7. OTHER CRYPTOSYSTEMS

Large number of interesting and important cryptosystems have already been developed. In this chapter we present several such cryptosystems in order to illustrate:

- Principles and techniques that can be used to design cryptosystems.

For example, we present several cryptosystems security of which is based on the fact that computation of discrete logarithms is infeasible in some groups.

Finally, we discuss one of the key questions of modern cryptography: when can a cryptosystem be considered as (computationally) perfectly secure?

In order to do that we will:

- discuss the role randomness play in the cryptography;
- introduce the very fundamental defintions of perfect security of cryptosystem
- present some examples of perfectly secure cryptosystems.

Rabin cryptosystem

Primes p, q of the form $4k+3$ are chosen and kept secret; $n = pq$ is the public key.

Encryption: of a plaintext $w < n$

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

Decryption: of a cryptotext c 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

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

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 the Appendix.

If the plaintext w is a meaningful English text, it should be then easy to determine w.

However, if w is a random string (say, for a key exchange) it is impossible to determine w.

Rabin did not propose this system as a to-be-used cryptosystem.

Generalized Rabin cryptosystem

Public key: n, B $(0 \le B \le n-1)$
Trapdoor: primes p, q $(such that n=pq)$ of the form $4k+3$

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

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

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

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

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

Indeed, the congruence

$$x^2 + Bx \equiv y \pmod n$$

can be transformed, by the substitution $x = x_1 - \frac{B}{2}$, into the congruence

$$x_1^2 \equiv \frac{B^2}{4} + y \pmod n$$

and, by defining $c = \frac{B^2}{4} + y$, into

$$x_1^2 \equiv c \pmod n$$

Decryption can be done by factoring n and solving congruences

$$x_1^2 \equiv c \pmod p$$

$$x_1^2 \equiv c \pmod q$$

Security of Rabin cryptosystem

We show that any hypothetical decryption algorithm $\mathcal{A}$ 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 such that, $1 \leq r \leq n-1$;
2. Compute $y = (r^2 - B^2/4) \bmod n$; $\{y = e_k(r - \frac{B}{2})\}$.
3. Call $\mathcal{A}(y)$, to obtain a decryption $x = (\sqrt{\frac{B^2}{4} + y} - \frac{B}{2}) \bmod n$;
4. Compute $x_1 = x + \frac{B}{2}$; $\{x_1^2 \equiv r^2 \pmod{n}\}$
5. **if** $x_1 = \pm r$ **then quit** (failure)
   else $gcd(x_1 + r, n) = p$ or q

Indeed, after Step 4, either $x_1 = \pm r \pmod{n}$ or $x_1 = \pm \omega r \pmod{n}$.

In the second case we have

$$n | (x_1 - r)(x_1 + r),$$

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.

ElGamal cryptosystem

Design: choose a large prime p — at least with 150 digits — choose two random integers $1 \leq q, x < p$ — where q is a primitive element of Z_p^* — calculate

$$y = q^x \bmod p.$$

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

Encryption: of a plaintext w: chose a random r and compute

$$a = q^r \bmod p, \;\; b = y^r w \bmod p$$

Cryptotext: $c = (a, b)$

(Cryptotext contains (in a) indirectly r and the plaintext W is masked by multiplying w with y^r (and taking modulo p)}

Decryption: $w = \frac{b}{a^x} \bmod p = ba^{-x} \bmod p.$

Proof of correctness: $a^x \equiv q^{rx} \bmod p$,

$$\frac{b}{a^x} \equiv \frac{y^r w}{a^x} \equiv \frac{q^{rx} w}{q^{rx}} \equiv w \pmod{p}$$

Note 1: Security of the ElGamal cryptosystem is based on infeasibility of the discrete logarithm problem.

Note 2: $p - 1$ should have at least one large factor.

Shank's algorithm for discrete logarithm

The following (simple but not efficient) algorithm computes $\lg_q y$ in $\mathbf{Z}_\mathbf{p}^*$.

Let $m = \lceil \sqrt{p-1} \rceil$.

1. Compute $q^{mj} \bmod p$, $0 \leq j \leq m-1$.
2. Create list L_1 of m pairs $(j, q^{mj} \bmod p)$, sorted by the second item.
3. Compute $yq^{-i} \bmod p$, $0 \leq i \leq m-1$.
4. Create list L_2 of pairs $(i, yq^{-i} \bmod p)$ 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

$$q^{mj} \bmod p = z = yq^{-i} \bmod p$$

and as the result

$$\lg_q y \equiv (mj + i) \pmod{p-1}$$

and therefore

$$q^{mj+i} \equiv y \pmod{p}$$

On the other hand, for any y we can write

$$\lg_q y = mj + i,$$

where $0 \leq i, j \leq m-1$. Hence the search in the Step 5 of the algorithm has to be successful.

Bit security of discrete logarithm

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

Result 1 $L_1(y)$ can be computed efficiently.

To show that we use the fact that the set $QR(p)$ has $\frac{p-1}{2}$ elements.

Let q be a primitive element of Z_p^*. Clearly, $q^a \in QR(p)$ if a is even. Since the elements

$$q^0 \bmod p, q^2 \bmod p, \ldots, q^{p-3} \bmod p$$

are all distinct, we have that

$$QR(p) = \{q^{2i} \bmod p \,|\, 0 \leq i \leq (p-3)/2\}.$$

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

By Euler's criterion y is a quadratic residue iff $y^{(p-1)/2} \equiv 1 \pmod p$

$L_1(y)$ can therefore be computed as follows:

$$L_1(y) = \begin{cases} 0, & \textbf{if } y^{(p-1)/2} \equiv 1 \pmod p; \\ 1, & \text{otherwise.} \end{cases}$$

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

A group version of discrete logarithm problem

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

$$\log_\alpha \beta \text{ that is a } k \text{ such that } \alpha^k = \beta$$

.

Group version of ElGamal cryptosystem

ElGamal cryptosystem can be implemented in any group where discrete logarithm problem is intractable.

Cryptosystem for $(G, \circ)$,:
Public key: α, β.
Trapdoor: k such that $\alpha^k = \beta$.

Encryption: of a plaintext w with a random $r \in \{1, \ldots, |\{\alpha^i \mid i \geq 0\}|\}$:

$$e(w, r) = (c_1, c_2) \text{ where } c_1 = \alpha^r, c_2 = w \circ \beta^r$$

Decryption: of cryptotext (c_1, c_2):

$$d(c_1, c_2) = y_2 \circ (c_1^k)^{-1} = w \circ \beta^r \circ \alpha^{-rk} = w \circ \beta^{rk-rk} = w$$

An interesting fact is that discrete logarithm problem is intractable in any group Z_p^*, where p is a prime, but it is easily computable in any group $Z_{\phi(p)}$, in spite of the fact that for any p these two groups are isomorphic.

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

Eliptic curves over $Z_p, p > 3$ be prime

The elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{Z}_p$ is a set of solutions (x, y) to the congruence

$$y^2 \equiv x^3 + ax + b \pmod p \qquad \{4a^3 + 27b^3 \not\equiv 0 \pmod p\}$$

together with a special point 0, called the point at infinity.

Examples of points of the elliptic curve $y^2 = x^3 + x + 6$ over $\mathbb{Z}_{11}$:

x	$x^3 + x + 6 \pmod{11}$	in QR_{11} ?	y
0	6	no	
1	8	no	
2	5	yes	4,7
3	3	yes	5,6
4	8	no	
5	4	yes	2,9
6	8	no	
7	4	yes	2,9
8	9	yes	3,8
9	7	no	
10	4	yes	2,9

The number of points of an elliptic curve over $\mathbb{Z}_p$ is in the interval

$$(p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p})$$

In order to find a subgroup in which the discrete logarithm problem is intractable it is useful to know more about the structure of the group of elliptic curves points.

Theorem Let E be an elliptic curve defined over $\mathbb{Z}_p$, where p is prime, $p > 3$. Then there exist integers n_1, n_2 such that E is isomorphic to $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$. Further $n_2 | n_1, n_2 | (p - 1)$.

There are two problems when implementing directly a group version of ElGamal cryptosystem on an elliptic curve: **expansion factor** is 4 and there is no deterministic method known to generate points (plaintexts) on the curve.

Menezes-Vanstone Elliptic Curve Cryptosystem

This is an elliptic curve group version of the ElGamal cryptosystem. A nice illustration of the idea how a change of the underlying algebra can create one cryptosystem from another one.

Let E be an elliptic curve over $\mathbb{Z}_p$, ($p > 3$, prime) such that E contains a cyclic group H in which the discrete logarithm problem is intractable.

Public key: $q, y \in E$.
Trapdoor information: x such that $y = xq$

Encryption of a plaintext $(w_1, w_2) \in \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star$

Choose a random $r \in \mathbb{Z}_{|H|}$

$$\begin{aligned} e(r, w_1, w_2) &= (c_0, c_1, c_2); \\ & \quad c_0 = rq, \qquad (z_1, z_2) = ry; \\ & \quad c_1 = z_1 w_1 \bmod p; \\ & \quad c_2 = z_2 w_2 \bmod p. \end{aligned}$$

Decryption of a cryptotext (c_0, c_1, c_2):

$$d(c_0, c_1, c_2) = (c_1 z_1^{-1} \bmod p, c_2 z_2^{-1} \bmod p),$$

where

$$xc_0 = (z_1, z_2).$$

Williams cryptosystem – basics

Similar to RSA, but number operations are performed in a quadratic field. Cryptoanalysis of Williams cryptosystem is equivalent to factoring.

Consider numbers of the form

$$\alpha = a + b\sqrt{c}$$

where a, b, c are integers.

If c remains fixed α can be viewed as a pair (a, b).

$$\begin{aligned} \alpha_1 + \alpha_2 &= (a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2) \\ \alpha_1\alpha_2 &= (a_1, b_1)(a_2, b_2) = (a_1a_2 + cb_1b_2, a_1b_2 + b_1a_2) \end{aligned}$$

The **conjugate** $\bar{\alpha}$ of α is defined by

$$\bar{\alpha} = a - b\sqrt{c}$$

Auxiliary functions:

$$\begin{aligned} X_i(\alpha) &= \frac{(\alpha^i + \bar{\alpha}^i)}{2} \\ Y_i(\alpha) &= \frac{b(\alpha^i - \bar{\alpha}^i)}{(\alpha - \bar{\alpha})} \quad \left(= \frac{\alpha^i - \bar{\alpha}^i}{2\sqrt{c}}\right) \end{aligned}$$

Hence

$$\begin{aligned} \alpha^i &= X_i(\alpha) + Y_i(\alpha)\sqrt{c} \\ \bar{\alpha}^i &= X_i(\alpha) - Y_i(\alpha)\sqrt{c} \end{aligned}$$

Williams cryptosystem - efficient exponentiation

Assume now

$$a^2 - cb^2 = 1$$

Then $\alpha\bar{\alpha} = 1$ and consequently

$$X_i^2 - cY_i^2 = 1$$

Moreover, for $j \geq i$

$$\begin{aligned} X_{i+j} &= 2X_iX_j - X_{j-i} \\ Y_{i+j} &= 2X_iY_j - Y_{j-i} \end{aligned}$$

From these and following equations:

$$\begin{aligned} X_{i+j} &= X_iX_j + cY_iY_j \\ Y_{i+j} &= Y_iX_j + X_iY_j \end{aligned}$$

we get the recursive formulas:

$$\begin{aligned} X_{2i} &= X_i^2 + cY_i^2 = 2X_i^2 - 1 \\ Y_{2i} &= 2X_iY_i \\ X_{2i+1} &= 2X_iX_{i+1} - X_1 \\ Y_{2i+1} &= 2X_iY_{i+1} - Y_1 \end{aligned}$$

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.

Congruences on numbers of type $a + b\sqrt{c}$ are defined:

$$a_1 + b_1\sqrt{c} \equiv a_2 + b_2\sqrt{c} (\mathrm{mod}\, n) \Leftrightarrow a_1 \equiv a_2 (\mathrm{mod}\, n), b_1 \equiv b_2 (\mathrm{mod}\, n)$$

Instead of $a^2 - cb^2 = 1$ we will consider congruence $a^2 - cb^2 \equiv 1 (\mathrm{mod}\, n)$

Basic Lemma: Let $n = p \cdot q$ (both primes) and let a, b, c be such that $a^2 - cb^2 \equiv 1 \pmod{n}$. Moreover, let the Jacobi-Legendre symbols

$$\varepsilon_p = (c|p), \varepsilon_q = (c|q)$$

satisfy the congruence

$$\varepsilon_i \equiv -i (\mathrm{mod}\, 4) \text{ for } i \in \{p, q\}.$$

Assume also that $\gcd(cb, n) = 1$ and $(2(a + 1)|n) = 1$.
Denote

$$m = \frac{(p - \varepsilon_p)(q - \varepsilon_q)}{4}$$

and assume that e and d satisfy the congruence

$$ed \equiv \frac{(m + 1)}{2} \mathrm{mod}\, m.$$

Under these assumptions

$$\alpha^{2ed} \equiv \pm\alpha (\mathrm{mod}\, n),$$

where

$$\alpha = a + b\sqrt{c}.$$

This lemma plays the same role in showing correctness of the Williams cryptosystem as Euler's theorem plays for showing correctness of the RSA cryptosystem.

Design of cryptosystem

Choose p, q, compute $n = pq$.

Choose c such that Jacobi-Legendre symbols $\varepsilon_p, \varepsilon_q$ satisfy congruences of previous lemma (c can be chosen by a trial).

Choose (by trial) a number s such that

$$(s^2 - c|n) = -1, \quad \gcd(n, s) = 1.$$

Let m be as in Basic lemma and d be such that $\gcd(m, d) = 1$ and let e be such that

$$ed \equiv \frac{(m+1)}{2} \bmod m.$$

Public key: n, e, c, s
Secret key: p, q, m, d

Encryption: A plaintext $0 < w < n$ will first be encoded as a number α_w of the form $a + b\sqrt{c}$.

Denote

$$b_1 = 0, \gamma = w + \sqrt{c} \text{ if } (w^2 - c|n) = 1$$

$$b_1 = 1, \gamma = (w + \sqrt{c})(s + \sqrt{c}) \text{ if } (w^2 - c|n) = -1$$

In both cases:

$$(\gamma\bar{\gamma}|n) = 1.$$

Define: $\alpha = \gamma\bar{\gamma}^{-1} = \frac{\gamma}{\bar{\gamma}}$
(1) if $b_1 = 0$, then $\alpha \equiv \frac{w^2+c}{w^2-c} + \frac{2w}{w^2-c}\sqrt{c} \pmod n$
(2) if $b_1 = 1$ then $\alpha \equiv a + b\sqrt{c}$ where $a = \frac{(w^2+c)(s^2+c)+4csw}{(w^2-c)(s^2-c)}$ $\quad b = \frac{2s(w^2+c)+2w(s^2+c)}{(w^2-c)(s^2-c)}$

In both cases $\alpha\bar{\alpha} = a^2 - cb^2 \equiv 1 \pmod n$ and $(2(a+1)|n) = 1$.

Decryption: cryptotext: (E, b_1, b_2), where $E = (X_e(\alpha)Y_e(\alpha)^{-1} \bmod n)$, $b_2 \in \{0,1\}$, depending whether a is even or odd.

Decryption: Using E the receiver may compute:

$$\begin{aligned}\alpha^{2e} &\equiv \frac{\alpha^{2e}}{(\alpha\bar{\alpha})^e} \equiv \frac{\alpha^e}{\bar{\alpha}^e} = \frac{X_e(\alpha)+Y_e(\alpha)\sqrt{c}}{X_e(\alpha)-Y_e(\alpha)\sqrt{c}} \\ &\equiv \frac{E+\sqrt{c}}{E-\sqrt{c}} = \frac{E^2+c}{E^2-c} + \frac{2E}{E^2-c}\sqrt{c} (\bmod n)\end{aligned}$$

(The above computation can perform also a cryptanalyst. Trapdoor is needed for the next computation.)

$$\alpha^{2ed} = X_{2ed}(\alpha) + Y_{2ed}(\alpha)\sqrt{c} = X_d(\alpha^{2e}) + Y_d(\alpha^{2e})\sqrt{c}$$

Now all assumptions of Basic lemma are satisfied and, consequently

$$\alpha^{2ed} \equiv \pm\alpha (\bmod n)$$

b_2 is then used to determine which of the above signs is correct.

w is now obtained as follows:

Denote:

$$\alpha' = \begin{cases} \alpha & \text{if } b_1 = 0 \\ \alpha\frac{s-\sqrt{c}}{s+\sqrt{c}} & \text{if } b_1 = 1 \end{cases}$$

Then

$$\alpha' \equiv \frac{w+\sqrt{c}}{w-\sqrt{c}} (\bmod n)$$

and

$$w \equiv \frac{\alpha'+1}{\alpha'-1}\sqrt{c}\ (\bmod n)$$

WHEN is a CRYPTOSYSTEM (perfectly) SECURE?

In the rest of this lecture we deal with the above key 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 not acceptable that some information about plaintext could be obtained.

- Intuitively, a cryptosystem is (perfectly) secure if one cannot get any (new) information about the corresponding plaintext from any cryptotext, even if some a priori information about plaintext is known.
- It is very nontrivial to define correctly when a cryptosystem is (computationally) perfectly secure.
- It has been shown that perfectly secure cryptosystem have to use randomized encryptions.

Cryptography and Randomness

Randomness and cryptography are deeply related.

1. Prime importance of encryption methods is to transform 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

$$x_i = e_k(x_{i-1}), i \geq 1.$$

It is intuitive clear that if e_k is "cryptographically secure", then it is likely that the sequence $x_0 x_1 x_2 x_3$ is quite random.

Perfect encryption can therefore produce quite perfect (pseudo)randomness

2. 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 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 really SECURE?

First some very basic and very technical concepts:

Definition A function $f : \mathbf{N} \rightarrow \mathbf{R}$ is a negligible function if for any polynomial $p(n)$ and almost all n, it holds

$$f(n) \leq \frac{1}{p(n)}.$$

Definition – computational distinguishibility Let $X = \{X_n\}_{n\in\mathbf{N}}$ and $Y = \{Y_n\}_{n\in\mathbf{N}}$ 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

$$d_A(n) = |\text{Pr}[A(X_n) = 1] - \text{Pr}[A(Y_n) = 1]|$$

is a negligible function in n.

Secure encryptions – pseudorandom generators

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

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

$$G(s) = b(s) \cdot b(f(s)) \cdots b(f^{l(|s|)-1}(s))$$

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 $f(x)$ when x is uniformly chosen, to predict $b(x)$ substantially better than with probability $1/2$.)

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

Cryptographically strong pseudo-random generators

Fundamental question: when a pseudo-random generator is 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 foundation of computing.

Indeed, a cryptographically strong pseudo-random generator exists if and only if a one-way function exists what is equivalent with $P \neq UP$ and what implies $P \neq NP$.

Examples of cryptographically strong pseudo-random generators

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 $x \bmod n$, coin-tossing is the best possible way to estimate the least significant bit of x after seeing $x^2 \bmod n$.)

Let n be a Blum integer. Choose a random quadratic residue x_0 (modulo n).
For $i \geq 0$ let

$$x_{i+1} = x_i^2 \bmod n, \; b_i = \text{the least significant bit of } x_i$$

For each integer i, let

$$BBS_{n,i}(x_0) = b_0 \dots b_{i-1}$$

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

BBS pseudo-random generator – analysis

Choose random x, relatively prime to n, compute $x_0 = x^2 \bmod n$

$$x_{i+1} = x_i^2 \bmod n,\ b_i = \text{the least significant bit of } x_i$$

$$BBS_{n,i}(x_0) = b_0 \dots b_{i-1}$$

Assume that the BBS pseudo-randomize generator with a Blum integer is not unpredictable to the left.

Let y be a quadratic residue from $\mathbb{Z}_n^\star$.

Compute $BBS_{n,i-1}(y)$ for some $i > 1$.

Let us pretend that last $(i-1)$ of $BBS_{n,i}(x)$ are actually the first $(i-1)$ bits of $BBS_{n,i-1}(y)$, where x is the principal square root of y.

Hence, if the BBS pseudo-random generator is not unpredictable to the left, then there exists a better method than coin-tossing to determine the least significant bit of x, what is, as mentioned above, impossible.

Randomized encryptions

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

A cryptoanalyst who knows the public encryption function e_k and a cryptotext c can 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: Given:	plaintext-space	$\mathcal{P}$
	cryptotext	$\mathcal{C}$
	key-space	$\mathcal{K}$
	random-space	$\mathcal{R}$

encryption: $e_k : \mathcal{P} \times \mathcal{R} \rightarrow \mathcal{C}$
decryption: $d_k : \mathcal{C} \rightarrow \mathcal{P}$ or $\mathcal{C} \rightarrow 2^{\mathcal{P}}$
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 with encryption function e is semantically secure if for every feasible algorithm A, there exists a feasible algorithm B so that for every two functions

$$f, h : \{0,1\}^* \to \{0,1\}^n$$

and all probability ensembles $\{X_n\}_{n\in\mathbf{N}}$, where X_n ranges over $\{0,1\}^n$

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

where μ is a negligible function.

It can be shown that any 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 integers $n = p \times q$ – key
Plaintex-space – all binary strings
Random-space – QR_n
Crypto-space – $QR_n \times \{0, 1\}^\star$

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

$$x_{i+1} = x_i^2 \bmod n$$

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

A hash function

$$h : \{0,1\}^* \rightarrow \{0,1\}^n$$

maps messages (inputs) into hash-codes or hash-values or (digital) fingerprints or message digests).

Hash values are used as compact representations of images.

A hash function maps infinite domain into a finite one. Therefore there have to be many collisions.

A pair (m_1, m_2) is called a collision of a hash function h if $h(m_1) = h(m_2)$.

In some applications it is crucial that given an $h(m_1)$ an adversary has practically no chance to find an m_2 such that $h(m_1) = h(m_2)$.

Definition A hash function h is collision free if it is computationally infeasible to find two messages m_1 and m_2 such that $h(m_1) = h(m_2)$.

There are candidates for collision free hash function.

Randomized version of RSA-like cryptosystems

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

$$f : D \to D, D \subset \{0,1\}^n,$$

for any pseudorandom generator

$$G : \{0,1\}^k \to \{0,1\}^l, k << l$$

and any hash function

$$h : \{0,1\}^l \to \{0,1\}^k,$$

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:

1. A random string $r \in \{0,1\}^k$ is chosen.
2. Set $x = (m \oplus G(r))||(r \oplus h(m \oplus G(r)))$. (If $x \notin D$ go to step 1.)
3. Compute $c = f(x)$.

Decryption of a cryptotext c.

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

Comment Operation "$||$" stands for a concatenation of strings.

Bloom-Goldwasser cryptosystem

Private key: Blum primes p and q.
Public key: $n = pq$.

Encryption of $m \in \{0,1\}^k$.

1. Randomly choose $s_0 \in \{0, 1, \ldots, n\}$.
2. For $i = 1, 2, \ldots, k+1$ compute

$$s_i \leftarrow s_{i-1}^2 \bmod n$$

   and $\sigma_i = \mathsf{lsb}(s_i)$.

   The cryptotext is (s_{m+1}, c), where $c = m \oplus \sigma_1\sigma_2\ldots\sigma_k$.

Decryption: of the cryptotext (r, c):

   Let $d = 2^{-k} \bmod \phi(n)$.

1. Let $s_1 = r^d \bmod n$.
2. For $i = 1, \ldots, k$, compute $\sigma_i = \mathsf{lsb}(s_i)$ and $s_{i-1} \leftarrow s_i^2 \bmod n$.

The plaintext m can then be computed as $c \oplus \sigma_1\sigma_2\ldots\sigma_m$.

{"lsb" stands for "least significant bit"}

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 devise its attacks after the scheme has been specified, makes design of such 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 needed to deal with general cryptography problems as zero-knowledge proofs will be presented and discussed.

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