape.

$\square$ is the **blank** symbol, $\triangleright$ is the **start** symbol

q_{start}, q_{halt} are the **start** and **halting** states

A **register** contains a current state

A **tape head** reads/write symbols, moves Left, Right or Stays

Explain these notions using fig. 4.

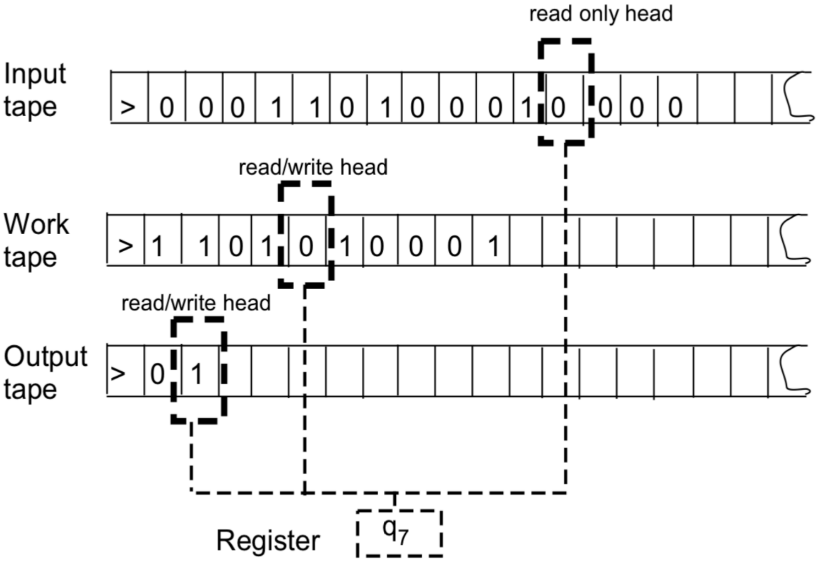

Figure 4: A Turing machine with 3 tapes (Arora and Barak 2009)

4.1.2 Probabilistic Turing machine

A Turing machine that may choose at every step a move at random according to a probability distribution.

Note that such a Turing machine is thus **non-deterministic**.

A probabilistic Turing machine M computes a function $f : \{0,1\}^* \to \{0,1\}^*$ if

$$P[f(x) = M(x)] \geq \tfrac{2}{3}$$

4.2 Running time

M computes $f : \{0,1\}^* \to \{0,1\}^*$ in $T(n)$-time, if for every input $x \in \{0,1\}^*$, M halts after at most $T(|n|)$ steps with output $f(x)$.

4.2.1 Running times of arithmetic operations

Operation	Complexity
Addition of two n-bit numbers	$O(n)$
Multiplication of two n-bit numbers	$O(n^2)$
Raising a number to an n-bit power	$O(n^3)$
Exhaustive key search for an n-bit key	$O(2^n)$

4.3 Decidablity and Verifiability

4.3.1 Decidablity

A language $L \subset \{0,1\}^*$ is **decidable** if there exists a deterministic Turing machine computing

$$\chi_L(x) := \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}.$$

4.3.2 Verifiability

A language $L \subset \{0,1\}^*$ is **recursively enumerable** if there exists a deterministic Turing machine computing a function

$$\nu_L : \{0,1\}^* \times \{0,1\}^* \to \{0,1\},$$

with the property that

$$x \in L \quad \Leftrightarrow \quad \exists c \in \{0,1\}^* : \nu_L(x,c) = 1.$$

We call c a **certificate** for x.

Note: Definiton in C2 is wrong!

4.4 Complexity classes

A problem instance x lies in the **complexity class**

- P if x is solvable by a polynomial **deterministic** algorithm.
- BPP if x is solvable by a polynomial **probabilistic** algorithm.
- BQP if x is solvable by a polynomial deterministic algorithm on a **quantum** computer.
- NP if x is **verifiable** by a polynomial deterministic algorithm.

It is conjectured that $P = BPP$ and Discrte Logarithm Problem as well as Factorization do not lie in $NP \cap BPP$.

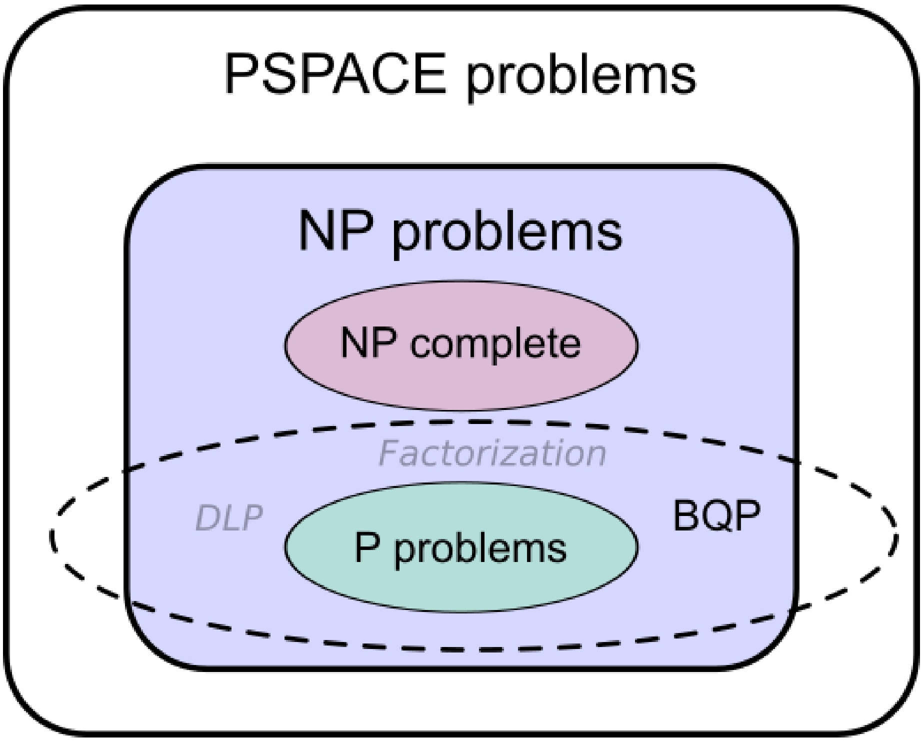

Figure 5: Known relations of complexities

5 Three types of security

Three types of security. Perfect secrecy: definition, examples, equivalent formulations (with proof). Perfect secrecy: Shannon's Theorem (with proof).

5.1 Types of security

- **Computational security:** computational complexity is high.
- **Provable security:** breaking the cryptosystem would solve a problem known to be hard.
- **Unconditional security:** breaking is not possible even if computational resources are unlimited.

5.2 Perfect secrecy

A cryptosystem has **perfect secrecy** if seeing the ciphertext gives not extra information about the plaintext.

A cryptosystem with perfect secrecy is **unconditionally secure** against a ciphertext only attack.

Let $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a cryptosystem and probability distributions are given on $\mathcal{P}$ and $\mathcal{K}$:

- $Pr[\mathbf{p} = p]$ denotes the probability that a plaintext $p \in \mathcal{P}$ occurs,
- $Pr[\mathbf{k} = k]$ denotes the probability that a key $k \in \mathcal{K}$ is chosen.

Analogously, $Pr[\mathbf{c} = c]$ denotes the probability that a ciphertext $c \in \mathcal{C}$ is transmitted.

5.2.1 Assumptions

- the key and the plaintext are independent random variables;
- each key is used for only one encryption.

5.2.2 Propability distribution on $\mathcal{C}$

By assumption we have for all $c \in \mathcal{C}$ that

$$Pr[\mathbf{c} = c] = \sum_{k \text{ with } c \in C(k)} Pr[[k] = k] Pr[\mathbf{p} = D_k c]$$

where $C(k) := \{E_k(p) : p \in \mathcal{P}\}$. This yields

$$Pr[\mathbf{c} = c \mid \mathbf{p} = p] = \sum_{k \text{ with } c = D_k(p)} Pr[\mathbf{k} = k]$$

Bayes' theorem implies (if $Pr[\mathbf{c} = c] > 0$) that

$$Pr[\mathbf{p} = p \mid \mathbf{c} = c] = \frac{Pr[\mathbf{p} = p]\ Pr[\mathbf{c} = c \mid \mathbf{p} = p]}{Pr[\mathbf{c} = c]}.$$

5.2.3 Definition

A cryptosystem has **perfect secrecy** if $Pr[\mathbf{p} = p \mid \mathbf{c} = c] = Pr[\mathbf{p} = p]$ for all $p \in \mathcal{P}, c \in \mathcal{C}$.

5.2.4 Equivalent formulations

TFAE:

1. $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ has perfect secrecy;
2. random variables p and c are independent;
3. $Pr[\mathbf{c} = c \mid \mathbf{p} = p] = Pr[\mathbf{c} = c]$;
4. $\forall p_1, p_2 \in \mathcal{P} : Pr[\mathbf{c} = c \mid \mathbf{p} = p_1] = Pr[\mathbf{c} = c \mid \mathbf{p} = p_2]$

The distribution of $\mathcal{P}$ is irrelevant!

Proof: WLOG, we assume that $Pr[\mathbf{c} = c], Pr[\mathbf{p} = p] > 0$ for all $c \in \mathcal{C}, p \in \mathcal{P}$. Otherwise, restrict $\mathcal{C}$ or $\mathcal{P}$.

$(1 \Rightarrow 2)$:

$$Pr[\mathbf{p} = p \cap \mathbf{c} = c] = Pr[\mathbf{p} = p \mid \mathbf{c} = c] Pr[\mathbf{c} = c] = Pr[\mathbf{p} = p] Pr[\mathbf{c} = c]$$

$(2 \Rightarrow 3)$:

$$Pr[\mathbf{c} = c \mid \mathbf{p} = p] = \frac{Pr[\mathbf{p} = p \cap \mathbf{c} = c]}{Pr[\mathbf{p} = p]} = \frac{Pr[\mathbf{p} = p] Pr[\mathbf{c} = c]}{Pr[\mathbf{p} = p]} = Pr[\mathbf{c} = c]$$

(3 ⇒ 4): $\forall c \in \mathcal{C}$ and $\forall p_1, p_2 \in \mathcal{P}$ we have

$$Pr[\mathbf{c} = c \mid \mathbf{p} = p_1] = Pr[\mathbf{c} = c] = Pr[\mathbf{c} = c \mid \mathbf{p} = p_2].$$

(4 ⇒ 1): By Bayes' Theorem we have

$$\begin{aligned}\frac{Pr[\mathbf{c} = c]Pr[\mathbf{p_1} = p \mid \mathbf{c} = c]}{Pr[\mathbf{p} = p_1]} &= Pr[\mathbf{c} = c \mid \mathbf{p} = p_1] = \\ &= Pr[\mathbf{c} = c \mid \mathbf{p} = p_2] = \\ &= \frac{Pr[\mathbf{c} = c]Pr[\mathbf{p_2} = p \mid \mathbf{c} = c]}{Pr[\mathbf{p} = p_2]}\end{aligned}$$

Thus, there exists a constant $\lambda \in \mathbb{R}$ such that

$$Pr[\mathbf{p} = p \mid \mathbf{c} = c] = \lambda Pr[\mathbf{p} = p].$$

We prove $\lambda = 1$ and are finished

$$\begin{aligned}Pr[\mathbf{p} = p] &= \sum_{c \in \mathcal{C}} Pr[\mathbf{p} = p \mid \mathbf{c} = c]Pr[\mathbf{c} = c] = \\ &= \sum_{c \in \mathcal{C}} Pr[\mathbf{p} = p]Pr[\mathbf{c} = c] = \\ &= \lambda Pr[\mathbf{p} = p] \underbrace{\sum_{c \in \mathcal{C}} Pr[\mathbf{c} = c]}_{=1}.\end{aligned}$$

5.2.5 Example: One-time pad

Let $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}/2\mathbb{Z})^n$ and $E_k(p) = k + p \mod 2$.

The key is k is chosen at random and may never be reused.

The One-time pad has perfect secrecy as

$$\forall p \in \mathcal{P}, \forall c \in \mathcal{C} : Pr[c \mid p] = \frac{1}{|\mathcal{K}|}$$

5.2.6 Shannon's theorem

Let $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a cryptosystem with $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$. Then it has perfect secrecy if and only if every key is used with equal probability $1/|\mathcal{K}|$, and $\forall p \in \mathcal{P}, \forall c \in \mathcal{C}$, there is a unique key $k \in \mathcal{K}$ such that $E_k(p) = c$.

Proof: (⇒): We can assume that $\forall p \in \mathcal{P} : Pr[p] > 0, \forall c \in \mathcal{C} Pr[c] > 0$. We have $Pr[c|p] = Pr[c] > 0$, that is, $\forall c \in \mathcal{C}$ there is at least one $p \in \mathcal{P}$ with $E_k(p) = c$.

Therefore, $|\mathcal{C}| = |\{E_k(p) \mid k \in \mathcal{K}\}| \leq |K|$ and, as $|\mathcal{K}| = |\mathcal{C}|$, there is no distinct $k_1 \neq k_2$ with $E_{k_1}(p) = E_{k_2}(p) = c$. That is, $\forall p \in \mathcal{P}, \forall c \in \mathcal{C}$, there is a unique key $k \in \mathcal{K}$ such that $E_k(p) = c$.

(Analogously, $|\mathcal{P}| \leq |\mathcal{K}|$.)

Let $n = |\mathcal{K}|, \mathcal{P} = \{p_1, ..., p_n\}$, and $c \in \mathcal{C}$ be fixed. Let $k_i \in \mathcal{K}$ be so that $E_{k_i}(p_i) = c$. Using Bayes' theorem:

$$Pr[p_i|c] = \frac{Pr[c|p_i]Pr[pi]}{Pr[c]} = \frac{Pr[k_i]Pr[p_i]}{Pr[c]}.$$

Perfect secrecy implies that $\forall i : Pr[k_i] = Pr[c]$, all keys are used with equal probability. Since there are $|\mathcal{K}|$ keys, the probability is $1/|\mathcal{K}|$.

($\Leftarrow$) $\forall p \in \mathcal{P}$, $\forall c \in \mathcal{C} : Pr[c|p] = 1/|K|$, hence, we conclude by the Proposition.

6 RSA cryptosystem

RSA cryptosystem: definition, examples, correctness (encryption and decryption are inverse operations). Parameter generation, its complexity. Main attacks.

7 One-way functions

One-way function, with trapdoor. Theorem: RSA keys vs Factoring (formulation and sketch of proof).

8 Hash functions

Hash function: definition, types of resistance, (non)-examples. Optimal asymmetric encryption padding.

9 Discrete logarithm problem

Discrete logarithm problem. The DLP assumption. The DLP in (Z/(p-1)Z, +) Is breaking the ECC cryptosystem equivalent to solving the DLP?

10 ElGamal cryptosystem

ElGamal cryptosystem and parameter generation: definition, correctness (encryption and decryption are inverse operations). Theorem: ElGamal keys versus DLP (with proof).

11 Elliptic curves

Elliptic curve: definition, singularities, normal forms, tangents. Theorem: intersection of E with a projective line (with proof).

12 Group structure on elliptic curves

> Group structure on the elliptic curve over the algebraic closure, geometrically: definition and theorem (with proof).

13 Cayley-Bacharach's theorem

> Cayley-Bacharach's theorem (with proof).

14 Associativity

> Associativity (sketch of proof).

15 Elliptic curves over finite fields

> Elliptic curves over finite fields: theorems (without proof) and examples. Check that for a prime q, each natural number in the Hasse interval occurs as the order of the el@liptic curve group over the field of q elements.

16 Diffie-Hellman key agreement protocol

> Diffie-Hellman key agreement: protocol, attacks. The DHP problem. The ECDHE.

17 Digital signature scheme

> Digital Signature Scheme. RSA signature algorithm. Attacks: definitions and examples.

To ensure the **non-repudiation** of data

17.1 Definition

Signature scheme is a 5-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, satisfying:

- $\mathcal{P}$ is a finite set of possible **messages**;
- $\mathcal{A}$ is a finite set of possible **signatures**;
- $\mathcal{K}$, the keyspace, is a finite set of possible **keys**;
- $\mathcal{S} = \{sig_k : k \in \mathcal{K}\}$ consists of polynomial signing algorithms $sig_k : \mathcal{P} \to \mathcal{A}$;
- $\mathcal{V} = \{ver_k : k \in \mathcal{K}\}$ consists of polynomial verification algorithms $ver_k : \mathcal{P} \times \mathcal{A} \to \{\texttt{true}, \texttt{false}\}$;

$$\forall x \in \mathcal{P}, \forall y \in A : ver_k(x, y) = \begin{cases} \texttt{true}, & \text{if } y = sig_k(x) \\ \texttt{false}, & \text{otherwise} \end{cases}$$

A pair (x, y) with $x \in \mathcal{P}$, $y \in \mathcal{A}$ is called a **signed message**.

17.2 RSA signature algorithm

Public-key cryptosystem	Digital Signature
Encrypt with E_k	Sign with D_k
Decrypt with D_k	Verify with E_k

RSA signature scheme is a 5-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ such that:

- $n = pq$, where p, q are primes,
- $\mathcal{P} = \mathcal{A} = \mathbb{Z}/n\mathbb{Z}$, and
- $\mathcal{K} = \{(n, p, q, d, e) : de = 1 \mod \phi(n)\}$
- For $k = (n, p, q, d, e)$, we define

$$sigk(x) = x^d \mod n \quad \text{and}$$

$$ver_k(x, y) = \begin{cases} \texttt{true}, & \text{if } x = y^e \mod n \\ \texttt{false}, & \text{otherwise.} \end{cases}$$

- Public-key is (n, e) and private-key is (p, q, d).

Note: By the defintion of DSS we should have:

$$\begin{aligned} ver_k(x, y) = \texttt{true} &\Leftrightarrow y = sig_k(x) = x^d \mod n \\ &\Leftrightarrow x = y^e \mod n \end{aligned}$$

Since $de = 1 \mod \phi(n)$, we have $de = t\phi(n) + 1$ for some $t \in \mathbb{Z}$. If $x \in (\mathbb{Z}/n\mathbb{Z})^*$, then

$$\begin{aligned} (x^e)^d = x^{t\phi(n)+1} \mod n = (x^{\phi(n)})^t x \mod n = \\ \underset{|(\mathbb{Z}/n\mathbb{Z})^*|=\phi(n)}{=} 1^t x \mod n = x \mod n \end{aligned}$$

If $x \notin (\mathbb{Z}/n\mathbb{Z})^*$, we know that $x \equiv 0 \mod p$ or $x \equiv 0 \mod q$ and uses Fermat's little theorem and the Chinese remainder theorem as in sec. 6.

17.3 Attacks on DSS

- **Key-only:** The attacker knows the public verification key, hence, verk .
- **Known message:** The attacker knows some messages (not selected by him) and their signatures.
- **Chosen message:** The attacker knows some messages (selected by him) and their signatures.

17.4 Goals of attacks on DSS

- **Total break:** The attacker determines Alice's private key, hence, sig_k.
- **Selective forgery:** With a non-negligible probability, the attacker creates a valid signature on a message chosen by someone else.

- **Existential forgery:** Forge a signature for some message (without the ability to do this for any message).
- **Universal forgery:** Forge signatures of any message.

17.5 Examples of attacks

1. *Existential forgery using key-only attack* is always possible: Choose an arbitrary signature y, then compute the message x given by $x := E_k(y)$.
   $\Rightarrow$ use **redundancy** or **hashing**.

2. *Universal forgery under a chosen message attack* is possible, if one-way function with trapdoor is multiplicative (e.g. RSA $(xy)^e = x^e \cdot y^e$). To sign $x = x_1 x_2$ trick Alice into signing x_1, x_2 to obtain y_1, y_2 and compute $(x, y) = (x, y_1 y_2)$.

18 DSS with hashing

> DSS with hashing. Hash functions from block ciphers: definition and example, with proof (the example where $(x, y) \to a^x b^y$).

DSS + Hashing = Hash-then-sign

18.1 Definition

DSS with hashing is a DSS 5-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ such that:

- $\mathcal{P} = \{0,1\}^*$ and
- $\mathcal{A} = \{0,1\}^l$ for some $l \in N$;
- $h : \mathcal{P} \to \mathcal{A}$ a public **hash function** given by a polynomial algorithm;
- $sig_k(x) = f_k^{-1}(h(x))$, where $f_k : A \to A$ is a one-way function with trapdoor.
- $\forall x \in P, \forall y \in A : ver_k(x, y) = \begin{cases} \texttt{true}, & \text{if } f_k(y) = h(x) \\ \texttt{false}, & \text{otherwise.} \end{cases}$

To avoid the attacks h must be a one-way **non-multiplicative** function. h is **collision resistant** if it is infeasible to find $x_1 \neq x_2$ with $h(x_1) = h(x_2)$.

A **block cipher** encodes blocks of bits at a time (e.g. Vigenère, Feistel).

18.2 Definition

Hash functions from block ciphers:

Let $\mathcal{P} = \mathcal{K} = \mathcal{C} = \{0,1\}^l$ for some $l \in \mathbb{N}$ and E be a block cipher:

$$E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}, \quad (x, e) \mapsto E_e(x).$$

Define $h(x_1, ..., x_r) \in \{0,1\}^l$ with $x_i \in \{0,1\}^l$ recursively (on r), by $h(\emptyset) = 0$, and

$$h(x_1, ..., x_r) = E_{e_h}(x_r) + e_h, \text{where } e_h = h(x_1, ..., x_{r-1}).$$

SHA-1 is an example of such a hash function.

19 DSS and Public-key cryptosystems

> DSS and Public-key cryptosystem: sign-then-encrypt versus encrypt-versus-sign.

19.1 Problem

> The use of symmetric keys involves an implicit indication of the originator and intended recipient of a message. By their very nature, this is not the case for use of public keys. — (Martin 2012, 244)

19.2 Sign-then-encrypt

19.2.1 Algorithm

1. Given $x \in \mathcal{P}$, Alice computes her signature $y = sig_{d_{Alice}}(x)$.
2. She encrypts both x and y using Bob's public key $z = E_{e_{Bob}}(x, y)$.
3. She sends z to Bob, who decrypts it $D_{d_{Bob}}(z) = (x, y)$.
4. He uses her public verification function to check whether $ver_{e_{Alice}}(x, y) =$ `true`.

19.2.2 Attack

Bob can forward messages form Alice to Charlie pretending that Alice wrote them directly.

1. Alice sends Bob a signed and encrypted Message z.
2. Bob decrypts the message and recovers the signature $D_{d_{Bob}}(z) = (x, y)$.
3. Bob encrypts the message and the signature using Charlie's public key $\tilde{z} = E_{e_{Charlie}}(x, y)$
4. Bob sends the message to Charlie, who decrypts it and verifies Alice's signature $ver_{e_{Alice}}(x, y) =$ `true`.

Charlie thinks

- that Alice was the origin of the data and (**true**)
- that nobody except Alice knows the content of the message (**false**)

19.3 Solution

Include the receiver's identity in the signed data.

19.4 Encrypt-then-sign

19.4.1 Algorithm

1. Alice encrypts the plaintext using Bob's public key $c = E_{e_{Bob}}(x)$
2. She then signs the ciphertext $y = sig_{d_{Alice}}(c)$.
3. She sends both c and y to Bob, who decrypts the ciphertext $D_{d_{Bob}}(c) = (x)$.
4. He uses her public verification function to check whether $ver_{e_{Alice}}(x, y) =$ `true`.

19.5 Attack

Charlie can intercept the message from Alice to Bob and pretend the message came from him.

1. Charlie intercepts the message form Alice to Bob and signs the ciphertext $\tilde{y} = sig_{d_{Charlie}}(c)$
2. Charlie sends both c and $\tilde{y}$ to Bob, who decrypts the ciphertext $D_{d_{Bob}}(c) = (x)$.
3. Bob uses Charlie's public verification function to check whether $ver_{e_{Charlie}}(x, y) = \texttt{true}$.

Bob thinks

- that Charlie was the origin of the data and (**false**)
- that nobody except Charlie knows the content of the message. (**false**)

19.6 Solution

Include the sender's identity in the encrypted data.

20 ElGamal variant of DSS (Definition)

> ElGamal variant of DSS: definition and correctness. Security assumptions. Example of misuse (with proof).

20.1 Definition

- Let p be a prime and g a primitive element $\mod p$.
- Let $\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^*, \mathcal{A} = (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/(p-1)\mathbb{Z})$ and define

$$\mathcal{K} = \{(p, g, d, y) : y = g^d \mod p\}.$$

- For $k = (p, g, d, y)$, and for a secrete random $r \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$, define

$$sig_k(x; r) = (y1, y2),$$

where

$$y_1 = g^r \mod p, \quad \text{and} \quad y_2 = (x - dy_1)r^{-1} \mod p - 1.$$

For $x, y_1 \in (\mathbb{Z}/p\mathbb{Z})^*$ and $y_2 \in \mathbb{Z}/(p-1)\mathbb{Z}$, define

$$ver_k(x, (y_1, y_2)) = \texttt{true} \Leftrightarrow y^{y_1}(y_1)^{y_2} \equiv g^x \mod p$$

- Public key is (p, g, y) and private key is d.

20.2 Correctness

We have to prove that

$$ver_k(x, sig_k(x; r)) = \texttt{true}$$

for all $k \in \mathcal{K}$, all $x \in \mathcal{P}$, and all $r \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$. Thus,

$$y^{y_1}(y_1)^{y_2} \equiv (g^d)^{y_1} g^{ry_2} \tag{1}$$

Since g is primitive $\mod p$ it has order $p-1$. Therefore,

$$g^{y_2} \equiv g^{(x-dy_1)r^{-1}} \mod p,$$

since

$$y_2 \equiv (x - dy_1)r^{-1} \mod p-1.$$

Plugging this into the RHS of eq. 1 yields

$$y^{y_1}(y_1)^{y_2} \equiv g^{dy_1} g^{r(x-dy_1)r^{-1}} \equiv g^x \mod p$$

as claimed.

20.3 Security assumptions

1. Approach: Choose arbitrary $y_1 \in (\mathbb{Z}/p\mathbb{Z})^*$ and try to find $y_2 \in (\mathbb{Z}/(p-1)\mathbb{Z})$. To do this one needs to solve

   $$y_2 \equiv \log_{y_1} g^x y^{-y_1} \mod p,$$

   i.e. one solves the DLP in $(\mathbb{Z}/p\mathbb{Z})^*$.
2. Approach: Choose arbitrary $y_2 \in (\mathbb{Z}/(p-1)\mathbb{Z})$ and try to find $y_1 \in (\mathbb{Z}/p\mathbb{Z})^*$. To do this one needs to solve

   $$y^{y_1}(y_1)^{y_2} \equiv g^x \mod p.$$

Assumption: Both problems do not lie in BPP.

20.4 Example of misuse

If the same k is used twice a total break is possible.

Proof: Let (y_1, y_2) a signature of x_1 and (y_1, z_2) a signature of x_2. Then

$$y^{y1}(y_1)^{y_2} \equiv g^{x_1} \mod p, \quad y^{y1}(y_1)^{z_2} \equiv g^{x_2} \mod p,$$

thus,

$$g^{x_1 - x_2} \equiv (y_1)^{y_2 - z_2)} modp.$$

Substituting $y_1 \equiv g^r \mod p$ yields an equation in the single **unknown** r. By Fermat's little theorem this is equivalent to

$$x_1 - x_2 \equiv r(y_2 - z_2) \mod p-1. \tag{2}$$

If $(y_2 - z_2)$ is invertible $\mod p-1$, we divide by $(y_2 - z_2)$ and are done. Otherwise, set $s := \gcd(y_2 - z - 2, p-1)$ and note that $s \mid x_1 - x_2$. We set

$$x' := \frac{x_1 - x_2}{s}, \quad y' := \frac{y_2 - z_2}{s}, \quad p' := \frac{p-1}{s}.$$

Then eq. 2 becomes

$$x' \equiv ry' \mod p'.$$

Since $\gcd(y', p') = 1$, we obtain $r \equiv x'(y')^{-1} \mod p'$.

This yields s possible values for $r \mod p-1$, namely

$$r_i := x'(y')^{-1} + ip' \mod p-1, \text{ for } 0 \leq i \leq s-1.$$

The correct value is obtained by testing

$$y_1 \equiv g^{r_i} \mod p.$$

To determine the private key d, we modify the defining equation for y_2 and obtain

$$dy_1 \equiv x - ry_2 \mod p-1.$$

If y_1 is invertible $\mod p-1$ we divide by y_1, otherwise we proceed as above.

21 ElGamal and ECDSA variants of DSS

ElGamal variant of DSS: example of misuse (with proof). ECDSA: definition and correctness.

21.1 Definition

- Let p be a prime and g a primitive element $\mod p$.
- Let $\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^*, \mathcal{A} = (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/(p-1)\mathbb{Z})$ and define

  $$\mathcal{K} = \{(p, g, d, y) : y = g^d \mod p\}.$$

- For $k = (p, g, d, y)$, and for a secrete random $r \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$, define

  $$sig_k(x; r) = (y1, y2),$$

  where

  $$y_1 = g^r \mod p, \quad \text{and} \quad y_2 = (x - dy_1)r^{-1} \mod p-1.$$

  For $x, y_1 \in (\mathbb{Z}/p\mathbb{Z})^*$ and $y_2 \in \mathbb{Z}/(p-1)\mathbb{Z}$, define

  $$ver_k(x, (y_1, y_2)) = \texttt{true} \Leftrightarrow y^{y_1}(y_1)^{y_2} \equiv g^x \mod p$$

- Public key is (p, g, y) and private key is d.

21.2 Example of misuse

If the same k is used twice a total break is possible.

Proof: Let (y_1, y_2) a signature of x_1 and (y_1, z_2) a signature of x_2. Then

$$y^{y1}(y_1)^{y_2} \equiv g^{x_1} \mod p, \quad y^{y1}(y_1)^{z_2} \equiv g^{x_2} \mod p,$$

thus,

$$g^{x_1 - x_2} \equiv (y_1)^{y_2 - z_2)} mod p.$$

Substituting $y_1 \equiv g^r \mod p$ yields an equation in the single **unknown** r. By Fermat's little theorem this is equivalent to

$$x_1 - x_2 \equiv r(y_2 - z_2) \mod p - 1. \tag{3}$$

If $(y_2 - z_2)$ is invertible $\mod p - 1$, we divide by $(y_2 - z_2)$ and are done. Otherwise, set $s := \gcd(y_2 - z - 2, p - 1)$ and note that $s \mid x_1 - x_2$. We set

$$x' := \frac{x_1 - x_2}{s}, \quad y' := \frac{y_2 - z_2}{s}, \quad p' := \frac{p-1}{s}.$$

Then eq. 3 becomes

$$x' \equiv ry' \mod p'.$$

Since $\gcd(y', p') = 1$, we obtain $r \equiv x'(y')^{-1} \mod p'$.

This yields s possible values for $r \mod p - 1$, namely

$$r_i := x'(y')^{-1} + ip' \mod p - 1, \text{ for } 0 \leq i \leq s - 1.$$

The correct value is obtained by testing

$$y_1 \equiv g^{r_i} \mod p.$$

To determine the private key d, we modify the defining equation for y_2 and obtain

$$dy_1 \equiv x - ry_2 \mod p - 1.$$

If y_1 is invertible $\mod p - 1$ we divide by y_1, otherwise we proceed as above.

21.3 Definition: ECDSA, hash-and-sign

- p prime, $\mathbf{k} = (\mathbb{Z}/p\mathbb{Z})$, $E = E(\mathbf{k})$, $P \in E$ of prime order q.
- $\mathcal{P} = \{0,1\}^*, \mathcal{A} = (\mathbb{Z}/q\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*$ and

  $$\mathcal{K} = \{(p, q, E, P, d, Q) : Q = dP\},$$

  where $0 \leq d \leq q - 1$.
- For $k = (p, q, E, P, dQ)$ and a secret random $r \in \{1, ..., q - 1\}$, define

  $$sig_k(x, r) = (t, s)$$

  where $rP = (u, v)$ and

  $$t = u \mod q,$$
  $$s = r^{-1}(h(x) + dt) \mod q.$$
- If either $t = 0$ or $s = 0$, a new random value of r is chosen.
- The public key is (p, q, E, P, Q) and the private key is d.

21.3.1 Verification of ECDSA

For $x \in \{0,1\}^*$ and $t, s \in (\mathbb{Z}/q\mathbb{Z})^*$, we compute

$$\begin{aligned} w &= s^{-1} \mod q \\ i &= wh(x) \mod q \\ j &= wt \mod q \\ (u,v) &= iP + jQ \end{aligned}$$

and define

$$ver_k(x,(t,s)) = \texttt{true} \Leftrightarrow u \mod q = t$$

To prove correctness, show that $iP + jQ = rP$.

22 Digital currency

Digital currency: definition and security requirements. Distributed ledgers. Blockchain. Security assumptions underlying the generation of the bitcoin address.

23 Bitcoin transactions

Bitcoin transaction and its verification. Merkle tree. Bitcoin mining.

24 Bit generators

Bit generator. Linear feedback shift register: definition, periods, security. RSA bit generator.

25 Distinguisher and next bit predictors

Distinguisher. Next bit predictor. Yao's theorem (sketch of proof).

26 Error-correcting codes and expander graphs

Error-correcting codes and expander graphs

27 Probabilistic pidgeonhole principle

Describe the probabilistic pidgeonhole principle and explain, with examples, why it is relevant in cryptography (i.e hash functions, birthday paradox etc).

28 Attacks on cryptosystems relying on structural weaknesses

Describe a variety of attacks that rely on structural weaknesses in respective cryptosystems (for instance, known message attacks for multiplicative systems, or weaknesses of El Gamal under weak random choices).

29 Shanks algorithm

Describe Shanks algorithm, give examples of its use and outline how to use Shanks Algorithm to compute the order of an elliptic curve of prime order in combination with Hasse's bound.

30 TODO-s:

- □ Sec. 18: Example where $(x, y) \to a^x b^y$.
- □ Sec. 3: If the given key of a Vigenère Cipher has repeated letters, does it make it any easier to break?

References

Arora, Sanjeev, and Boaz Barak. 2009. *Computational Complexity*. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511804090.

Arzhantseva, Goulnara. 2019. "Exam Questions." https://www.mat.univie.ac.at/~gagt/crypto2019/ExamQuestions.pdf.

"Caesar Cipher in Cryptography - Geeksforgeeks." 2020. January 23, 2020. https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/.

"Frequency Analysis: Breaking the Code - Crypto Corner." 2020. January 23, 2020. https://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html.

Martin, Keith M. 2012. *Everyday Cryptography: Fundamental Principles and Applications*. Oxford University Press.