IV054

CHAPTER 8: Elliptic Curves Cryptography and factorization

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

For example, US-government has recommended to use elliptic curve cryptography.

The main advantage of elliptic curves cryptography is that to achieve a certain level of security shorter keys are required 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 available for cryptography based on factorization and discrete logarithm do not work for elliptic curves cryptography.

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

Elliptic Curves

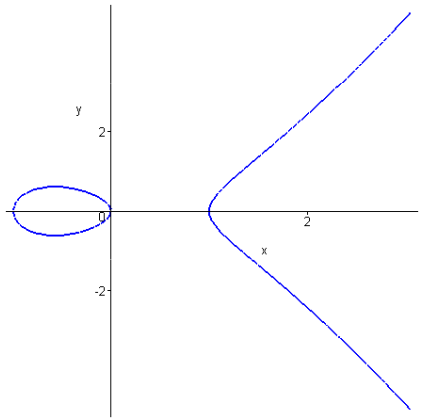
An elliptic curve *E* is the graph of the relation defined by the equation

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

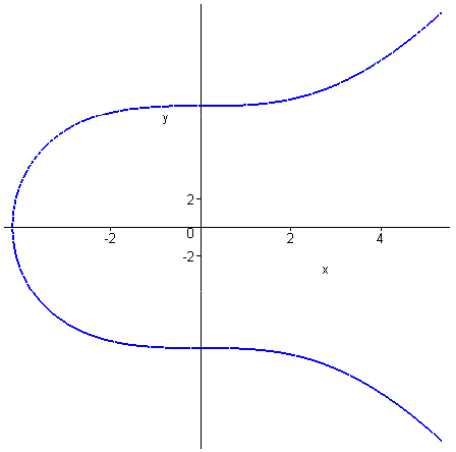
(where *a, b* will be either rational numbers or integers (and computation may be done modulo some *n*)) extended by a "point at infinity", denoted usually as ∞ (or 0) that can be regarded as sitting, at the same time, at the very top and very bottom of the *y*-axis.

We will consider mainly only those elliptic curves that have no multiple roots - what 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 form shown in the following figure that depends on whether polynomial x^3+ax+b has three or one real root.



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



$y^2=x^3+73$

Historical Remarks

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 not to consider 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 our general equation can be transformed to our special case. In other cases, it may be necessary to consider the most general form of equation.

Addition of Points on Elliptic Curves (1)

Geometry

On elliptic curves we can define addition of points in such a way that points of the corresponding curve with such an addition form an Abelian group.

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 $P+\infty = P$.)

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.

ELIPTIC CURVES - GENERALITY

An 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 ***0***, called the point of infinity.

If *p≠2* Weierstrass equation can be simplified by transformation

$$y \rightarrow y - (ux + v) / 2$$

to get the equation

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

for some constants *d,e,f* and if *p≠3* by transformation

$$x \rightarrow x - d / 3$$

to get equation

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

Addition of Points on Elliptic Curves (2)

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

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

All that holds for the case that λ is finite; otherwise $P_3 = \infty$.

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

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 $y^2=x^3+2x+3 \pmod 5$ has points

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

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

Hasse's theorem If an elliptic curve $E \pmod p$ has N points then $|N-p-1|<2\sqrt{p}$

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 $(1,4)+(3,1)=(2,0)$; $(1,4)+(2,0)=(?,?)$.

Elliptic Curve Discrete Logarithm

Let *E* be an elliptic curve and *A, B* be its points such that $B = kA = (A + A + \ldots + A)$ - 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.

A general procedure for changing a discrete logarithm based cryptographic protocols to a cryptographic protocols based on elliptic curves:

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

Mapping Messages into Points of Elliptic Curves (1)

Problem and basic idea

The problem of assigning messages to points on an elliptic curve 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 (mod 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 mod p.

Mapping Messages into Points of Elliptic Curves (2)

Technicalities

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

For j from 0 to K verify whether for $x = mK + j$, $x^3 + ax + b \pmod p$ is a square $\pmod p$ of an integer y.

If such an j is found, encoding is done; if not the algorithm fails (with probability $1/2^K$ 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:

$$\left\lceil \frac{x}{K} \right\rceil$$

Elliptic Curve Key Exchange

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

Let Alice and Bob agree on a prime p, on an elliptic curve $E \pmod p$ 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 Curve Version of ElGamal Cryptosystem

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

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

$$y_1 = q^r \; ; \; y_2 = my^r$$

and sends it to Bob who decrypts by calculating $m = y_2 y_1^{-a} \pmod p$

Elliptic curve version of ElGamal: Bob chooses a prime p, an elliptic curve $E \pmod p$, a point P on E, an integer a, computes $Q = aP$, makes $E, p,$ and Q public and keeps a secret.

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

$$y_1 = rP \; ; \; y_2 = X + rQ$$

And sends the pair (y_1, y_2) to Bob who decrypts by calculating $X = y_2 - ay_1$.

Elliptic Curve Digital Signature

Eliptic 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 $E \pmod p$, a point P on E and calculates the number of points n on $E \pmod p$ – 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 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 $P + t\,\infty = P$

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 i*n doing that 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=(2,2); 4P=(0,22); 8P=(16,19) and at the attempt to compute 9P one needs to compute gcd(15,35)=5 and factorization is done. The only things that remains to be explored is how efficient is this method 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 (1)

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

Problem: How to choose k such that for a given point P we should try to compute points iP or 2^iP 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 $k=2^7\cdot 3^4\cdot 5^3\cdot 7^2\cdot 11\cdot 2\cdot 13\cdot \ldots \cdot 139$. The following table shows B and the number of elliptic curves one has to test:

Practicality of Factoring Using ECC (2)

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 curve 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 curve 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 $p_j P \equiv O \pmod{n}$ 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 $B =$

$$e^{\sqrt{\ln F (\ln \ln F)/2}}$$

and in such a case running time is

$$O(e^{\sqrt{2+o(1 \ln F (\ln \ln F))} \ln^2 n})$$

Elliptic Curves: 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 $(E \bmod n)$ 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.

FACTORIZATION

Factorization of integers is a very important problem.

A variety of techniques has been developed to deal with this problem.

So far the fastest classical factorization algorithms work in time

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

The fastest quantum algorithm for factorization works in both quantum and classical polynomial time.

In the rest of chapter several factorization methods will be presented and discussed.

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 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)

$$F_7 = 340282366920938463463374607431768211457 =$$
$$= 57046892006851290547 21 \cdot 5964958912749 7217$$

1980 Brent+Pollard found factorization for F_8

1990 A. K. Lenstra+… found factorization for F_9 (155 digits)

$$x^{n-1} \neq 1 \pmod{n}$$

FERMAT TEST

It follows from the Little Fermat Theorem that if p is a prime, then for all 0<b<p, we have

$$b^{p-1} \equiv 1 (\mathrm{mod}\, p)$$

Can we say that n is prime if and only if for all 0<b<n, we have

$$b^{n-1} \equiv 1 (\mathrm{mod}\, n)?$$

No, there are composed numbers n, so-called Carmichael numbers, such that for all 0<b<n that are primes with n it holds

$$b^{n-1} \equiv 1 (\mathrm{mod}\ n)$$

Such number is, for example, n=561.

Pollard ρ-Method

A variety of factorization algorithms, of complexity around $O(p^{1/2})$ 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

$$x_{i+1} = f(x_i) \bmod n$$

  is performed (this modulo n computation actually "hides" modulo p computation in the following sense: if $x'_0 = x_0$, $x'_{i+1} = f(x'_i) \bmod n$, then $x'_i = x_i \bmod p$)
- 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 $O(p^{1/2})$.

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.

First Pollard ρ-algorithm

Input: an integer *n* with a factor smaller than B

Complexity: $O(B^{1/2})$ of arithmetic operations

$x_0 \leftarrow$ random; $a \leftarrow x_0$; $b \leftarrow x_0$;
do
    $a \leftarrow f(a) \bmod n$;
    $b \leftarrow f(f(b) \bmod n) \bmod n$;
until $\gcd(a - b, n) \neq 1$
output $\gcd(a - b, n)$

The proof that complexity of the first Pollard-ρ factorization algorithm is given by $O(n^{1/4})$ arithmetic operations is based on the following result:

Lemma Let x_0 be random and f be "random" in Z_p, $x_{i+1} = f(x_i)$. The probability that all elements of the sequence

$$x_0, x_1, \ldots, x_t$$

are pairwise different when $t = 1 + \text{floor}((2\lambda p)^{1/2})$ is less than $e^{-\lambda}$.

Second Pollard ρ-algorithm

Basic idea 1. Choose an easy to compute f: $Z_n \to Z_n$ 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,… and $\gcd(x_j - x_k, n)$, $k \leq j$.

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

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?)

Basic lemma

Given: n, $f:Z_n \to Z_n$ and $x_0 \in Z_n$

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 $\lambda > 0$, $l = 1 + \lfloor \sqrt{2\lambda r} \rfloor$. 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,…

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 $r^{-l-1} \coprod_{j=0}^{l} (r-j)$.

For $l = 1 + \lfloor \sqrt{2}lr \rfloor$ we have $\ln\left(r^{-l-1} \coprod_{j=0}^{l} (r-j)\right) = \ln\left(\coprod_{j=0}^{l} \left(1 - \frac{j}{r}\right)\right) \leq \sum_{j=1}^{l} -\frac{j}{r} = -\frac{l(l+1)}{2r}$

$$< -\frac{l^2}{2r} < -\frac{\left(\sqrt{2\lambda r}\right)^2}{2r} < -\lambda.$$

RHO-ALGORITHM

A simplification of the basic idea: For each k compute $\gcd(x_k - x_j, n)$ for just one $j < k$.

Choose $f:Z_n \rightarrow Z_n$, 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 $\gcd(x_k, x_{2\text{^}h-1})$.

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

$x_1 = f(2) = 7$,	$\gcd(x_1 - x_0, n) = 1$
$x_2 = f(7) = 57$,	$\gcd(x_2 - x_1, n) = \gcd(57 - 7, n) = 1$
$x_3 = f(57) = 3307$,	$\gcd(x_3 - x_1, n) = \gcd(3307 - 7, n) = 1$
$x_4 = f(3307) = 2745$,	$\gcd(x_4 - x_3, n) = \gcd(2745 - 3307, n) = 1$
$x_5 = f(2746) = 1343$,	$\gcd(x_5 - x_3, n) = \gcd(1343 - 3307, n) = 1$
$x_6 = f(1343) = 2626$,	$\gcd(x_6 - x_3, n) = \gcd(2626 - 3307, n) = 1$
$x_7 = f(2626) = 3734$,	$\gcd(x_7 - x_3, n) = \gcd(3734 - 3307, n) = 61$

Disadvantage We likely will not detect the first case such that for some k_0 there is a $j_0 < k_0$ such that $\gcd(x_{k0} - x_{j0}, n) > 1$.

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, $\gcd(x_k - x_j, n) > 1$

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

IV054 RHO-ALGORITHM

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 $O(\sqrt[4]{n}\log^3 n)$ 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) mod 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 $x_{k0} \equiv x_{j0} \bmod r$, then the rho-algorithm finds r in $k \le 4k_0$ 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 $k_0 \le 1+\sqrt{2\lambda r}$, then the number of bits operations needed to find r is bounded by

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

If we choose C > 4sqrt(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 $e^{-\lambda}$.

COMMENTS

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

Next method, so called Pollard (p-1)-method, works fine for n having a prime factor p such that all prime factors of p-1 are small.

When all prime factors of p-1 are smaller than a B, we say that p-1 is B-smooth.

POLLARD 's *p-1* algorithm

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

a := 2;

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

$f := \gcd(a-1,n);$ $f=\gcd(2^{b!}-1,n)\}$

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

$$p \mid \gcd(a-1,n)$$

Important Observations (2)

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 works fine if integer has two factors of almost the same size.

Fermat factorization

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

$$x = a^2 - n$$

for $a = \lceil\sqrt{n}\rceil + 1, \lceil\sqrt{n}\rceil + 2, \ldots, (n-1)/2$
until a perfect square is found.

IV054 FERMAT FACTORIZATION

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

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

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

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

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

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

IV054 Method of Quadratic Sieve to factorize an integer *n*

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

Example

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

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

$$(87\cdot 88)^2 \equiv (87^2 - 7429)(88^2 - 7429) \bmod 7429$$
$$227^2 \equiv 210^2 \bmod 7429$$

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

$$\begin{aligned} \lambda 1 &\equiv 0 \bmod 2 \\ \lambda 1 + \lambda 2 + \lambda 3 &\equiv 0 \bmod 2 \\ \lambda 2 + \lambda 3 &\equiv 0 \bmod 2 \end{aligned}$$

$$\lambda 1 = 0,\ \lambda 2 = \lambda 3 = 1$$

Method of quadratic sieve to factorize *n*

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.

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

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 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, 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 - $10^{10\text{^}100}$

FACTORIZATION of very large NUMBERS

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

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