Part V

Public-key cryptosystems, I. Key exchange, knapsack, RSA

The main problem of secret key (or symmetric) cryptography is that in order to send securely

A secret message

we need to send at first securely

a secret key

and therefore **secret key cryptography is clearly not a sufficiently good tool for massive communication capable to protect secrecy, privacy and anonymity.**

SECURE ENCRYPTION - a PRACTICAL POINT OF VIEW

From practical point of view encryptions by a cryptosystem can be considered as secure if they cannot be broken by many (thousands) superomputeers with exaflop performance working for some years.

CONTENT

In this chapter we describe the birth of public key cryptography, that can better manage key distribution problem, and three of its cryptosystems, especially **RSA**.

The basic idea of a public key cryptography:

In a public key cryptosystem not only the encryption and decryption algorithms are public, but for each user U also the key e_U for encrypting messages (by anyone) for U is public.

Moreover, each user U keeps secret another (decryption) key, d_U, that can be used for decryption of messages that were addressed to him and encrypted with the help of the public encryption key e_U.

Encryption and decryption keys could (and should) be different - we can therefore speak also about **asymmetric cryptography**. Secret key cryptography, that has the same key for encryption and for decryption is also called **symmetric cryptography**.

KEYS DISTRIBUTION PROBLEM

- **The main problem of secret-key cryptography**: Before two people can exchange a secret (a message) they must already share a secret (the encryption/decryption key).
- Key distribution has been a big problem for 2000 of years, especially during both World Wars.
- Around 1970 the vision of an internet started to appear (ARPAnet was created in 1969) and it started to be clear that an enormous communication potential that a whole world connecting network could provide, could hardly be fully utilized unless secrecy of communication can be established and therefore **key distribution problem** **started to be seen as problem of immense importance**.
- For example around 1970 only Us government institutions needed to distribute daily tons of keys (on discs, tapes,...) to users they planned to communicate.
- Big banks had special employees that used to travel all the time around the world and to deliver keys, in special padlock briefcases, to everyone who had to get a message next week.
- Informatization of society was questioned because if governments had problems with key distribution how smaller companies could handle key distribution problem without bankrupting?
- At the same time, the **key distribution problem** used to be considered, practically by all, as an **unsolvable problem**.

FIRST INGENIOUS IDEA - KEY PLAYERS

Whitfield Diffie (1944), graduated in mathematics in 1965, and started to be obsessed with the key distribution problem - he realized that whoever could find a solution of this problem would go to history as one of the all-time greatest cryptographers.

In 1974 Diffie convinced **Martin Hellman** (1945), a professor in Stanford, to work together on the key distribution problem - Diffie was his graduate student.

In 1975 they got a basic idea that the key distribution may not be needed that can be now illustrated as follows

A padlock protocol

- **If Alice was to send securely a message to Bob, she puts the message into a box, lock the box with a padlock and sends the box to Bob.**
- **Bob has no key to open the box, so he uses another padlock to double-lock the box and sends this now doubly padlocked box back to Alice.**
- **Alice uses her key to unlock her padlock (but she cannot unlock Bob's padlock) and sends the box back to Bob.**
- **Bob uses his key to unlock his (now single) padlock and reads the message.**

Great idea was born. The problem now was to find a computational realization of such an idea - if possible. Mathematically, the problem was to find a simple one-way

FIRST ATTEMPT to DIGITALIZE PADLOCK PROTOCOL

Let us replace uses of padlocks by substitution encryptions.

Alice's key.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
H	F	S	U	G	T	A	K	V	D	E	O	Y	J	B	P	N	X	W	C	Q	R	I	M	Z	L

Bob's key.

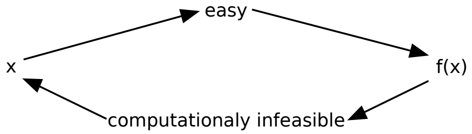
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
C	P	M	G	A	T	N	O	J	E	F	W	I	Q	B	U	R	Y	H	X	S	D	Z	K	L	V

Message	m	e	e	t	m	e	a	t	n	o	o	n
Alice's encrypt.	Y	G	G	C	Y	G	H	C	J	B	B	J
Bob's encrypt.	L	N	N	M	L	N	O	M	E	P	P	E
Alice's decrypt.	Z	Q	Q	X	Z	Q	L	X	K	P	P	K
Bob's decrypt.	w	n	n	t	w	n	y	t	x	b	b	x

The idea does not work.Why?

ONE-WAY FUNCTIONS

Informally, a function $F : N \rightarrow N$ is said to be a **one-way function** if it is easily computable - in polynomial time - but any computation of its inverse is infeasible.



A **one-way permutation** is a 1-1 one-way function.

A more formal approach

Definition A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is called a strongly one-way function if the following conditions are satisfied:

1. **f can be computed in polynomial time;**
2. **there are $c, \varepsilon > 0$ such that $|x|^\varepsilon \leq |f(x)| \leq |x|^c$;**
3. **for every randomized polynomial time algorithm A, and any constant $c > 0$, there exists an n_c such that for $|x| = n > n_c$**

$$P_r(A(f(x)) \in f^{-1}(f(x))) < \frac{1}{n^c}.$$

Candidates: **Modular exponentiation:** $f(x) = a^x \bmod n$
Modular squaring $f(x) = x^2 \bmod n$, $n - a$ Blum integer
Prime number multiplication $f(p, q) = pq$.

APPLICATION - COMPUTER PASSWORDS SECRECY PROBLEM

A **naive solution** to the password secrecy problem is to keep in computer a file with entries as

login CLINTON password BUSH,

that is a list of login names and corresponding passwords. This is not **safe** enough.

A **more safe method** is to keep in the computer a file with entries as

login CLINTON password BUSH one-way function f_c

where BUSH is a "public" password and CLINTON is the only one that knows a "secret" password, say MADONNA, such that

$$f_c(\text{MADONNA}) = \text{BUSH}$$

PUBLIC ESTABLISHMENT of SECRET KEYS

Main problem of the secret-key cryptography: is a need to make a secure distribution (establishment) of secret keys ahead of intended transmissions.

Diffie+Hellman solved this problem of key distribution first in 1976 by designing **a protocol for secure key establishment (distribution) over public communication channels.**

Diffie-Hellman Protocol: If two parties, Alice and Bob, want to create a common secret key, then they first agree, somehow, on a large prime p and a $q < p$ of large order in Z_p^* and then they perform, using a public channel, the following activities.

- Alice chooses, randomly, a large $1 \leq x < p-1$ and computes
$$X = q^x \bmod p.$$
- Bob also chooses, again randomly, a large $1 \leq y < p-1$ and computes
$$Y = q^y \bmod p.$$
- Alice and Bob exchange **X** and **Y**, through a public channel, but keep x, y secret.
- Alice computes $Y^x \bmod p$ and Bob computes $X^y \bmod p$ and then each of them has the same key
$$k = q^{xy} \bmod p.$$

An eavesdropper seems to need, in order to determine x from **X, q, p** and y from **Y, q, p**, a capability to compute discrete logarithms, or to compute q^{xy} from q^x and q^y, what is believed to be infeasible.

MAN-IN-THE-MIDDLE ATTACKS

The following attack, called " a man-in-the-middle attack, is possible against the Diffie-Hellman key establishment protocol.

1. Eve chooses an integer (exponent) z.
2. Eve intercepts q^x and q^y – when they are sent from Alice to Bob and from Bob to Alice.
3. Eve sends q^z to both Alice and Bob. (After that Alice believes she has received q^y and Bob believes he has received q^x.)
4. Eve computes $K_A = q^{xz}$ (mod p) and $K_B = q^{yz}$ (mod p).
   Alice, not realizing that Eve is in the middle, also computes K_A and
   Bob, not realizing that Eve is in the middle, also computes K_B.
5. When Alice sends a message to Bob, encrypted with K_A, Eve intercepts it, decrypts it, then encrypts it with K_B and sends it to Bob.
6. Bob decrypts the message with K_B and obtains the message. At this point he has no reason to think that communication was insecure.
7. Meanwhile, Eve enjoys reading Alice's message.

GENERALISATION - BLOOM's KEY PRE-DISTRIBUTION PROTOCOL

allows a trusted authority (Trent - TA) to distribute secret keys to $\frac{n(n-1)}{2}$ pairs of n users.

Let a large prime $p > n$ be publicly known. Steps of the protocol follow:

1. Each user U in the network is assigned, by Trent, a unique public number $r_U < p$.
2. Trent chooses three random numbers a, b and c, smaller than p.
3. For each user U, Trent calculates two numbers

   $$a_U = (a + br_U) \bmod p, \qquad b_U = (b + cr_U) \bmod p$$

   and sends them via his secure channel to U.
4. Each user U creates the polynomial

   $$g_U(x) = a_U + b_U(x).$$
5. If Alice (A) wants to send a message to Bob (B), then Alice computes her key $K_{AB} = g_A(r_B)$ and Bob computes his key $K_{BA} = g_B(r_A)$.
6. It is easy to see that $K_{AB} = K_{BA}$ and therefore Alice and Bob can now use their (identical) keys to communicate using some secret-key cryptosystem.

SECURE COMMUNICATION with SECRET-KEY CRYPTOSYSTEMS

and without any need for secret key distribution

The idea contained in the above mention padlock protocol has been later materialized by Shamir.

(**Shamir's "no-key algorithm"**)

Basic assumption: Each user X has its own

secret encryption function e_X

secret decryption function d_X

and all these functions commute (to form a commutative cryptosystem).

Communication protocol

with which Alice can send a message w to Bob.

1. Alice sends $e_A(w)$ to Bob
2. Bob sends $e_B(e_A(w))$ to Alice
3. Alice sends $d_A(e_B(e_A(w))) = e_B(w)$ to Bob
4. Bob performs the decryption to get $d_B(e_B(w)) = w$.

Disadvantage: 3 communications are needed (in such a context 3 is a much too large number).

BIRTH of PUBLIC KEY CRYPTOGRAPHY I

Diffie and Hellman demonstrated their discovery of the hey establishment protocol at the National Computer Conference in June 1976 and astonished the audience.

Next year they filled for a patent.

However, the solution of the key distribution problem through Diffie-Hellman protocol could still be seen as not good enough. Why?

The protocol required still too much communication and cooperation of both parties for quite a time.

BIRTH of PUBLIC KEY CRYPTOGRAPHY II

Already in 1975 Diffie got the an idea for key distribution that seemed to be better: To design asymmetric cryptosystems - public key cryptosystems.

The basic idea was that in a public key cryptosystem not only the encryption and decryption algorithms would be public, but for each user U also the key e_U for encrypting messages (by anyone) for U would be public, and each user U would keep secret another key, d_U, that could be used for decryption of messages that were addressed to him and encrypted with the help of public encryption key e_U.

The realization that a cryptosystem does not need to be symmetric can be seen nowadays as the single most important breakthrough in modern cryptography.

Diffie published his idea in the summer of 1975 in spite of the fact that he had no idea how to design such a system.

To turn asymmetric cryptosystems from a great idea into a practical invention, somebody had to discover an appropriate mathematical function.

Mathematically, the problem was to find a simple enough one-way trapdoor function.

A search (hunt) for such a function started.

CRYPTOGRAPHY and COMPUTATIONAL COMPLEXITY

Modern cryptography uses such encryption methods that no “enemy” can have enough computational power and time to do decryption (even those capable to use thousands of supercomputers during tens of years for encryption).

Modern cryptography is based on negative and positive results of complexity theory – on the fact that for some algorithm problems no efficient algorithm seem to exists, surprisingly, and for some “small” modifications of these problems, surprisingly, simple, fast and good (randomized) algorithms do exist. Examples:

Integer factorization: Given an integer $n(=pq)$, it is, in general, unfeasible, to find p, q.

There is a list of “most wanted to factor integers”. Top recent successes, using thousands of computers for months.

(*) Factorization of $2^{2^9}+1$ with 155 digits (1996)

(**) Factorization of a “typical” 232 digits integer RSA-768 (2009)

Primes recognition: Is a given n a prime? – fast randomized algorithms exist (1977). The existence of polynomial deterministic algorithms for primes recognition has been shown only in 2002

COMPUTATIONALLY INFEASIBLE PROBLEMS

Discrete logarithm problem: Given integers x, y, n, determine an integer a such that $y \equiv x^a \pmod{n}$ – infeasible in general.

Discrete square root problem: Given integers y, n, compute an integer x such that $y \equiv x^2 \pmod{n}$ – infeasible in general, but easy if factorization of n is known

Knapsack problem: Given a (knapsack - integer) vector $X = (x_1, \ldots, x_n)$ and an (integer capacity) c, find a binary vector $(b_1, \ldots, b_n)$ such that

$$\sum_{i=1}^{n} b_i x_i = c.$$

Problem is NP-hard in general, but easy if $x_i > \sum_{j=1}^{i-1} x_j, 1 < i \leq n$.

TRAPDOOR ONE-WAY FUNCTIONS

The key concept for design of public-key cryptosystems is that of trapdoor one-way functions.

A function $f : X \to Y$ is trapdoor one-way function

- if f and its inverse can be computed efficiently,
- yet even the complete knowledge of the algorithm to compute f does not make it feasible to determine a polynomial time algorithm to compute the inverse of f.
- To compute inverse of f efficiently some special, "trapdoor", knowledge is needed.

A candidate: modular squaring $\sqrt{y} \bmod n$ with a fixed modulus n.

- computation of discrete square roots is unfeasible in general, but quite easy if the decomposition of the modulus **n** into primes is known.

A way to design a trapdoor one-way function is to transform an easy case of a hard (one-way) function to a hard-looking case of such a function, that can be, however, solved easily by those knowing how the above transformation was performed.

GENERAL KNAPSACK PROBLEM – UNFEASIBLE

KNAPSACK PROBLEM: Given an integer-vector $X = (x_1, \ldots, x_n)$ and an integer c. Determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

Knapsack problem with superincreasing vector – easy

Problem Given a **superincreasing integer-vector** $X = (x_1, \ldots, x_n)$ (i.e. $x_i > \sum_{j=1}^{i-1} x_j, i > 1$) and an integer c,

determine a binary vector $B = (b_1, \ldots, b_n)$ (if it exists) such that $XB^T = c$.

Algorithm – to solve knapsack problems with superincreasing vectors:

```
for $i = n \leftarrow$ downto 2 do
    if $c \geq 2x_i$ then terminate {no solution}
        else if $c \geq x_i$ then $b_i \leftarrow 1; c \leftarrow c - x_i$;
            else $b_i = 0$;
if $c = x_1$ then $b_1 \leftarrow 1$
    else if $c = 0$ then $b_1 \leftarrow 0$;
        else terminate {no solution}
```

Example $X = (1,2,4,8,16,32,64,128,256,512)$, $c = 999$
$X = (1,3,5,10,20,41,94,199)$, $c = 242$

KNAPSACK ENCODING – BASIC IDEAS

Let a (knapsack) vector

$$A = (a_1, \dots, a_n)$$

be given.

Encoding of a (binary) message $B = (b_1, b_2, \dots, b_n)$ by A is done by the vector $\times$ vector multiplication:

$$AB^T = c$$

and results in the cryptotext c.

Decoding of c requires to solve the knapsack problem for the instant given by the knapsack vector A and the cryptotext c.

The problem is that decoding seems to be infeasible.

Example
If $A = (74, 82, 94, 83, 39, 99, 56, 49, 73, 99)$ and $B = (1100110101)$ then

$$AB^T =$$

DESIGN of KNAPSACK CRYPTOSYSTEMS

1. Choose a superincreasing raw vector $X = (x_1, \ldots, x_n)$.
2. Choose **m, u** such that $m > 2x_n,\ gcd(m, u) = 1$.
3. Compute u^{-1} mod $m, X' = (x'_1, \ldots, x'_n)$, $x'_i = \underbrace{\underbrace{ux_i}_{\text{diffusion}} \mod m}_{\text{confusion}}$.

Cryptosystem: X' – public key
X, u, m – trapdoor information

Encryption: of a binary raw vector w of length n: $c = X'w^T$

Decryption: compute $c' = u^{-1}c$ mod m
and solve the knapsack problem with X and c'.

Lemma Let X, m, u, X', c, c' be as defined above. Then the knapsack problem instances (X, c') and (X', c) have at most one solution, and if one of them has a solution, then the second one has the same solution.

Proof Let $X'w^T = c$. Then

$$c' \equiv u^{-1}c \equiv u^{-1}X'w^T \equiv u^{-1}uXw^T \equiv Xw^T (\text{mod } m).$$

Since X is superincreasing and $m > 2x_n$ we have

$$(Xw^T) \text{ mod } m = Xw^T$$

and therefore

$$c' = Xw^T.$$

DESIGN of KNAPSACK CRYPTOSYSTEMS – EXAMPLE

Example **X** = (1,2,4,9,18,35,75,151,302,606)
m = 1250, u = 41
X' = (41,82,164,369,738,185,575,1191,1132,1096)

In order to encrypt an English plaintext, we first encode its letters by 5-bit numbers _ - 00000, A - 00001, B - 00010,. . . and then divide the resulting binary strings into blocks of length 10.

Plaintext: Encoding of AFRICA results in vectors

$$w_1 = (0000100110) \qquad w_2 = (1001001001) \qquad w_3 = (0001100001)$$

Encryption:
$c_{1'} = X'w_1^T = 3061 \qquad c_{2'} = X'w_2^T = 2081 \qquad c_{3'} = X'w_3^T = 2203$

Cryptotext: (3061,2081,2203)

Decryption of cryptotexts: (2163, 2116, 1870, 3599)

By multiplying with $u^{-1} = 61 \pmod{1250}$ we get new cryptotexts (several new c')

$$(693, 326, 320, 789)$$

And, in the binary form, solutions B of equations $XB^T = c'$ have the form

$$(1101001001, 0110100010, 0000100010, 1011100101)$$

Therefore, the resulting plaintext is:

ZIMBABWE

STORY of KNAPSACK

Invented: 1978 - Ralph C. Merkle, Martin Hellman
Patented: in 10 countries
Broken: 1982: Adi Shamir

New idea: to use iterated knapsack cryptosystem with **hyper-reachable vectors**.

Definition A knapsack vector $X' = (x_{1'}, \ldots, x_{n'})$ is obtained from a knapsack vector $X = (x_1, \ldots, x_n)$ by strong modular multiplication if

$$x'_i = ux_i \bmod m, i = 1, \ldots, n,$$

where

$$m > 2\sum_{i=1}^{n} x_i$$

and $gcd(u, m) = 1$. A knapsack vector X' is called hyper-reachable, if there is a sequence of knapsack vectors $Y = X_0, X_1, \ldots, X_k = X'$,

where X_0 is a super-increasing vector, and for $i = 1, \ldots, k$ X_i is obtained from X_{i-1} by a strong modular multiplication.

Iterated knapsack cryptosystem was broken **in 1985 - by E. Brickell**

New idea: to use **knapsack cryptosystems with dense vectors**. Density of a knapsack vector $X = (x_1, \ldots, x_n)$ is defined by $d(x) = \frac{n}{log(max\{x_i | 1 \leq i \leq n\})}$

Remark. Density of super-increasing vectors of length n is $\leq \frac{n}{n-1}$

KNAPSACK CRYPTOSYSTEM – COMMENTS

The term "knapsack" in the name of the cryptosystem is quite misleading.

By **Knapsack problem** one usually understands the following problem:

Given n items with weights $w_1, w_2, \ldots, w_n$, values $v_1, v_2, \ldots, v_n$ and a knapsack limit c, the task is to find a bit vector $(b_1, b_2, \ldots, b_n)$ such that $\sum_{i=1}^{n} b_i w_i \leq c$ and $\sum_{i=1}^{n} b_i v_i$ is as large as possible.

The term **subset problem** is usually used for problems deployed in our construction of knapsack cryptosystems. It is well-known that the decision version of this problem is NP-complete.

For our version of the **knapsack problem** the term **Merkle-Hellman (Knapsack) Cryptosystem** is often used.

McELIECE CRYPTOSYSTEM

McEliece cryptosystem is based on a similar design principle as the Knapsack cryptosystem. McEliece cryptosystem is formed by transforming an easy to break cryptosystem (based on an easy to decode linear code) into a cryptosystem that is hard to break (because it seems to be based on a linear code that is, in general, *NP*-hard).

The underlying fact is that the decision version of the decryption problem for linear codes is in general *NP*-complete. However, for special types of linear codes polynomial-time decryption algorithms exist. One such a class of linear codes, the so-called Goppa codes, are often used to design McEliece cryptosystem.

Goppa codes are $[2^m, n - mt, 2t + 1]$-codes, where $n = 2^m$.

(McEliece suggested to use $m = 10, t = 50$.)

McELIECE CRYPTOSYSTEM – DESIGN

Goppa codes are $[2^m, n - mt, 2t + 1]$-codes, where $n = 2^m$.

Design of McEliece cryptosystems. Let

- G be a generating matrix for an $[n, k, d]$ Goppa code C;
- S be a $k \times k$ binary matrix invertible over Z_2;
- P be an $n \times n$ permutation matrix;
- $G' = SGP$.

Plaintexts: $P = (Z_2)^k$; cryptotexts: $C = (Z_2)^n$, key: $K = (G, S, P, G')$, message: w
G' is made public, G, S, P are kept secret.

Encryption: $e_K(w, e) = wG' + e$, where e is any binary vector of length n & weight t.

Decryption of a cryptotext $c = wG' + e \in (Z_2)^n$.

1. Compute $c_1 = cP^{-1} = wSGPP^{-1} + eP^{-1} = wSG + eP^{-1}$
2. Decode c_1 to get $w_1 = wS$,
3. Compute $w = w_1 S^{-1}$

COMMENTS on McELIECE CRYPTOSYSTEM I

1. Each irreducible polynomial over Z_2^m of degree t generates a Goppa code with distance at least $2t+1$.
2. In the design of McEliece cryptosystem the goal of matrices S and C is to modify a generator matrix G for an easy-to-decode Goppa code to get a matrix that looks as a random generator matrix for a linear code for which the decoding problem is **NP**-complete.
3. An important novel and unique trick is an introduction, in the encoding process, of a random vector e that represents an introduction of up to t errors – such a number of errors that are correctable using the given Goppa code and this is the basic trick of the decoding process.
4. Since P is a permutation matrix eP^{-1} has the same weight as e.
5. As already mentioned, McEliece suggested to use a Goppa code with $m = 10$ and $t = 50$. This provides a $[1024, 524, 101]$-code. Each plaintext is then a 524-bit string, each cryptotext is a 1024-bit string. The public key is an 524×1024 matrix.
6. Observe that the number of potential matrices S and P is so large that probability of guessing these matrices is smaller that probability of guessing correct plaintext!!!
7. It can be shown that it is not safe to encrypt twice the same plaintext with the same public key (and different error vectors).

COMMENTS on McELIECE CRYPTOSYSTEM II

- Cryptosystem was invented in 1978 by Robert McEliece.
- Cryptosystem is a candidate for post-quantum cryptography - all attempts to break it using quantum computers failed.
- There are nowadays various variant of the cryptosystem that use different easy to decode linear codes. Some are known not to be secure.
- McEliece cryptosystem was the first public key cryptosystem that used randomness - a very innovative step.
- Observe that public key is derived from private key by disguising the selected code as a general linear code.
- For a standard selection of parameters the public key is more than 521 000 bits long.
- Thata is why cryptosystem is rarely used in practise in spite of the fact that it has some advantages comparing with RSA cryptosystem discussed next - it has more easy encoding and decoding.

FINAL COMMENTS

1. **Deterministic public-key cryptosystems can never provide absolute security.** This is because an eavesdropper, on observing a cryptotext c can encrypt each possible plaintext by the encryption algorithm e_A until he finds c such that $e_A(w) = c$.
2. One-way functions exist if and only if **P = UP**, where UP is the class of languages accepted by **unambiguous polynomial time bounded nondeterministic Turing machine.**
3. There are actually two types of keys in practical use: A session key is used for sending a particular message (or few of them). A master key is usually used to generate several session keys.
4. **Session keys** are usually generated when actually required and discarded after their use. Session keys are usually keys of a secret-key cryptosystem.
5. **Master keys** are usually used for longer time and need therefore be carefully stored. Master keys are usually keys of a public-key cryptosystem.

RSA CRYPTOSYSTEM

The most important public-key cryptosystem is the RSA cryptosystem on which one can also illustrate a variety of important ideas of modern public-key cryptography.

For example, we will discuss various possible attacks on the security of RSA cryptosystems.

A special attention will be given in Chapter 7 to the problem of factorization of integers that play such an important role for security of RSA.

In doing that we will illustrate modern distributed techniques to factorize very large integers.

DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

Basic idea: prime multiplication is very easy, integer factorization seems to be unfeasible.

Design of RSA cryptosystems

1. Choose two large s-bit primes p,q, where $s \in [512, 1024]$, and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\bmod\ \phi(n))$$

Public key: n (modulus), e (encryption exponent)
Trapdoor information: p, q, d (decryption exponent)

Plaintext w
Encryption: cryptotext $c = w^e \bmod n$
Decryption: plaintext $w = c^d \bmod n$

Details: A plaintext is first encoded as a word over the alphabet $\{0, 1, \ldots, 9\}$, then divided into blocks of length $i-1$, where $10^{i-1} < n < 10^i$. Each block is taken as an integer and decrypted using modular exponentiation.

HOW TO DO EFFICIENTLY RSA COMPUTATIONS

How to compute w^e mod n? Use the method of exponentiation by squaring - see the Appendix - and perform all operations modulo n

How to compute d^{-1} mod $\phi(n)$? :

Method 1 Use Extended Euclid algorithm, see the Appendix, that shows how to find, given integers $0 < m < n$ with $GCD(m, n) = 1$, integers x, y such that

$$xm + yn = 1$$

Once this is done, $x = m^{-1} \bmod n$

Method 2 It follows from Euler Totient Theorem, see the Appendix, that

$$m^{-1} \equiv m^{\phi(n)-1} \bmod \phi(n)$$

if $m < n$ and $GCD(m, n) = 1$

PROOF of the CORRECTNESS of RSA

Let $c = w^e \bmod n$ be the cryptotext for a plaintext w, in the cryptosystem with

$$n = pq, ed \equiv 1 \pmod{\phi(n)}, \gcd(d, \phi(n)) = 1$$

In such a case

$$w \equiv c^d \bmod n$$

and, if the decryption is unique, $w = c^d \bmod n$.

Proof Since $ed \equiv 1 \pmod{\phi(n)}$, there exists a $j \in N$ such that $ed = j\phi(n) + 1$.

- Case 1. Neither p nor q divides w.
  In such a case $\gcd(n, w) = 1$ and by the Euler's Totient Theorem we get that
  $$c^d = w^{ed} = w^{j\phi(n)+1} \equiv w \pmod{n}$$
- **Case 2. Exactly one of p, q divides w – say p.**
  In such a case $w^{ed} \equiv w \pmod{p}$ and by Fermat's Little theorem $w^{q-1} \equiv 1 \pmod{q}$
  $$\begin{aligned}\Rightarrow w^{q-1} \equiv 1 \pmod{q} &\Rightarrow w^{\phi(n)} \equiv 1 \pmod{q}\\ &\Rightarrow w^{j\phi(n)} \equiv 1 \pmod{q}\\ &\Rightarrow w^{ed} \equiv w \pmod{q}\end{aligned}$$
  Therefore: $w \equiv w^{ed} \equiv c^d \pmod{n}$
- **Case 3.** Both p, q divide w.
  This cannot happen because, by our assumption, $w < n$.

DESIGN and USE of RSA CRYPTOSYSTEM

Example of the design and of the use of RSA cryptosystems.

- By choosing $p = 41, q = 61$ we get $n = 2501, \phi(n) = 2400$
- By choosing $d = 2087$ we get $e = 23$
- By choosing $d = 2069$ we get $e = 29$
- By choosing other values of d we would get other values of e.

Let us choose the first pair of encryption/decryption exponents ($e = 23$ and $d = 2087$).

Plaintext: KARLSRUHE **First encoding (letters–int.):** 100017111817200704

Since $10^3 < n < 10^4$, the numerical plaintext is divided into blocks of 3 digits $\Rightarrow$ therefore 6 integer plaintexts are obtained

$$100, 017, 111, 817, 200, 704$$

Encryptions:

$$\begin{array}{ccc} 100^{23} \bmod 2501, & 17^{23} \bmod 2501, & 111^{23} \bmod 2501 \\ 817^{23} \bmod 2501, & 200^{23} \bmod 2501, & 704^{23} \bmod 2501 \end{array}$$

provide cryptotexts:

$$2306, 1893, 621, 1380, 490, 313$$

Decryptions:

$$2306^{2087} \bmod 2501 = 100, 1893^{2087} \bmod 2501 = 17$$
$$621^{2087} \bmod 2501 = 111, 1380^{2087} \bmod 2501 = 817$$

RSA CHALLENGE

The first public description of the RSA cryptosystem was in the paper.

Martin Gardner: Mathematical games, Scientific American, 1977

and in this paper RSA inventors presented the following challenge.

Decrypt the cryptotext:

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

encrypted using the RSA cryptosystem with 129 digit number, called also RSA129

n: 114 381 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 830 597 123 513 958 705 058 989 075 147 599 290 026 879 543 541.

and with $e = 9007$.

The problem was solved in 1994 by first factorizing n into one 64-bit prime and one 65-bit prime, and then computing the plaintext

THE MAGIC WORDS ARE SQUEMISH OSSIFRAGE

RSA SECURITY

Security of RSA is based on the fact that for the following two problems no classical polynomial time algorithms seem to exist.

- Integer factorization problem.
- RSA problem: Given a public key (n, e) and a cryptotext c find an m such that $c = m^e \pmod n$.

PRIMES

- A prime p is an integer with exactly two divisors - 1 and p.
- Primes play very important role in mathematics.
- Already Euclid new that there are infinitely many primes.
- Probability that an n-bit integer is prime is $\frac{1}{2.3n}$. (The accuracy of this estimate is closely related to the *Rieman Hypothesis* considered often as the most important open problem of mathematics.)
- Each integer has a uniquer decomposition as a product of primes.
- Golbach conjecture: says that every even integer n can be written as the sum of two primes (verified for $n \leq 4 \cdot 10^{14}$).
- Vinogradov Theorem: Every odd integer $n > 10^{43000}$ is the sum of three primes.
- There are fast ways to determine whether a given integer is prime or not.
- However, if an integer is not a prime then it is very hard to find its factors.

HOW to DESIGN REALLY GOOD RSA CRYPTOSYSTEMS?

1. How to choose large primes p, q?
   Choose randomly a large integer p and verify, using a randomized algorithm, whether p is prime. If not, check $p+2, p+4, \ldots$ for primality. From the Prime Number Theorem it follows that there are approximately
   $$\frac{2^d}{\log 2^d} - \frac{2^{d-1}}{\log 2^{d-1}}$$
   d bit primes. (A probability that a 512-bit number is prime is 0.00562.)
2. **What kind of relations should be between p and q?**
   - 2.1 Difference $|p-q|$ should be neither too small nor too large.
   - 2.2 $\gcd(p-1, q-1)$ should not be large.
   - 2.3 Both $p-1$ and $q-1$ should not contain small prime factors.
   - 2.4 Quite ideal case: q, p should be safe primes -such that also $(p–1)/2$ and $(q-1)/2$ are primes. ($83, 107, 10^{100} - 166517$ are examples of safe primes).
3. **How to choose e and d?**
   - 3.1 Neither d nor e should be small.
   - 3.2 d should not be smaller than $n^{\frac{1}{4}}$. (For $d < n^{\frac{1}{4}}$ a polynomial time algorithm is known to determine d).

PRIMES RECOGNITION and INTEGERS FACTORIZATION

The key problems for the development of RSA cryptosystem are that of primes recognition and integers factorization.

On August 2002, the first polynomial time algorithm was discovered that allows to determine whether a given m bit integer is a prime. Algorithm works in time $O(m^{12})$.

Fast randomized algorithms for prime recognition has been known since 1977. One of the simplest one is due to Rabin and will be presented later.

For integer factorization situation is somehow different.

- No polynomial time classical algorithm is known.
- Simple, but not efficient factorization algorithms are known.
- Several sophisticated distributed factorization algorithms are known that allowed to factorize, using enormous computation power, surprisingly large integers.
- Progress in integer factorization, due to progress in algorithms and technology, has been recently enormous.
- Polynomial time quantum algorithms for integer factorization are known since 1994 (P. Shor).

Several simple and some sophisticated factorization algorithms will be presented and illustrated in the following.

LARGEST PRIMES

Largest known prime, from 25.1.2013, is Mersenne prime

$$2^{57,885,161} - 1$$

that has $17,425,170$ digits

The last 15 record primes were Mersenne primes (of the form $2^p - 1$).

Record was obtained by **Great Internet Mersenne Prime Search (GIMPS)** consortium established in 2006.

Next goal is to find prime that would have more than 100 millions of digits - there is a special price for that.

RABIN-MILLER's PRIME RECOGNITION

Rabin-Miller's Monte Carlo prime recognition algorithm is based on the following result from the number theory.

Lemma Let $n \in N$. Denote, for $1 \leq x \leq n$, by $C(x)$ the condition:
Either $x^{n-1} \neq 1 \pmod{n}$, or there is an $m = \frac{n-1}{2^i}$ for some i, such that $\gcd(n, x^m - 1) \neq 1$
If $C(x)$ holds for some $1 \leq x \leq n$, then n is not a prime. If n is not a prime, then $C(x)$ holds for at least half of x between 1 and n.

Algorithm:

Choose randomly integers $x_1, x_2, \ldots, x_m$ such that $1 \leq x_i \leq n$.
For each x_i determine whether $C(x_i)$ holds.

Claim: If $C(x_i)$ holds for some i, then n is not a prime for sure. **Otherwise n is declared to be prime. Probability that this is not the case is 2^{-m}.**

MILLER-RABIN PRIMALITY TESTING

Input: an odd integer $n > 3$;
Parameter k - an integer;

```
Write $n = 2^s t + 1$ for an odd $t$;
repeat $k$ times
   choose a random $0 < b < n$;
   $x \leftarrow b^t \bmod n$; $i \leftarrow 0$;
   if $x \neq 1$ then
                 if $i = s$ or $x = 1$ then output "composite", stop;
Output: prime
```

Theorem If n is prime, then the above algorithm always outputs "prime". If n is composite then the above algorithm outputs "composite" with probability greater than $1 - 4^{-k}$.

FACTORIZATION of 512-BITS and 708-BITS NUMBERS

On August 22, 1999, a team of scientists from 6 countries found, after 7 months of computing, using 300 very fast SGI and SUN workstations and Pentium II, factors of the so-called RSA-155 number with 512 bits (about 155 digits).

RSA-155 was a number from a Challenge list issue by the US company RSA Data Security and "represented" 95% of 512-bit numbers used as the key to protect electronic commerce and financial transmissions on Internet.

Factorization of RSA-155 would require in total 37 years of computing time on a single computer.

When in 1977 Rivest and his colleagues challenged the world to factor RSA-129, they estimated that, using knowledge of that time, factorization of RSA-129 would require 10^{16} years.

In 2005 RSA-640 was factorized - this took approximately 30 2.2GHz-Opteron-CPU years - over five months of calendar time.

In 2009 RSA-768, a 768-bits number, was factorized by a team from several institutions. Time needed would be 2000 years on a single 2.2 GHz AND Opterons. Cash price obtained - 30 000 $.

LARGE NUMBERS

Hindus named many large numbers - one having 153 digits.

Romans initially had no terms for numbers larger than 10^4.

Greeks had a popular belief that no number is larger than the total count of sand grains needed to fill the universe.

Large numbers with special names:

duotrigintillion=googol-10^{100} **googolplex**$-10^{10^{100}}$

FACTORIZATION of very large NUMBERS

W. Keller factorized F_{23471} which has 10^{7000} digits.

J. Harley factorized: $10^{10^{1000}} + 1$.

One factor: $316, 912, 650, 057, 350, 374, 175, 801, 344, 000, 001$

In 1992 E. Crandal, Doenias proved, using a computer that F_{22}, which has more than million of digits, is composite (but no factor of F_{22} is known).

Number $10^{10^{10^{34}}}$ was used to develop a theory of the distribution of prime numbers.

DESIGN OF GOOD RSA CRYPTOSYSTEMS

Claim 1. **Difference $|p-q|$ should not be small.**

Indeed, if $|p-q|$ is small, and $p > q$, then $\frac{(p+q)}{2}$ is only slightly larger than $\sqrt{n}$ because

$$\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$$

In addition, $\frac{(p+q)^2}{4} - n$ is a square, say y^2.
In order to factor n, it is then enough to test $x > \sqrt{n}$ until x is found such that $x^2 - n$ is a square, say y^2. In such a case

$$p + q = 2x, p - q = 2y \qquad \text{and therefore } p = x + y, q = x - y.$$

Claim 2. $\gcd(p-1, q-1)$ should not be large.

Indeed, in the opposite case $s = \operatorname{lcm}(p-1, q-1)$ is much smaller than $\phi(n)$ If

$$d'e \equiv 1 \bmod s,$$

then, for some integer k,

$$c^d \equiv w^{ed} \equiv w^{ks+1} \equiv w \bmod n$$

since $p-1|s, q-1|s$ and therefore $w^{ks} \equiv 1 \bmod p$ and $w^{ks+1} \equiv w \bmod q$. Hence, d' can serve as a decryption exponent.
Moreover, in such a case s can be obtained by testing.
Question Is there enough primes (to choose again and again new ones)?
No problem, the number of primes of length 512 bit or less exceeds 10^{150}.

WHAT SMALL REALLY MEANS

If $n = pq$ and $p - q$ is "small", then factorization can be quite easy.

For example, if $p - q < 2n^{0.25}$

(which for even small 1024-bit values of n is about $3 \cdot 10^{77}$)

then factoring of n is quite easy.

HOW IMPORTANT is FACTORIZATION for BREAKING RSA?

1. If integer factorization is feasible, then RSA is breakable.
2. There is no proof that factorization is indeed needed to break RSA.
3. If a method of breaking RSA would provide an effective way to get a trapdoor information, then factorization could be done effectively.

   Theorem Any algorithm to compute $\phi(n)$ can be used to factor integers with the same complexity.

   Theorem Any algorithm for computing d can be converted into a break randomized algorithm for factoring integers with the same complexity.
4. There are setups in which RSA can be broken without factoring modulus n.

   Example An agency chooses p, q and computes a modulus $n = pq$ that is publicized and common to all users $U_1, U_2, \ldots$ and also encryption exponents $e_1, e_2, \ldots$ are publicized. Each user U_i gets his decryption exponent d_i.

   In such a setting any user is able to find in deterministic quadratic time another user's decryption exponent.

BASIC RSA SECURITY PROBLEM

- RSA problem Given cryptotext c, public key n, e find plaintext w such that $c = w^e \pmod n$.
- RSA problem is not equivalent to the problem of factorization of the module n
- Computation of the secret key exponent and factorization of moduli are equivalent problems.

CHOSEN CRYPTOTEXT ATTACK

To encrypt a cryptotext $c = w^e$ the attacker can ask the holder of the decryption exponent d to decrypt an innocently looking message $c' = cr^e$ for some value r chosen by the attacker.

In such a case c' is an encryption of wr

If w' is outcome of such a decryption, then

$$w = w'r^{-1} \bmod n$$

This attack is based on the fact that $w_1^e w_2^e = (w_1 w_2)^e \bmod n$ - Therefore c' is an encryption of wr

SECURITY of RSA in PRACTICE

None of the numerous attempts to develop attacks on any RSA cryptosystem has turned out to be successful.

There are various results showing that it is impossible to obtain even only partial information about the plaintext from the cryptotext produced by the RSA cryptosystem.

We will show that were the following two functions, that are computationally polynomially equivalent, be efficiently computable, then the RSA cryptosystem with the encryption (decryption) exponents $e_k(d_k)$ would be breakable.

$$\textbf{parity}_{e_k}(c) = \text{the least significant bit of such an } w \text{ that } k(w) = c;$$
$$\textbf{half}_{e_k}(c) = 0 \text{ if } 0 \leq w < \tfrac{n}{2} \text{ and } \textbf{half}_{e_k}(c) = 1 \text{ if } \tfrac{n}{2} \leq w \leq n-1$$

We show two important properties of the functions *half* and *parity*.

1. Polynomial time computational equivalence of the functions **half** and **parity** follows from the following identities

$$\textbf{half}_{e_k}(c) = \textbf{parity}_{e_k}((c \times e_k(2)) \bmod n$$

$$\textbf{parity}_{e_k}(c) = \textbf{half}_{e_k}((c \times e_k(\frac{1}{2})) \bmod n$$

   and from the multiplicative rule $e_k(w_1)e_k(w_2) = e_k(w_1 w_2)$.

2. There is an efficient algorithm, on the next slide, to determine the plaintexts w from the cryptotexts c obtained from w by an RSA-encryption provided the efficiently computable function **half** can be used as the oracle:

BREAKING RSA USING THE ORACLE half

Algorithm:

```
for $i = 0$ to $\lceil \lg n \rceil$ do
    $c_i \leftarrow \mathbf{half}_{e_k}(c); c \leftarrow (c \times e_k(2)) \bmod n$
$l \leftarrow 0; u \leftarrow n$
for $i = 0$ to $\lceil \lg n \rceil$ do
    $m \leftarrow (i + u)/2;$
    if $c_i = 1$ then $i \leftarrow m$ else $u \leftarrow m$;
output $\leftarrow [u]$
```

The algorithm does the job. Indeed, in the first cycle

$$c_i = \mathbf{half}_{e_k}(c \times (e_k(2))^i) = \mathbf{half}_{e_k}(e_k(2^i w)),$$

is computed for $0 \leq i \leq \lg n$.

In the second part of the algorithm binary search is used to determine interval in which w lies. For example, we have that

$$\mathbf{half}_{e_k}(e_k(w)) = 0 \equiv w \in [0, \frac{n}{2})$$
$$\mathbf{half}_{e_k}(e_k(2w)) = 0 \equiv w \in [0, \frac{n}{4}) \cup [\frac{n}{2}, \frac{3n}{4})$$
$$\mathbf{half}_{e_k}(e_k(4w)) = 0 \equiv w \in [0, \frac{n}{8}) \cup [\frac{n}{4}, \frac{3n}{8}) \cup [\frac{n}{2}, \frac{5n}{8}) \cup [\frac{3n}{4}, \frac{7n}{8})$$

SECURITY of RSA in PRACTICE II

There are many results for RSA showing that certain parts are as hard as whole. For example, any feasible algorithm to determine the last bit of the plaintext can be converted into a feasible algorithm to determine the whole plaintext.

Example Assume that we have an algorithm H to determine whether a plaintext x for a cryptotext y designed by RSA with the public key e, n is smaller than $\frac{n}{2}$.

We construct an algorithm A to determine in which of the intervals $(\frac{jn}{8}, \frac{(j+1)n}{8}), 0 \leq j \leq 7$ the plaintext lies.

Basic idea: algorithm H will be used to decide whether the plaintexts for cryptotexts $x^e \bmod n, 2^e x^e \bmod n, 4^e x^e \bmod n$ are smaller than $\frac{n}{2}$.

Let us summarize answers all possible outcomes of tests imply:

yes, yes, yes	$0 < x < \frac{n}{8}$	no, yes, yes	$\frac{n}{2} < x < \frac{5n}{8}$
yes, yes, no	$\frac{n}{8} < x < \frac{n}{4}$	no, yes, no	$\frac{5n}{8} < x < \frac{3n}{4}$
yes, no, yes	$\frac{n}{4} < x < \frac{3n}{8}$	no, no, yes	$\frac{3n}{4} < x < \frac{7n}{8}$
yes, no, no	$\frac{3n}{8} < x < \frac{n}{2}$	no, no, no	$\frac{7n}{8} < x < n$

COMMON MODULUS ATTACK

Let a message w be encoded with a modulus n and two encryption exponents e_1 and e_2 such that $\gcd(e_1, e_2) = 1$. Therefore

$$c_1 = w^{e_1} \bmod n, \qquad c_2 = w^{e_2} \bmod n;$$

Then

$$w = c_1^a c_2^b,$$

where, a, b are such that

$$a \cdot e_1 + b \cdot e_2 = 1$$

PRIVATE-KEY versus PUBLIC-KEY CRYPTOGRAPHY

- The prime advantage of public-key cryptography is increased security – the private keys do not ever need to be transmitted or revealed to anyone.
- Public key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure.
- Example RSA and DES (AES) are usually combined as follows
  1. The message is encrypted with a random DES key
  2. DES-key is encrypted with RSA
  3. DES-encrypted message and RSA-encrypted DES-key are sent.

  This protocol is called RSA digital envelope.
- In software (hardware) DES is generally about 100 (1000) times faster than RSA.

  If n users communicate with secrete-key cryptography, they need n (n - 1) / 2 keys.

  If n users communicate with public-key cryptography 2n keys are sufficient.

  Public-key cryptography allows spontaneous communication.

KERBEROS

We describe a very popular key distribution protocol with trusted authority TA with which each user A shares a secret key K_A.

- To communicate with user B the user A asks TA for a session key (K)
- TA chooses a random session key K, a time-stamp T, and a lifetime limit L.
- TA computes

  $$m_1 = e_{K_A}(K, ID(B), T, L); \quad m_2 = e_{K_B}(K, ID(B), T, L);$$

  and sends m_1, m_2 to A.
- A decrypts m_1, recovers $K, T, L, ID(B)$, computes $m_3 = e_K(ID(B), T)$ and sends m_2 and m_3 to B.
- B decrypts m_2 and m_3, checks whether two values of T and of $ID(B)$ are the same. If so, B computes $m_4 = e_K(T+1)$ and sends it to A.
- A decrypts m_4 and verifies that she got $T+1$.

APPENDIX I

APPENDIX I

RSA with a composite "to be a prime"

Let us explore what happens if some integer p used, as a prime, to design a RSA is actually not a prime.

Let $n = pq$ where q be a prime, but $p = p_1p_2$, where p_1, p_2 are primes. In such a case

$$\phi(n) = (p_1 - 1)(p_2 - 1)(q - 1)$$

but assume that the RSA-designer works with $\phi_1(n) = (p - 1)(q - 1)$
Let $u = \text{lcm}(p_-1, p_2 - 1, q - 1)$ and let $\gcd(w, n) = 1$. In such a case

$$w^{p_1-1} \equiv 1 \pmod{p_1}, w^{p_2-1} \equiv 1 \pmod{p_2}, w^{q-1} \equiv 1 \pmod{q}$$

and as a consequence $\quad w^u \equiv 1 \pmod{n}$

In such a case u divides $\phi(n)$ and let us assume that also u divides $\phi_1(n)$ Then

$$w^{\phi_1(n)+1} \equiv w \pmod{n}.$$

So if $e_d \equiv 1 \bmod \phi_1(n)$, then encryption and decryption work as if p were prime.

Example $p = 91 = 7 \cdot 13, q = 41, n = 3731, \phi_1(n) = 3600, \phi(n) = 2880, \text{lcm}(6, 12, 40) = 120, 120 | \phi_1(n)$.
If $\gcd(d, \phi_1(n)) = 1$, then $\gcd(d, \phi(n)) = 1 \Rightarrow$ one can compute e using $\phi_1(n)$. However, if u does not divide $\phi_1(n)$, then the cryptosystem does not work properly.

TWO USERS SHOULD not USE THE SAME MODULUS

Otherwise, users, say A and B, would be able to decrypt messages of each other using the following method.

Decryption: B computes

$$f = \gcd(e_B d_B - 1, e_A), m = \frac{e_B d_B - 1}{f}$$

$$e_B d_B - 1 = k\phi(n) \text{ for some } k$$

It holds:

$$\gcd(e_A, \phi(n)) = 1 \Rightarrow \gcd(f, \phi(n)) = 1$$

and therefore

$$m \text{ is a multiple of } \phi(n).$$

m and e_A have no common divisor and therefore there exist integers u, v such that

$$um + ve_A = 1$$

Since m is a multiple of $\phi(n)$, we have

$$ve_A = 1 - um \equiv 1 \bmod \phi(n)$$

and since $e_A d_A \equiv 1 \bmod \phi(n)$, we have

$$(v - d_A)e_A \equiv 0 \bmod \phi(n)$$

and therefore

$$v \equiv d_A \bmod \phi(n)$$

is a decryption exponent of A. Indeed, for a cryptotext c:

$$c^v \equiv w^{e_A v} \equiv w^{e_A d_A + c\phi(n)} \equiv w \bmod (n)$$

KEY DISTRIBUTION versus KEY AGREEMENT

One should distinguish between key distribution and key agreement

- Key distribution is a mechanism whereby one party chooses a secret key and then transmits it to another party or parties.
- Key agreement is a protocol whereby two (or more) parties jointly establish a secret key by communication over a public channel.

The objective of key distribution or key agreement protocols is that, at the end of the protocols, the two parties involved both have possession of the same key k, and the value of k is not known to any other party (except possibly the TA).

ANOTHER APPLICATION - LAMPORT's ONE-TIME PASSWORDS

One-way functions can be used to create a sequence of passwords:

1. Alice chooses a random w and computes, using a one-way function h, a sequence of passwords
$$w, h(w), h(h(w)), \ldots, h^n(w)$$
2. Alice then transfers securely "the initial secret" $w_0 = h^n(w)$ to Bob.
3. The i-th authentication, $0 < i < n + 1$, is performed as follows:

- - - - - - - Alice sends $w_i = h^{n-i}(w)$ to Bob for I = 1, 2,. . . ,n-1

- - - - - - - Bob checks whether $w_{i-1} = h(w_i)$.

When the number of identifications reaches n, a new w has to be chosen.

SECURITY POTENTIAL of McELIECE CRYPTOSYSTEM

McEliece cryptosystem is one of those cryptosystems that has not been yet shown to be breakable by quantum computers.

McEliece cryptosystem is not practical, because for the recommended security parameters the public key size is 2^{19} bits; and therefore its security was not much scrutinised.

Big problem of cryptography is to find practical public-key cryptosystem that could not be broken even with quantum computers.

Big question? What comes first, powerful quantum computers or practical public-key cryptosystem secure also against quantum computers.

COMPLEXITY of RSA

Let modulus be product of two s-bit primes.

- Setting cryptosystem requires $O(s^4)$ operations.
  - generation of two s bit primes requires $O(s^4)$;
  - multiplication of primes p and q requires $O(s^2)$;
  - finding exponent e requires one GCD-computation - $O(s^2)$;
  - finding exponent d requires computation of generalized GCD - $O(s^2)$.
- Encryption requires one exponentiation - $O(s^3)$;
- Decryption requires one exponentiation - $O(s^3)$.

IMPLICATIONS of HARD INVERTABILITY of RSA

- Under the assumption that RSA is hard to invert we can design:
  - cryptographically perfect pseudorandom generators;
  - zero-knowledge proofs for any **NP** statement;
  - multiparty protocols for computing securely any multi-variant function
- The fact that RSA is hard to invert does not imply that RSA is secure cryptosystem.

HISTORY of RSA

- Diffie published his idea of asymmetric cryptosystem in summer 1975, though he had no example of such a cryptosystem.
- The problem was to find a one-way function with a backdoor.
- Rivest, Shamir and Adleman, from MIT, started to work on this problem in 1976.
- Rivest and Shamir spent a year coming up with new ideas and Adleman spent a year shooting them down.
- In April 1977 they spent a holiday evening drinking quite a bit of wine.
- At night Rivest could not sleep, mediated and all of sudden got an idea. In the morning the paper about

HISTORY of RSA REVISITED

- Around 1960 British militaries started to worry about the key distribution problem.
- At the beginning of 1969 James Ellis from secrete Government Communications Headquarters (GCHQ) was asked to look into the problem.
- By the end of 1969 Ellis discovered the basic idea of public key cryptography.
- For next three years best minds of GCHQ tried to unsuccessfully find a suitable trapdoor function necessary for useful public-key cryptography.
- In September 1973 a new member of the team, Clifford Cocks, who graduated in number theory from Cambridge, was told about problem and solved it in few

BREAKING RSA using PERIOD FINDING ALGORITHM I

We show that if Eve has an efficient algorithm to determine the period of functions $f(x) = b^x \bmod n$ in the group $\mathbf{Z}_n^*$, where $n = pq$ for primes p, q, that is to find order of elements in $\mathbf{Z}_n^*$, then she can break RSA cryptosystem with modulus n.

Let us have RSA cryptosystem with modulus n, a public encryption exponent e and secret decryption exponent d.

If Eve gets the cryptotext $c = w^e$ of an unknown plaintext w, she will use her order finding algorithm to determine the order r of c in $\mathbf{Z}_n^*$.

Order r of c in $\mathbf{Z}_n^*$ is the same as the order of w.

BREAKING RSA using PERIOD FINDING ALGORITHM II

Indeed, the subgroup of $\mathbf{Z}_n^*$ generated by w contains clearly c and therefore it contains subgroup generated by c. However, the subgroup generated by c contains $c^d = w$ and therefore contains subgroup generated by w. Since each subgroup contains the other they have to be the same.

Therefore if Eve can find the order of the known cryptotext c she will have also the order of unknown plaintext w.

Since e was chosen to have no factor with $\phi(n) = (p-1)(q-1)$ and since the order r has to divide the order $\phi(n) = (p-1)(q-1)$ of $\mathbf{Z}_n^*$, the encoding exponent e can have no factor in common with r.

This means that e is congruent modulo r to a member e' of $\mathbf{Z}_r$, which has an inverse d' in $\mathbf{Z}_r$ and d' is also a modulo-r inverse of e, that is

$$ed' \equiv 1 (\bmod r)$$

Therefore, given a cryptotext c and publicly known modulus n, Eve can calculate easily, using the extended Euclid algorithm, d' on a classical computer.

This implies, that for some integer m it holds

$$c^{d'} \equiv w^{ed'} \equiv w^{1+mr} \equiv w(w^r)^m \equiv w (\bmod n)$$

and therefore an efficient period finding algorithm allows Eve to find the plaintext w