Part VIII

Elliptic curves cryptography and factorization

ELLIPTIC CURVE CRYPTOGRAPHY and FACTORIZATION

Cryptography based on manipulation of points of so called **elliptic curves** is currently getting momentum and has a tendency to replace public key cryptography based on the infeasibility of factorization of integers, or on infeasibility of the computation of discrete logarithms.

For example, the US-government has recommended to its governmental institutions to use mainly **elliptic curve cryptography - ECC.**

The main advantage of elliptic curves cryptography is that to achieve a certain level of security shorter keys are sufficient than in case of "usual cryptography". Using shorter keys can result in a considerable savings in hardware implementations.

The second advantage of the elliptic curves cryptography is that quite a few of attacks developed for cryptography based on factorization and discrete logarithm do not work for the elliptic curves cryptography.

It is amazing how practical is the elliptic curve cryptography that is based on very strangely looking theoretical concepts.

ELLIPTIC CURVES

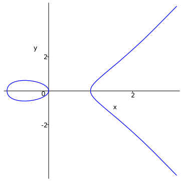
An elliptic curve E is the graph of points of the plane curve defined by the Weierstrass -equation

$$E : y^2 = x^3 + ax + b$$

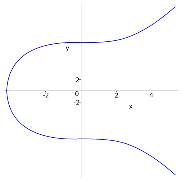
(where a, b are either rational numbers or integers (and computation is done modulo some integer n)) extended by a "point at infinity", denoted usually as ∞ (or 0) that can be regarded as being, at the same time, at the very top and very bottom of the y-axis.

We will consider mainly only those elliptic curves that have no multiple roots - which is equivalent to the condition $4a^3 + 27b^2 \neq 0$.

In case coefficients and x, y can be any rational numbers, a graph of an elliptic curve has one of the forms shown in the following figure. The graph depends on whether the polynomial $x^3 + ax + b$ has three or only one real root.



$y^2 = x(x+1)(x-1)$



$y^2 = x^3 + 73$

MORE PRECISE DEFINITION

A more precise definition of elliptic curves requires that it is the curve of points of the equation

$$E : y^2 = x^3 + ax + b$$

in the case the curve is non-singular.

Geometrically, this means that the graph has no cusps, self-interactions, or isolated points.

Algebraically a curve is non-singular if and only if the discriminant

$$\Delta = -16(4a^3 + 27b^3 0r) \neq 0$$

The graph of a non-singular curve has two components if its discriminant is positive, and one component if it is negative.

HISTORICAL REMARKS on ELLIPTIC CURVES

Elliptic curves are not ellipses and therefore it seems strange that they have such a name. Elliptic curves actually received their names from their relation to so called elliptic integrals

$$\int_{x1}^{x2} \frac{dx}{\sqrt{x^3 + ax + b}} \qquad \int_{x1}^{x2} \frac{xdx}{\sqrt{x^3 + ax + b}}$$

that arise in the computation of the arc-length of ellipses.

It may also seem puzzling why to consider curves given by equations

$$E : y^2 = x^3 + ax + b$$

and not curves given by more general equations

$$y^2 + cxy + dy = x^3 + ex^2 + ax + b$$

The reason is that if we are working with rational coefficients or mod p, where $p > 3$ is a prime, then such a general equation can be transformed to our special case of equation. In other cases, it may be indeed necessary to consider the most general form of equation.

ELLIPTIC CURVES - GENERALITY

A general elliptic curve over Z_{p^m} where p is a prime is the set of points (x, y) satisfying so-called Weierstrass equation

$$y^2 + uxy + vy = x^3 + ax^2 + bx + c$$

for some constants u, v, a, b, c together with a single element $\mathbf{0}$, called the point of infinity.

If $p \neq 2$ Weierstrass equation can be simplified by transformation

$$y \to \frac{y - (ux + v)}{2}$$

to get the equation

$$y^2 = x^3 + dx^2 + ex + f$$

for some constants d, e, f and if $p \neq 3$ by transformation

$$x \to x - \frac{d}{3}$$

to get equation

$$y^2 = x^3 + fx + g$$

IMPORTANCE of ELLIPTIC CURVES

- Elliptic curves are currently an important area of mathematical research with importance for many other areas.
- Recently, in 1995, elliptic curves played an important role in proving, by Andrew Wiles, Fermat's Last Theorem (formulated in 1635) , what could be considered as one of the most important mathematical achievements of the last 50 years.
- Elliptic curves have also close relation to BSD Conjecture (Birch and Swinnerton-Dyer Conjecture), one of the Millennium problems of the Clay Mathematics institute.
- **Elliptic curves are currently behind practically most preferred methods of cryptographic security.**
- Elliptic curves are a basis of very important factorization method.

ADDITION of POINTS on ELLIPTIC CURVES - GEOMETRY

Geometry

On any elliptic curve we can define addition of points in such a way that points of the corresponding curve with such an operation of addition form an Abelian group. in which the point in infinite, denoted by ∞, is plying the role of the identity element

If the line through two different points P_1 and P_2 of an elliptic curve E intersects E in a point $Q = (x, y)$, then we define $P_1 + P_2 = P_3 = (x, -y)$.(This also implies that for any point P on E it holds $P + \infty = P$.) ∞ therefore indeed play a role of the null element of the group

If the line through two different points P_1 and P_2 is parallel with y-axis, then we define $P_1 + P_2 = \infty$.

In case $P_1 = P_2$, and the tangent to E in P_1 intersects E in a point $Q = (x, y)$, then we define $P_1 + P_1 = (x, -y)$.

It should now be obvious how to define subtraction of two points of an elliptic curve.

It is now easy to verify that the above addition of points forms Abelian group with ∞ as the identity (null) element.

ADDITION of POINTS on ELLIPTIC CURVES - FORMULAS)

Formulas

Addition of points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ of an elliptic curve $E : y^2 = x^3 + ax + b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

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

All that holds for the case that $\lambda \neq \infty$; otherwise $P_3 = \infty$.

Example For curve $y^2 = x^3 + 73$ and $P_1 = (2, 9)$, $P_2 = (3, 10)$ we have $\lambda = 1$, $P_1 + P_2 = P_3 = (-4, -3)$ and $P_3 + P_3 = (72, 611)$. – $\{\lambda = -8\}$

ELLIPTIC CURVES mod n

The points on an elliptic curve

$$E : y^2 = x^3 + ax + b \pmod{n}$$

are such pairs (x,y) mod n that satisfy the above equation, along with the point ∞ at infinity.

Example Elliptic curve $E : y^2 = x^3 + 2x + 3 \pmod{5}$ has points

$$(1,1), (1,4), (2,0), (3,1), (3,4), (4,0), \infty.$$

Example For elliptic curve $E : y^2 = x^3 + x + 6 \ (\mod 11)$ and its point $P = (2,7)$ it holds $2P = (5,2)$; $3P = (8,3)$. Number of points on an elliptic curve (mod p) can be easily estimated.

The addition of points on an elliptic curve mod n is done by the same formulas as given previously, except that instead of rational numbers c/d we deal with cd^{-1}

Example For the curve $E : y^2 = x^3 + 2x + 3$ it holds
$(1,4) + (3,1) = (2,0); (1,4) + (2,0) = (?,?)$.

EXAMPLE

On the elliptic curve

$$y^2 \equiv x^3 + x + 6 \pmod{11}$$

lies the point $P = (2,7)$

Indeed, $49 \equiv 16 \mod 11$.

To compute $2P$ we have

$$\lambda \equiv (3 \cdot 2^2 + 1)/(14) \equiv 13/14 \equiv 2/3 \equiv 2 \cdot 4 \equiv 8 \equiv \bmod 11$$

Therefore

$$x_3 \equiv 8^2 - 2 - 2 \equiv 60 \equiv 5 \bmod 11$$

and

$$y_3 \equiv 8(2-5) - 7 \equiv -31 \equiv -9 \equiv 2 \mod 11$$

PROPERTIES of ELLIPTIC CURVES MODULO n

- Elliptic curves have finitely many points and are finitely generated - all points can be obtained from few given points using the operation of addition.
- Hasse's theorem If an elliptic curve $E(\bmod p)$ has $|E|$ points then $||E| - p - 1| < 2\sqrt{p}$

ELLIPTIC CURVES DISCRETE LOGARITHM

Let E be an elliptic curve and A, B be its points such that $B = kA = (A + A + \dots A + A)$ – k times – for some k. The task to find such a k is called the discrete logarithm problem for elliptic curves.

No efficient algorithm to compute discrete logarithm problem for elliptic curves is known and also no good general attacks. Elliptic curves based cryptography is based on these facts.

There is the following general procedure for changing a discrete logarithm based cryptographic protocols to a cryptographic protocols based on elliptic curves:

- Assign given message (plaintext) to a point on a given elliptic curve E.
- Change, in the cryptographic protocol, modular multiplication to addition of points on E.
- Change, in the cryptographic protocol, exponentiation to multiplication of points of the elliptic curve E by integers.
- To the point of the elliptic curve E that results from such a protocol assigns a message (cryptotext).

MAPPING MESSAGES into POINTS of ELLIPTIC CURVES (I)

Problem and basic idea

The problem of assigning messages to points on elliptic curves is difficult because there are no polynomial-time algorithms to write down points of an arbitrary elliptic curve.

Fortunately, there is a fast randomized algorithm, to assign points of any elliptic curve to messages, that can fail with probability that can be made arbitrarily small.

Basic idea: Given an elliptic curve $E(\bmod p)$, the problem is that not to every x there is an y such that (x, y) is a point of E.

Given a message (number) m we therefore adjoin to m few bits at the end of m and adjust them until we get a number x such that $x^3 + ax + b$ is a square $\bmod p$.

MAPPING MESSAGES into POINTS of ELLIPTIC CURVES (II)

Technicalities

Let K be a large integer such that a failure rate of $\frac{1}{2^K}$ is acceptable when trying to encode a message by a point.

For $j \in \{0, \dots, K-1\}$ verify whether for $x = mK + j$, $x^3 + ax + b \pmod{p}$ is a square (mod p) of an integer y.

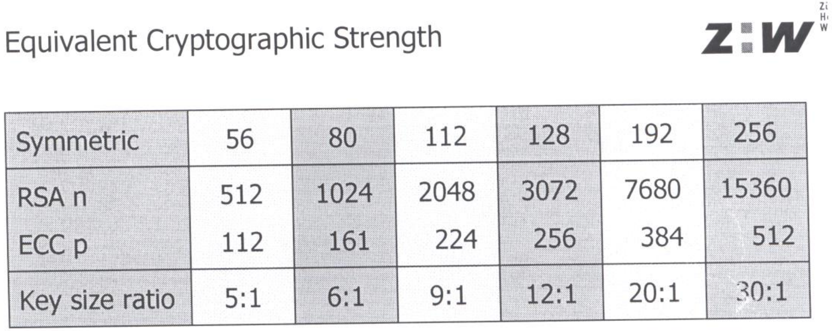
If such an j is found, encoding is done; if not the algorithm fails (with probability $\frac{1}{2^K}$ because $x^3 + ax + b$ is a square approximately half of the time).

In order to recover the message m from the point (x, y), we compute:

$$\left\lfloor \frac{x}{K} \right\rfloor$$

EFFICIENCY of various CRYPTOGRAPHIC SYSTEMS

The following pictures show how many bits needed keys of different cryptographic systems to achieve the same security.



Equivalent Cryptographic Strength

Symmetric	56	80	112	128	192	256
RSA n	512	1024	2048	3072	7680	15360
ECC p	112	161	224	256	384	512
Key size ratio	5:1	6:1	9:1	12:1	20:1	30:1

ELLIPTIC CURVES KEY EXCHANGE

Elliptic curve version of the Diffie-Hellman key generation protocol goes as follows:

Let Alice and Bob agree on a prime p, on an elliptic curve E (mod p) and on a point P on E.

- Alice chooses an integer n_a, computes n_aP and sends it to Bob.
- Bob chooses an integer n_b, computes n_bP and sends it to Alice.
- Alice computes $n_a(n_bP)$ and Bob computes $n_b(n_aP)$. This way they have the same key.

ELLIPTIC CURVES VERSION of ElGamal CRYPTOSYSTEM

Standard version of ElGamal: Bob chooses a prime p, a generator $q < p$, an integer x, computes $y = q^x \pmod p$, makes public p, q, y and keeps x secret.

To send a message m Alice chooses a random r, computes:

$$a = q^r \; ; \; b = my^r$$

and sends it to Bob who decrypts by calculating $m = ba^{-x} \pmod p$

Elliptic curve version of ElGamal: Bob chooses a prime p, an elliptic curve E (mod p), a point P on E, an integer x, computes Q = xP, makes E, p, and Q public and keeps x secret.

To send a message m ALice expresses m as a point X on E, chooses random r, computes

$$a = rP \; ; \; b = X + rQ$$

and sends the pair (a, b) to Bob who decrypts by calculating $X = b - xa$.

ELLIPTIC CURVES DIGITAL SIGNATURES

Elliptic curves version of ElGamal digital signatures has the following form for signing (a message) m, an integer, by Alice and to have the signature verified by Bob:

Alice chooses p and an elliptic curve E (mod p), a point P on E and calculates the number of points n on E (mod p) – what can be done, and we assume that $0 < m < n$.

Alice then chooses a random integer a and computes Q = aP. She makes public p, E, P, Q and keeps secret a.

To sign a message m Alice does the following:

- Alice chooses a random integer $r, 1 \leq r < n$ such that gcd(r,n) = 1 and computes R = rP = (x,y).
- Alice computes $s = r^{-1}(m - ax) \pmod{n}$
- Alice sends the signed message (m,R,s) to Bob.

Bob verifies the signature as follows:

- Bob declares the signature as valid if xQ + sR = mP

  The verification procedure works because

$$xQ + sR = xaP + r^{-1}(m - ax)(rP) = xaP + (m - ax)P = mP$$

Warning Observe that actually $rr^{-1} = 1 + tn$ for some t. For the above verification procedure to work we then have to use the fact that $nP = \infty$ and therefore $P + t \cdot \infty = P$

COMMENT

Federal (USA) elliptic curve digital signature standard (ECDSA) was introduced in 2005.

ELLIPTIC CURVES DETAILS

- A special definition is needed for an addition of a point to itself, that is for doubling of a point. Calculation of $P + P$ is defined in principle in a similar way as for two different points with the only difference that this time the tangent to the curve at the point P is constructed.
- In order to be able to avoid brute force attacks on elliptic curve cryptosystem the underlying elliptic curve must be considered in a large field. This means, when an implementation is considered, that much larger integers have to be considered as the size of the computer words and on these special arithmetic has to be implemented. An efficient implementation is offered by so called Montgomery representation of field elements.
- Every implementation of an elliptic curve cryptosystem has to cope with the problem of selecting/generating a good elliptic curve. (One way is to use www.kurvenfabrik.de to get such a curve.)

SECURITY of ELLIPTIC CURVE CRYPTOGRAPHY

- Security of ECC depends on the difficulty of solving the discrete logarithm problem over elliptic curves.
- Two general methods of solving such discrete logarithm problems are known.
- The square root method and Silver-Pohling-Hellman (SPH) method.
- SPH method factors the order of a curve into small primes and solves the discrete logarithm problem as a combination of discrete logarithms for small numbers.
- Computation time of the square root method is proportional to $O(\sqrt{e^n})$ where n is the order of the based element of the curve.

KEY SIZE

- All fastest known algorithms to solve elliptic curves discrete logarithm problem need $O(\sqrt{n})$ steps.
- This implies that the size of the underlying field (number of points on the chosen elliptic curve) should be roughly twice the security parameter.
- For example, for 128-bit security one needs a curve over $\underline{F}_q$, where $q \approx 2^{256}$.
- This can be contrasted with RSA cryptography that requires 3072 public and private keys.

- The hardest ECC scheme (publicly) broken to date had a 112-bit key for the prime field case and a 109-bit key for the binary field case.
- The prime field case was broken in July 2009 using 200 PlayStation 3 game consoles and could be finished in 3.5 months.
- The binary field case was broken in April 2004 using 2600 computers for 17 months.

GOOD ELLIPTIC CURVES

- NIST recommended 5 elliptic curves for prime fields, one for prime sizes 192, 224, 256, 384 and 521 bits.
- NIST also recommended five elliptic curves for binary fields $\mathbf{F}_{2^m}$ one for m equal 163, 233, 283, 409 and 571.

INTEGER FACTORIZATION - PROBLEM I

Two very basic questions concerning integers are large theoretical and also practical cryptographical importance.

- **Can a given integer *n* be factorized? (Or, is *n* prime?)**
- **If *n* can be factorized, find its factors.**

Till around 1977 no polynomial algorithm was know to determine primality of integers. In spite of the fact that this problem bothered mathematicians since antique ancient times.

In 1977 several very simple and fast randomized algorithms for primality testing were discovered - one of them is on the next slide.

So called Fundamental theorem of arithmetic, known since Euclid, claims that factorization of an integer n into a power of primes

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

is unique when primes p_i are ordered. However, theorem provides no clue how to find such a factorization.

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}.

INTEGER FACTORIZATION - PROBLEM II

In 2002 a deterministic, so called ASK, polynomial time algorithm, with complexity $O(n^{12})$ were discovered by three mathematicians from IIT Kanpur.

For factorization no polynomial deterministic algorithm is known and development of methods that would allow to factorized large integers is one of mega challenges for the development of computing algorithms and technology.

Largest recent success was factorization of so called RSA-768 number that has 232 digits (and 768 bits). Factorization took 2 years using several hundred of fast computers all over the world. On a single computer it would take 2000 years.

There is a lot of heuristics to factorized integers - some are very simple, other sophisticated. A method based on elliptic curves presented later, is one of them.

So far the fastest classical factorization algorithms work in time

$$e^{O((\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}})}$$

Factorization could be done in polynomial time using Shor's algorithm and a powerful quantum computer, as discussed later.

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

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

are primes.

This is indeed 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

$$F_6 = 18446744073709551617 = 274177 \cdot 67280421310721$$

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

$$F_7 = 340282366920938463463374607431768211457 =$$
$$= 5704689200685129054721 \cdot 5964958912749 7217$$

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

FACTORIZATION BASICS

- Not all numbers of a given length are equally hard to factor. The hardest instances are **semi-primes** - products of two primes of similar length.
- Concerning complexity classes it holds. **Function version of the factorization problem is known to be in FNP and it is not known to be in FP**.

  Decision version of the factorization problem: Does an integer n has a factor smaller than d? is known to be in NP and not known to be in P. Moreover it is known to be both in **NP** and **co-NP** as well both in **UP** and **co-UP**.

VERY SIMPLE FACTORIZATION METHODS

Euler's factorization method

The idea is to factorize an integer n by writing it as a sum of two different integers in two different ways, that is

$$n = a^2 + b^2 = c^2 + d^2 - - - - - - - - 1000009 = 1000^2 + 3^2 = 972^2 + 235^2$$

.

Fermat's factorization method If n = pq, $p < \sqrt{n}$, then

$$n = \left(\frac{q+p}{2}\right)^2 - \left(\frac{q-p}{2}\right)^2 = a^2 - b^2$$

Therefore, in order to find a factor of n, we need only to investigate the values

$$x = a^2 - n$$

$$\text{for } a = \left\lceil\sqrt{n}\right\rceil + 1, \left\lceil\sqrt{n}\right\rceil + 2, \ldots, \frac{(n-1)}{2}$$

until a perfect square is found.

Pollard's factorization methods They are discussed next and in the Appendix.

FIRST Pollard ρ-ALGORITHM I

A simple factorization algorithm, invented by John Pollard in 1975, has its efficiency based on two facts.

- **Fact 1** For a given prime p, as in birthday problem, two numbers are congruent modulo p, with probability 0.5 after $1.177\sqrt{p}$ numbers have been randomly chosen.
- **Fact 2** If p is a factor of an n, then $p < gcd(x-y, n)$ since p divides both n and $x-y$.

POLLARD's ρ-algorithm

```
Input: An integer $n$ to factorize. $x_0 \leftarrow random$; $a \leftarrow x_0$; $b \leftarrow x_0$; $d \leftarrow 1$;
while $d = 1$
  $a \leftarrow f(a) \mod n$;
  $b \leftarrow f(f(b) \mod n) \mod n$;
  $d \leftarrow gcd(|a-b|, n)$;
If $d = n$ return failure else return $d$.
```

Algorithm is fast in the case of at least one small factor. For example, it is reported that that on a 3 GHz processor, the factor 274177 of the sixth Fermat number (18446744073709551617) was found in 26 milliseconds.

An improvement of the algorithm, due to Pollard and R. Brent: Instead of computing $gcd(|a-b|)$ at every iteration, z is defined as the product of several, say 100 consecutive $|a-b|$ terms modulo n and then a single $gcd(z, n)$ is computed.

Algorithm is fast for small factors

ρ-ALGORITHM - EXAMPLE

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

$$n = 18923; \quad a = b = x_0 = 2347$$

$$a \leftarrow f(a) \bmod n; b \leftarrow f(f(b)) \bmod n$$

$$\begin{array}{lllllllll}
a & = & 1817 & b & = & 8888 & gcd & = & 1 \\
a & = & 8888 & b & = & 12599 & gcd & = & 1 \\
a & = & 11943 & b & = & 13068 & gcd & = & 1 \\
a & = & 12599 & b & = & 1342 & gcd & = & 1 \\
a & = & 8678 & b & = & 10137 & gcd & = & 1 \\
a & = & 13068 & b & = & 7978 & gcd & = & 1 \\
a & = & 11473 & b & = & 8232 & gcd & = & 1 \\
a & = & 1342 & b & = & 16487 & gcd & = & 1 \\
a & = & 3280 & b & = & 11407 & gcd & = & 1 \\
a & = & 10137 & b & = & 11280 & gcd & = & 127
\end{array}$$

FACTORING with ELLIPTIC CURVES

Basis idea: To factorize an integer n choose an elliptic curve E, a point P on E and compute, modulo n, either iP for $i = 2, 3, 4, \ldots$ or $2^j P$ for $j = 1, 2, \ldots$. **The point is that in doing such calculations one needs to compute gcd(k,n) for various k. If one of these values is between 1 and n we have a factor of n.**

Factoring of large integers: The above idea can be easily parallelised and converted to using an enormous number of computers to factor a single very large n. Each computer gets some number of elliptic curves and some points on them and multiplies these points by some integers according to the rule for addition of points. If one of computers encounters, during such a computation, a need to compute $1 < gcd(k, n) < n$, factorization is finished.

Example: If curve $E : y^2 = x^3 + 4x + 4 \pmod{2773}$ and its point $P = (1, 3)$ are used, then $2P = (1771, 705)$ and in order to compute 3P one has to compute $gcd(1770, 2773) = 59$ – factorization is done.

Example: For elliptic curve $E : y^2 = x^3 + x - 1 \pmod{35}$ and its point $P = (1, 1)$ we have $2P = (2, 32); 4P = (25, 12); 8P = (6, 9)$ and at the attempt to compute 9P one needs to compute $gcd(15, 35) = 5$ and factorization is done.

The only things that remain to be explored is how efficient this method is and when it is more efficient than other methods.

IMPORTANT OBSERVATIONS (1)

- If $n = pq$ for primes p, q, then an elliptic curve $E \pmod n$ can be seen as a pair of elliptic curves $E \pmod p$ and $E \pmod q$.
- It follows from the Lagrange theorem that for any elliptic curve $E \pmod n$ and its point P there is an $k < n$ such that $kP = \infty$.
- In case of an elliptic curve $E \pmod p$ for some prime p, the smallest positive integer m such that $mP = \infty$ for some point P divides the number N of points on the curve $E \pmod p$. Hence $NP = \infty$.

  If N is a product of small primes, then b! will be a multiple of N for a reasonable small b. Therefore, $b!P = \infty$.
- The number with only small factors is called smooth and if all factors are smaller than an b, then it is called b-smooth.

It can be shown that the density of smooth integers is so large that if we choose a random elliptic curve $E \pmod n$ then it is a reasonable chance that n is smooth.

PRACTICALITY of FACTORING USING ECC I

Let us continue to discuss the following key problem for factorization using elliptic curves:

Problem: How to choose integer k such that for a given point P we should try to compute points iP or $2^i P$ for all multiples of P smaller than kP?

Idea: If one searches for m-digits factors, one chooses k in such a way that k is a multiple of as many as possible of those m-digit numbers 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 the elliptic curve modulo n.

Method 1: 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:

PRACTICALITY of FACTORING USING ECC - II

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 elliptic curves method depends on the size of factors.

ELLIPTIC CURVES FACTORIZATION - DETAILS

Given an n such that gcd(n, 6) = 1 and let the smallest factor of n be expected to be smaller than an F. One should then proceed as follows:
Choose an integer parameter r and:

1. Select, randomly, an elliptic curve
$$E : y^2 = x^3 + ax + b$$
such that $gcd(n, 4a^2 + 27b^2) = 1$ and a random point P on E.
2. Choose integer bounds A,B,M such that
$$M = \prod_{j=1}^{l} p_j^{a_{p_j}}$$
for some primes $p_1 < p_2 < \ldots < p_l \leq B$ and a_{p_j}, being the largest exponent such that $p_j^{a_j} \leq A$.
Set j = k = 1
3. Calculate $p_j P$.
4. Computing gcd.
   - If $p_j P \neq O \pmod{n}$, then set $P = p_j P$ and reset $k \leftarrow k + 1$
     1. If $k \leq a_{p_j}$, then go to step (3).

ELLIPTIC CURVES FACTORIZATION - DETAILS II

2. If $k > a_{p_j}$, then reset $j \leftarrow j+1$, $k \leftarrow 1$.
   If $j \leq l$, then go to step (3); otherwise go to step (5)

- If $p_j P \equiv O(\mod n)$ and no factor of n was found at the computation of inverse elements, then go to step (5)

5. Reset $r \leftarrow r-1$. If $r > 0$ go to step (1); otherwise terminate with "failure".
   The "smoothness bound" B is recommended to be chosen as
   $$B = e^{\sqrt{\frac{lnF(lnlnF)}{2}}}$$
   and in such a case running time is
   $$O(e^{\sqrt{2+o(1lnF(lnlnF))}ln^2n})$$

ELLIPTIC CURVES FACTORIZATION: FAQ

- How to choose (randomly) an elliptic curve E and point P on E? An easy way is first choose a point $P(x, y)$ and an a and then compute $b = y^2 - x^3 - ax$ to get the curve $E : y^2 = x^3 + ax + b$.
- What happens at the factorization using elliptic curve method, if for a chosen curve $E \pmod{n}$ the corresponding cubic polynomial $x^3 + ax + b$ has multiple roots (that is if $4a^3 + 27b^2 = 0$) ? No problem, method still works.
- What kind of elliptic curves are really used in cryptography? Elliptic curves over fields $GF(2^n)$ for $n > 150$. Dealing with such elliptic curves requires, however, slightly different rules.
- History of ECC? The idea came from Neal Koblitz and Victor S. Miller in 1985. Best known algorithm is due to Lenstra.
- How secure is ECC?No mathematical proof of security is know.
- How about patents concerning ECC?There are patents in force covering certain aspects of ECC technology.

FACTORIZATION on QUANTUM COMPUTERS

In the following we present the basic idea behind a polynomial time algorithm for quantum computers to factorize integers.

Quantum computers works with superpositions of basic quantum states on which very special (unitary) operations are applied and and very special quantum features (non-locality) are used.

Quantum computers work not with bits, that can take on any of two values 0 and 1, but with qubits (quantum bits) that can take on any of infinitely many states $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

REDUCTIONS

Shor's polynomial time quantum factorization algorithm is based on an understanding that factorization problem can be reduced

1. first on the problem of solving a simple modular quadratic equation;
2. second on the problem of finding period of functions $f(x) = a^x \bmod n$.

FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

Proof. Let $a \neq \pm 1$ be such that $a^2 \equiv 1 \pmod{n}$. Since

$$a^2 - 1 = (a+1)(a-1),$$

if n is not prime, then a prime factor of n has to be a prime factor of either $a+1$ or $a-1$. By using Euclid's algorithm to compute

$$gcd(a+1, n) \text{ and } \quad gcd(a-1, n)$$

we can find, in $O(\lg n)$ steps, a prime factor of n.

SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer r such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any k, i.e. the smallest r such that

$$x^r \equiv 1 \pmod n.$$

AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod n$.

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a,n)$. *If* $gcd(a,n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod n$,*then go to step 1; otherwise stop.*

If this algorithm stops, then $a^{r/2}$ is a non-trivial solution of the equation

$$x^2 \equiv 1 \pmod n.$$

EXAMPLE

Let $n = 15$. Select $a < 15$ such that $gcd(a, 15) = 1$.
$\{$The set of such a is $\{2, 4, 7, 8, 11, 13, 14\}\}$

Choose $a = 11$. Values of 11^x mod 15 are then

$$11, 1, 11, 1, 11, 1$$

which gives $r = 2$.

Hence $a^{r/2} = 11 \pmod{15}$. Therefore

$$gcd(15, 12) = 3, \qquad gcd(15, 10) = 5$$

For $a = 14$ we get again $r = 2$, but in this case

$$14^{2/2} \equiv -1 \pmod{15}$$

and the following algorithm fails.

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod{n}$,*then go to step 1; otherwise stop.*

EFFICIENCY of REDUCTION

Lemma If $1 < a < n$ satisfying $gcd(n, a) = 1$ is selected in the above algorithm randomly and n is not a power of prime, then

$$Pr\{r \text{ is even and } a^{r/2} \not\equiv \pm 1\} \geq \frac{9}{16}.$$

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* r *of function* $a^k \bmod n$.
4. *If* r *is odd or* $a^{r/2} \equiv \pm 1 \pmod{n}$*,then go to step 1; otherwise stop.*

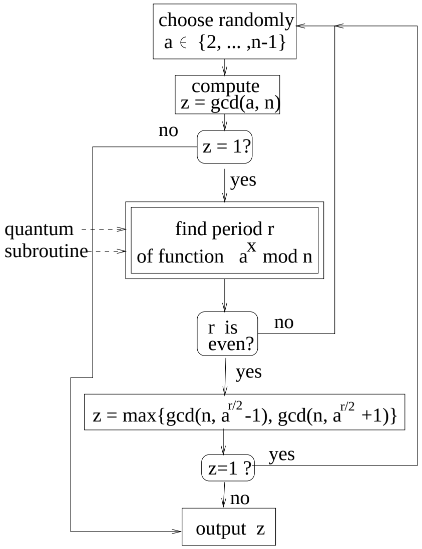
Corollary If there is a polynomial time randomized [quantum] algorithm to compute the period of the function

$$f_{n,a}(k) = a^k \bmod n,$$

then there is a polynomial time randomized [quantum] algorithm to find non-trivial solution of the equation $a^2 \equiv 1 \pmod{n}$ (and therefore also to factorize integers).

A GENERAL SCHEME for Shor's ALGORITHM

The following flow diagram shows the general scheme of Shor's quantum factorization algorithm


choose randomly a ∈ {2, ... ,n-1}
compute z = gcd(a, n)
z = 1?
no
yes
quantum subroutine
find period r of function a^x mod n
r is even?
no
yes
z = max{gcd(n, a^{r/2} -1), gcd(n, a^{r/2} +1)}
z=1 ?
yes
no
output z

POLLARD ρ-METHOD in GENERAL

A variety of factorization algorithms, of complexity around $O(\sqrt{p})$ where p is the smallest prime factor of n, is based on the following idea:

- A function f is taken that "behaves like a randomizing function" and $f(x) \equiv f(x \bmod p) \pmod p$ for any factor p of n – usually $f(x) = x^2 + 1$
- A random x_0 is taken and iteration
$$x_{i+1} = f(x_i) \mod n$$
is performed (this modulo n computation actually "hides" modulo p computation in the following sense: if $x_0' = x_0$, $x_{i+1}' = f(x_i') \mod n$, then $x_i' = x_i \mod p$)
- Since $\mathbf{Z}_p$ is finite, the shape of the sequence x_i' will remind the letter ρ, with a tail and a loop. Since f is "random", the loop modulo n rarely synchronizes with the loop modulo p
- The loop is easy to detect by GCD-computations and it can be shown that the total length of tail and loop is $O(\sqrt{p})$.

LOOP DETECTION

In order to detect the loop it is enough to perform the following computation:

```
$a \leftarrow x_0$; $b \leftarrow x_0$;
**repeat**
    $a \leftarrow f(a)$;
    $b \leftarrow f(f(b))$;
**until** a = b
```

Iteration ends if $a_t = b_{2t}$ for some t greater than the tail length and a multiple of the loop length.

SECOND Pollard ρ-ALGORITHM

Basic idea

1. Choose an easy to compute $f : Z_n \rightarrow Z_n$ and $x_0 \in Z_n$.

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

2. Keep computing $x_{i+1} = f(x_j)$, $j = 0, 1, 2, \ldots$ and $gcd(x_j - x_k, n)$, $k \leq j$. (Observe that if $x_j \equiv x_k \mod p$ for a prime factor p of n, then $gcd(x_j - x_k, n) \leq p$.)

   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 ρ-method, it is important to choose a function f in such a way that f maps Z_n into Z_n in a "random" way.

Basic question: How good is the ρ-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?)

POLLARD's p-1 algorithm

Pollard's algorithm (to factor n given a bound b on factors).

```
a := 2;
for j=2 to b do $a := a^j \mod n$;
$f := gcd(a-1, n)$;          – { $f = gcd(2^{b!} - 1, n)$}
if $1 < f < n$ then f is a factor of n otherwise failure
```

Indeed, let p be a prime divisor of n and $q < b$ for every prime $q|(p-1)$.
(Hence $(p-1)|b!$).

At the end of the **for**-loop we have

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

and therefore

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

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

$$p|gcd(a-1, n)$$

IMPORTANT OBSERVATIONS II

Pollard ρ-method works fine for numbers with a small factor.

The p-1 method requires that p-1 is smooth. The elliptic curve method requires only that there are enough smooth integers near p and so at least one of randomly chosen integers near p is smooth.

This means that the elliptic curves factorization method succeeds much more often than p-1 method.

Fermat factorization and Quadratic Sieve method discussed later work fine if integer has two factors of almost the same size.

QUADRATIC SIEVE METHOD of FACTORIZATION - BASIC IDEAS

Step 1 To factorize an n one finds many integers x such that $x^2 - n$, $n = 7429$, has only small factors and decomposition of $x^2 - n$ into small factors.

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

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

$$(87^2 - 7429)(88^2 - 7429) \equiv (87 \cdot 88)^2 \equiv 7656^2 \equiv 227^2 \mod 7429$$
$$\text{and therefore } 227^2 \equiv 210^2 \mod 7429$$

Hence 7429 divides $227^2 - 210^2$ and therefore $17 = 227 - 210$ is a factor of 7429.
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 square the following equations have to hold
$$\begin{array}{rl} \lambda_1 & \equiv 0 \mod 2 \\ \lambda_1 + \lambda_2 + \lambda_3 & \equiv 0 \mod 2 \\ \lambda_2 + \lambda_3 & \equiv 0 \mod 2 \\ \lambda_1 = 0, & \lambda_2 = \lambda_3 = 1 \end{array}$$

QUADRATIC SIEVE FACTORIZATION - SKETCH of METHODS

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 $-k \leq u \leq k$ 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 - see table for n=7429 and m=86.

u	-3	-2	-1	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

QUADRATIC SIEVE (QS) FACTORIZATION - SUMMARY I

- Method was invented Carl Pomerance in 1981.
- It is currently second fastest factorization method known and the fastest one for factoring integers under 100 decimal digits.
- It consists of two phases: data collection and data processing.
- In data collection phase for factoring n a huge set of such integers x is found that numbers $(x+\lceil\sqrt{n}\rceil)^2-n$ have only small factors as well all these factors. This phase is easy to parallelise and can use methods called **sieving** for finding all required integers with only small factors.
- In data processing phase a system of linear congruences is formed on the basis of factorizations obtained in the data collection phase and this system is solved to reach factorization. This phase is much memory consuming for storing huge matrices and so hard to parallelise.
- The basis of sieving is the fact that if $y(x)=x^2-n$, then for any prime p it holds $y(x+kp)\equiv y(x) \pmod p$ and therefore solving $y(x)\equiv 0 \mod p$ for x generate a whole sequence of y which are divisible by p.
- The general running time of QS, to factor n, is

$$e^{(1+o(1))\sqrt{\lg n \lg\lg n}}$$

- The current record of QS is a 135-digit co-factor of $2^{803}-2^{402}-1$.

FACTORING ALGORITHMS RUNNING TIMES

Let p denote the smallest factor of an integer n and p^* the largest prime factor of $p-1$.

Pollard's Rho algorithm	$O(\sqrt{p})$
Pollard's $p-1$ algorithm	$O(p^*)$
Elliptic curve method	$\O(e^{(1+o(1))\sqrt{2\ln p \ln\ln p)}})$
Quadratic sieve method	$\O(e^{1+o(1))\sqrt{(\ln n \ln\ln n)}})$
Number field sieve method	$\O(e^{(\frac{64}{9}\ln n)^{1/3}(\ln\ln n)^{2/3}})$

Fastest is the general number field sieve method; the second one the quadratic sieve method.

APPENDIX

HISTORY of ELLIPTIC CURVES CRYPTOGRAPHY

- The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985.
- Behind this method is a believe that the discrete logarithm of a random elliptic curve element with respect to publicly known base point is infeasible.
- At first Elliptic curves over a prime finite field were used for ECC. Later also elliptic curves over the fields $GF(2^m)$ started to be used.
- In 2005 the US NSA endorsed to use ECC (Elliptic curves cryptography) with 384-bit key to protect information classified as "top secret".
- There are patents in force covering certain aspects of ECC technology.
- Elliptic curves have been first used for factorization by Lenstra.
- Elliptic curves played an important role in perhaps most celebrated mathematical proof of the last hundred years - in the proof of Fermat's Last Theorem - due to A. Wiles and R. Taylor.

MODULO COMPUTATION

For special types of primes p computation of modulo p can be done much faster.

For example, for

$$p = 2^{251} - 1$$

or

$$p = 2^{256} - 1$$

or

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1.$$

FACTORING ALGORITHMS RUNNING TIMES

Let p denote the smallest factor of an integer n and p^* the largest prime factor of $p-1$.

Pollard's Rho algorithm	$O(\sqrt{p})$
Pollard's $p-1$ algorithm	$O(p^*)$
Elliptic curve method	$\emptyset(e^{(1+o(1))\sqrt{2\ln p\ln\ln p)}})$
Quadratic sieve method	$\emptyset(e^{1+o(1))\sqrt{(\ln n\ln\ln n)}})$
Number field sieve method	$\emptyset(e^{(\frac{64}{9}\ln n)^{1/3}(\ln\ln n)^{2/3}})$

Fastest is the general number field sieve method; the second one the quadratic sieve method.

RSA FACTORING CHALLENGES

- In 1991 RSA Laboratories published a list of semi-primes (numbers that are product of two primes) and prizes for their decoding.
- Numbers are named as RSA-x, where x is number of decimal or binary digits of the number.
- The largest price cashed so far was 30 000 $ for factorization of RSA-704.
- The largest price offered was 200 000 $ for factorization of RSA-2024.
- Challenge is no longer active - no longer are prices given.
- Numbers were generated on a computer with no network connections and after their generation hard drive was destroyed and therefore nobody knows their factorization.

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} 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

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.

Fermat FACTORIZATION - DETAILS

Basic idea: Factorization is easy if one finds x, y such that $n|(x^2 - y^2)$

Proof: 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

$n = 7429 = 227^2 - 210^2$, $\quad x = 227$, $y = 210$

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

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

How to find such x and y?

First idea: one tries all t starting with $\sqrt{n}$ until $t^2 - n$ is a square S^2.

Second idea: One forms a system of (modular) linear equations and determines x and y from the solutions of such a 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