RABIN CRYPTOSYSTEM

Primes $p,q$ of the form $4k + 3$, so called Blum primes, are kept secret, $n = pq$ is the public key.

Encryption: of a plaintext $w < n$

$$c = w^2 \mod n$$

Decryption: It is easy to verify, using Euler’s criterion which says that if $c$ is a quadratic residue modulo $p$, then $c^{(p-1)/2} \equiv 1 \pmod{p}$, that

$$\pm c^{(p+1)/4} \mod p \quad \text{and} \quad \pm c^{(q+1)/4} \mod q$$

are two square roots of $c$ modulo $p$ and $q$. One can now obtain four square roots of $c$ modulo $n$ using the method shown in Appendix.

In case the plaintext $w$ is a meaningful English text, it should be easy to determine $w$ from $w_1, w_2, w_3, w_4$.

However, if $w$ is a random string (say, for a key exchange) it is impossible to determine $w$ from $w_1, w_2, w_3, w_4$.

Rabin did not propose this system as a practical cryptosystem.

GENERALIZED RABIN CRYPTOSYSTEM

PUBLIC KEY: $n, B$ ($0 \leq B \leq n - 1$)

TRAPDOOR: Blum primes $p, q$ ($n = pq$)

Encryption: $e(x) = x(x + B) \mod n$

Decryption: $d(y) = \left(\sqrt{\frac{B^2}{4} + y - \frac{B}{2}}\right) \mod n$

It is easy to verify that if $\omega$ is a nontrivial square root of $1$ modulo $n$, then there are four decryptions of $e(x)$:

$$x, \quad -x, \quad \omega (x + \frac{B}{2}) - \frac{B}{4}, \quad -\omega (x + \frac{B}{2}) - \frac{B}{4}$$

Example

$$e \left( \omega (x + \frac{B}{2}) - \frac{B}{4} \right) = \left( \omega (x + \frac{B}{2}) - \frac{B}{4} \right) \left( \omega (x + \frac{B}{2}) + \frac{B}{4} \right) = \omega^2 (x + \frac{B}{2})^2 - \left( \frac{B}{4} \right)^2 = x^2 + Bx = e(x)$$

Decryption of the generalized Rabin cryptosystem can be reduced to the decryption of the original Rabin cryptosystem.

Indeed, the equation $x^2 + Bx \equiv y \pmod{n}$ can be transformed, by the substitution $x = x_1 - B/2$, into $x_1^2 \equiv B^2/4 + y \pmod{n}$ and, by defining $c = B^2/4 + y$, into $x_1^2 \equiv c \pmod{n}$.

Decryption can be done by factoring $n$ and solving congruences $x_1^2 \equiv c \pmod{p}$, $x_1^2 \equiv c \pmod{q}$.
We show that any hypothetical decryption algorithm $A$ for Rabin cryptosystem, can be used, as an oracle, in the following Las Vegas algorithm, to factor an integer $n$.

**Algorithm:**
1. Choose a random $r$, $1 \leq r \leq n - 1$;
2. Compute $y = (r^2 - B^2/4) \mod n$; \( \{y = e_3(r - B/2)\} \).
3. Call $A(y)$, to obtain a decryption $x = \left(\sqrt{B^2/4 + y - r/2}\right) \mod n$;
4. Compute $x_1 = x + B/2$; \( \{x_1^2 \equiv r^2 \mod n\} \).
5. If $x_1 = \pm r$ then quit (failure)
   else $\gcd(x_1 + r, n) = p$ or $q$.

Indeed, after Step 4, either $x_1 = \pm r \mod n$ or $x_2 = \pm \omega r \mod n$.

In the second case we have
\[
n \mid (x_1 - r)(x_1 + r),
\]
but $n$ does not divide either factor $x_1 - r$ or $x_1 + r$.

Therefore computation of $\gcd(x_1 + r, n)$ or $\gcd(x_1 - r, n)$ must yield factors of $n$.

**SHANKS’ ALGORITHM for DISCRETE ALGORITHM**

Let $m = \lceil \sqrt{(p - 1)} \rceil$. The following algorithm computes $\log_p y$ in $\mathbb{Z}_p^*$.

1. Compute $q^m \mod p$, $0 \leq j \leq m - 1$.
2. Create list $L_1$ of $m$ pairs $(j, q^m \mod p)$, sorted by the second item.
3. Compute $yq^{-i} \mod p$, $0 \leq i \leq m - 1$.
4. Create list $L_2$ of pairs $(i, yq^{-i} \mod p)$ sorted by the second item.
5. Find two pairs, one $(j, z) \in L_1$ and second $(i, z) \in L_2$.

If such a search is successful, then
\[
q^m \mod p = z = yq^{-i} \mod p
\]
and as the result
\[
\log_y x \equiv (mj + i) \mod (p - 1).
\]

Therefore
\[
q^{mj+i} \equiv y \pmod{p}
\]
On the other hand, for any $y$ we can write
\[
\log_y x \equiv mj + i,
\]
For some $0 \leq i, j \leq m - 1$. Hence the search in the Step 5 of the algorithm has to be successful.

**ElGamal CRYPTOSYSTEM**

**Design:** choose a large prime $p$ – (with at least 150 digits).

choose two random integers $1 \leq q, x < p$ – where $q$ is a primitive element of $\mathbb{Z}_p^*$

calculate $y = q^r \mod p$.

**Public key:** $p, q, y$;  
**trapdoor:** $x$  

**Encryption** of a plaintext $w$: choose a random $r$ and compute
\[
a = q^r \mod p, \quad b = y^{w} \mod p
\]

**Cryptotext:** $c = (a, b)$

(Cryptotext contains indirectly $r$ and the plaintext is "masked" by multiplying with $y^r$ (and taking modulo $p$))

**Decryption:** $w = b_p^{-1} \mod p = ba^{-x} \mod p$.

**Proof of correctness:**
\[
a^x \equiv q^{r} \mod p
\]
\[
\frac{b}{a^{x}} = \frac{y^{w} \mod p}{q^{rx} \mod p} \equiv w \mod p
\]

**Note:** Security of the ElGamal cryptosystem is based on infeasibility of the discrete logarithm computation.

**BIT SECURITY of DISCRETE LOGARITHM**

Let us consider problem to compute $L_i(y) = i$-th least significant bit of $\log_a y$ in $\mathbb{Z}_p^*$.

**Result 1** $L_i(y)$ can be computed efficiently.
To show that we use the fact that the set $QR(p)$ has $(p - 1)/2$ elements.

Let $q$ be a primitive element of $\mathbb{Z}_p^*$. Clearly, $q^i \in QR(p)$ if $i$ is even. Since the elements
\[
q^0 \mod p, q^2 \mod p, \ldots, q^{3} \mod p
\]
are all distinct, we have that
\[
QR(p) = \{q^i \mod p \mid 0 \leq i \leq (p - 3)/2\}
\]

**Consequence:** $y$ is a quadratic residue iff $\log_y y$ is even, that is iff $L_1(y) = 0$.

By Euler’s criterion $y$ is a quadratic residue if $y^{(p-1)/2} \equiv 1 \mod p$

$L_1(y)$ can therefore be computed as follows:
\[
L_i(y) = 0 \quad \text{if} \quad y^{(p-1)/2} \equiv 1 \mod p; \quad L_i(y) = 1 \quad \text{otherwise}
\]

**Result 2** Efficient computability of $L_i(y), i > 1$ in $\mathbb{Z}_p^*$ would imply efficient computability of the discrete logarithm in $\mathbb{Z}_p^*$. 

**SECURITY of RABIN CRYPTOSYSTEM**

We show that any hypothetical decryption algorithm $A$ for Rabin cryptosystem, can be used, as an oracle, in the following Las Vegas algorithm, to factor an integer $n$.

Algorithm:
1. Choose a random $r$, $1 \leq r \leq n - 1$;
2. Compute $y = (r^2 - B^2/4) \mod n$; \( \{y = e_3(r - B/2)\} \).
3. Call $A(y)$, to obtain a decryption $x = \left(\sqrt{B^2/4 + y - r/2}\right) \mod n$;
4. Compute $x_1 = x + B/2$; \( \{x_1^2 \equiv r^2 \mod n\} \).
5. If $x_1 = \pm r$ then quit (failure)
   else $\gcd(x_1 + r, n) = p$ or $q$.

Indeed, after Step 4, either $x_1 = \pm r \mod n$ or $x_2 = \pm \omega r \mod n$.

In the second case we have
\[
n \mid (x_1 - r)(x_1 + r),
\]
but $n$ does not divide either factor $x_1 - r$ or $x_1 + r$.

Therefore computation of $\gcd(x_1 + r, n)$ or $\gcd(x_1 - r, n)$ must yield factors of $n$.
A group version of discrete logarithm problem

Given a group \((G, \circ)\), \(\alpha \in G\), \(\beta \in \{\alpha^i \mid i \geq 0\}\). Find

\[ \log_\alpha \beta = k \] such that \(\alpha^k = \beta\)

**GROUP VERSION of ElGamal CRYPTOSYSTEM**

ElGamal cryptosystem can be implemented in any group in which discrete logarithm problem is infeasible.

**Cryptosystem** for \((G, \circ)\)

**Public key**: \(\alpha, \beta\)

**Trapdoor**: \(k\) such that \(\alpha^k = \beta\)

**Encryption**: of a plaintext \(w\) and a random integer \(k\)

\[ e(w, k) = (y_1, y_2) \quad \text{where} \quad y_1 = \alpha^k, y_2 = w \circ \beta^k \]

**Decryption**: of cryptotext \((y_1, y_2)\):

\[ d(y_1, y_2) = y_2 \circ y_1^{-k} \]

An important special case is that of computation of discrete logarithm in a group of points of an elliptic curve defined over a finite field.

**WILLIAMS CRYPTOSYSTEM – EFFICIENT EXPONENTIATION**

Assume now

\[ a^2 - cb^2 = 1 \]

Then \(\alpha \pi = 1\) and consequently

\[ X_i^2 - cY_i^2 = 1 \]

Moreover, for \(j \geq i\)

\[ X_{i+j} = 2X_iX_j + X_{j-1} \]
\[ Y_{i+j} = 2Y_iX_j + Y_{j-1} \]

From these and following equations:

\[ X_{i+j} = 2X_iX_j + cY_iY_j \]
\[ Y_{i+j} = 2Y_iX_j + X_iY_j \]

we get the recursive formulas:

\[ X_{2i} = X_i^2 + cY_i^2 = 2X_i^2 - 1 \]
\[ Y_{2i} = 2X_iY_i \]
\[ X_{2i+1} = 2X_iY_{i+1} - X_i \]
\[ Y_{2i+1} = 2X_iY_{i+1} - Y_i \]

Consequences: 1. \(X_i\) and \(Y_i\) can be, given \(i\), computed fast.

**Remark** Since \(X_0 = 1, X_1 = a, X_i\) does not depend on \(b\).

**WILLIAMS CRYPTOSYSTEM – BASICS**

This cryptosystem is similar to RSA, but with number operations performed in a quadratic field. Complexity of the cryptanalysis of the Williams cryptosystem is equivalent to factoring.

Consider numbers of the form

\[ \alpha = a + b\sqrt{c} \]

where \(a, b, c\) are integers.

If \(c\) is fixed, \(\alpha\) can be viewed as a pair \((a, b)\).

\[ \alpha_1 + \alpha_2 = (a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2) \]
\[ \alpha_1\alpha_2 = (a_1, b_1) \cdot (a_2, b_2) = (a_1a_2 + c b_1b_2, a_1b_2 + b_1a_2) \]

The conjugate \(\overline{\alpha}\) of \(\alpha\) is defined by

\[ \overline{\alpha} = a - b\sqrt{c} \]

**Auxiliary functions:**

\[ X_i(\alpha) = \frac{\alpha^i + \alpha^{-i}}{2} \]
\[ Y_i(\alpha) = \frac{b(\alpha^i - \alpha^{-i})}{2} \left( = \frac{\alpha - \frac{1}{\alpha}}{2\sqrt{c}} \right) \]

Hence

\[ \alpha^i = X_i(\alpha) + Y_i(\alpha)\sqrt{c} \]
\[ \overline{\alpha} = X_i(\alpha) - Y_i(\alpha)\sqrt{c} \]

**WHEN is a CRYPTOSYSTEM (perfectly) SECURE?**

First question: Is it enough for perfect security of a cryptosystem that one cannot get a plaintext from a cryptotext?

**NO, NO, NO**

**WHY**

For many applications it is crucial that no information about the plaintext could be obtained.

\[ \boxed{\text{Intuitively, a cryptosystem is (perfectly) secure if one cannot get any (new) information about the corresponding plaintext from any cryptotext.}} \]

\[ \boxed{\text{It is very nontrivial to define fully precisely when a cryptosystem is (computationally) perfectly secure.}} \]

\[ \boxed{\text{It has been shown that perfectly secure cryptosystems have to use randomized encryptions.}} \]

prof. Jozef Gruska
Randomness and cryptography are deeply related.

**Prime goal of any good encryption method is to transform even a highly nonrandom plaintext into a highly random cryptotext.** (Avalanche effect.)

**Example** Let \( e_k \) be an encryption algorithm, \( x_0 \) be a plaintext. And

\[
x_i = e_k(x_{i-1}), \quad i \geq 1.
\]

It is intuitively clear that if encryption \( e_k \) is "cryptographically secure", then it is very, very likely that the sequence \( x_0, x_1, x_2, x_3 \) is (quite) random.

Perfect encryption should therefore produce (quite) perfect (pseudo)randomness.

**The other side of the relation is more complex.** It is clear that perfect randomness together with ONE-TIME PAD cryptosystem produces perfect secrecy. The price to pay: a key as long as plaintext is needed.

The way out seems to be to use an encryption algorithm with a pseudo-random generator to generate a long pseudo-random sequence from a short random seed and to use the resulting sequence with ONE-TIME PAD.

**Basic question:** When is a pseudo-random generator good enough for cryptographical purposes?

---

**Secure Encryptions – Basic Concepts I**

We now start to discuss a very nontrivial question: when is an encryption scheme computationally perfectly **SECURE**?

At first, we introduce two very basic technical concepts:

**Definition** A function \( f : N \rightarrow R \) is a negligible function if for any polynomial \( p(n) \) and for almost all \( n \):

\[
f(n) \leq \frac{1}{p(n)}
\]

**Definition – computational distinguishibility** Let \( X = \{ X_n \}_{n \in N} \) and \( Y = \{ Y_n \}_{n \in N} \) be probability ensembles such that each \( X_n \) and \( Y_n \) ranges over strings of length \( n \). We say that \( X \) and \( Y \) are computationally indistinguishable if for every feasible algorithm \( A \) the difference

\[
d_A(n) = | \Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1] |
\]

is a negligible function in \( n \).

---

**Secure Encryptions – Pseudorandom Generators**

In cryptography random sequences can be usually be well enough replaced by pseudorandom sequences generated by (cryptographically perfect) pseudorandom generators.

**Definition - pseudorandom generator.** Let \( l(n) : N \rightarrow N \) be such that \( l(n) > n \) for all \( n \). A (computationally indistinguishable) pseudorandom generator with a stretch function \( l \), is an efficient deterministic algorithm which on the input of a random \( n \)-bit seed outputs a \( l(n) \)-bit sequence which is computationally indistinguishable from any random \( l(n) \)-bit sequence.

**Theorem** Let \( f \) be a one-way function which is length preserving and efficiently computable, and \( b \) be a hard core predicate of \( f \), then

\[
G(s) = b(s) \cdot b(f(s)) \cdots b(f^{(l(n) - 1)}(s))
\]

is a (computationally indistinguishable) pseudorandom generator with stretch function \( l(n) \).

**Definition** A predicate \( b \) is a hard core predicate of the function \( f \) if \( b \) is easy to evaluate, but \( b(x) \) is hard to predict from \( f(x) \). (That is, it is unfeasible, given \( f(x) \) where \( x \) is uniformly chosen, to predict \( b(x) \) substantially better than with the probability 1/2.)

It is conjectured that the least significant bit of the modular squaring function \( x^2 \mod n \) is a hard-core predicate.

---

**Cryptographically Strong Pseudorandom Generators**

**Fundamental question:** when is a pseudo-random generator good enough for cryptographical purposes?

**Basic concept:** A pseudo-random generator is called cryptographically strong if the sequence of bits it produces, from a short random seed, is so good for using with ONE-TIME PAD cryptosystem, that no polynomial time algorithm allows a cryptanalyst to learn any information about the plaintext from the cryptotext.

A cryptographically strong pseudo-random generator would therefore provide sufficient security in a secret-key cryptosystem if both parties agree on some short seed and never use it twice.

As discussed later: Cryptographically strong pseudo-random generators could provide perfect secrecy also for public-key cryptography.

**Problem:** Do cryptographically strong pseudo-random generators exist?

**Remark:** The concept of a cryptographically strong pseudo-random generator is one of the key concepts of the foundations of computing.

Indeed, a cryptographically strong pseudo-random generator exists if and only if a one-way function exists which is equivalent with \( P \neq UP \) and what implies \( P \neq NP \).
So far there are only candidates for cryptographically strong pseudo-random generators. For example, cryptographically strong are all pseudo-random generators that are unpredictable to the left in the sense that a cryptanalyst that knows the generator and sees the whole generated sequence except its first bit has no better way to find out this first bit than to toss the coin.

It has been shown that if integer factoring is intractable, then the so-called BBS pseudo-random generator, discussed below, is unpredictable to the left.

(We make use of the fact that if factoring is unfeasible, then for almost all quadratic residues \( x \mod n \), coin-tossing is the best possible way to estimate the least significant bit of \( x \) after seeing \( x^2 \mod n \).)

Let \( n \) be a Blum integer. Choose a random quadratic residue \( x_0 \) (modulo \( n \)).

For \( i \geq 0 \) let

\[
    x_{i+1} = x_i^2 \mod n, \quad b_i = \text{the least significant bit of } x_i
\]

For each integer \( i \), let

\[
    BBS_n(x_0) = b_0 \ldots b_{i-1}
\]

be the first \( i \) bits of the pseudo-random sequence generated from the seed \( x_0 \) by the BBS pseudo-random generator.

From security point of view, public-key cryptography with deterministic encryptions has the following serious drawback:

A cryptoanalyst who knows the public encryption function \( e_k \) and a cryptotext \( c \) can try to guess a plaintext \( w \), compute \( e_k(w) \) and compare it with \( c \).

The purpose of randomized encryptions is to encrypt messages, using randomized algorithms, in such a way that one can prove that no feasible computation on the cryptotext can provide any information whatsoever about the corresponding plaintext (except with a negligible probability).

**Formal setting:**

<table>
<thead>
<tr>
<th>plaintext-space</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>cryptotext</td>
<td>C</td>
</tr>
<tr>
<td>key-space</td>
<td>K</td>
</tr>
<tr>
<td>random-space</td>
<td>R</td>
</tr>
</tbody>
</table>

definition: \( e_k : P \times R \rightarrow C \)

decryption: \( d_k : C \rightarrow P \) or \( C \rightarrow 2^P \) such that for any \( p, r : \)

\[
    d_k(e_k(p, r)) = p.
\]

- \( d_k, e_k \) should be easy to compute.
- Given \( e_k \), it should be unfeasible to determine \( d_k \).

---

**Secure Encryption – First Definition**

Definition – semantic security of encryption A cryptographic system is semantically secure if for every feasible algorithm \( A \), there exists a feasible algorithm \( B \) so that for every two functions

\[
    f, h : \{0, 1\}^* \rightarrow \{0, 1\}^a
\]

and all probability ensembles \( \{X_n\}_{n \in N} \), where \( X_n \) ranges over \( \{0, 1\}^a \)

\[
    Pr[A(E(X_n), h(X_n)) = f(X_n)] < Pr[B(h(X_n)) = f(X_n)] + \mu(n),
\]

where \( \mu \) is a negligible function.

It can be shown that any semantically secure public-key cryptosystem must use a randomized encryption algorithm.

RSA cryptosystem is not secure in the above sense. However, randomized versions of RSA are semantically secure.
SECURE ENCRYPTIONS – SECOND DEFINITION

Definition A randomized-encryption cryptosystem is polynomial time secure if, for any $c \in \mathbb{N}$ and sufficiently large $s \in \mathbb{N}$ (security parameter), any randomized polynomial time algorithms that takes as input $s$ (in unary) and the public key, cannot distinguish between randomized encryptions, by that key, of two given messages of length $c$, with the probability larger than $\frac{1}{2} + \frac{1}{s^c}$.

Both definitions are equivalent.

Example of a polynomial-time secure randomized (Bloom-Goldwasser) encryption:

- $p, q$ - large Blum primes
- $n = p \times q$ - key
- Plaintext-space - all binary strings
- Random-space - $QR_n$
- Crypto-space - $QR_n \times \{0, 1\}^*$

Encryption: Let $w$ be a $t$-bit plaintext and $x_0$ a random quadratic residue modulo $n$.

Compute $x_t$ and $BBS_n, t(x_0)$ using the recurrence

$$x_{t+1} = x_t^2 \mod n$$

Cryptotext: $(x_t, w \oplus BBS_n, t(x_0))$

Decryption: Legal user, knowing $p, q$, can compute $x_0$ from $x_t$, then $BBS_n, t(x_0)$, and finally $w$.

HASH FUNCTIONS

Another very simple, fundamental and important cryptographic concept is that of hash functions.

- Hash functions $h : \{0, 1\}^* \rightarrow \{0, 1\}^m$; $h : \{0, 1\}^n \rightarrow \{0, 1\}^m$, $n >> m$

map (very) long messages $w$ into short ones, called usually messages digests or hashes or fingerprints of $w$, in a way that has important cryptographic properties.

Digital signatures are one of important applications of hash functions.

In most of the digital signature schemes, to be discussed in the next chapter, the length of a signature is at least as long as of the message being signed. This is clearly a big disadvantage.

To remedy this situation, signing procedure is applied to a hash of the message, rather than to the message itself. This is OK provided the hash function has good cryptographic properties, discussed next.

PROPERTIES GOOD HASH FUNCTIONS SHOULD HAVE I.

We now derive basic properties cryptographically good hash functions should have – by analysing several possible attacks on their use.

**Attack 1** If Eve gets a valid signature $(w, y)$, where $y = \text{sig}_k(h(w))$ and she would be able to find $w'$ such that $h(w') = h(w)$, then also $(w', y)$, a forgery, would be a valid signature.

Cryptographically good hash function should therefore have the following weak collision-free property

**Definition 1.** Let $w$ be a message. A hash function $h$ is weakly collision-free for $w$, if it is computationally infeasible to find a $w'$ such that $h(w) = h(w')$.

PROPERTIES GOOD HASH FUNCTIONS SHOULD HAVE II.

**Attack 2** If Eve finds two $w$ and $w'$ such that $h(w') = h(w)$, she can ask Alice to sign $h(w)$ to get signature $s$ and then Eve can create a forgery $(w', s)$.

Cryptographically good hash function should therefore have the following strong collision-free property

**Definition 2.** A hash function $h$ is strongly collision-free if it is computationally infeasible to find two elements $w \neq w'$ such that $h(w) = h(w')$. 
**Properties Hash Functions Should Have**

**Attack 3** If Eve can compute signature $s$ of a random $z$, and then she can find $w$ such that $z = h(w)$, then Eve can create forgery $(w, s)$.

To exclude such an attack, hash functions should have the following one-wayness property.

**Definition 3.** A hash function $h$ is one-way if it is computationally infeasible to find, given $z$, an $w$ such that $h(w) = z$.

One can show that if a hash function has strongly collision-free property, then it has one-wayness property.

**Examples**

**Example 1** For a vector $a = (a_1, \ldots, a_k)$ of integers let

$$H(a) = \sum_{i=0}^{k} a_i \mod n$$

where $n$ is a product of two large integers.

This hash function does not meet any of the three properties mentioned on the last slide.

**Example 2** For a vector $a = (a_1, \ldots, a_k)$ of integers let

$$H(a) = \left(\sum_{i=0}^{k} a_i \right)^2 \mod n$$

This function is one-way, but it is not weakly collision-free.

**Finding Collisions with Inversion Algorithm**

**Theorem** Let $h : X \rightarrow Z$ be a hash function where $X$ and $Z$ are finite and $|X| \geq 2|Z|$. If there is an inversion algorithm $A$ for $h$, then there exists randomized algorithm to find collisions.

**Sketch of the proof.** One can easily show that the following algorithm

1. Choose a random $x \in X$ and compute $z = h(x)$; Compute $x_1 = A(z)$;
2. if $x_1 \neq x$, then $x_1$ and $x$ collide (under $h$ – success) else failure

has probability of success

$$p(\text{success}) = \frac{1}{|X|} \sum_{x \in X} \frac{[x] - 1}{|[x]|} \geq \frac{1}{2}$$

where, for $x \in X$, $[x]$ is the set of elements having the same hash as $x$. 
VARIATIONS on BIRTHDAY PARADOX

It is well known that if there are 23 (29) \{40\} \{57\} < 100 > people in one room, then the probability that two of them have the same birthday is more than 50% (70%) [89%] \{99%\} < 99.99997% — this is called a Birthday paradox.

More generally, if we have \(n\) objects and \(r\) people, each choosing one object (so that several people can choose the same object), then if
\[
r \approx 1.177 \sqrt{n} (r \approx \sqrt{2n\lambda}),
\]
then probability that two people choose the same object is 50% \((1 - e^{-\lambda})%\).

Another version of the birthday paradox: Let us have \(n\) objects and two groups of \(r\) people. If
\[
r \approx \sqrt{n\lambda},
\]
then probability that someone from one group chooses the same object as someone from the other group is \((1 - e^{-\lambda})\).

HASH FUNCTION DOMAIN LOWER BOUND

Birthday paradox imposes a lower bound on the sizes of message digests (fingerprints)

For example a 40-bit message would be insecure because a collision could be found with probability 0.5 with just over 20\(^{20}\) random hashes.

Minimum acceptable size of message digest seems to be 128 and therefore 160 are used in such important systems as DSS – Digital Signature Schemes (standard).

BASIC DERIVATIONS related to BIRTHDAY PARADOX

For probability \(\bar{p}(n)\) that all \(n\) people in a room have birthday in different days, it holds
\[
\bar{p}(n) = \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right) = \frac{365!}{365^n (365 - n)!}
\]
This equation expresses the fact for no person to share a birthday, the second person cannot have the same birthday as the first one, third person cannot have the same birthday as first two,......

Probability \(p(n)\) that at least two person have the same birthday is therefore
\[
p(n) = 1 - \bar{p}(n)
\]
This probability is larger than 0.5 first time for \(n = 23\).

AN ALMOST GOOD HASH FUNCTION

We show an example of the hash function (so called Discrete Log Hash Function) that seems to have as the only drawback that it is too slow to be used in practice:

Let \(p\) be a large prime such that \(q = \frac{p+1}{2}\) is also prime and let \(\alpha, \beta\) be two primitive roots modulo \(p\). Denote \(a = \log_{\alpha} \beta\) (that is \(\beta = \alpha^a\)).

\(h\) will map two integers smaller than \(q\) to an integer smaller than \(p\), for \(m = x_0 + x_1q, 0 \leq x_0, x_1 \leq q - 1\) as follows,
\[
h(x_0, x_1) = h(m) = \alpha^{x_0} \beta^{x_1} \pmod{p}
\]
To show that \(h\) is one-way and collision-free the following fact can be used:

**FACT**: If we know different messages \(m_1\) and \(m_2\) such that \(h(m_1) = h(m_2)\), then we can compute \(\log_{\alpha} \beta\).
EXTENDING HASH FUNCTIONS

Let \( h : \{0,1\}^m \rightarrow \{0,1\}^t \) be a strongly collision-free hash function, where \( m > t + 1 \).

We design now a strongly collision-free hash function

\[
h^* : \bigoplus_{i=m}^{\infty} \{0,1\}^j \rightarrow \{0,1\}^t.
\]

Let a bit string \( x, |x| = n > m \), have decomposition

\[x = x_1 \parallel x_2 \cdots \parallel x_k,
\]

where \(|x_i| = m - t - 1 \) if \( i < k \) and \(|x_k| = m - t - 1 - d \) for some \( d \). (Hence \( k = \left\lceil \frac{n}{(m-t-1)} \right\rceil \).)

\( h^* \) will be computed as follows:

- for \( i = 1 \) to \( k-1 \) do \( y_i := x_i \);
- \( y_k := x_k \parallel 0^d \); \( y_{k+1} := \text{binary representation of } d \);
- \( g_1 := h(0^{l+1} \parallel y_1) \);
- for \( i = 1 \) to \( k \) do \( g_{i+1} := h(g_i \parallel 1 \parallel y_{i+1}) \);
- \( h^*(x) := g_{k+1} \).

PRACTICALLY USED HASH FUNCTIONS

A variety of hash functions has been constructed. Very often used hash functions are MD4, MD5 (created by Rivest in 1990 and 1991 and producing 128 bit message digest).

NIST even published, as a standard, in 1993, SHA (Secure Hash Algorithm) – producing 160 bit message digest – based on similar ideas as MD4 and MD5.

A hash function is called secure if it is strongly collision-free.

One of the most important cryptographic results of the last years was due to the Chinese Wang who has shown that MD4 is not cryptographically secure.

RANDOMIZED VERSION of RSA-LIKE CRYPTOSYSTEM

The scheme works for any trapdoor function (as in case of RSA),

\[ f : D \rightarrow D, D \subseteq \{0,1\}^n, \]

for any pseudorandom generator

\[ G : \{0,1\}^k \rightarrow \{0,1\}^l, k << l \]

and any hash function

\[ h : \{0,1\}^l \rightarrow \{0,1\}^k, \]

where \( n = l + k \). Given a random seed \( s \in \{0,1\}^k \) as input, \( G \) generates a pseudorandom bit-sequence of length \( l \).

Encryption of a message \( m \in \{0,1\}^l \) is done as follows:

- A random string \( r \in \{0,1\}^k \) is chosen.
- Set \( x = (m \oplus G(r)) || (r \oplus h(m \oplus G(r))) \). (If \( x \notin D \) go to step 1.)
- Compute encryption \( c = f(x) \) – length of \( x \) and of \( c \) is \( n \).

Decryption of a cryptotext \( c \).

- Compute \( f^{-1}(c) = a || b \), \( |a| = l \) and \( |b| = k \).
- Set \( r = h(a) \oplus b \) and get \( m = a \oplus G(r) \).

Comment Operation \( "\parallel" \) stands for a concatenation of strings.

HASH FUNCTIONS from CRYPTOSYSTEMS

Let us have computationally secure cryptosystem with plaintexts, keys and cryptotexts being binary strings of a fixed length \( n \) and with encryption function \( e_k \).

If

\[ x = x_1 \parallel x_2 \cdots \parallel x_k \]

is decomposition of \( x \) into substrings of length \( n \), \( g_0 \) is a random string, and

\[ g_i = f(x_i, g_{i-1}) \]

for \( i = 1, \ldots, k \), where \( f \) is a function that "incorporates" encryption function \( e_k \) of the cryptosystem, then

\[ h(x) = g_k. \]

For example such good properties have these two functions:

\[ f(x_i, g_{i-1}) = e_{g_{i-1}}(x_i) \oplus x_i \]
\[ f(x_i, g_{i-1}) = e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1} \]
Private key: Blum primes p and q.
Public key: \( n = pq \).

**Encryption** of \( x \in \{0, 1\}^m \):
- Randomly choose \( s_0 \in \{0, 1, \ldots, n\} \).
- For \( i = 1, 2, \ldots, m + 1 \) compute
  \[
  s_i \leftarrow s_{i-1}^2 \mod n
  \]
  and \( \sigma_i = \text{lsb}(s_i) \).

The ciphertext is \( (s_{m+1}, y) \), where \( y = x \oplus \sigma_1\sigma_2\ldots\sigma_m \).

**Decryption** of the ciphertext \( (r, y) \):
- Let \( d = 2^{-m} \mod \phi(n) \).
- Let \( s_1 = r^d \mod n \).
- For \( i = 1, \ldots, m \), compute \( \sigma_i = \text{lsb}(s_i) \) and \( s_{i+1} \leftarrow s_i^2 \mod n \)

The plaintext \( x \) can then be computed as \( y \oplus \sigma_1\sigma_2\ldots\sigma_m \).

---

**GLOBAL GOALS of CRYPTOGRAPHY**

Cryptosystems and encryption/decryption techniques are only one part of modern cryptography.

**General goal of modern cryptography is construction of schemes which are robust against malicious attempts to make these schemes to deviate from their prescribed functionality.**

The fact that an adversary can design its attacks after the cryptographic scheme has been specified, makes design of such cryptographic schemes very difficult – schemes should be secure under all possible attacks.

In the next chapters several of such most important basic functionalities and design of secure systems for them will be considered. For example: digital signatures, user and message authentication, ...

Moreover, also such basic primitives as zero-knowledge proofs, needed to deal with general cryptography problems will be presented and discussed.

We will also discuss cryptographic protocols for a variety of important applications. For example for voting, digital cash, ...

---

**BLUM INTEGERS**

- An integer \( n \) is a **Blum integer** if \( n = pq \), where \( p, q \) are primes congruent 3 modulo 4, that is primes of the form \( 4k + 3 \) for some integer \( k \).
- If \( n \) is a Blum integer, then each \( x \in \text{QR}(n) \) has 4 square roots and exactly one of them is in \( \text{QR}(n) \) – so called principal square root of \( x \) modulo \( n \).
- Function \( f : \text{QR}(n) \rightarrow \text{QR}(n) \) defined by \( f(x) = x^2 \mod n \) is a permutation.