6. RSA CRYPTOSYSTEM

The most important public-key cryptosystem is the RSA cryptosystem at 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 RSA cryptosystem and various other problems related to security of the RSA cryptosystems.

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

Several factorization methods will be presented and discuss. In doing that we will illustrate modern distributed techniques to factorize very large integers.

DESIGN and USE of RSA CRYPTOSYSTEM

Invented: 1978 – Rivest, Shamir, Adleman

Basic idea: multiplication of primes is very easy, factorization of integerss is unfeasible (on classical computers).

Design of RSA cryptosystems

1. Choose two large (512 – 1024 bits) primes p, q and denote

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

2. Choose a large d

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

and compute

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

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

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.

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 $e = 23$, we get $d = 2087$
- By choosing $e = 29$, we get $d = 2069$
- By choosing other values of e we get other values of d.

Let us choose the first pair of encryption/decryption exponents.
Plaintext: KARLSRUHE

Encoding: 100017111817200704
Since $10^3 < n < 10^4$, the numerical plaintext is divided into blocks of 3 digits $\Rightarrow$ 6 plaintext integers are obtained

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

Encryption:

$$100^{23} \bmod 2501, 17^{23} \bmod 2501, 111^{23} \bmod 2501$$

$$817^{23} \bmod 2501, 200^{23} \bmod 2501, 704^{23} \bmod 2501$$

provides cryptotexts:

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

Decryption:

$$2306^{2087} \bmod 2501 = 100, 1893^{2087} \bmod 2501 = 17$$

$$621^{2087} \bmod 2501 = 111, 1380^{2087} \bmod 2501 = 817$$

$$490^{2087} \bmod 2501 = 200, 313^{2087} \bmod 2501 = 704$$

Correctness of RSA

Theorem Let $c = w^e \bmod n$ be the cryptotext obtained by encrypting a plaintext w, by the RSA-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 $ed \equiv 1(\bmod\, \phi(n))$, there exist a $j \in \mathbb{N}$ 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 Euler's Totien Theorem we get that

  $$c^d = w^{ed} = w^{j\phi(n)+1} \equiv w(\bmod\, n)$$

- Case 2. Exactly one of primes p, q divides w – say p.
  In such a case $w^{ed} \equiv w(\bmod\, p)$ and by Fermat's Little theorem $w^{q-1} \equiv 1(\bmod\, q)$

  $$\begin{aligned} \Rightarrow w^{q-1} \equiv 1(\bmod\, q) \quad &\Rightarrow \quad w^{\phi(n)} \equiv 1(\bmod\, q) \\ &\Rightarrow \quad w^{j\phi(n)} \equiv 1(\bmod\, q) \\ &\Rightarrow \quad w^{ed-1} \equiv 1(\bmod\, q) \\ &\Rightarrow \quad w^{ed} \equiv w(\bmod\, q) \end{aligned}$$

  Since $w^{ed} \equiv w \pmod{p}$ and also $w^{ed} \equiv w \pmod{q}$ we get

  $$w \equiv w^{ed} \equiv c^d \pmod{pq}$$

- Case 3. Both p, q divide w.
  This cannot happen because, by our assumption, $w < n$.

RSA challenge

One of the first description of RSA 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
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

How to design a good RSA cryptosystem

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 $p+2, p+4, \ldots$

From the Prime Number Theorem if 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 not too large.
2.2 $\gcd(p-1, q-1)$ should not be large.
2.3 Both $p-1$ and $q-1$ should contain large prime factors.
2.4 Quite ideal case: q, p should be safe primes – such that also $\frac{p-1}{2}$ and $\frac{q-1}{2}$ 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 $n^{1/4}$. (For $d < n^{1/4}$ a polynomial time algorithm is known to determine d.

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 such x is found that x^2-n is a square, say y^2. In such a case

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

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 $w^{ks+}\equiv w \pmod p$ and $w^{ks+1}\equiv w \pmod q$. 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}.

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 needed to break RSA.

3. If a method to 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 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 and also encryption exponents $e_1, e_2, \dots$ 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.

Security of RSA

None of the numerous attempts to develop attacks on RSA 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 produces by the RSA cryptosystem.

We show that were the following functions *parity* and *half* (computationally equivalent in polynomial time), be efficiently computable for an RSA cryptosystem with the encryption (decryption) algorithm e_k (d_k), then this RSA would be breakable.

$$\mathit{parity}_{e_k}(c) = \text{the least significant bit of such an } w \text{ that } e_k(w) = c;$$

$$\mathit{half}_{e_k}(c) = 0 \text{ if } 0 \leq w < \frac{n}{2} \text{ and } \mathit{half}(c) = 1 \text{ if } \frac{n}{2} \leq w \leq n-1.$$

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

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

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

   and from the multiplicative rule $e_k(w_1)e_k(w_2) = e_k(w_1w_2)$.

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

BREAKING RSA USING AN ORACLE

Algorithm:

for $i = 0$ **to** $\lfloor \lg n \rfloor$ **do**
$c_i \leftarrow \mathit{half}(c); c \leftarrow (c \times e_k(2)) \bmod n$
$l \leftarrow 0; u \leftarrow n$
for $i = 0$ **to** $\lfloor \lg n \rfloor$ **do**
$m \leftarrow (l + u)/2;$
if $c_i = 1$ **then** $l \leftarrow m$ **else** $u \leftarrow m;$
$w \leftarrow \lceil u \rceil$

Indeed, in the first cycle

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

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

$$\mathit{half}(e_k(w)) = 0 \equiv w \in \left[0, \tfrac{n}{2}\right)$$
$$\mathit{half}(e_k(w)) = 1 \equiv w \in \left[\tfrac{n}{2}, 1\right)$$
$$\mathit{half}(e_k(2w)) = 0 \equiv w \in \left[0, \tfrac{n}{4}\right) \cup \left[\tfrac{n}{2}, \tfrac{3n}{4}\right)$$
$$\mathit{half}(e_k(2w)) = 1 \equiv w \in \left[\tfrac{n}{4}, \tfrac{n}{2}\right) \cup \left[\tfrac{3n}{4}, n\right)$$
$$\mathit{half}(e_k(4w)) = 0 \equiv w \in \left[0, \tfrac{n}{8}\right) \cup \left[\tfrac{2n}{8}, \tfrac{3n}{8}\right) \cup \left[\tfrac{4n}{8}, \tfrac{5n}{8}\right) \cup \left[\tfrac{6n}{8}, \tfrac{7n}{8}\right)$$

....

RSA with a composite “to be a prime”

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

Let w be a plaintext and $n = pq$ where q is 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 $u = \mathrm{lcm}(p_1 - 1, p_2 - 1, q - 1)$ and let $\gcd(w, n) = 1$.

In such a case

$$w^{p_1-1} \equiv 1 (\mathrm{mod}\, p_1), w^{p_2-1} \equiv 1 (\mathrm{mod}\, p_2), w^{q-1} \equiv 1 (\mathrm{mod}\, q)$$

and as a consequence

$$w^u \equiv 1 (\mathrm{mod}\, n).$$

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

$$w^{\phi_1(n)+1} \equiv w (\mathrm{mod}\, n).$$

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

Example $p = 91 = 7 \cdot 13, q = 41, n = 3731, \phi_1(n) = 3600, \phi(n) = = 2880, \mathrm{lcm}(6, 12, 40) = 120, 120 | \phi_1(n)$.

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

If two users, say A and B, use the same modulus for RSA, then they are able to decrypt messages of each other using the following method.

Decryption: B computes

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

. Since

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

, it holds

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

and therefore

$$m = \tfrac{k\phi(n)}{f} \text{ is a multiple of } \phi(n).$$

m and e_A have no common divisor (otherwise $F \neq \gcd(e_B d_B - 1, e_A)$) 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:

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

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 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 $\frac{n(n-1)}{2}$ keys. In the case they use public key cryptography $2n$ keys are sufficient.

Public-key cryptography allows spontaneous communication.

Prime recognition and factorization

The key problems for the development of RSA cryptosystem are that of prime recognition and integer factorization.

In August 2002 first polynomial time algorithm has been 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.

Rabin-Miller's prime recognition

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

Lemma Let $n \in N$. Denote, for $1 \leq x \leq n$, by $C(x)$ the condition:

Either $x^{n-1} \not\equiv \pmod{n}$, or thre 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 whther $C(x_i)$ holds.

Claim: If $C(x_i)$ holds for some i, then n is not a prime for sure. Otherwise n is prime, with probability of error 2^{-n}.

Fermat numbers factorization

Factorization of so-called Fermat numbers $2^{2^i}+1$ is a good example to illustrate progress that has been made in the area of factorization.
Pierre de Fermat (1601-65) expected that all numbers

$$F_i = 2^{2^i} + 1 \quad i \geq 1$$

are primes.

This is true for $i = 1, \ldots, 4$. $F_1 = 5, F_2 = 17, F_3 = 257, F_4 = 65537$.

1732 L. Euler found that $F_5 = 4294967297 = 641 \cdot 6700417$
1880 Landry+LeLasser found that

$$F_6 = 18446744073709551617 = 274177 \times 67280421310721$$

1970 Morrison+Brillhart found factorization for F_7 (39 digits)

$$F_7 = 340282366920938463463374607431768211457 =$$

$$= 5704689200685129054721 \times 59649589127497217$$

1980 Brent+Pollard found factorization for F_8
1990 A. K. Lenstra+... found factorization for F_9 (155 digits)

Simple Factorization methods

Two simple method to find factors of an integer n will be shown.

1. **Trial division** Divide n by all odd integers (primes) smaller than $\sqrt{n}$.
2. Pollard's $p-1$ – algorithm – to factor n given a bound b.

   $a \leftarrow 2;$

   for $j = 2$ **to** b **do** $a \leftarrow a^j \bmod n;$

   $f \leftarrow gcd(a-1, n)$

   if $1 < f < n$ **then** f is a factor **otherwise** failure

   Proof of correctness: Let p be a prime divisor of n and $q \leq b$ for every prime power $q|(p-1)$. Then $(p-1)|b!$.

   At the end of the for-loop we have

   $$a \equiv 2^{b!} \pmod{n}$$

   what implies that

   $$a \equiv 2^{b!} \pmod{p}.$$

   By Fermat theorem $2^{p-1} \equiv 1 \pmod{p}$ and since $(p-1)|b!$ we have $1 \equiv 2^{b!} \pmod{n}$. Hence $a \equiv 1 \pmod{p}$. Therefore,

   $$p|(a-1)$$

   and

   $$p|d = gcd(a-1, n).$$

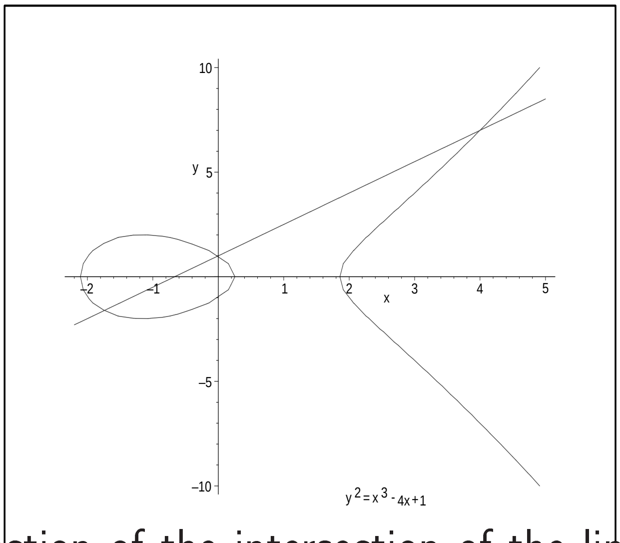
Elliptic curve method for integer factorization

Basic idea: To factorize an integer n one keeps choosing randomly elliptic curves and performs certain computations that include $\gcd(x, n)$ evaluations, for various x, and that these computations continue only if all $\gcd(x, n) = 1$. If n is a prime, these evaluations are always successful. However if n is composite, some of the evaluated $\gcd(x, n)$ will be different from 1, providing a factor of n.

Indeed, H. W. Lenstra has shown that if n is not a prime, then there is an elliptic curve such that the above computations provide a factor of n.

An elliptic curve is a set of points (x, y) of an equation

$$y^2 = x^3 + ax + b \text{ where } 4a^3 + 27b^2 \neq 0.$$



A crucial idea is that one can define addition of two points of an elliptic curve and an "inverse element" of a point of the curve, in such a way that one gets an additive group of points, with a special "null" point $\mathbf{0}$ in the infinity.

Sum of two points P_1, P_2 on the curve that do not have the same x coordinate is defined as the x-axis reflection of the intersection of the line through these points with the curve. .

Formulas for operations on points

If $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$, then

$-P_1 = (x_1, -y_1)$

$$P_1 + P_2 = \begin{cases} 0, & \text{if } P_1 = -P_2; \\ P_2, & \text{if } P_1 = \mathbf{0}; \\ P_1, & \text{if } P_2 = \mathbf{0}. \end{cases}$$

Otherwise

$$P_1 + P_2 = (x_3, y_3) \qquad \begin{aligned} x_3 &= -x_1 - x_2 + \lambda^2; \\ y_3 &= -y_1 + \lambda(x_1 - x_3); \\ \lambda &= \begin{cases} \frac{y_1 - y_2}{x_1 - x_2}, & \text{if } P_1 \neq P_2; \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P_1 = P_2. \end{cases} \end{aligned}$$

New key idea: All points and operations are taken modulo an integer p. $\{$In this case it has to hold that $4a^3 + 27b^2 \not\equiv 0 \bmod p.\}$

Example $p = 11, a = 1, b = 6 \quad (y^2 = x^3 + x + 6)$,

$P_1 = (2, 7), P_1 + P_1 = (5, 2),\ 3P_1 = (P_1 + P_1) + P_1 = (8, 3)$.

According to the Lagrange Theorem, to every point P there is a $k \leq p$ such that

$$\underbrace{P + P + \ldots + P}_{k} = \mathbf{0}.$$

EXAMPLE

An example to see how one can use elliptic curves to factor an integer.

Let $n = 35$.
Choose an elliptic curve: i.e. $y^2 \equiv x^3 + x - 1$
Choose a point: $P = (1, 1)$
Compute $9P$: $2P = (2, 2)$

$4P = (0, 22)$

$8P = (16, 19)$

$\gcd(15, 35) = 5$ is a factor of $n = 35$.

In order to compute $P + 8P$ one has to compute $15^{-1} \bmod 35$ and in order to do that we need first to compute $\gcd(15, 35) \neq 1$.

Now we can formulate the basic idea of factorization using elliptic curve method.

Generate many elliptic curves, choose many points P on them and for sufficiently large integer k compute kP.

In realizing the above strategy, what can be done in a very distributed way, provided a root generates and distributed elliptic curves and points, one often needs to compute gcd(x, n) for various x. If at least once such a gcd$(x, n) \neq 1$ we have a factor of n.

Problem: How to choose k?

Idea: If one searches for m-digit factors, one chooses k in such a way that k is a multiple of as many of m-digit numbers as possible which do not have too large prime factors. In such a case one has a good chance that k is a multiple of the number of elements of the group of points of elliptic curves modulo n.

Method: One chooses an integer B and takes as k the product of all maximal powers of primes smaller than B.

Example In order to find a 6-digit factor one chooses $B = 147$ and $k = 2^7 \cdot 3^4 \cdot 5^3 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot \ldots \cdot 139$.

Factoring using elliptic curves I

Let us continue to discuss the following problem:

Problem: How to choose k?

Idea: If one searches for m-digit factors, one chooses k in such a way that k is a multiple of as many of m-digit numbers as possible which do not have too large prime factors. In such a case one has a good chance that k is a multiple of the number of elements of the group of points of elliptic curves modulo n.

Method: One chooses an integer B and takes as k the product of all maximal powers of primes smaller than B.

Example In order to find a 6-digit factor one chooses $B = 147$ and $k = 2^7 \cdot 3^4 \cdot 5^3 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot \ldots \cdot 139$.

The following table shows B and the number of elliptic curves one has to test:

number of digits of to-be-factors	6	9	12	18	24	30
B	147	682	2462	23462	162730	945922
number of curves	10	24	55	231	833	2594

Computation time by the eliptic curves method depends on the size of factors.

Method of quadratic sieve to factorize n

Basic idea: One finds x, y such that $n|(x^2 - y^2)$

Reasoning: If n divides $(x+y)(x-y)$ and n does not divide neither $x + y$ nor $x - y$, then one factor of n has to divide $x + y$ and another one $x - y$.

Example $n = 7429 = 227^2 - 210^2, X = 227, Y = 210$

$x - y = 17 \quad x + y = 437$

$\gcd(17, 7429) = 17 \quad \gcd(437, 7429) = 437.$

How to find X and Y? One forms a system of (modular) linear equations and determines X and Y from the solutions of the system.

number of digits of n	50	60	70	80	90	100	110	120
number of equations	3000	4000	7400	15000	30000	51000	120000	245000

Step 1 One finds numbers x such that $x^2 - n$ is small and has small factors.

Example

$$\left.\begin{array}{l} 83^2 - 7429 = -540 = (-1) \cdot 2^2 \cdot 3^3 \cdot 5 \\ 87^2 - 7429 = \ 140 \ = \qquad\quad 2^2 \cdot 5 \cdot 7 \\ 88^2 - 7429 = \ 315 \ = \qquad\quad 3^2 \cdot 5 \cdot 7 \end{array}\right\} \text{relations}$$

Step 2 One multiplies some of the relations if their product is a square. For example

$$(87^2 - 7429)(88^2 - 7429) = 2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 = 210^2$$

Now

$$\begin{aligned} (87 \cdot 88)^2 &\equiv (87^2 - 7429)(88^2 - 7429) \bmod 7429 \\ 227^2 &\equiv 210^2 \bmod 7429 \end{aligned}$$

Hence 7429 divides $227^2 - 210^2$.

Formation of equations: For the i-th relation one takes a variable λ_i and forms the expression

$$((-1) \cdot 2^2 \cdot 3^3 \cdot 5)^{\lambda_1} \cdot (2^2 \cdot 5 \cdot 7)^{\lambda_2} \cdot (3^2 \cdot 5 \cdot 7)^{\lambda_3} =$$

$$= (-1)^{\lambda_1} \cdot 2^{2\lambda_1 + 2\lambda_2} \cdot 3^{2\lambda_1 + 2\lambda_2} \cdot 5^{\lambda_1 + \lambda_2 + \lambda_3} \cdot 7^{\lambda_2 + \lambda_3}$$

If this is to form a quadrat the following equations have to hold

$$\begin{array}{lllll} \lambda_1 & & & \equiv 0 \pmod 2 \\ \lambda_1 & + \lambda_2 & + & \lambda_3 \equiv 0 \pmod 2 \\ & \ \ \lambda_2 & + & \lambda_3 \equiv 0 \pmod 2 \end{array}$$

$$\lambda_1 = 0, \lambda_2 = \lambda_3 = 1.$$

Problem How to find relations?
Using the algorithm called Quadratic sieve method.

Step 1 One chooses a set of primes that can be factors – a so-called factor basis.

One chooses an m such that $m^2 - n$ is small and considers numbers $(m+u)^2 - n$ for $-k \le u \le k$ for small k.

One then tries to factor all $(m+u)^2 - n$ with primes from the factor basis, from the smallest to the largest.

u	-3	-3	-3	0	1	2	3
$(m+u)^2 - n$	-540	-373	-204	-33	140	315	492
Sieve with 2	-135		-51		35		123
Sieve with 3	-5		-17	-11		35	41
Sieve with 5	-1				7	7	
Sieve with 7					1	1	

In order to factor a 129-digit number from the RSA challenge they used

8 424 486 relations
569 466 equations
544 939 elements in the factor base

The rho method of integer factorization

Basic idea 1. Choose an easy to compute $f : \mathbb{Z}_n \to \mathbb{Z}_n$ and $x_0 \in \mathbb{Z}_n$.

Example $f(x) = x^2 + 1$

2. Keep computing $x_{j+1} = f(x_j) \quad j = 0, 1, 2, \ldots$
and $\gcd(x_j - x_k, n) \quad k \leq j$.

(Observe that iff $x_j \equiv x_k \bmod r$ for a prime factor r of n, then $\gcd(x_j - x_k, n) \geq r$.)

Example $n = 91, f(x) = x^2 + 1, x_0 = 1, x_1 = 2, x_2 = 5, x_3 = 26$
$\gcd(x_3 - x_2, n) = \gcd(26 - 5, 91) = 7$

Remark: In the rho method it is important to choose f in such a way that f maps $\mathbb{Z}_n$ into $\mathbb{Z}_n$ in a "random" way.

Basic question: How good is the rho method?
(How long we expect to have to wait before we get two values x_j, x_k such that $\gcd(x_j - x_k, n) \neq 1$ if n is not a prime?)

Basic lemma

Given: $n, f : \mathbb{Z}_n \to \mathbb{Z}_n, x_0 \in \mathbb{Z}_n$
We ask how many iterations are needed to get $x_j \equiv x_k \bmod r$ where r is a prime factor of n.

Lemma Let S be a set, $r = |S|$. Given a map $f : S \to S, x_0 \in S$, let $x_{j+1} = f(x_j), j \geq 0$. Let $\lambda > 0, l = 1 + \left\lceil \sqrt{2\lambda r} \right\rceil$. Then the proportion of pairs (f, x_0) for which $x_0, x_1, \ldots, x_l$ are distinct, where f runs over all mappings from S to S and x_0 over all S, is less than $e^{-\lambda}$.

Proof Number of pairs (x_0, f) is r^{r+1}.
How many pairs (x_0, f) are there for which $x_0, \ldots, x_l$ are distinct?

r choices for x_0, $r-1$ for x_1, $r-2$ for x_2, $\ldots$.

The values of f for each of the remaining $r - l$ values are arbitrary – there are r^{r-l} possibilities for those values.

Total number of ways of choosing x_0 and f such that $x_0, \ldots, x_l$ are different is

$$r^{r-l} \prod_{j=0}^{l} (r-j)$$

and the proportion of pairs with such a property is $r^{-l-1} \prod_{j=0}^{l} (r-j)$.
For $l = 1 + \left\lceil \sqrt{2} l r \right\rceil$ we have

$$\begin{aligned}
\ln\left(r^{-l-1} \prod_{j=0}^{l} (r-j)\right) = \ln\left(\prod_{j=0}^{l}\left(1 - \frac{j}{r}\right)\right) &\leq \sum_{j=1}^{l} -\frac{j}{r} = -\frac{l(l+1)}{2r} \\
&< -\frac{l^2}{2r} < -\frac{(\sqrt{2\lambda r})^2}{2r} = -\lambda.
\end{aligned}$$

A simplification of the basic idea:

For each k compute $\gcd(x_k - x_j, n)$ for just one $j < k$.

RHO-ALGORITHM]

Choose $f : \mathbb{Z}_n \to \mathbb{Z}_n, x_0$, compute $x_k = f(x_{k-1}), k > 0$.

If k is an $(h+1)$-bit integer, i.e. $2^h \leq k \leq 2^{h+1}$, then compute $\gcd(x_k, x_{2^h-1})$.

Example $\quad n = 4087, f(x) = x^2 + x + 1, x_0 = 2$

$x_1 = f(2) = 7,\qquad \gcd(x_1 - x_0, n) = 1$

$x_2 = f(7) = 57,\qquad \gcd(x_2 - x_1, n) = \gcd(57 - 7, n) = 1$

$x_3 = f(57) = 3307,\qquad \gcd(x_3 - x_1, n) = \gcd(3307 - 7, n) = 1$

$x_4 = f(3307) = 2745,\qquad \gcd(x_4 - x_3, n) = \gcd(2745 - 3307, n) = 1$

$x_5 = f(2746) = 1343,\qquad \gcd(x_5 - x_3, n) = \gcd(1343 - 3307, n) = 1$

$x_6 = f(1343) = 2626,\qquad \gcd(x_6 - x_3, n) = \gcd(2626 - 3307, n) = 1$

$x_7 = f(2626) = 3734,\qquad \gcd(x_7 - x_3, n) = \gcd(3734 - 3307, n) = 61$

Disadvantage We likely will not detect the first case such that for some k_0 there is a $j_0 < k_0$ such that $\gcd(x_{k_0} - x_{j_0}, n) > 1$.

This is no real problem! Let k_0 has $h + 1$ bits. Set $j = 2^{h+1} - 1$, $k =$
$= j + k_0 - j_0$. k has $(h + 2)$ bits, $\gcd(x_k - x_j, n) > 1$

$$k < 2^{h+2} = 4 \cdot 2^h \leq 4k_0.$$

Theorem Let n be odd + composite and $1 < r < \sqrt{n}$ its factor. If f, x_0 are chosen randomly, then rho algorithm reveals r in $\mathcal{O}(\sqrt[4]{n}\log^3 n)$ bit operations with high probability. More precisely, there is a constant $C > 0$ such that for any $\lambda > 0$, the probability that the rho algorithm fails to find a nontrivial factor of n in $C\sqrt{\lambda}\sqrt[4]{n}\log^3 n$ bit operations is less than $e^{-\lambda}$.

Proof Let C_1 be a constant such that $\gcd(y-z, n)$ can be computed in $C_1 \log^3 n$ bit operations whenever $y, z < n$.

Let C_2 be a constant such that $f(x) \bmod n$ can be computed in $C_2 \log^2 n$ bit operations if $x < n$.

If k_0 is the first index for which there exists $j_0 < k_0$ with $x_{k_0} \equiv x_{j_0} \bmod r$, then the rho-algorithm finds r in $k \leq 4k_0$ steps.
The total number of bit operations is bounded by

$$4k_0(C_1 \log^3 n + C_2 \log^2 n)$$

By Lemma the probability that k_0 is greater than $1 + \sqrt{2\lambda r}$ is less than $e^{-\lambda}$.

If $k_0 \leq 1 + \sqrt{2\lambda r}$, then the number of bits operations needed to find r is bounded by

$$4(1+\sqrt{2\lambda r})(C_1 \log^3 n - C_2 \log^2 n) < 4(1+\sqrt{2\lambda}\sqrt[4]{n})(C_1 \log^3 n + C_2 \log^2 n)$$

If we choose $C > 4\sqrt{2}(C_1 + C_2)$, then we have that r will be found in $C\sqrt{\lambda}\sqrt[4]{n}\log^3 n$ bit operations – unless we made uniformed choice of (f, x_0) the probability of what is at most $e^{-\lambda}$.

Simple factorization strategy to factor an integer n

1.For $i = 3, 5, \ldots$ till $\lceil 10 \log n \rceil$ check whether $i|n$.
If such an i is found we have a factor. Otherwise:

2.Fermat test:
Verify whether $2^{n-1} \equiv 1 \bmod n$.
If yes, n is probably prime. To confirm it use Lucas test.

3.Lucas test:
Lucas sequence: $U_0 = 0, U_1 = 1, U_{i+1} = U_i - qU_{i-1}, i \geq 1$.

Lucas theorem: If n is prime, $n > q, (1 - 4q|n) = -1$, then $n|U_{n+1}$.

Test: Find the smallest D such that $(D|n) = -1$, put $D = 1 - 4q$, check whether $U_{n+1} \equiv 0 \bmod n$. If not, n is composite. Otherwise n is prime with large probability.

Remark No composite integer is known that would satisfy both Fermat and Lucas tests. (A proof of this fact exists for $n < 25 \cdot 10^9$.)

Homework: Factorize: 7500596246954111183.

Computation of U_{n+1}

$$V_0 = 2 \qquad U_0 = 2$$
$$U_{2t} = U_t V_t \qquad V_{2t} = V_t^2 - 2g^t$$
$$U_{2t+1} = \frac{U_{2t} + V_{2t}}{2} \qquad V_{2t+1} = \frac{DU_{2t} + V_{2t}}{2}$$

Homework

1. Factor $2^{77} - 3$.

2. Factor $2^{79} - 3$.

Factorization of a 512-bit number

On August 22, 1999, a team of scientifists 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 financinal 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, he estimated that, using knowledge of that time, factorization of RSA-129 would require 10^{16} years.

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:
googol – 10^{100} golplex $10^{10^{100}}$

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

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.

EXERCISES 6

1. For the RSA cryptosystem with $p = 101$, $q = 113$, $e = 3533$ decrypt $c = 5761$.
2. Taking small primes and large blocks can lead¡to a confusion at RSA. Indeed, if $p = 17$, $q = 23$ and $e = 29$, $d = 85$, then the plaintets 100, 017, 111, 817, 200 and 704 are encrypted as 104, 204, 314, 154, 064 and 295. The decryption then provides the plaintexts 100, 017, 111, 035, 200, 313. Where is the problem?
3. Consider RSA with $p = 47$, $q = 71$, $e = 79$. (a) Compute d; (b) Encrypt the plaintext "The truth is more certain than probable"; (c) Decrypt 3301, 1393, 2120, 1789, 1701, 2639, 895, 1150, 742, 1633, 1572, 1550, 2668, 2375, 1643, 108.
4. Suppose the cryptoanalyst has learned that $n = 84773093$, and $\phi(n) = 84754668$. Find factors of n.
5. Compute Legendre-Jacobi symbols: (a) $(6278|9975)$; (b) $(7411|9283)$; $(m^2|m)$, where m is your "rodné čıslo" – the first part of it.
6. Factorize integers: (a) 262063; (b) 9420457; (c) 1577070841; (d) $2^{77}-3$; (e) $2^{79} - 3$.