RABIN CRYPTOSYSTEM

Public: $n, 0 \leq b \leq n-1$.

Private: p, q such that $n = pq$ and $p \equiv q \equiv 3 \bmod 4$.

Encryption:

$$M \to E(M) = M(M+b) \bmod n.$$

Decryption:

$$N \to D(N) = \left(\sqrt{\frac{b^2}{4} + N} - \frac{b}{2} \right) \bmod n$$

There is an ambiguity in the decryption because there are four possible square roots modulo n. If ω is a nontrivial square root of 1 modulo n then it is easy to see that the four square roots are

$$M, -M-b, \omega(M+b/2)-b/2, -\omega(M+b/2)-b/2$$

If $p \equiv 1 \bmod 4$ no algorithm is known for finding square roots.

If $p \equiv 3 \bmod 4$ then there is a formula. Indeed, let $x \in QR_n$. Then $x \in QR_p$ and $x \in QR_q$. Since $p \equiv 3 \bmod 4$ we have that $4|(p+1)$. Therefore

$$\begin{aligned}(\pm x^{\frac{p+1}{4}})^2 &\equiv x^{\frac{p+1}{2}} \bmod p \\ &\equiv x^{\frac{p-1}{2}} \cdot x \bmod p \\ &\equiv x \bmod p,\end{aligned}$$

(Note $x^{\frac{p-1}{2}} \equiv 1 \bmod p$, since $x \in QR_p$.) Hence, $\pm x^{\frac{p+1}{4}}$ are the square roots of $x \bmod p$. With similar reasoning we conclude $\pm x^{\frac{q+1}{4}}$ are the square roots of $x \bmod q$.

Thus if we have the two square roots modulo p and the two square roots modulo q then (as explained before) we can use the Chinese remainder theorem to find the four square roots modulo n.

Example: $n = 77 = 7 \cdot 11, b = 9$. Note $7 \equiv 11 \equiv 3 \bmod 4$. Note $2^{-1} = 39 \bmod 77$ and $4^{-1} \equiv 58 \bmod 77$.

Encryption: $M \to E(M) = M(M+9) = M^2 + 9 \cdot M \bmod 77$.

Decryption: $N \to D(N) = \sqrt{N + \frac{81}{4}} - \frac{9}{2} = (\sqrt{N+1} - 43) \bmod 77$.

Say the ciphertext $N = 22$. Must compute $\sqrt{23} \bmod 77$. We have

$$23^{(p+1)/4} \equiv 23^2 \equiv 2^2 \equiv 4 \bmod 7$$
$$23^{(q+1)/4} \equiv 23^3 \equiv 1^3 \equiv 1 \bmod 11$$

Use the Chinese remainder theorem to find the four roots of 23 modulo 77. These are

$$\pm 10 \bmod 77, \pm 32 \bmod 77.$$

The four possible plaintexts are

$$(10 - 43) \equiv 44 \bmod 77, (-10 - 43) \equiv 24 \bmod 77$$
$$(32 - 43) \equiv 66 \bmod 77, (-32 - 43) \equiv 2 \bmod 77.$$

Decryption: Given N must find x such that $x(x+b) \equiv N \bmod n$. To do this "we complete squares".

$$\begin{aligned} x(x+b) &\equiv N \bmod n \\ x^2 + 2x\tfrac{b}{2} + \tfrac{b^2}{4} &\equiv N + \tfrac{b^2}{4} \bmod n \\ (x+\tfrac{b}{2})^2 &\equiv N + \tfrac{b^2}{4} \bmod n \end{aligned}$$

Given N, n, b compute $N + \frac{b^2}{2}$ which must a quadratic residue modulo n. As before we can compute all four square roots modulo n.

Secutity of Encryption: Is there an efficient algorithm A which given ciphertext N it outputs $x = A(N)$ such that $x(x+b) \equiv N \bmod n$?

Assuming such an algorithm A exists we give a Las Vegas algorithm for factoring n with probability $\geq 1/2$.

Idea: If $x^2 \equiv y^2 \bmod n$ then $n|(x-y)(x+y)$. Hence if $x \not\equiv \pm y \bmod n$ then $\gcd(x-y, n)$ is a nontrivial factor of n.

LAS VEGAS FACTORING ALGORITHM

Input: n, b

1. Choose a random $1 \leq r < n$
2. Compute $y \equiv (r^2 - b^2/4) \bmod n$
3. Compute $x = A(y)$
4. Compute $x_1 = x + b/2$

5a. **if** $x_1 \equiv \pm r \bmod n$ **then** **quit (failure)**

5b. **if** $x_1 \not\equiv \pm r \bmod n$ **then** **output** $\gcd(x_1 + r, n)$

Theorem: Pr[success] $\geq 1/2$.

Proof: Define an equivalence relation

$$r \approx s \Leftrightarrow r^2 \equiv s^2 \bmod n.$$

The equivalence classes consist of four elements $[y] = \{\pm y, \pm \omega y\}$, where ω is a nontrivial root of unity modulo n. Consider the value returned by the oracle A on input y, i.e., $x = A(y)$. If $r = \pm y$ then the algorithm fails while it succeeds if $r = \pm \omega y$. Since r is random the theorem is proved.

FACTORING ALGORITHMS

Problem: Factor a given n.

This is a very important problem. No efficient algorithm (i.e., running in time polylogarithmic in n) is known.

The 1996 challenge referred to an RSA challenge with a key length of 130 decimal digits. Implementation was done on the Internet.

Decimal Digits	*Year Achieved*	*MIPS Years*	*Algorithm*
100	1991	7	*Q Sieve*
110	1992	75	*Q Sieve*
120	1993	830	*Q Sieve*
130	1996	500	*Gen. Num. Field*

MIPS-Years is Millions of Instructions Per Second counted in Years, e.g. a Pentium 200 is a 50 MIPS machine.

Existing Factoring Algorithms

Sieve of Eratosthenes (oldest known factoring algorithm).
Number Field Sieve (NFS).
Special Number Field Sieve (SNFS).
General Number Field Sieve (GNFS). Is the fastest known factoring algorithm for numbers with 110 digits or more.

Quadratic Sieve (QS) is the fastest known algorithm for numbers less than 110 digits.

Elliptiv Curve Method (ECM) (works well for numbers with less than 40 digits).

Pollard's $p-1$.
Pollard's ρ.
Pollard's Monte Carlo Algorithm.

Continued Fraction Method (only of intellectual interest).

Experimental Running Times

Key length selection for RSA depends on intended security and expected key lifetime. E.g., if you want your keys to remain secure for 20 years a key 1,024 bits long is too short!

Table for factoring times in NFS and SNFS.

# of Bits	NFS-MIPS	SNFS-MIPS
512	$3 \cdot 10^4$	< 200
768	$2 \cdot 10^8$	$1 \cdot 10^5$
1024	$3 \cdot 10^{11}$	$3 \cdot 10^7$
1280	$1 \cdot 10^{14}$	$3 \cdot 10^9$
1536	$3 \cdot 10^{16}$	$2 \cdot 10^{11}$
2048	$3 \cdot 10^{20}$	$4 \cdot 10^{14}$

To be sure, certainly you can use very large keys, but remember your computation time will become unreasonable! Here are some predictions in bit lengths:

Year	Individual	Corporation	Government
2000	1024	1280	1538
2005	1280	1538	2048
2010	1280	1538	2048
2015	1538	2048	2048

Theoretical Running Times

Here is the "expected running times" of some algorithms used in practice:

$Quadratic\ Sieve$	$O(e^{(1+o(1)\sqrt{\ln n \ln\ln n}})$
$Elliptic\ Curve$	$O(e^{(1+o(1)\sqrt{2\ln p \ln\ln p}})$
$Num.\ Field\ Sieve$	$O(e^{(1.92+o(1))\ln^{1/3} n \ln\ln^{2/3} n})$

To avoid attacks by the Pollard $p-1$ method RSA uses $p = 2p_1 + 1, q = 2q_1 + 1$, with p_1, q_1 primes od roughly the same size.

For RSA factors of equal size the Quadratic sieve is the most successful.

The elliptic curve method is useful when the number has prime factors of differng size.

The Number Field Sieve is the most recent test.

POLLARD'S $p-1$ METHOD

Input: n,B

1. $a = 2$
2. **for** $j = 2$ **to** B **do** $a \equiv a^j \bmod n$
3. $d = \gcd(a-1, n)$
4. $1 < d < n$ **then** d is a factor of n
   else failure

Idea: Let p be a prime divisor of n. Let B be chosen so that "every power of a prime dividing $p-1$ is $\leq B$". It follows that $(p-1)|B!$. However, at the end of the for loop $a = 2^{B!} \equiv 1 \bmod n$. Since $p|n$ we also have $a = 2^{B!} \equiv 1 \bmod p$. It follows that $p|a-1$ and hence also $p|\gcd(a-1, n) = d$. Next we proceed to factor $d, n/d$, and so on.

Running Time: Each modular exponentiation requires $O(\log B)$ modular multiplications each taking time $O(\log^2 n)$. The gcd can be done in time $O(\log n)$. Hence complexity is $O(B \log B \log^2 n + \log^3 n)$.

Example

Disadvantage: Requires n to have a prime factor p such that $p-1$ has small factors.

Let $n = 15770708441$. We do not know any prime factor of n. We assume that there is one, say p, such that $p-1$ has small factors.

We guess that all prime factors of $p-1$ are $\leq B = 180$.

Iterate and in step 3 we compute the value $a = 11620221425$.

From this we compute $d = \gcd(a-1, n) = 135979$.

The complete factorization of n is found to be

$$n = 135979 \cdot 115979$$

If $p = 135979$ then we see that $p - 1 = 2 \cdot 3 \cdot 131 \cdot 173$. Hence our choice of $B = 180$ was correct. We could have picked something smaller, say $B = 173$!

DIXON'S ALGORITHM

Select a factor base $\mathcal{B} = \{p_1, p_2, \ldots, p_B\}$ of primes and a constant $C \geq B + 10$.

For $j \leq C$ consider the vector $a_j = (a_{1,j} \bmod 2, a_{2,j} \bmod 2, \ldots, a_{B,j} \bmod 2) \in (Z_2)^B$ so that a subset of them adds to $(0, 0, \ldots, 0)$ modulo 2. This amounts to finding a linear dependence among these vectors.

For $j \leq C$ suppose we have obtained congruences: $x_j^2 \equiv p_1^{a_{1,j}} p_2^{a_{2,j}} \cdots p_B^{a_{B,j}} \bmod n$

Multiplying these we get the congruence $x^2 \equiv y^2 \bmod n$ which can be used to factor n.

Finding vectors $a_j, j \leq C$ corresponds to finding a linear dependence oner Z_2 (which exists since $C > B$).

To generate integers x_j such that $x_j^2 \bmod n$ factors completely over the base $\mathcal{B}$ one uses the quadratic sieve algorithm.

Example of Dixon's Algorithm

Let $n = 15770708441$ and $\mathcal{B} = \{2, 3, 5, 7, 11, 13\}$. Take the congruences:

$$\begin{aligned} 8340934156^2 &\equiv 3 \cdot 7 \bmod n \\ 12044942944^2 &\equiv 2 \cdot 7 \cdot 13 \bmod n \\ 2773700011^2 &\equiv 2 \cdot 3 \cdot 13 \bmod n \end{aligned}$$

Multiply these out and reduce the product to obtain

$$9703435785^2 \equiv (2 \cdot 3 \cdot 7 \cdot 13)^2 \equiv 546 \bmod n$$

Hence we obtain the factor

$$\gcd(9703435785 - 546, 15770708441) = 115759$$

The three vectors are

$$\begin{aligned} a_1 &= (0, 1, 0, 1, 0, 0) \\ a_2 &= (1, 0, 0, 1, 0, 1) \\ a_3 &= (1, 1, 0, 0, 0, 1) \end{aligned}$$

and satisfy

$$a_1 + a_2 + a_3 = (0, 0, 0, 0, 0, 0)$$

CRYPTOSYSTEMS BASED ON GROUPS

A group is a pair $(G, \cdot)$ with a distinguished element e (identity) where G is a finite set and $\cdot : G \times G \to G$ is a binary operation such that

1. $\forall u, v, w(u \cdot (v \cdot w) = (u \cdot v) \cdot w)$
2. $\forall u \exists! v(u \cdot v = e)$

For each u the unique v such that $u \cdot v = e$ is called inverse of u and is denoted by u^{-1}.

Why are groups of interest to cryptography?

Starting with an element $u \in G$ we can iterate integer powers of u, namely $u, u^2, u^3, \ldots$ and generate new elements of G.

Given a "base" $u \in G$ and an element $v \in G$ which is an integer power of u, say u^k, can we compute k?

Examples

Example: In the group Z_{19}^* we can form the powers 7^i mod 19.

$$\begin{aligned} 7^0 &\equiv 1 \bmod 19 \\ 7^1 &\equiv 7 \bmod 19 \\ 7^2 &\equiv 11 \bmod 19 \\ 7^3 &\equiv 1 \bmod 19 \\ 7^4 &\equiv 7 \bmod 19 \\ 7^5 &\equiv 11 \bmod 19 \end{aligned}$$

Now given an element, say 11, can you compute the exponent 5 which satisfies $7^5 \equiv 11$ mod 19?

The bigger the prime involved the more complicated the problem!

Example: In the group Z_{2579}^*, the number 435 is a power of 2! Can you find the exponent?

Classes of Groups

- Z_n^* (set of integers relatively prime to n, modulo n).

- The multiplicative group of the Galois Field $GF(p^n)$. Most frequently used for $p = 2$.

- Elliptic curve groups over Z_p^*.

- Groups of symmetry (e.g., the set of symmetries of a regular polygon).

- Groups of permutations.

- Groups formed by words and relations.

El Gamal Cryptosystem

For any group $(G, \cdot)$ we define the following cryptosystem:

Cryptosystem ElGamal (G, α, β) :
Public: $(G, \cdot), \alpha, \beta \in G$
Private: Exponent $a \in Z$ s.t. $\alpha^a = \beta$
Encryption: To encrypt M choose some random integer k and encrypt $M \to E(M) = (\alpha^k, M\beta^k)$
Decryption: $(y_1, y_2) \to D(y_1, y_2) = y_2(y_1^a)^{-1}$

In the sequel we will relate the security of this cryptosystem to the difficulty of computing Discrete Logarithms in the group $(G, \cdot)$.

Let $\alpha \in G$.

Discrete Logarithm Problem(α): Given a $\beta \in < \alpha >$ find an integer s such that $\beta = \alpha^s$.

ElGamal is applicable to any cyclic subgroup $< \alpha >$ of $(G, \cdot)$ such that the discrete logarith problem is difficult on this subgroup.

ElGamal is usually applied to the multiplicative group of Z_p^*, for p a large prime.

Example of ElGamal$(2579, 2, 949)$:

Private: $a = 765$
Public: $p = 2579, \alpha = 2$
$\beta = \alpha^{765} \bmod 2579 = 949$

To send message $M = 1299$ we choose random $k = 853$ and compute

$$y_1 = 2^{853} \bmod 2579 = 435$$
$$y_2 = 1299 \cdot 949^{853} \bmod 2579 = 2396$$

Ciphertext is $(435, 2396)$. To decrypt we compute

$$2396 \cdot (435^{765})^{-1} \bmod 2579 = 1299$$

The Elliptic curve cryptosystem is essentially ElGamal on a specific class of groups called Elliptic groups.

REPRESENTATIONS OF GROUPS

How difficult is it to solve the Discrete Logarithm Problem?

Depends on the group representation!

Consider the groups $(Z_p^*, \cdot)$ and $(Z_{p-1}, +)$. Take a generator g of Z_p^* and define the mapping

$$(Z_{p-1}, +) \rightarrow (Z_p^*, \cdot) : x \rightarrow \phi(x) = g^x \bmod p.$$

Clearly,

1. $\phi(x + y \bmod p) = (\phi(x) \cdot \phi(y)) \bmod (p-1)$
2. $a\phi(\alpha) = \phi(\alpha^a \bmod p) \bmod (p-1)$
3. $\beta \equiv \alpha^a \bmod p \Leftrightarrow \phi(\beta) \equiv a\phi(\alpha) \bmod (p-1)$

It is easy to solve linear congruences! Does this mean that it is easy to solve the discrete logarithm problem?

No! Finding the isomorphism means we must find the generator! But there are too many possible generators!

DISCRETE LOGS in Z_{11}^*

x	2^x	3^x	5^x	7^x
1	2	3	5	7
2	4	9	3	5
3	8	5	4	2
4	5	4	9	3
5	10	1	1	10
6	9	3	5	4
7	7	9	4	6
8	3	5	9	9
9	6	4	1	8
10	1	1	5	1
β	$\log_\alpha$	$\log_3$	$\log_5$	$\log_7$
1	0			0
2	1			3
3	8			4
4	2			6
5	4			2
6	9			7
7	7			1
8	3			9
9	6			8
10	5			5

Discrete Log Attacks

The following attacks will be considered, where

$$p - 1 = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

Method	Time (# of Operations)
Exhaustive Search	$O(p)$
Schank's Algorithm	$O(\sqrt{p})$
Pohlig-Hellman	$O(\sum_{i=1}^{k} e_i(\log(p-1)\sqrt{p_i}))$
Index Calculus	$O(e^{1/2+o(1)\sqrt{\ln p \ln \ln p}})$

Thus, the Pohlig-Hellman algorithm requires the factorization of $p - 1$.

A General Method

Consider a permutation $f : X \to X$ of an n-element set X without fixed points, i.e. $f(x) \neq x$, for all x. **Example:** $f : Z_p^* \to Z_p^* : u \to \beta \cdot u$.

Let the orbit of $x \in X$ be defined by $Orb_f(x) = \{x, f(x), f^2(x), f^3(x), \ldots, f^n(x)\}$. Given that $y \in Orb_f(x)$ find an integer k such that $y = f^k(x)$.

Put $m = \sqrt{n}$. By Euclidean division, every integer $a < n$ is of the form $a = qm + r$, where $0 \leq q, r < n$. Hence,

$$\begin{aligned} f^a(x) &= f^{qm+r}(x) = f^{qm}(f^r(x)) \\ &= f^m \circ \cdots \circ f^m(f^r(x) \cdots) \end{aligned}$$

Compute $(r, f^{-r}(y))$, for $r \leq m$, and sort by second coordinate. Also, compute $(q, f^{qm}(x))$, for $q \leq m$, and sort by second coordinate.

Find two pairs $(q, f^{qm}), (r, f^{-r}(y)$ with identical second coordinates, i.e., $f^{-r}(y) = f^{qm}(x)$. It follows that $y = f^{qm+r}(x)$.
Search time is $O(\sqrt{n})$.

DISCRETE LOGS IN Z_p^*

Problem Instance: p prime, α generator of Z_p^*, and β an element of Z_p^*.

Objective: Find unique $k \leq p-1$ such that $\beta = \alpha^k \bmod p$ (we denote this by $k := \log_\alpha \beta$).

One way to do this is to generate all powers of β:

$$\beta, \beta^2, \beta^3, \ldots$$

and find the correct exponent by exhaustive search.
This takes time $O(p)$.

Another way is to use the previous general method in order to reduce the search time to $O(\sqrt{p})$. This is known as Schank's algorithm.

SCHANK'S ALGORITHM

Schank's Algorithm(p, α, β): If $m = \lfloor \sqrt{p-1} \rfloor$ then notice that every integer $a < p - 1$ is of the form $a = qm + r$, where $0 \leq q, r \leq m - 1$. Hence $\alpha^a \equiv (\alpha^m)^q \alpha^r \bmod p$.

1. Compute $(r, \beta\alpha^{-r} \bmod p)$, for $r = 0, 1, \ldots, m-1$ and sort them by the second coordinate.

2. Compute $(q, \alpha^{mq} \bmod p)$, for $q = 0, 1, \ldots, m-1$ and sort them by second coordinate.

3. Find two pairs with identical second coordinates, i.e., $(r, \beta\alpha^{-r} \bmod p)$, and $(q, \alpha^{mq} \bmod p)$ such that $\beta\alpha^{-r} \equiv \alpha^{mq} \bmod p$.

Running time of the algorithm is $O(\sqrt{p})$.

Example: Let $p = 809$ and we want to compute $\log_3(525)$. We have $\alpha = 3, \beta = 525, m = \lfloor\sqrt{808}\rfloor = 29$ and $\alpha^{29} \bmod 809 = 99$. We then tabulate $(i, 99^i \bmod 809)$ and $(i, 525 \cdot (3^i)^{-1} \bmod 809)$, for $i = 0, \ldots, 28$.

$(0,1)$	$(1,99)$	$(2,93)$	$(3,308)$
$(4,559)$	$(5,329)$	$(6,211)$	$(7,664)$
$(8,207)$	$(9,268)$	$(10,644)$	$(11,654)$
$(12,26)$	$(13,147)$	$(14,800)$	$(15,727)$
$(16,781)$	$(17,464)$	$(18,632)$	$(19,276)$
$(20,528)$	$(21,496)$	$(22,564)$	$(23,15)$
$(24,676)$	$(25,586)$	$(26,575)$	$(27,295)$
$(28,81)$			

$(0,525)$	$(1,175)$	$(2,328)$	$(3,379)$
$(4,396)$	$(5,132)$	$(6,44)$	$(7,554)$
$(8,724)$	$(9,511)$	$(10,440)$	$(11,686)$
$(12,768)$	$(13,256)$	$(14,355)$	$(15,388)$
$(16,399)$	$(17,133)$	$(18,314)$	$(19,644)$
$(20,754)$	$(21,521)$	$(22,713)$	$(23,777)$
$(24,259)$	$(25,356)$	$(26,658)$	$(27,489)$
$(28,163)$			

$(10,644)$ and $(19,644)$ have same second coordinate. Hence, $\log_3 525 = 29 \cdot 10 + 19 = 309$.

POHLIG-HELLMAN(p, α, β)

Let $p-1 = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. If $\beta = \alpha^a \bmod p$ then $a = \log_\alpha \beta$ is uniquely determined modulo $p-1$. So enough to compute $a \bmod p_i^{e_i}$, $i = 1, \ldots, k$ and use the Chinese Remainder Theorem.

Assume q is prime s.t. $p \equiv 1 \bmod q^c$ and $p \not\equiv 1 \bmod q^{c+1}$. We want to compute $x = a \bmod q^c = \log_\alpha \beta \bmod q^c$. Express $x = \sum_{i=1}^{c-1} \alpha_i q^i$, where $0 \leq a_i \leq q-1$. Write $a = x + sq^c$. We

Claim: $\beta^{\frac{p-1}{q}} \equiv \alpha^{\frac{\alpha_0(p-1)}{q}} \bmod p$.

Proof: $\beta^{\frac{p-1}{q}} \equiv \alpha^{a\frac{p-1}{q}} \equiv \alpha^{(x+sq^c)\frac{p-1}{q}} \bmod p$. Hence enough to show $\alpha^{(x+sq^c)\frac{p-1}{q}} \equiv \alpha^{\alpha_0 \frac{p-1}{q}} \bmod p$. This is equivalent to proving $(x+sq^c)\frac{p-1}{q} \equiv \alpha_0 \frac{p-1}{q} \bmod (p-1)$, i.e.,

$$\begin{aligned}
\tfrac{p-1}{q}(x + sq^c - \alpha_0) &= \tfrac{p-1}{q}(sq^c - \alpha_0 + \textstyle\sum_{i=0}^{c-1} \alpha_i q^i) \\
&= \tfrac{p-1}{q}(sq^c + \textstyle\sum_{i=1}^{c-1} \alpha_i q^i) \\
&= (p-1)(sq^c + \textstyle\sum_{i=1}^{c-1} \alpha_i q^i) \\
&\equiv 0 \bmod (p-1)
\end{aligned}$$

Now, compute $\beta^{\frac{p-1}{q}}$. If $\beta^{\frac{p-1}{q}} \equiv 1 \bmod p$ then $\alpha_0 = 0$. If not then compute

$$\alpha^{\frac{p-1}{q}} \bmod p, \alpha^{\frac{p-1}{q}2} \bmod p, \ldots, \alpha^{\frac{p-1}{q}i} \bmod p, \ldots$$

until $\beta^{\frac{p-1}{q}} \equiv \alpha^{\frac{p-1}{q}i} \bmod p$. It follows that $\alpha_0 = i$. If $c = 1$ we are done.

Else $c > 1$. Define $\beta_1 = \beta\alpha^{-\alpha_0}$ and let $x_1 = \log_\alpha \beta_1 \bmod q^c$. It follows $x_1 = \sum_{i=1}^{c-1} \alpha_i q^i$ and $\beta_1^{\frac{p-1}{q^2}} \equiv \alpha^{\frac{(p-1)\alpha_0}{q^2}} \bmod p$. We can compute α_1 as before. If $c = 2$ we are done, else $c > 2$ and continue as before.

1. compute $\gamma_i = \alpha^{\frac{(p-1)i}{q}} \bmod p$, for $i < q$.
2. set $j = 0$ and $\beta_j = \beta$
   while $j \leq c - 1$ **do**
   compute $\delta = \beta^{\frac{p-1}{q^{j+1}}} \bmod p$
   find i s.t. $\delta = \gamma_i$
   $\alpha_J := i$
   $\beta_{j+1} = \beta_j \alpha^{-\alpha_j q^j} \bmod p$
   $j = j + 1$

Example of Pohlig-Hellman$(29, 2, 18)$:

$p-1 = 28 = 2^2 \cdot 7, \alpha = 2, \beta = 18$; want to compute the discrete logarithm $\log_2 18$. The exponent a such that $2^a \equiv 18 \bmod 29$ is unknown. Compute separately $a \bmod 2^2$ and $a \bmod 7$.

$$
\begin{array}{ll}
q = 2, c = 2: & \alpha^{28/2} = 2^{14} = 28 \bmod 29 \\
& \beta^{28/2} = 18^{14} = 28 \bmod 29 \\
& \text{Hence } \alpha_0 = 1 \\
q = 2^2, c = 2: & \beta_1 = \beta_0 \alpha^{-1} = 9 \bmod 29 \\
& \beta_1^{28/4} = 9^7 = 28 \bmod 29 \\
& \text{Hence } \alpha_1 = 1 \text{ and } a = 3 \\
q = 7, c = 1: & \alpha^{28/4} = 2^4 = 16 \bmod 29 \\
& \beta^{28/4} = 18^4 = 25 \bmod 29 \\
& \text{Hence } \alpha_0 = 4 \text{ and } a = 4
\end{array}
$$

Use the Chinese Remainder Theorem to solve the system $x \equiv 3 \bmod 4, x \equiv 4 \bmod 7$ to get $x = 11$. Hence, $\log_2 18 = 11$.

INDEX CALCULUS METHOD

Method uses a factor base $\mathcal{B} = \{p_1, p_2, \ldots, p_B\}$.

Step 1: Compute the logarithms of the B primes in the base $\mathcal{B}$, i.e. $\log_\alpha p_1, \ldots, \log_\alpha p_B$.

Explanation of Step 1: Take $C = B+10$ congruences $\alpha^{x_j} \equiv p_1^{\alpha_{1,j}} \cdots p_B^{\alpha_{B,j}} \bmod p$, $j \leq C$, i.e., $x_j \equiv \alpha_{1,j} \log_\alpha p_1 + \cdots + \alpha_{B,j} \log_\alpha p_B \bmod (p-1)$. This gives a system of C linear congruences in the unknowns $\log_\alpha p_i$, $i \leq C$.

Step 2: Compute $\log_\alpha \beta$ using the knowledge from Step 1.

Explanation of Step 2: This is done with a Las Vegas algorithm. Assume we were successful in Step 1 and we know the values

$$\log_\alpha p_1, \ldots, \log_\alpha p_B$$

Choose a random $1 \leq s \leq p-2$.

Compute $\beta\alpha^s \bmod p$.

Factor $\beta\alpha^s \bmod p$ in the given factor base $\mathcal{B} = \{p_1, p_2, \ldots, p_B\}$, say

$$\beta\alpha^s \equiv p_1^{c_1} p_2^{c_2} \cdots p_B^{c_B} \bmod p$$

It follows that

$$\log_\alpha \beta = -s + c_1 \log_\alpha p_1 + \cdots + c_B \log_\alpha p_B.$$

Analysis shows that precomputation has asymptotic running time

$$O(e^{(1+o(1))\sqrt{\ln p \ln\ln p}})$$

and the time to find the discrete log is

$$O(e^{(1/2+o(1))\sqrt{\ln p \ln\ln p}})$$

EXAMPLE

$p = 10007, \alpha = 5, \beta = 9451, \mathcal{B} = \{2, 3, 5, 7\}$. First we compute $\log_5 2, \log_5 3, \log_5 5, \log_5 7$. Try to factor α^x for $x = 4063, 5136, 9865$.

$$\begin{aligned} 5^{4063} &\equiv 42 = 2 \cdot 3 \cdot 7 \mod 10007 \\ 5^{5136} &\equiv 54 = 2 \cdot 3^3 \mod 10007 \\ 5^{9865} &\equiv 189 = 3^3 \cdot 7 \mod 10007 \end{aligned}$$

which give the linear congruences

$$\begin{aligned} \log_5 2 + \log_5 3 + \log_5 7 &\equiv 4063 \bmod 10006 \\ \log_5 2 + 3\log_5 3 &\equiv 5136 \bmod 10006 \\ 3\log_5 3 + \log_5 7 &\equiv 9865 \bmod 10006 \end{aligned}$$

with unique solution $\log_5 2 = 6578, \log_5 3 = 6190, \log_5 7 = 1301$. To find $\log_5 9451$ we choose a random exponent $s = 7736$ and compute $9451 \cdot 5^{7736} \equiv 8400 \bmod 10007$. Over $\mathcal{B}$ this factors to $2^4 3^1 5^2 7^1$. We derive $\log_5 9451 \equiv -7736 + 4\log_5 2 + \log_5 3 + 2\log_5 5 + \log_5 7 \equiv 4 \cdot 6678 + 6190 + 2 \cdot 1 + 1301 \equiv 6057$.

DISCRETE LOGARITHM BITS

Let α be a generator of Z_p^*.

Least Significant Bit LSB (β):

$$\begin{aligned} \textbf{LSB}(\beta) = 0 \quad &\Leftrightarrow \log_\alpha \beta \text{ is even} \\ &\Leftrightarrow \beta \in QR_p \\ &\Leftrightarrow \beta^{\frac{p-1}{2}} \equiv 1 \bmod p \end{aligned}$$

Thus the least significant bit is easy by Euler's test.

Other Bits: Write $p - 1 = 2^s t$, with t odd.

Theorem 1: It is easy to compute the i-th least significant bit for $i \leq s$.

Theorem 2: If there is an "efficient" algorithm for computing the $s + 1$-st least significant bit then it can be used to compute efficiently $\log_\alpha \beta$.

Proof of Theorem 2 for $s = 1$

Assume there is an "efficient" algorithm for computing the 2nd least significant bit.

$p - 1 = 2t$, where t is odd. Hence, $p - 3 = 2(t - 1)$ which implies $p \equiv 3 \bmod 4$. In this case, if β is a quadratic residue mod p then $\pm\beta^{\frac{p+1}{4}}$ are the two square roots of β mod p.

Claim: For any γ, $\log_\alpha \gamma$ is even $\Leftrightarrow$ $\log_\alpha(p-\gamma)$ is odd.

Proof: ($\Rightarrow$) Assume $\log_\alpha \gamma$ is even. Since α is a generator of Z_p^*, $\alpha^{\frac{p-1}{2}} \not\equiv \bmod p$. By Fermat's theorem, $\alpha^{\frac{p-1}{2}} \equiv -1 \bmod p$. Assume $\gamma = \alpha^a \bmod p$, for some a even. Hence,

$$p - \gamma \equiv -\gamma \equiv \alpha^{\frac{p-1}{2}}\alpha^a \equiv \alpha^{\frac{p-1}{2}+a} \bmod p,$$

which implies $\log_\alpha(p-\gamma) \equiv \frac{p-1}{2}+a \bmod (p-1)$, i.e., $\log_\alpha(p-\gamma)$ is odd. Reverse the steps to prove ($\Leftarrow$).

Let the input be α, β, p. We want to output $\log_\alpha \beta$.

Let $L_2(\gamma) =$ the 2-nd least significant bit of $\log_\alpha \gamma$.

If $L_1(\beta)$ is odd then replace $\beta := \beta\alpha^{-1} \bmod p$. This new β satisfies $L_1(\beta) = 0$.

Let $\beta \equiv \alpha^a \bmod p$, for some even a. Then

$$\sqrt{\beta} \equiv \beta^{\frac{p+1}{4}} \equiv \pm\alpha^{a/2} \bmod p.$$

By assumption, the value $L_2(\beta)$ can be computed efficiently and moreover, $L_2(\beta) = L_1(\alpha^{a/2})$. Hence, $L_1(\alpha^{a/2})$ can also be computed efficiently. Hence we can determine which of the two possibilities $\pm$ is correct.

If $L_2(\beta) = L_1(\beta^{\frac{p+1}{4}})$ **then** set $\beta := \beta^{\frac{p+1}{4}}$, **else** set $\beta = p - \beta^{\frac{p-1}{4}}$. Next set $\beta := \beta \cdot \alpha^{-L_2(\beta)}$ and iterate.

GALOIS FIELDS (used in ElGamal)

$(Z_p, +, \cdot)$ is a Galois Field for p prime. It has p elements.

Let $Z_p[x]$ be the polynomials in the variable x and coefficients over Z_p. We can add and multiply such polynomials. So $(Z_p[x], +, \cdot)$ becomes a ring.

We also define an equivalence relation $g(x) \equiv h(x) \bmod f(x)$ iff $f(x)|g(x) - h(x)$. The set of these equivalence classes becomes a ring, denoted by $\frac{Z_p[x]}{(f(x))}$.

If the polynomial $f(x)$ is **irreducible**, i.e., it has no nontrivial polynomial divisors, then $\frac{Z_p[x]}{(f(x))}$ is in fact a finite field.

If $f(x)$ is an irreducible polynomial of degree n then $\frac{Z_p[x]}{(f(x))}$ is the Galois field of size p^n.

How do we construct irreducible polynomials?

EXAMPLE

Look at $Z_2[x]$. There are four polynomials of degree 3 and constant term equal to 1. $x^3+1, x^3+x+1, x^3+x^2+1, x^3+x^2+x+1$ Of these, $x^3+1=(x+1)(x^3+x+1)$ and $x^3+x^2+x+1=(x+1)(x^2+1)$. But the other two are irreducible. This gives rise to the fields $\frac{Z_p[x]}{(x^3+x+1)}$ and $\frac{Z_p[x]}{(x^3+x^2+1)}$ They are both identical with 2^3 elements. Look at x^3+x+1.

	001	010	011	100	101	110	111
001	001	010	011	100	101	110	111
010	010	100	110	011	001	111	101
011	011	110	101	111	100	001	010
100	100	011	111	110	010	101	001
101	101	001	100	010	111	011	110
110	110	111	011	101	011	010	100
111	111	101	010	001	110	100	011

To obtain its elements we divide by x^3+x+1 and keep the remainder which is a degree two polynmial) (represented by its coefficients).

PROPERTIES OF FIELDS

There is an irreducible polynomial in $Z_p[x]$ of degree n, for each $n \geq 1$.

Hence, there is a (unique) Galois field with p^n elements, denoted by $GF(p^n)$, for each $n \geq 1$. Of course, $GF(p)$ is the same as Z_p.

The multiplicative group of $GF(p^n)$ is cyclic and has $p^n - 1$ elements.

$GF(2^n)$ is the most studied field. Previous algorithms (e.g., Shanks and Pohlig-Hellman) can be modified to work here. The discrete logarithm is considered intractable for large n provided that $2^n - 1$ has a large prime factor.

Hardware implementation of $GF(2^{127})$ exist, where $p(x) = x^{127} + x + 1$.

KNAPSACK SYSTEMS

Cryptosystems based on NP-hard problems are not always secure. We give two such examples: the first insecure and the second secure.

Based on **Subset Sum** problem:

Instance: $s_1, s_2, \ldots, s_n, T$ are positive integers.

Question: Is there a $0-1$ vector $(x_1, x_2, \ldots, x_n)$ such that $\sum_{i=1}^{n} x_i s_i = T$?

Subset sum problem is easy for superincreasing instances, i.e. $\sum_{i<j} s_i < s_j$, for all j.

for $i = n$ **downto** 1 **do**
    if $T \geq s_i$ **then** $T = T - s_i, x_i = 1$
    else $x_i = 0$
if $\sum_{i=1}^{n} x_i s_i = T$ **then**
    $x_1, x_2, \ldots, x_n$ is the solution
else there is no solution

The above representation of T is unique if the given sequence $s_1, \ldots, s_n$ is superincreasing.

MERKLE-HELLMAN KNAPSACK

For $s = (s_1, s_2, \ldots, s_n)$ consider the encryption function $E_s : \{0,1\}^n \to [0, \sum_{i=1}^n s_i]$:

$$x = (x_1, x_2, \ldots, x_n) \to E_s(x) = \sum_{i=1}^{n} x_i s_i$$

Note that E_s is $1-1$ and is also easy to decrypt if s is superincreasing. **Strategy:** Permute the s_i's so that the sequence is no longer superincreasing. Let p be a prime $> \sum_{i=1}^n s_i$.

	Knapsack Cryptosystem:
s	superincreasing
f	one-way linear transformation
	e.g., $x \to ax$ mod p
Public :	$(f(s_1), \ldots, f(s_n))$
Private :	s and trapdoor f
Encryption :	$E_{(f(s_1),\ldots,f(s_n))}$
Decryption :	Given N solve Subset Sum problem for $f^{-1}(N)$

This cryptosystem has only "historical interest" since it was already broken in the 1980s. However, there might be other one-way functions that may make it secure!

Example

The input sequence

$$2, 5, 9, 21, 45, 103, 215, 450, 946$$

is easily checked to be superincreasing. The transformation f is defined as follows: Choose a modulus $p = 2003$ and an integer $a = 1289$. f is the one-way linear transformation

$$x \to (a \cdot x) \bmod p : x \to (1289 \cdot x) \bmod 2003.$$

The integer a is the trapdoor. This gives rise to the following public key

$$575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570$$

The plaintext 101100111 is encrypted as

$$575+1586+1030+721+1183+1570 = 6665$$

To recover the plaintext we compute

$$a^{-1}y \equiv 317 \cdot 6665 \equiv 1643 \bmod 2003.$$

From this we obtain the plaintext 101100111.

CODES

An $[n,k]$-**code** is a k-dimensional subspace of $(Z_2)^n$. A **generating matrix** G for an $[n,k]$-code is a $k \times n$ matrix whose rows form a basis for C. $d(x,y)$ is the **Hamming Distance** of x,y. $d(C) = \min_{x,y \in C, x \neq y} d(x,y)$. An $[n,k,d]$-code is an $[n,k]$-code C s.t. $d(C) = d$.

Correcting Errors: Let G be the generating matrix of an $[n,k,d]$-code C.

- A **Transmits** $x \in (Z_2)^k$: Encode $y = xG \in (Z_2)^n$, and transmit y through channel.

- B **Receives** $r \in (Z_2)^n$: B finds $y' \in C$ s.t. $d(y',r) = \min_{u \in C} d(u,r)$ (nearest neighbor decoding) and outputs $x' = y'G$.

Claim: If #(errors) in r is $\leq \frac{d-1}{2}$ then $y' = y$ and hence $x' = x$. **Proof:** $d(y',y) \leq d(y',r) + d(r,y) \leq 2d(r,y) \leq 2\frac{d-1}{2} = d - 1 < d$. Hence, $d(y',y) = 0$.

Nearest-Neighbor Decoding

Syndrome: Consider the **parity-check** matrix H for C, i.e. an $(n-k)\times n$, $0-1$ matrix which forms a basis for $C^{\perp}$ (orthogonal complement for C). The **syndrome** of r is the vector $Hr^{\top}$. Note for $x \in (Z_2)^n$ we have that: (1) $x \in C \Leftrightarrow Hx^{\top} = 0$, and (2) if $r = x + e$, where $x \in C$ and $e \in (Z_2)^n$, then $Hr^{\top} = He^{\top}$. Hence, syndrome depends only on the errors and note on the particular word transmitted.

Syndrome Decoding: Compute $s = Hx^{\top}$. If $s = 0$ then no errors and decode r as r. If $s \neq 0$ generate all possible error vectors e of weight 1 and for each such e compute $He^{\top}$. If it happens that for any of these $He^{\top} = s$ then decode r as $r - e$. Otherwise continue to generate all vectors of weight $2, 3, \ldots, (d-1)/2$. Thus we decode in at most

$$1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{(d-1)/2}.$$

Note that if B compares r to all code words, time 2^k will be needed, since $|C| = 2^k$.

McELIECE SYSTEM

G generating matrix of $[n, k, d]$ Goppa code C (this is class of codes for which decoding is easy); $n = 2^m, d = 2t + 1, k = n - mt$. S is a $k \times k$ invertible matrix, P an $n \times n$ permutation matrix and $G' = SGP$.

	McEliece System
Public :	G'
Private :	G, S, P
Encryption :	$x \to xG' + e$, for some random vector of weight t
Decryption :	$y \to D(y)$
	Compute: $y_1 = yP^{-1}$
	Decode: y_1 as $y_1 = x_1 + e_1$ where $x_1 \in C$
	Compute: x_0 s.t. $x_0 G = x_1$
	Compute: $x = x_0 S^{-1}$
Correctness: $xG' + e$	$= x_0 S^{-1} SGP + e$
	$= x_0 GP + e$
	$= x_1 P + e$
	$= y_1 P - e_1 P + e$
	$= y - e_1 P + e$
	$= y$

A Hamming [7,4,3]-Code

The matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

generates a $[7, 4, 3]$ code. Choose matrices

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}, P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The public generating matrix is

$$G' = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The plaintext $x = (1, 1, 0, 0)$ is encrypted as $y = xG' + e = (0, 1, 1, 0, 1, 1, 0)$, where e is the error vector $(0, 0, 0, 0, 1, 0, 0)$.