Introduction to Modern Cryptography Summary

WS20/21

Valentin Knappich

February 1, 2021

Contents

1 Symmetric encryption

Kerkhoffs Principle: The security of a system should only depend on whether the actual key is secret, not on the system itself. The whole system is assumed to be public. No "Security by obscurity".

1.1 Scenario 1

One message with constant length

1.1.1 Cryptosystems

A cryptosystem is a tuple $\mathcal{S} = (X, K, Y, e, d)$ with

- X: set of plaintexts
- K: finite set of keys
- Y: set of ciphertexts
- e: encryption function
- d: decryption function

Perfect correctness:	$d(e(x,k),k) \quad \forall x \in X, k \in K$
No unnecessary ciphertexts:	$Y = \{e(x,k) \mid x \in X, k \in K\}$

1.1.2 Vernam system

The Vernam cryptosystem of length l is defined as $(\{0,1\}^l, \{0,1\}^l, \{0,1\}^l, e, d)$ where

$e(x,k) = x \oplus k$ and $d(y,k) = y \oplus k$.

A vernam system of length $l > 0$ provides perfect secrecy for every uniform P_K. It is the perfect system for Scenario 1.

1.1.3 Perfect Secrecy

A cryptosystem with key distribution $\mathcal{V} = \mathcal{S}[P_k]$ provides perfect secrecy if for all plaintext distributions P_X, the probability of every plaintext remains the same after the ciphertext is seen, i.e.:

$$P(x) = P(x|y) \quad \forall x \in X, y \in Y, P(y) > 0$$

Example Proof:

We need to show the criteria above for all plaintext distributions P_X. Therefore we use variable probabilities for the plaintexts $P_X(a) = p, P_X(b) = 1 - p$ (for 2 plaintexts, else $p_1, ..., p_n$).

$K \backslash X$		a	b
$\frac{1}{2}$	k_0	A	B
$\frac{1}{2}$	k_1	B	A

$$
\begin{aligned}
P(a|A) &= \frac{P(a,A)}{P(A)} &&= \frac{\frac{1}{2} * p}{\frac{1}{2} * p + \frac{1}{2} * (1-p)} = p = P(a) \\
P(a|B) &= \frac{P(a,B)}{P(B)} &&= \frac{\frac{1}{2} * p}{\frac{1}{2} * p + \frac{1}{2} * (1-p)} = p = P(a) \\
P(b|A) &= \frac{P(b,A)}{P(A)} &&= \frac{\frac{1}{2} * (1-p)}{\frac{1}{2} * (1-p) + \frac{1}{2} * p} = 1-p = P(b) \\
P(b|B) &= \frac{P(b,B)}{P(B)} &&= \frac{\frac{1}{2} * (1-p)}{\frac{1}{2} * (1-p) + \frac{1}{2} * p} = 1-p = P(b)
\end{aligned}
$$

Theorem:

Let $\mathcal{S} = (X, K, Y, e, d)$ be a cryptosystem providing perfect secrecy, then it holds $|K| \geq |Y| \geq |X|$.

Shannons Theorem:

Let $\mathcal{V} = \mathcal{S}[P_k]$ be a cryptosystem with key distribution P_K and $|K| = |Y| = |X|$. The system provides perfect secrecy if and only if

1. P_K is a uniform distribution
2. $\forall x \in X, y \in Y \exists k \in K$ with $e(x,k) = y$ (There must be a key for every plaintext/ciphertext pair)

1.2 Scenario 2

Multiple messages with constant length, no repetition

1.2.1 Vernam in Scenario 2

Vernam is not a secure cryptosystem anymore, since from 2 ciphertexts, Eve can learn non-trivial information about the plaintexts:

$$y_0 \oplus y_1 = x_0 \oplus k \oplus x_1 \oplus k = x_0 \oplus x_1$$

Also with 1 plaintext-ciphertext pair (CPA), the key can be calculated as $k = x \oplus y$.

1.2.2 Substitution Cryptosystem

Let X be a non-empty finite set. A substitution cryptosystem over X is a tuple (X, P_X, X, e, d) where P_X is the set of all permutations of X.

$$e(x, \pi) = \pi(x) \quad d(y, \pi) = \pi^{-1}(y) \quad \forall x, y \in X, \pi \in P_X$$

Substitution cryptosystems provide “perfect security” in scenario 2, but they are impractical because the substitution table (π) has a size of $2^l * l$.

1.2.3 l-Block Cipher

Let $l : \mathbb{N} \to \mathbb{N}$ be a polynomial. An l-block cipher B is a cryptosystem of the form $\left(\{0,1\}^{l(\eta)}_{\eta\in\mathbb{N}},\ Gen(1^\eta),\ \{0,1\}^{l(\eta)}_{\eta\in\mathbb{N}},\ E,\ D\right)$ or simplified: $\left(\{0,1\}^{l},\ Gen(1^\eta),\ \{0,1\}^{l},\ E,\ D\right)$

1.2.4 Substitution-Permutation Cryptosystem (SPCS)

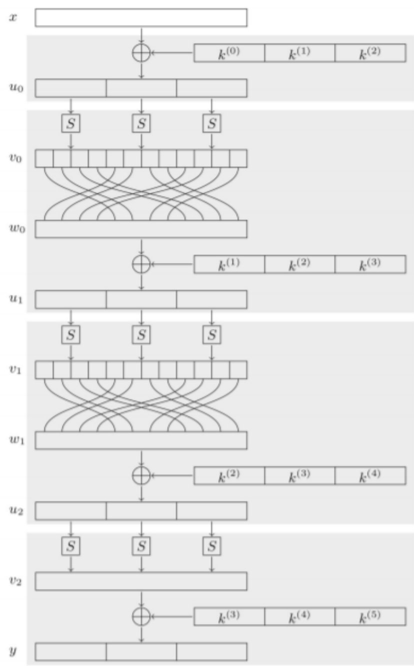
Notation:

- plaintexts are split into m words with length n with $l = m * n$, $x^{(i)}$ denotes the i'th word
- $[r] = \{0, 1, ..., r-1\}$
- $\beta \in \mathcal{P}_{[l]}$, then $x^\beta(i) = x(\beta(i))$

General Principle: Over r rounds, (round) key additions, word substitutions and bit permutations are applied, including an initial step that just applies key addition and shortened last round without bit permutation.

$E(x : \{0,1\}^{mn}, k : \{0,1\}^{s}) : \{0,1\}^{mn}$

1. *initial white step (round key addition)*
   $u = x \oplus K(k, 0)$
2. $r-1$ *regular rounds*
   for $i = 1$ to $r-1$ do
   a. *word substitutions*
      for $j = 0$ to $m-1$ do
         $v^{(j)} = S(u^{(j)})$
   b. *bit permutation*
      $w = v^\beta$
   c. *round key addition*
      $u = w \oplus K(k, i)$
3. *shortened last round (without bit permutation)*
   for $j = 0$ to $m-1$ do
      $v^{(j)} = S(u^{(j)})$
   $y = v \oplus K(k, r)$; return y



Known Attacks:

- Brute Force Attack
- Linear Cryptanalysis
- Differential Cryptanalysis

Linear Cryptanalysis:

- Relies on a set T of plaintext-ciphertext pairs
- Instead of brute forcing the whole key, get small parts of the key at a time
- Exploit linear dependencies

AES (Advanced encryption standard): basically SPCS with modifications

1.2.5 Algorithmic Security of Block Ciphers

We consider a block cipher secure if it is almost as good as a substitution cryptosystem w.r.t. resource-bound adversaries. Therefore no adversary U should be able to distinguish BCS and SCS. Formally, we use the BCS for $b = 1$ (real world) and the SCS for $b = 0$ (random world) in the security game.
The winning probability is $Pr[\mathbb{E}(1^n) = 1]$. Since a random guesser already has a probability of 0.5, the advantage is normalized.

$\mathbb{S}(1^\eta) : \{0,1\}$

1. *Choose real world or random world.*
   $b \xleftarrow{\$} \{0,1\}$
   if $b = 1$ then
   $\quad k \xleftarrow{\$} \mathsf{Gen}(1^\eta)$ and $F = E(\cdot, k)$
   else
   $\quad F \xleftarrow{\$} \mathcal{P}_{\{0,1\}^{l(\eta)}}$
2. *Guess phase.*
   $b' \xleftarrow{\$} U(1^\eta, F)$
3. *Output.*
   return b'.

$$Adv_{U,B}(\eta) = 2 * \left(Pr[\mathbb{E}_U^B(1^\eta) = 1] - \frac{1}{2} \right) \in [-1, 1] \qquad suc_{U,B}(\eta) = Pr[\mathbb{S}_U^B\langle b = 1\rangle(1^\eta) = 1]$$

$$Adv_{U,B}(\eta) = suc_{U,B}(\eta) - fail_{U,B}(\eta) \qquad fail_{U,B}(\eta) = Pr[\mathbb{S}_U^B\langle b = 0\rangle(1^\eta) = 1]$$

1.2.6 PRP/PRF Switching Lemma

Since substitution cryptosystems cannot be distinguished from (secure) l-Block cryptosystems, we can see l-Block cryptosystems as pseudo-random permutations (PRP). Anyway, for proving purposes, it can be easier to see them as pseudo-random functions. The PRP/PRF Switching Lemma says, that we can use them interchangeably, since the difference of advantages is negligible:

Let B be an l-block cipher and U be an l-distinguisher with runtime bound $q(\eta)$ where q is a positive polynomial and $\eta \in \mathbb{N}$. Then the following holds true:

$$|Adv_{U,B}^{PRP}(\eta) - Adv_{U,B}^{PRF}(\eta)| \leq \frac{q(\eta)^2}{2^{l(\eta)+1}}$$

1.3 Scenario 3

Arbitrary messages with any length (possibly with repetition)

1.3.1 Symmetric Encryption Scheme

A symmetric encryption scheme is a tuple $S = (Gen(^\eta), E, D)$ with

- security parameter η
- ppt key generation algorithm $Gen(1^\eta)$
- ppt encryption algorithm $E(x : \{0,1\}^*, k : K) : \{0,1\}^*$
- dpt decryption algorithm $D(y : \{0,1\}^*, k : K) : \{0,1\}^*$
- and $D(E(x,k),k) = x$

E cannot be deterministic, because else we wouldn't be able to send the same message multiple times, i.e. the same plaintext encrypted under the same key should result in a different ciphertext (with a high probability).

1.3.2 Encryption Schemes from Stream Ciphers

Idea: Vernam is safe if we use every key just once. So using the key as seed of a random number generator, that generates a stream of random numbers, enables the usage of the vernam system for arbitrarily long messages.

1.3.2.1 Number generator

A number generator (NG) is a dpt algorithm of the Form $G : (s : \{0,1\}^\eta) : \{0,1\}^{p(\eta)}$ where p is the expansion factor.

1.3.2.2 PRNG-Distinguisher

TODO

1.3.3 Encryption Schemes from Block Ciphers

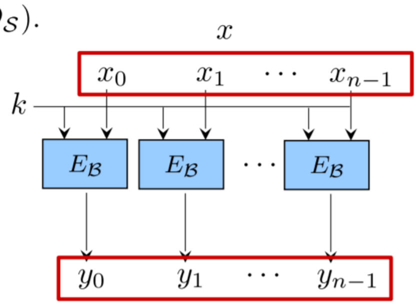
1.3.3.1 ECB Mode

Idea: Split the message in blocks of constant length and encrypt each block under the given key using the underlying block cipher.

$$\mathcal{S} = \textsf{ECB-}\mathcal{B} = (\textsf{Gen}_{\mathcal{B}}(1^\eta), E_{\mathcal{S}}, D_{\mathcal{S}}).$$

$E_{\mathcal{S}}(x : \{0,1\}^{l(\eta)+}, k : K_{\mathcal{B}}) : \{0,1\}^*$:

1. Split x into several blocks of length $l(\eta)$:
   $x =: x_0 || \cdots || x_{n-1}, n \in \mathbb{N}, x_i \in \{0,1\}^{l(\eta)}$
2. $y_i = E_{\mathcal{B}}(x_i, k) \quad \forall i \in \{0, \dots, n-1\}$
3. **return** $y := y_0 || \dots || y_{n-1}$

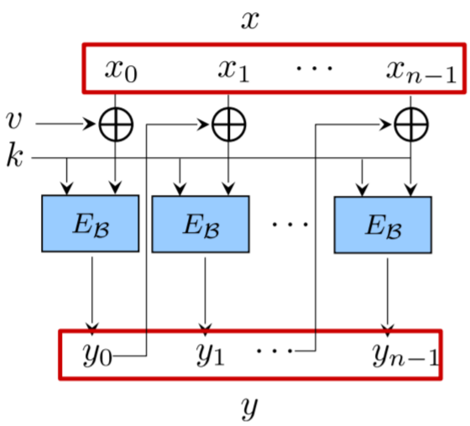


Security: It's not secure, since the ciphertext carries non-trivial information about the plaintext: for $y = y_0||y_1$, then $y_0 = y_1$ if $x_0 = x_1$.

1.3.3.2 CBC Mode

Idea: Add and initialization vector v that is `xor`'ed with the plaintext before encrypting. That v is part of the key.

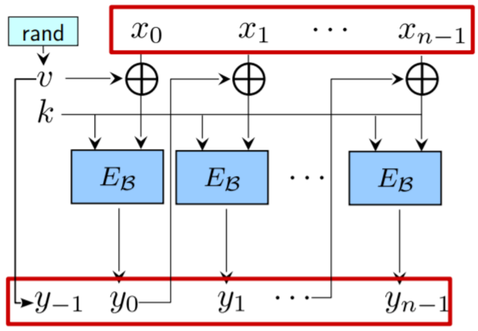
Problem: Still deterministic, so every plaintext can be sent just once.



1.3.3.3 R-CBC Mode

Idea: To solve the issues of CBC-Mode, R-CBC moves the initialization vector v out of the key and generates a random one while decryption. The vector is appended as first block of the ciphertext to enable decryption.

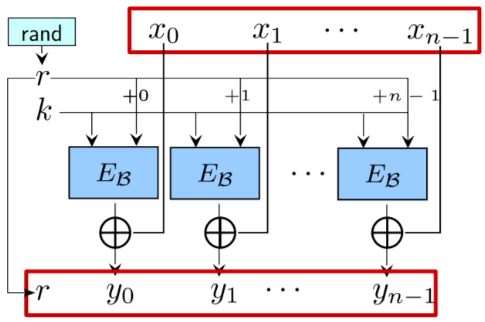
Security: Its secure if the underlying block cipher is secure.



1.3.3.4 R-CTR Mode

Idea: Alternative to R-CBC. Generate a random number r (comparable to v of R-CBC), encrypt this random number under the key and xor it with the plaintext. The counter is increased by 1 for each block. The counter r is appended as first block of y to enable decryption.

Security: Its secure if the underlying block cipher is secure.



1.3.4 CPA-Security

CPA: Chosen-Plaintext-Attack

Game: Adversary A consists of finder AF and guesser AG. The finder chooses 2 plaintexts z_0, z_1. One of them is encrypted. The guesser has to determine which of them is the corresponding plaintext.

Advantage, success and failure are defined as for block ciphers.

$\mathbb{E}(1^\eta) : \{0,1\}$

1. *Choose cipher.*
   $k \xleftarrow{\$} \mathsf{Gen}(1^\eta)$; $H = E(\cdot, k)$
2. *Find phase.*
   $(z_0, z_1) \xleftarrow{\$} AF(1^\eta, H)$
3. *Selection.*
   $b \xleftarrow{\$} \{0,1\}$; $y \xleftarrow{\$} H(z_b)$
4. *Guess phase.*
   $b' \xleftarrow{\$} AG(1^\eta, H, y)$
5. *Evaluation.*
   if $b' = b$, return 1, otherwise 0.

1.3.5 CCA-Security

CCA: Chosen-Ciphertext-Attack

Game: In addition to the encryption oracle H from the CPA-game, the adversary also gets a decryption oracle H^{-1}.

Advantage, success and failure are defined as for block ciphers.

$\mathbb{E}(1^\eta) : \{0,1\}$

1. *Choose cipher.*
   $k \xleftarrow{\$} \mathsf{Gen}(1^\eta)$; $H = E(\cdot, k)$
2. *Find phase.*
   $(z_0, z_1) \xleftarrow{\$} AF(1^\eta, H)$
3. *Selection.*
   $b \xleftarrow{\$} \{0,1\}$; $y \xleftarrow{\$} H(z_b)$
4. *Guess phase.*
   $b' \xleftarrow{\$} AG(1^\eta, H, y)$
5. *Evaluation.*
   if $b' = b$, return 1, otherwise 0.

1.3.6 Vaudenay's Padding Attack

- TODO

2 Number Theory

2.1 Fundamental Theorem of Arithmetic

Every natural number $n \in \mathbb{N}, n \geq 2$ has exactly one combination of prime factors.

$$n = p_1 * \dots * p_k \quad \text{with} k \leq log(n)$$

2.2 Modulo

Let $n \in \mathbb{N} \backslash \{0\}, a \in \mathbb{Z}$. Then $\exists ! q \in \mathbb{Z}, r \in \{0, \dots, n-1\}$ such that $a = n * q + r$.

$$a \text{ div } n := q \qquad \text{and} \qquad a \bmod n := r$$

2.3 $\mathbb{Z}_n$

Let $n \geq 1$. We define the set $\mathbb{Z}_n := \{0, \dots, n-1\}$ of remainders of divisions by n. Let $a, b \in \mathbb{Z}_n$, then

$$a +_n b := (a + b) \bmod n \qquad \text{and} \qquad a *_n b := (a * b) \bmod n$$

2.4 Group

A tuple $(\mathcal{G}, \cdot)$ is called group if $\mathcal{G}$ is a non-empty set and $\cdot : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}$ is a function such that:

- $(x \cdot y) \cdot z = x \cdot (y \cdot z) \quad \forall x, y, z \in \mathcal{G}$ (associativity)
- $\exists e \in \mathcal{G} : e \cdot x = x \cdot e = x \quad \forall x \in \mathcal{G}$ (neutral element)
- $\forall x \in \mathcal{G} \exists x^{-1} \in \mathcal{G} : x \cdot x^{-1} = e$ (inverse element)

The *order* of a group is the number of elements in $\mathcal{G}$.

The exponentiation is defined as usual. For a finite group $(\mathcal{G}, \cdot)$ with order n and neutral element e, the following holds true:

$$g^n = e \qquad \text{and} \qquad g^a = g^{a \bmod n}$$

2.5 Ring

A Ring is the tuple $(\mathcal{R}, +, \cdot)$ if $(\mathcal{R}, +)$ is an abelian (commutative) group and the function $\cdot : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$ is associative, distributive and has a neutral element.

The set of invertible elements in $\mathcal{R}$ is denoted by $\mathcal{R}^*$. The tuple $(\mathcal{R}^*, \cdot)$ is an abelian group called group of units.

2.6 Greatest common divisor

We say a divides b or $a|b$ if $\exists c \in \mathbb{Z} : b = c \cdot a$. The greatest common divisor is definded as

$$gcd(a,b) = max\{c : c|a \text{ and } c|b\} \text{ where } gcd(0,0) := 0$$

The set of invertible elements of $\mathbb{Z}_n$ can be determined by the gcd.

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | gcd(a,n) = 1\}$$

2.7 Eurler's Totient Function

Let $n \geq 2$. The Euler's totient function is defined by

$$\Phi(n) = |\mathbb{Z}_n^*| = (p_0 - 1) \cdot p_0^{\alpha_0 - 1} \dots (p_{r-1} - 1) \cdot p_{r-1}^{\alpha_{r-1}-1}$$

where $p_1, \dots, p_{r-1}$ are primes and $n = p_0^{\alpha_0} \dots p_{r-1}^{\alpha_{r-1}}$. Let p be a prime, then $\Phi(p) = p - 1$.

It can also be used to calculate the number of generators in a cyclic group as $\Phi(n)$ where $n = |\mathcal{G}|$.

2.8 Euclids Algorithm

Algorithm to calculate the gcd. Can be extended to calculate the inverse of an element in $\mathbb{Z}_n^*$.

1. Initialize loop:
   $a' = a, b' = b$

2. Loop: compute gcd by using that gcd(a,b) = gcd$(b, a \bmod b)$ for $b > 0$.
   Invariant: gcd(a,b) = gcd(a', b') and $a' \geq b' \geq 0$.
   while $b' \neq 0$:
   #Do one reduction step.
   $q = a'$ div b', $r = a'$ mod b'
   $a' = b', b' = r$

2.9 Fast Exponentiation

Algorithm to efficiently compute the exponentiation of a group element. Let $\mathcal{G}$ be a group and $g \in \mathcal{G}, m \in \mathbb{N}$. It uses the fact, that $g^{2k} = (g^k)^2$. Instead of doing $2k$ multiplications, we can do $k+1$. This is applied recursively to minimize the number of exponentiations that need to be computed. To make the algorithm work with any k (not just powers of 2), we use the binary representation of the exponent, e.g.

$$13 = 2^0 + 2^2 + 2^3 = (1101)_2 \quad \Rightarrow \quad g^{13} = g^{2^0} \cdot g^{2^2} \cdot g^{2^3}$$

To compute g^m, the algorithm iterates over the bits of m. If the bit is one, multiply the result with the current factor. In any case, square the current factor.
The algorithm has a complexity of $\mathcal{O}(log(m))$.

1. Initialization:
   $i = l; h = 1; k = g$
2. Iterated squaring:
   while $i \geq 0$:
   if $b(i) = 1$
   $h = kh$
   $k = k^2$
   $i = i - 1$
3. Output:
   return h

Postcondition: $g^m = h$

2.10 Cyclic Groups

A group $\mathcal{G}$ is called cyclic, iff $\exists g \in \mathcal{G}$ such that $\langle g \rangle = \mathcal{G}$.

If $p = |\mathcal{G}|$ is prime, then $\mathcal{G}$ is a cyclic group.

$\mathbb{Z}_p^*$ is a cyclic group if p is prime.

2.10.1 Subgroups

Let $(\mathcal{G}, \cdot)$ be a finite group and $U \subseteq \mathcal{G}$.

Definition: The tuple $(U, \cdot)$ is a subgroup of $\mathcal{G}$ iff U is a group.

Lemma: The tuple $(U, \cdot)$ is a subgroup of $\mathcal{G}$ iff $1 \in U$ and $a \cdot b \in U \quad \forall a, b \in U$

Lagranges Theorem: If U is a subgroup of $\mathcal{G}$, then it holds true that $|U| \mid |\mathcal{G}|$.

2.10.2 Generated Groups and Generators

Let $\mathcal{G}$ be a group and $g \in \mathcal{G}$. By $\langle g \rangle$ we denote the smallest subgroup of $\mathcal{G}$ that contains g.

$$\langle g \rangle = \{1, g, g^{-1}, g^2, g^{-2}, \dots\} \quad \text{and if } \mathcal{G} \text{ if inite: } \langle g \rangle = \{1, g, g^2, \dots, g^{|\langle g \rangle|-1}\}$$

We call g a generator of $\mathcal{G}$ if $\langle g \rangle = \mathcal{G}$.

2.10.3 Finding Generators

We find generators for a group by guessing a group element and checking whether or not it is a generator. This can be evaluated by the equation

$$g^{n/p} \neq 1 \quad \forall \quad p \in P \text{ (prime factors of } n) \text{ and } n = |\mathcal{G}|$$

FindGenerator($\mathcal{G}$)
Precondition: $\mathcal{G}$ is a cyclic group.
loop
  1. Select an element out of $\mathcal{G}$ at random
  $g \xleftarrow{\$} \mathcal{G}$
  2. Test whether g is a generator of $\mathcal{G}$.
  if GeneratorTest(g) outputs "g is a generator of $\mathcal{G}$." **then**
    return n
  end if
end loop

GeneratorTest($\mathcal{G}, g, n, P$)
Precondition: $\mathcal{G}$ a finite group, $g \in \mathcal{G}$, $n = |\mathcal{G}|$, $P =$ set of prime factors of $|\mathcal{G}|$.
  For $p \in P$ do
    $h =$ FastExponentiation($\mathcal{G}, g, n/p$)
    If $h = 1$
      break and **return** "g is not a generator of $\mathcal{G}$."
  return: "g is generator of $\mathcal{G}$".

This only works, because the probability of finding a generator when randomly sampling is relatively high, concretely it is upper-bounded by $Pr[\langle X_g \rangle = \mathcal{G}] \geq \frac{1}{1+log(n)}$ with $X_g \xleftarrow{\$} \mathcal{G}$.

2.10.4 Quadratic Residues

A number $a \in \mathbb{Z}_n^*$ *quadratic residue modulo n* iff there exists $b \in \mathbb{Z}_n$ such that $b^2 \bmod n = a \bmod n$. b is called root of a. All quadratic residues modulo n are denoted by $QR(n)$ and all elements that are *quadratic nonresidue modulo n* $QNR(n)$ with $\mathbb{Z}_n^* = QR(n) \cup QNR(n)$. For prime p, both sets have the same order:

$$|QR(p)| = |QNR(p)| = \frac{p-1}{2}$$

2.10.5 Eulers criterion

Let $p > 2$ be a prime number, $a \in \mathbb{Z}$ and $e = a^{(p-1)/2} \bmod p$

1. $e \in \{0, 1, p-1\}$
2. $a \bmod p = 0 \Leftrightarrow e = 0$
3. $a \bmod p \in QR(p) \Leftrightarrow e = 1$
4. $a \bmod p \in QNR(p) \Leftrightarrow e = -1 \bmod p$

We also call the criterion *Legendre symbol of a and p* $L_p(a) = a^{(p-1)/2}$.

3 Asymmetric Encryption

Symmetric encryption is efficient, but requires a key exchange that makes it impractical for most usecases. Asymmetric encryption works without key exchange, be splitting the key in public and private.

3.1 Asymmetric Encryption Scheme

As with symmetric encryption, the scheme is a tuple $\mathcal{S} = (X, Gen(1^\eta), E, E)$. The key gen algorithm now outputs a tuple $(k, \hat{k})$ (public, private), the encryption E uses the public key k and decrpytion D the private key $\hat{k}$.

3.2 Asymmetric CPA-Security

The security game is defined analogously to symmetric encryption. The key is now a tuple and the adversary knows the public key. Advantage, success and failure are defined as for Block Ciphers.

$\mathbb{E}(1^\eta) : \{0, 1\}$

1. *Generate keys.*
   $(k, \hat{k}) \xleftarrow{\$} \mathsf{Gen}(1^\eta)$
2. *Find phase.*
   $(z_0, z_1) \xleftarrow{\$} AF(1^\eta, k)$
3. *Selection.*
   $b \xleftarrow{\$} \{0, 1\}$; $y \xleftarrow{\$} E(z_b, k)$
4. *Guess phase.*
   $b' \xleftarrow{\$} AG(1^\eta, k, y)$
5. *Evaluation.*
   if $b' = b$, return 1, otherwise 0.

3.3 RSA

RSA is an asymmetric encryption scheme $\mathcal{S}_{RSA} = (X, Gen(1^\eta), E, D)$. It uses the problem of computing prime factors as oneway function.

- The keygen algorithm randomly selects 2 different primes p and q of binary length η.

$$n = p \cdot q \qquad m = (p-1)\cdot(q-1) = \Phi(n)$$

$$e \xleftarrow{\$} \mathbb{Z}_m^* \qquad d = e^{-1} \mod m \qquad k = ((n,e),(n,d))$$

- $E(x,(n,e)) = x^e \mod n$
- $D(y,(n,d)) = y^d \mod n$

$\mathbb{E}(1^\eta) : \{0,1\}$

1. *Generate keys.*
   $((n,e),(n,d)) \xleftarrow{\$} \mathsf{Gen}(1^\eta)$
2. *Message selection.*
   $x \xleftarrow{\$} \mathbb{Z}_n$
   $y = x^e \mod n$
3. *Guess phase.*
   $x' \xleftarrow{\$} I(1^\eta,(n,e),y)$
4. *Evaluation.*
   if $x' = x$, return 1, otherwise 0.

It has not been proven, that the RSA encryption cannot be inverted. An algorithm that attempts that is called inverter I. The advantage is assumed to be negligible, which is called RSA-Assumption.
$|Adv_{I,S}^{RSA}(\eta)| = Pr[\mathbb{E}_{I,S}^{RSA}(1^\eta) = 1]$

Since the encryption function is deterministic, this textbook RSA is not secure. The scheme can be modified to be secure, by adding randomness. E.g. PKCS#1 v1.5 defines a random padding that fixes that issue:

$$0^{14}||10||r||0^8||x$$

While this is assumed to be sufficient for CPA-Security, RSA does not hold CCA-Security. In fact one doesn't even need a full decryption oracle as demonstrated in the Bleichenbacher Attack.