Introduction to Cryptography

Franz Lemmermeyer

December 15, 2006

Contents

Chapter 1

Naive Cryptography

1.1 Cryptograms

Cryptograms, like Sudoku these days, are puzzles that can be solved using a few basic techniques. In a typical cryptogram, a plain text has been encrypted by replacing each letter by a (usually different) letter.

The basic strategy are best explained by an example:

```
CAKWQAHZ, OKARZ, IMR SAILLIA EMGPQJYEKMZ JKMWQU YFQ

CXHPEJ ZQMZQ OEYF LKAQ CXAEYU IMR CAQJEZEKM, YFIM YFQ

OEZQZY EMREWERXIP.

AIPCF OIPRK QLQAZKM
```

We observe that the word `YFQ` occurs twice. The most common three-letter word in English is `THE` (the runner-up is AND). If `YFQ` corresponds to `THE`, then the combination `YF` should occur often, since `TH` is a very common two-letter combination. This is indeed the case, so we assume that `Y`, `F` and `Q` correspond to `T`, `H` and `E`, respectively.

Next we look at the word `OEYF`; the only four-letter word ending with `TH` we can think of is `WITH`, so we guess that `O` and `E` correspond to `W` and `I`. The word `OEZQZY` now reads `WI*E*T`, suggesting that `Z` is an `S`. Now `ZQMZQ` clearly must be `SENSE`, hence `M` is an `N`.

Now the word `CAQJEZEKM` ends with `ISI*N`, so `K` must be an `O`. Next `YFIM` is `TH*N`, and since `I` cannot be an `E`, it must be an `A`. Similarly, `IMR` can only mean `AND`, and then `OKARZ` must be `WORDS`. Now it is getting easy: `EMREWERXIP` is `INDI*ID*A*`, that is, `INDIVIDUAL`.

After a few more steps, it is no problem at all to complete the decryption, and we end up with the following plaintext:

```
CAKWQAHZ, OKARZ, IMR SAILLIA EMGPQJYEKMZ JKMWQU YFQ CXHPEJ ZQMZQ
PROVERBS, WORDS, AND GRAMMAR INFLECTIONS CONVEY THE PUBLIC SENSE

OEYF LKAQ CXAEYU IMR CAQJEZEKM, YFIM YFQ OEZQZY EMREWERXIP.
WITH MORE PURITY AND PRECISION  THAN THE WISEST INDIVIDUAL.

AIPCF OIPRK QLQAZKM
RALPH WALDO EMERSON
```

For getting more practice in solving cryptograms, go to
`http://www.geocities.com/cryptogramcorner/`

1.2 The Gold Bug

Of course it helps a lot that we are given the empty spaces between the words as well as the interpunctuation. But even without this help, searching for three-letter combinations occurring frequently (`THE`), or simply for the most common letter (which, in English, usually stands for `E`), already gives us clues.

Edgar Allen Poe has given us a nice example in his short story *Gold Bug* (available online; just use google). In this story, the following encrypted text needs to be deciphered:

```
53++!305))6*;4826)4+.)4+);806*;48!8‘60))85;]8*:+*8!83(88)5*!;
46(;88*96*?;8)*+(;485);5*!2:*+(;4956*2(5*-4)8‘8*; 4069285);)6
!8)4++;1(+9;48081;8:8+1;48!85;4)485!528806*81(+9;48;(88;4(+?3
4;48)4+;161;:188;+?;
```

The most common symbol is `8`; if this stands for `E`, then there should be triples occurring frequently ending in `8`; one such triple is `;48`. If you want to find out how Poe's character decrypts the whole text, go read the story for yourself.

1.3 Frequency Analysis

Consider the cryptogram

```
LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFEMVEWHKVSTYLXZIXLIKII

XPIJVSZEYPERRGERIMWQLMGLMXQERIWGPSRIHMXQEREKIETXMJTPRGEVEKEIT

REWHEXXLEXXMZITWAWSQWXSWEXTVEPMRXRSJGSTVRIEYVIEXCVMUIMWERGMIW

XMJMGCSMWXSJOMIQXLIVIQIVIXQSVSTWHKPEGARCSXRWIEVSWIIBXVIZMXFSJ

XLIKEGAEWHEPSWYSWIWIEVXLISXLIVXLIRGEPIRQIVIIBGIIHMWYPFLEVHEWH

YPSRRFQMXLEPPXLIECCIEVEWGISJKTVWMRLIHYSPHXLIQIMYLXSJXLIMWRIGX

QEROIVFVIZEVAEKPIEWHXEAMWYEPPXLMWYRMWXSGSWRMHIVEXMSWMGSTPHLEV

HPFKPEZINTCMXIVJSVLMRSCMWMSWVIRCIGXMWYMX
```

A frequency analysis shows that the most frequent letter is `I`, the most frequent two-letter combination is `XL`, and the most frequent triple `XLI`. Use this information to decrypt the cipher text.

The point of these examples is to show that simple substitution ciphers can be broken easily using tools such as frequency analysis. This has led to the invention of much more sophisticated techniques for encrypting messages; one major weakness of all these systems up to the 1970s was the fact that sender and receiver both need to agree on a certain method of encrypting and decrypting messages; if knowledge about the encryption method is leaked, attackers usually can break the system easily.

For online communications between, say, a bank and its customers, these classical methods are useless. The fact that online banking is (more or less) secure these days was made possible by the invention of public-key cryptography. Before we can describe how it works, we have to review some basic elementary number theory.

Chapter 2

RSA

In this chapter we will first describe the basic idea behind RSA, and then discuss the resulting problems.

2.1 Background in Number Theory

The basic ingredient of RSA is Fermat's Little Theorem, according to which $a^{p-1} \equiv 1 \bmod p$ for all primes p and all integers a not divisible by p. Actually we need a special case of the more general Theorem of Euler-Fermat ($a^{\phi(m)} \equiv 1 \bmod m$ whenever $\gcd(a,m) = 1$, applied to $m = pq$): Let p and q be distinct odd primes. Then for any integer a coprime to pq, we have $a^{(p-1)(q-1)} \equiv 1 \bmod pq$. In fact, applying Fermat's Little Theorem twice we find

$$\begin{aligned} a^{(p-1)(q-1)} &= \big(a^{q-1}\big)^{p-1} \equiv 1 \bmod p, \\ a^{(p-1)(q-1)} &= \big(a^{p-1}\big)^{q-1} \equiv 1 \bmod q. \end{aligned}$$

Thus $p \mid (a^{(p-1)(q-1)} - 1)$ and $q \mid (a^{(p-1)(q-1)} - 1)$; since p and q are coprime, it follows from unique factorization that $pq \mid (a^{(p-1)(q-1)} - 1)$, and this implies the claim.

We will also use the Chinese Remainder Theorem. In its simplest form it claims that a system of congruences

$$\begin{aligned} x &\equiv a \bmod m \\ y &\equiv b \bmod n \end{aligned}$$

can always be solved if m and n are coprime. If, more generally, $\gcd(m,n) = d$, then the system has a solution if and only if $a \equiv b \bmod d$ (which is obviously necessary).

2.2 The Basic Idea

Now assume that Alice wants others to send her messages that cannot be read by anyone who's listening in. She picks two large prime numbers p and q (in practice, these should have about 150 digits each); she computes the product N and chooses an integer $1 < e < (p-1)(q-1)$ coprime to $(p-1)(q-1)$. Then she publishes the pair (N, e), her "public key" (for example by listing it on her homepage).

How does the encryption work? It is a simple matter to transform any text into a sequence of numbers, for example by using $a \longmapsto 01$, $b \to 02$, $\ldots$, with a couple of extra numbers for blanks, commas, etc. We may therefore assume that our message is a sequence of integers $m < n$ (if the text is longer, break it up into smaller pieces). Bob encrypts each integer m as $c \equiv m^e \bmod N$ and sends the sequence of c's to Alice (by email, say). Now Alice can decrypt the message as follows: since she knows p and q, she run the Euclidean algorithm on the pair $(e, (p-1)(q-1))$ to find integers $d, x > 0$ such that $de - x(p-1)(q-1) = 1$ (Bezout at work again). Now she takes the message c and computes $c^d \bmod N$. The result is $c^d \equiv (m^e)^d = m^{de} = m^{1+x(p-1)(q-1)} = m \cdot (m^{(p-1)(q-1)})^x \equiv m \bmod N$: thus Alice can compute the original text that Bob sent her.

Now assume that Eve is eavesdropping. Of course she knows the pair (N, e) (which is public anyway), and she also knows the message c that Bob sent to Alice. This does not suffice for decrypting the message, however, since one seems to need an inverse d of $e \bmod (p-1)(q-1)$ to do that; it is likely that one needs to know the factors of N in order to compute d.

Alice	Eve	Bob
picks two large primes p, q		
computes $N = pq$		
picks random $e \in (\mathbb{Z}/\phi(N)\mathbb{Z})^\times$		
publishes public key (N, e)	$\xrightarrow{(N,e)}$	
		computes $c \equiv m^e \bmod N$
	$\xleftarrow{c}$	sends c to Alice
solves $de \equiv 1 \bmod (p-1)(q-1)$		
computes $m \equiv c^d \bmod N$		

Figure 2.1: The RSA protocol

If Bob and Alice want to exchange messages, each of them has to pick their own key (so Alice picks primes p_A, q_A, and Bob p_B and q_B etc.)

2.3 An Example

Here is a worked example of RSA-encryption and decryption using `pari`, which you can get from

$$\texttt{http : //pari.math.u - bordeaux.fr/download.html}$$

as `Pari-2-3-0.exe`.

We first need to pick two primes; for this example, we choose them quite small, say less than 2^{27}. The command

```
random(2^27)
```

gives you a random number between 1 and 2^{27}. The command

```
p = nextprime(random(2^27))
```

produces the next prime after this random number. In this way I have produced two random primes

$$p = 52431073, \quad q = 59140507.$$

Multiplying them (type in `N = p*q`) gives

$$N = 3100800239774011.$$

Next we have to pick a random e coprime to `phi = (p-1)*(q-1)`. In order to increase the probability that e is coprime to ϕ, we compute

```
e=random(phi/6)*6+1
```

This gives me $e = 2352184338477589$, and with `gcd(e,phi)` I check that this gcd is indeed trivial (if it is not, I would simply pick another e).

Now we can "publish" our public key (N, e).

If Fermat wants to send me the message

```
It is impossible to separate any power higher than the
second into two like powers
```

then he first has to translate this into a string of numbers:

```
I  T     I  S     I  M  P  O  S  S  I  B  L  E     T  O
09 20 27 09 19 27 09 13 16 15 19 19 09 02 12 05 27 20 15

S  E  P  E  R  A  T  E     A  N  Y     P  O  W  E  R
19 05 16 05 18 01 20 05 27 01 14 25 27 16 15 23 05 18
```

etc. Since N has 16 digits and the first two are > 27, we can concatenate the numbers into blocks of 8; we get

```
0920270919270913 1615191909021205 2720151905160518 0120052701142527
```

etc. These messages m_1, m_2, m_3, m_4 are now encrypted using the public key (N, e); with

```
m1 = 0920270919270913; c1=Mod(m1,N)^e
```

we find

$$\begin{aligned} c_1 &= 1747970118663570, \\ c_2 &= 313567858470709, \\ c_3 &= 2917959816510238, \\ c_4 &= 594119149929740. \end{aligned}$$

Fermat sends me these numbers; for decrypting the messages, I first have to use my private keys p and q to compute an integer d with $de \equiv 1 \bmod (p-1)(q-1)$; this can be done by `d = lift(1/Mod(e,phi))`, which gives $d = 1226987039051581$. Now I decrypt by computing

```
  m1 = Mod(c1,N)^d}
```

etc., and I find $m_1 = 920270919270913$. This message is broken up into numbers of length 2 starting from the right; I get

```
9 20 27 09 19 27 09 13
```

giving me the plaintext "IT IS IM". The other c_j are taken care of similarly.

For practical purposes, keys of the size of our N are useless since the command `factor(N)` almost immediately gives you the prime factors p and q. Using

```
nextprime(random(10^151))
```

you can find suitable primes (pseudoprimes actually; but these work just as well) within a second, and messages transmitted using keys of that size will be impossible to decrypt even for the guys at NSA.

2.4 Stupid Things To Do

RSA is not a foolproof system; there are many ways of screwing up the system by doing more or less stupid things. Here we will briefly discuss two of them.

One of the most stupid things you can do is to encrypt the letters one by one; then there will be something like 26 possible cipher text symbols c_i, and a frequency analysis will easily break your system.

A slightly less stupid thing would be to choose a very small encryption exponent e. Note that the computing time for encoding could be kept very small if you could simply pick $e = 3$ (of course this means that you have to choose your primes p and q of the form $3n + 2$ in order to avoid that $(p-1)(q-1)$ is divisible by 3). Eve cannot use this information alone to break RSA: even if $e = 3$, computing $d \equiv 1/e \bmod (p-1)(q-1)$ is impossible without knowing p and q. The problem is this: assume you would like to send one and the same message m to three users with public keys $(N_1, 3)$, $(N_2, 3)$ and $(N_3, 3)$. You then compute $c_j \equiv m^3 \bmod N_j$ for $j = 1, 2, 3$ and send them the c_j. Now Eve knows the c_j and the N_j, so she can solve the system of congruences

$$\begin{aligned} x &\equiv c_1 \bmod N_1, \\ x &\equiv c_2 \bmod N_2, \\ x &\equiv c_3 \bmod N_3 \end{aligned}$$

for a unique integer x with $1 \le x < N_1N_2N_3$. I claim that $x = m^3$. In fact, $x \equiv m^3 \bmod N_i$, and since $m < N_j$, we have $m^3 < N_1N_2N_3$. Thus the Chinese Remainder Theorem allows Eve to recover m^3; but then all she needs to do is compute the cube root[1] of m^3, and she will find m.

Another reason why too small encryption exponents (actually, $e = 2^{16} + 1 = 65537$ is quite a common choice; the problems described above only occur if someone sends out the same messsage to more than 65000 people) are a bad choice is the following: if the message m you want to send satisfies $m^e < N$, then not only do we have $c \equiv m^e \bmod N$, we actually have $c = m^e$, and this means that anyone can decrypt c by simply computing the e-th root of c. This problem is only a minor one, however, since it can be avoided by filling up the plaintext message with blanks, that is, appending zeros to m until it has about the same size as N.

2.5 Variations

The Chinese Remainder Theorem is often used to speed up calculations. Consider e.g. the problem of decrypting messages in the RSA system: given some $c \bmod N$ for $N = pq$, the decryption requires computing $m \equiv c^d \bmod N$, where d typically has hundreds of digits. If you want fast decryption on a system with small computing power, such as a cell phone, then it would be nice if we could use a very small decryption exponent d, such as $d = 3$. Picking $d \le 1000$, say, would be quite foolish, however, since someone who knows c and N could simply try all small values of d and just check whether the decrypted message makes sense.

Now this is where the Chinese Remainder Theorem comes in. Assume that $\gcd(p - 1, q - 1) = 2$, for example, and solve the system of congruences

$$d \equiv 3 \bmod p - 1, \qquad d \equiv 5 \bmod q - 1.$$

Note that if $p \equiv q \equiv 1 \bmod 4$, then these congruences would imply $d \equiv 3 \bmod 4$ and $d \equiv 5 \bmod 4$, which gives a contradiction; thus the condition $\gcd(p - 1, q - 1) = 2$ is necessary (and sufficient) for solvability.

Now d will be large, in general about the size of $(p-1)(q-1)$; yet decryption can be performed in a very efficient way: First, find a Bezout representation

[1]Computing the cube root of an integer is easy; computing the cube root of a residue class mod N is about as difficult as factoring N.

$1 = ap + bq$. Then, for each encrypted message c, compute $m_1 \equiv c^3 \bmod p$ and $m_2 \equiv c^5 \bmod q$, which is very fast since you need only $2 + 3 = 5$ multiplications modulo p or q. Finally, put $M \equiv bqm_1 + apm_2 \bmod N$: then

$$\begin{aligned} M &\equiv bqm_1 \equiv m_1 \equiv c^3 \equiv c^d \bmod p, \\ M &\equiv apm_2 \equiv m_2 \equiv c^5 \equiv c^d \bmod q, \end{aligned}$$

hence $M \equiv c^d \equiv m \bmod N$.

2.6 Some Questions

There are a number of immediate questions that come to mind:

1. (Complexity) Is fast encryption possible? After all, we have to compute $m^e \bmod N$ for quite large values of N and e.

2. Are there enough keys? In other words, if we pick primes p and q with about 150 digits, are there sufficiently many? If there are only some 1000 or so, someone could try them all in order to factor N.

3. Is it possible to find such primes sufficiently fast? If we would have to compute away for hours before finding such primes, RSA would be quite useless as an online encryption tool.

4. How hard is it to factor N?

5. Is it possible to break the RSA-system without factoring N?

6. RSA is based on the fact that factoring is hard. Can other hard problems in number theory and algebra also be used to set up cryptosystems?

7. In practice, one of the most important problems actually is the following: if you transmit plain text and if in the transmission a few bits get changed or lost, nothing serious happens: the text "somewhare ovr the raimbow" could probably still be read correctly as "somewhere over the rainbow". If, however, instead of the RSA-encrypted message c you receive a message c' that differs in a single bit from c, then $m' \equiv c'^d \bmod N$ will be total junk. Thus if you want RSA to work in practice, you also will have to implement strong error-correcting codes.

Chapter 3

Complexity

3.1 Landau's big-O notation

Next we will address the question of how fast we can perform certain calculations. To this end we introduce the big-O notation: for realvalued functions $f, g : \mathbb{N} \longrightarrow \mathbb{R}_{\geq 0}$ we say that $f = O(g)$ if there is a constant $C > 0$ such that $f(n) \leq Cg(n)$ for all sufficiently large n (more exactly: there are constants C, N such that $f(n) \leq Cg(n)$ for all $n \geq N$).

Examples: Let $f(n) = 2n^2 + 10^6$ and $g(n) = n^2$; then $f = O(g)$ since $f(n) < 3g(n)$ for all sufficiently large n. Also $f = O(g)$ for $g(n) = n^3$. It is not true, however, that $f = O(g)$ for $g(n) = n$.

The idea behind the big-O notation is that it often allows us to replace a complicated function $f(n)$ in some estimate by a simple function $g(n)$ that has about the same growth as f.

3.2 Complexity of Basic Arithmetic Algorithms

In the following we will carefully study how many operations are necessary for computing the sum and product of integers, or applying the Euclidean algorithm. To this end, we assume that the addition of two bits takes time $O(1)$ (i.e. there is a constant cost for this addition).

Now let a be an integer written in base 2, and let m be its length (the number of bits necessary to write it down); for example, $11 = (1011)_2$ has length 4. In general, the length of a is given by $\ell(a) = \lfloor \log(a) \rfloor + 1$, where log is the logarithm in base 2.

How long does it take to add two integers a and b of lengths m and n, respectively? If $m \geq n$, then there are at most m additions, plus another m carries, thus giving at most $2m$ bit additions. Thus addition takes $O(\max\{m, n\})$ bit operations.

What about multiplication? For computing $a * b$ we have to write down a at most $n = \ell(b)$ times and add; thus we have to add a number of length $m+1$ (the 1 is for shifting) exactly $n-1$ times; thus there are at most $O(mn)$ additions necessary, at least when we use this naive method.

Since multiplication is such a basic operation, any improvement in the complexity will be most welcome. Schönhage and Strassen found a method for multiplying two integers of length n in $O(n \log n \log\log n)$ bit operations. In practice, this algorithm becomes more practical than the naive method for numbers with more than 10^4 bits.

For finding the complexity of performing a division with remainder $a = bq + r$, let k denote the length of q. Then we have to subtract a number of size $n = \ell(b)$ at most k times, which takes $O(kn)$ bit operations. Since $k \approx m - n$, this is the same as $O((m-n)n)$, and in any case is bounded by $O(mn)$. Here's an example:

$$\begin{array}{rl}
10110 & = 11 \cdot 111 + 1 \\
\underline{11} & \\
101 & \\
\underline{11} & \\
100 & \\
\underline{11} & \\
1 &
\end{array}$$

Very often we are required not to add or multiply integers but residue classes modulo m. In such a case we assume that these integers are given by their lowest positive residue classes mod m, and therefore have size $\ell(m)$. Thus addition and multiplication mod m takes $O(\ell(m))$ and $O(\ell(m)^2)$ bit operations, respectively: for multiplication, we multiply a and b; the product is $< m^2$; then we divide $ab = mq + r$, which takes $O(\ell(m)\ell(q))$ operations; since $q < m$, this is also bounded by $O(\ell(m)^2)$.

Division modulo m is more difficult. For computing $a/b \bmod m$, we first assume that $\gcd(m, b) = 1$ (otherwise the division might not be possible). Then we apply the Euclidean algorithm to m and b and compute a Bezout representation $1 = mx + by$, and then find that $1/b \equiv y \bmod m$. Afterwards we multiply a and y mod m.

In order to find out how many operations this takes we have to analyze the Euclidean algorithm. Here is what we do:

$$\begin{aligned}
a &= q_0 b + r_1, && 0 < r_1 < b \\
b &= q_1 r_1 + r_2, && 0 < r_2 < r_1 \\
r_1 &= q_2 r_2 + r_3, && 0 < r_3 < r_2 \\
&\ldots \\
r_{n-2} &= q_{n-1} r_{n-1} + r_n, && 0 < r_n < r_{n-1} \\
r_{n-1} &= q_n r_n.
\end{aligned}$$

For computing the first line we need at most $O(\log b \log q_0)$ bit operations, for the second at most $O(\log r_1 \log q_1) \le O(\log b \log q_1)$, ..., and for the last at most $O(\log r_n \log q_n) \le O(\log b \log q_n)$. Thus we need at most $O(\log b(\log q_0 + \log q_1 + \ldots + \log q_n)) = O(\log b(\log q_0 q_1 \cdots q_n))$ bit operations. Now we claim that $q_0 q_1 \cdots q_n \le a$. This follows from

$$\begin{aligned} a = bq_0 + r_1 \qquad & \ge bq_0 \\ = (q_1 r_1 + r_2) q_0 \qquad & \ge r_1 q_0 q_1 \\ = \ldots \qquad & \ge r_n q_0 q_1 q_2 \cdots q_n, \end{aligned}$$

which implies the claim since $r_n \ge 1$. Thus for performing the Euclidean algorithm on the pair (a, b) we need at most $O(\log a \log b)$ bit operations. For going back up in order to compute the associated Bezout representation we need at most that many operations, and this shows that we can compute a Bezout representation for the gcd of a and b in at most $O(\log a \log b)$ bit operations.

It remains to estimate the size of the numbers r and s in a Bezout representations $d = ar + bs$ of $d = \gcd(a, b)$.

It is quite easy to show that these integers *can be chosen* sufficiently small: in fact, consider the equation $d = ar + bs$ for arbitrary integers r, s; by adding or subtracting $0 = ab + b(-a)$ sufficiently often we can make sure that $|r| \le \frac{b}{2}$. Then $d = ar + bs$ shows $|bs| = |d - ar| \le d + a|r| \le b + b\frac{a}{2}$, which implies $|s| \le 1 + \frac{a}{2}$. In any case, there exist solutions with $|r| < b$ and $|s| < a$.

The question, however, is this: do the integers r and s provided by the extended Euclidean algorithm also satisfy this bound? The answer is yes, but we will not go into the details here.

Back to division mod m: for computing the inverse of b mod m we compute integers x, y with $1 = bx + my$, which costs $O(\log b \log m)$ operations; then we multiply a and b^{-1} mod m, which takes at most $O((\log m)^2)$ operations. Since $b < m$, this means that division mod m can be performed in at most $O((\log m)^2)$ bit operations.

Summary

operation	complexity
$a + b$	$O(\max\{\log a, \log b\})$
ab	$O(\log a \cdot \log b)$
$a = bq + r$	$O(\log b \cdot \log q)$
$\gcd(a, b)$	$O(\log a \cdot \log b)$
$(a + b)$ mod m	$O(\log m)$
(ab) mod m	$O((\log m)^2)$
a/b mod m	$O((\log m)^2)$
a^d mod m	$O(\log d(\log m)^2)$

Polynomial Complexity

If an algorithm with input a and b requires $O((\log a)^r \log(b)^s)$ bit operations for integers $r, s \ge 0$, then the algorithm is said to run in polynomial time. The algorithms discussed above all run in polynomial time. A similar definition applies to algorithms with more numbers as input.

An algorithm that requires $O(n)$ bit operations is said to run in exponential time since $n = e^{\ln n}$. The same remark applies if the complexity is $O(n^r)$ for some constant $r > 0$ since $n^r = e^{r \ln n}$. For sufficiently large input, polynomial algorithms run fast than those with exponential running time. This need not be true for input of a given size: if n has 20 digits, then $n^{1/2} < (\log n)^{100}$.

Exercises

3.1 Let $f(n) = \sqrt{n} \log n$; show that $f = O(n^{\frac{1}{2}+\varepsilon}$ for any (fixed) $\varepsilon > 0$.

3.2 If $f_1 = O(g)$ and $f_2 = O(g)$, then $f_1 + f_2 = O(g)$ and $cf_1 = O(g)$ for any constant $c \in \mathbb{R}$.

3.3 Assume that m and n are coprime integers; for solving the system of congruences

$$\begin{aligned} x &\equiv a \bmod m, \\ x &\equiv b \bmod n, \end{aligned}$$

compute integers r, s with $mr + ns = 1$, and put $x = ans + bmr$.

1. Show that this x solves the system.
2. Show that the extended Euclidean algorithm provides you with integers r, s such that $|r| < n$ and $|s| < m$.
3. Estimate the complexity of this algorithm; here you may assume that $0 \le a < m$ and $0 \le b < n$.

3.4 Let $f, g \in \mathbb{Z}[X]$ be polynomials. What is the complexity for computing $f + g$ and fg?

3.5 Let f and g be polynomials in $(\mathbb{Z}/m\mathbb{Z})[X]$. What is the complexity for computing $f + g$ and fg?

3.6 Prove the following rules for gcd's of natural numbers:

$$\gcd(a, b) = \begin{cases} 2\gcd(\frac{a}{2}, \frac{b}{2}) & \text{if } 2 \mid a, 2 \mid b; \\ \gcd(\frac{a}{2}, b) & \text{if } 2 \mid a, 2 \nmid b; \\ \gcd(\frac{a-b}{2}, b) & \text{if } 2 \nmid ab. \end{cases}$$

3.7 Show how to compute $\gcd(91, 77)$ using these rules. This algorithm is due to Stein (1961).

3.8 Explain why, in binary arithmetic, Stein's algorithm can be performed using only shifting and subtracting.

Chapter 4

Primes

4.1 Prime Number Theorem

Let us now address the question of how many primes there are. The first answer is “infinitely many”, and the beautiful proof was already known to Euclid. If you have to look for primes with 150 digits, however, then you need to ask (and answer) more precise questions, such as “how many primes are there between the numbers n and m?”.

To answer this question, let $\pi(x)$ denote the number of primes below x. Table 4.1 gives you an idea of its growth:[1]

Already Gauss and Legendre conjectured that $\pi(x) \sim \frac{x}{\ln x}$ (this means that $\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1$); for example, $\pi(10^6) = 78,498$ and $\frac{10^6}{\ln 10^6} \approx 72382$. Similarly, $\pi(10^{149}) \approx 2.915 \cdot 10^{146}$ and $\pi(10^{150}) \approx 2.895 \cdot 10^{147}$, hence there must be about $\pi(10^{150}) - \pi(10^{149}) \approx 2.6 \cdot 10^{147}$ primes with 150 digits. This implies in particular that about one in every 200 odd numbers with 150 digits is a prime.

The proof of the prime number theorem $\pi(x) \sim \frac{x}{\ln x}$ was given independently by Hadamard and de la Vallée-Poussin. They used properties of the Riemann zeta function $\zeta(s) = \sum_{n\ge1} n^{-s}$, which converges for all $s \in \mathbb{C}$ with $\operatorname{Re} s > 1$. Euler’s formula $\zeta(s) = \prod(1 - p^{-s})^{-1}$, where the product is over all primes, shows that $\zeta(s)$ has something to do with primes.

As a matter of fact, the prime number theorem follows from the observation that $\zeta(s) \ne 0$ for all $s \in \mathbb{C}$ with $\operatorname{Re} s = 1$. This statement makes sense only once we have extended the zeta function to a meromorphic function in the whole complex plane; that this can be done was shown by Riemann.

Riemann also conjectured that the only zeros of $\zeta(s)$ in the critical strip $0 \le \operatorname{Re} s \le 1$ lie on the line $\operatorname{Re} s = \frac{1}{2}$; this is a lot stronger than what is required for proving the prime number theorem, and in fact its truth would imply sharp estimates for the error term $|\pi(x) - \frac{x}{\ln x}|$.

[1] `http://primes.utm.edu/howmany.shtml`

x	$\pi(x)$
10	4
100	25
$1,000$	168
$10,000$	$1,229$
$100,000$	$9,592$
$1,000,000$	$78,498$
$10,000,000$	$664,579$
$100,000,000$	$5,761,455$
$1,000,000,000$	$50,847,534$
$10,000,000,000$	$455,052,511$
$100,000,000,000$	$4,118,054,813$
$1,000,000,000,000$	$37,607,912,018$
$10,000,000,000,000$	$346,065,536,839$
$100,000,000,000,000$	$3,204,941,750,802$
$1,000,000,000,000,000$	$29,844,570,422,669$
$10,000,000,000,000,000$	$279,238,341,033,925$
$100,000,000,000,000,000$	$2,623,557,157,654,233$
$1,000,000,000,000,000,000$	$24,739,954,287,740,860$
$10,000,000,000,000,000,000$	$234,057,667,276,344,607$
$100,000,000,000,000,000,000$	$2,220,819,602,560,918,840$
$1,000,000,000,000,000,000,000$	$21,127,269,486,018,731,928$
$10,000,000,000,000,000,000,000$	$201,467,286,689,315,906,290$

Table 4.1: Prime Number Theorem

x	$pi(x)$	$x/\ln x$
1000	168	145
10000	1229	1086
100000	9592	8686
1000000	78498	72382
10000000	664579	620420
100000000	5761455	5428681

Table 4.2: $\pi(x)$ vs. $x/\ln x$

4.2 Primality Tests

Assume you have picked some random odd 150-digit integer p; how can you tell whether it is prime or not?[2] The first step is to divide p by all primes < 1000 and see whether it has any factors. The easiest way to do that is to multiply all of these primes together (you only have to do this once) and then apply the Euclidean algorithm to this product and p.

Fermat Tests and Carmichael Numbers

Once you have checked that p does not have any small prime factors, apply a Fermat test: pick some integer a coprime to p and check whether $a^{p-1} \equiv 1 \bmod p$. If p is prime, this condition will be satisfied; unfortunately it is also satisfied for some composite numbers such as $341 = 11 \cdot 31$ or $2^{11} - 1 = 23 \cdot 89$. In fact, if q is prime and $M_q = 2^q - 1$, then we always have $2^{M_q - 1} \equiv 1 \bmod M_q$ whether M_q is prime or not. Composite integers passing the Fermat test for base a are called Fermat-pseudoprimes for the base a.

It gets worse: there are integers n that are Fermat pseudo-prime for any basis a coprime to n, such as $561 = 3 \cdot 11 \cdot 17$ (use the pari program

```
n=561;for(a=2,20,print(a,"   ",Mod(a,n)^n-1))
```

to check this). Such integers are called Carmichael numbers. It is easy to see that numbers of the form $(6k + 1)(12k + 1)(18k + 1)$ are Carmichael if each factor is prime; for example, $7 \cdot 13 \cdot 19 = 1729$ is a Carmichael number. It is known that there are infinitely many Carmichael numbers.

Euler Pseudo-Primes

The Fermat test was based on Fermat's Little Theorem $a^{p-1} \equiv 1 \bmod p$ for primes p and integers a coprime to p. There is a similar test based on Euler's criterium $a^{\frac{p-1}{2}} \equiv (\frac{a}{p}) \bmod p$. For odd integers n the left hand side of this congruence is computed by repeated squaring, and the Jacobi symbol on the right is evaluated using the quadratic reciprocity law.

The Euler test is better than the Fermat test: clearly every integer passsing the Euler test also passes the Fermat test, and there are integers passing the Fermat test that the Euler test recognizes as composites: consider e.g. the Fermat pseudo-prime 561 for base 5; here pari tells us that $5^{560/2} = 5^{280} \equiv 67 \bmod 561$, so we don't even have to compute $(\frac{5}{561}) = +1$ to know that this number must be composite. Actually the congruence $67^2 \equiv 1 \bmod 561$ yields the factors $\gcd(67 - 1, 561) = 33$ and $\gcd(67 + 1, 561) = 17$. This is a fairly typical situation: if a number n passes the Fermat test but fails the Euler test, then one usually gets a factor of n for free.

Are there analogs of Carmichael numbers in this situation? The answer is no:

[2]See `http://primes.utm.edu/prove/`

Proposition 4.1. *Let $n \in \mathbb{N}$ be composite and odd. Then the number of integers a with $1 \le a < n$ coprime to n such that n passes the Euler test $a^{\frac{n-1}{2}} \equiv (\frac{a}{n}) \bmod n$ is less than $\frac{n}{2}$.*

This leads directly to the primality test (actually a compositeness test) of Solovay-Strassen: pick an $a \in \{2, \ldots, n-1\}$ at random and do an Euler test. If it fails, declare n composite; if n passes the test, repeat with a different choice of a. The probability that a composite number n passes k such tests is less than 2^{-k}.

If a composite number n passes the Euler test for base a, we say that n is an Euler pseudo-prime for base a and write that n is a epsp(a).

The test of Solovay-Strassen is not used anymore: the Miller-Rabin test discussed below is stronger and has the same complexity.

The Miller-Rabin Test

Assume that p is an odd integer and write $p - 1 = 2^r u$ for some odd integer u. Then we have

$$\begin{aligned} a^{p-1} - 1 = a^{2^r u - 1} - 1 &= (a^{2^{r-1}u} + 1)(a^{2^{r-1}u} - 1) \\ &= (a^{2^{r-1}u} + 1)(a^{2^{r-2}u} + 1)(a^{2^{r-2}u} + 1) \\ &= \ldots \\ &= (a^{2^{r-1}u} + 1)(a^{2^{r-2}u} + 1) \cdots (a^u + 1)(a^u - 1). \end{aligned}$$

If p is prime and $p \nmid a$, then p divides the left hand side. Since p is prime, it must divide one of the factors on the right hand side. This shows that

Proposition 4.2. *Let $p = 1 + 2^r u$ for some odd integer u; if p is prime, then either $a^u \equiv 1 \bmod p$, or there is a $j \in \{0, 1, \ldots, r-1\}$ such that $a^{2^j u} \equiv -1 \bmod p$.*

If n is an odd number, we say it passes the Miller-Rabin test for base a (or that n is a strong pseudo-prime for base a (we write n is a spsp(a)) if it has the property described in Prop. 4.2. The three nice properties of the Miller-Rabin test are:

1. It runs as fast as the Euler test.
2. An integer passing the Miller-Rabin test will also pass the Euler test: any spsp(a) is also a epsp(a) (in other words: Miller-Rabin is stronger).
3. The probability that a composite number n passes k Miller-Rabin tests (for randomly chosen a) is less than 4^{-k}.

The last property is a consequence of

Proposition 4.3. *Let $n \in \mathbb{N}$ be composite and odd. Then the number of integers a with $1 \le a < n$ coprime to n such that n passes the Miller-Rabin test is $\le \frac{\phi(n)}{4}$.*

Proof. `www.mat.uniroma2.it/~schoof/millerrabinpom.pdf` □

This bound is best possible: if $n = pq$ for primes p and q, let $d = \gcd(p-1, q-1)$ and write $d = gu$ for g a power of 2 and some odd u. Then it can be shown that the number of a for which n passes the Miller-Rabin test is $u^2(1 + \frac{g^2-1}{3})$. If $d = 2$, then there are only two such a, namely $a = 1$ and $a = n - 1$.

On the other hand, if $p = 2u + 1$ and $q = 4u + 1$ are prime and t is odd, then $d = 2u$, and in this case we get that the number of such a is equal to $2u^2 = \phi(n)/4$. If $u = 3$, then $n = pq = 7 \cdot 13$, and there should be 18 such bases. `pari` shows that n passes the Miller-Rabin test for the following bases: for $a = 1, 9, 16, 22, 29, 53, 74, 79, 81$ we have $a^{45} \equiv +1 \bmod 91$, and for $a = 10, 12, 17, 38, 62, 69, 75, 82, 90$ we have $a^{45} \equiv -1 \bmod 91$.

These probabilistic tests discussed above allow you to find large "probable" primes quite quickly (in polynomial time). The Miller-Rabin test can actually be transformed into a deterministic primality test, that is, into a test that either proves that n is composite or that it is prime. Unfortunately, this works only if the Generalized Riemann Hypothesis is true, according to which the zeros of certain generalized zeta functions in the critical strip all have real part $\frac{1}{2}$.

Theorem 4.4. *Assume that GRH holds. Let n be an odd composite number, and write $n = 1 + 2^r u$ with $2 \nmid u$. If, for all integers a with $1 < a \le 2(\ln n)^2$, the integer n passes the Miller-Rabin test, then n is prime.*

Since a single Miller-Rabin tests requires $O((\log n)^2)$ bit operations, this test requires $O((\log n)^4)$, and thus is polynomial. Actally we can replace GRH by the weaker assumption that the Riemann Hypothesis holds for a small class of zeta functions, namely those of the form $L(s, \chi) = \sum_{n \ge 1} \chi(n) n^{-s}$, where χ is a quadratic Dirichlet character (for example $\chi(n) = (\frac{n}{d})$ for odd integers $d \ge 1$).

4.3 Background from Number Theory

In the following we need a few results from elementary number theory.

Assume that we are given an integer m and an integer a coprime to m. The smallest exponent $n > 0$ such that $a^n \equiv 1 \bmod m$ is called the order of $a \bmod m$; we write $n = \operatorname{ord}_m(a)$.

Proposition 4.5. *Let a and m be coprime integers. If $a^d \equiv 1 \bmod m$ for some integer d, then $\operatorname{ord}_m(a) \mid d$. In particular, $\operatorname{ord}_m(a) \mid \phi(m)$.*

Elements $g \in (\mathbb{Z}/m\mathbb{Z})^\times$ with $\operatorname{ord}_m(g) = \phi(m)$ are called *primitive roots* modulo m.

Proposition 4.6. *If p is prime, there are exactly $\phi(p-1)$ primitive roots modulo p.*

These results should also be familiar from abstract algebra: the first proposition is a special case of the statement that the order of an element divides the order of a group, and the second one is a special case of the fact that the multiplicative group of a finite field is cyclic.

4.4 Primality Proofs

We have already seen that the converse of Fermat's Little Theorem fails. The following result goes back to Lucas (1891):

Proposition 4.7. *Let n be an integer > 1. Assume that for every prime factor q of $n-1$ there is an integer a such that*

$$a^{n-1} \equiv 1 \bmod n, \qquad \textit{and}$$

$$a^{(n-1)/q} \not\equiv 1 \bmod n.$$

Then n is prime.

Proof. The converse is obvious: if n is prime, just pick a primitive root a mod n.

Assume therefore that n satisfies the conditions above. We claim that $\phi(n) = n-1$, which in turn implies that n is prime (because every element of $\{1, 2, \ldots, n-1\}$ must be coprime to n).

In fact, if $\phi(n) < n-1$, then there is a prime q and some $r \geq 1$ such that $q^r \mid (n-1)$ but $q^r \nmid \phi(n)$. Since $q \mid (n-1)$ there is an integer a satisfying the conditions above. Let m be the order of a modulo n. Then $m \mid (n-1)$ since $a^{n-1} \equiv 1 \bmod n$, but $m \nmid \frac{n-1}{q}$ because $a^{(n-1)/q} \not\equiv 1 \bmod n$.

This implies $q^r \mid m$; but $m \mid \phi(n)$: contradiction. $\square$

Primes p for which the factorization of $p-1$ is known can therefore be proved prime quite easily. For your average 150-digit prime, however, this might already be a problem, since factoring $p-1$ is only easy if its prime factors are small.[3]

The following improvement (due to Pocklington 1914) of Lucas' converse of Fermat's Little Theorem is substantial:

Theorem 4.8. *Let s be a divisor of $n-1$ with $s > \sqrt{n}$. Assume that there is an integer a such that*

$$a^{n-1} \equiv 1 \bmod n, \qquad \gcd(a^{(n-1)/q} - 1, n) = 1$$

for every prime $q \mid s$. Then n is prime.

Thus we can prove n to be prime by factoring only a part of $n-1$.

Proof. Assume that n is not prime, and let $p \leq \sqrt{n}$ be a prime factor of n. Put $b \equiv a^{(n-1)/s} \bmod n$. Then $b^s \equiv a^{n-1} \equiv 1 \bmod n$, and in particular

$$b^s \equiv 1 \bmod p. \tag{4.1}$$

Next we claim that

$$b^{s/q} \not\equiv 1 \bmod p \tag{4.2}$$

for every prime $q \mid s$. In fact, assume that $b^{s/q} \equiv 1 \bmod p$ for some prime $q \mid s$. Then $p \mid (b^{s/q} - 1) = (a^{(n-1)/q} - 1)$, contradicting the condition on the gcd's.

Now (4.1) and (4.2) imply that $s = \operatorname{ord}_p(b)$. Since $b^{p-1} \equiv 1 \bmod p$, Prop. 4.5 tells us that $s \mid p-1$. But then $s < p \leq \sqrt{n}$: contradiction. $\square$

[3]As we will see, primes p for which $p-1$ has only small prime factors are useless for RSA.

This result has a number of classical corollaries:

Corollary 4.9 (Proth)**.** *Let $k, l \in \mathbb{N}$ with $3 \nmid k$ and $k \le 2^l + 1$. Then $n = k \cdot 2^l + 1$ is prime if and only if $3^{k2^{l-1}} \equiv -1 \bmod n$.*

Proof. Assume that n is prime. Then we have to show that $(\frac{3}{n}) = -1$; since $l \ge 2$ (if $l = 1$, then $k \le 3$, hence $k \le 2$ and $n \in \{3, 5\}$), we know that $n \equiv 1 \bmod 4$. The quadratic reciprocity law implies $(\frac{3}{n}) = (\frac{n}{3}) = (\frac{2}{3}) = -1$ because $n = k \cdot 2^l + 1 \equiv 2 \bmod 3$. Euler's criterium now shows that $3^{k2^{l-1}} \equiv -1 \bmod n$.

Conversely, assume that $3^{k2^l - 1} \equiv -1 \bmod n$. Put $s = 2^l$ and $a = 3$; then

$$a^{n-1} \equiv 1 \bmod n \qquad \text{and} \qquad a^{(n-1)/2} \equiv -1 \bmod n,$$

so by Pocklington's test n must be prime. □

Corollary 4.10. *A Fermat number $F_n = 2^{2^n} + 1$ is prime if and only if*

$$3^{(F_n - 1)/2} \equiv -1 \bmod F_n.$$

Proof. This follows directly from Proth's test. □

Using the arithmetic of quadratic number fields or of finite fields it is easy to devise primality tests based on the factorization of $n + 1$; it is even possible to combine these tests and to show that n is prime using the factorization of a sufficiently large part of $(n - 1)(n + 1)$. In the special case of Mersenne numbers $M_p = 2^p - 1$ (for which the factorization of $M_p + 1$ is known), there is a particularly simple and effective primality test:

Proposition 4.11. *Let $p \ge 2$ be prime; set $S_1 = 4$ and $S_{r+1} = S_r^2 - 2$. Then M_p is prime if and only if $S_{p-1} \equiv 0 \bmod M_p$.*

4.5 AKS

The primality proofs using the factorizations of $n-1$ and $n+1$ are only effective for small n or for integers of a special form (for example $k \cdot 2^n \pm 1$). In any case, they do not run polynomial time since we do not have any polynomial time factorization algorithm. The deterministic Miller-Rabin test, on the other hand, runs in polynomial time but needs to assume the validity of the GRH.

In August 2002, Manindra Agrawal, Neeraj Kayal and Nitin Saxena published their discovery of a deterministic primality test that runs in polynomial time by sending out emails to a couple of number theorists. Their article "PRIMES is in P" was published in the Annals of Math. **160**, 2004.

The basic idea behind their algorithm is the observation

Lemma 4.12. *Assume that n and a are coprime integers. Then n is prime if and only if $(X + a)^n \equiv X^n + a \bmod n$.*

Proof. If n is prime, then the binomial coefficients $\binom{n}{k}$ with $1 \le k \le n-1$ are divisible by n. Moreover, $a^n \equiv a \bmod n$ by Fermat's Little Theorem.

If n is not prime, let p be a prime divisor of n, and let p^k be the largest power of p dividing n. Then $p^k \nmid \binom{n}{p} = \frac{n(n-1)\cdots(n-p+1)}{1\cdot 2\cdots p}$ because $p^k \parallel n$, $p \nmid (n-1)\cdots(n-p+1)$, and $p \parallel p!$. Thus the coefficient of X^p in $(X+a)^n$ is not divisible by p^k, and therefore not divisible by n. $\square$

As a direct test, this is useless: we have to compute n binomial coefficients mod n. Here's the next idea: instead of checking $(X+a)^n \equiv X^n + a \bmod n$, verify that $(X+a)^n \equiv X^n + a \bmod (X^r - 1, n)$ for suitable values of a and r.

Here is the algorithm:

1. If $n = a^b$ for integers $a, b \ge 2$, then n is `composite`.
2. Find $r \in \mathbb{N}$ such that $\operatorname{ord}(n \bmod r) \ge 4(\log n)^2 + 2$, and set $\ell = \lfloor 2\sqrt{r}\log n \rfloor + 1$.
3. If $1 < \gcd(a, n) < n$ for $a = 2, 3, \ldots, \ell$, then n is `composite`. If $r > n$, then n is `prime`.[4]
4. If $(X-a)^n \not\equiv X^n - a \bmod (X^r - 1, n)$ for $a = 1, 2, \ldots, \ell$, then n is `composite`.
5. If n has not been declared composite in steps 1 - 4, then n is `prime`.

There are now three things to prove:

A. If n is prime, the algorithm declares that n is prime in step 5.

B. If n is composite, the algorithm declares that n is composite in step 1, 3 or 4.

C. The algorithm runs in polynomial time.

Assume first that n is prime. Then n is clearly not declared composite in steps 1, 3, or 4, hence is declared prime in step 5.

Next assume that n is composite, but that the algorithm declares n to be a prime. Let $p \le \sqrt{n}$ be a prime factor of n. Then we know that

$$\begin{aligned} (X-a)^n &\equiv X^n - a \bmod (X^r - 1, p) &\qquad& \text{for } a = 1, 2, \ldots, \ell \\ (X-a)^p &\equiv X^p - a \bmod (X^r - 1, p) && \text{whenever } p \nmid a \end{aligned}$$

The first congruence comes from the algorithm and even holds mod $(X^r - 1, n)$; the second congruence holds because p is prime.

Lemma 4.13. *Let a be an integer coprime to n, and fix an integer r and a prime p. Then the set of all integers m such that*

$$(X-a)^m \equiv X^m - a \bmod (X^r - 1, p) \tag{4.3}$$

is multiplicative.

[4]We will see below that $r \le 16(\log n)^5 + \ldots$, so this will only happen for $n < 10^9$.

Proof. Assume that m_1 and m_2 are integers such that

$$(X-a)^{m_1} \equiv X^{m_1} - a \bmod (X^r-1, p), \tag{4.4}$$
$$(X-a)^{m_2} \equiv X^{m_2} - a \bmod (X^r-1, p). \tag{4.5}$$

Then

$$(X-a)^{m_1 m_2} \equiv (X^{m_1}-a)^{m_2} \bmod (X^r-1, p). \tag{4.6}$$

Moreover, (4.5) means that

$$(X-a)^{m_2} - (X^{m_2}-a) \equiv (X^r-1)g(X) \bmod p$$

for some polynomial g. Replacing X by X^{m_1} then shows

$$(X^{m_1}-a)^{m_2} - (X^{m_1 m_2}-a) \equiv (X^{rm_1}-1)g(X^{m_1}) \bmod p.$$

Since X^r-1 divides $X^{rm_1}-1$), this implies

$$(X^{m_1}-a)^{m_2} - (X^{m_1 m_2}-a) \equiv 0 \bmod (X^r-1, p).$$

Together with (4.6) this implies

$$(X-a)^{m_1 m_2} \equiv X^{m_1 m_2} - a \bmod (X^r-1, p),$$

which is what we wanted to prove. □

Since we know that $m = n$ and $m = p$ satisfy (4.3), the multiplicativity of the exponent implies that (4.3) holds for every integer of the form $m = p^i n^j$ with $i, j \geq 0$. Now consider the subgroup G of $(\mathbb{Z}/r\mathbb{Z})^\times$ generated by the residue classes of p and n modulo r, let $t = \#G$ denote its order, and set $L = \{p^i n^j : 0 \leq i, j \leq \sqrt{t}\}$. The elements of L are all $< n^{2\sqrt{t}}$. The residue classes of the elements in L are in G; since there are $(\lfloor\sqrt{t}\rfloor + 1)^2 > t = \#G$ elements in L, at least two of them must be congruent modulo r. Thus there exist $m_1 = p^{i_1} n^{j_1}$ and $m_2 = p^{i_2} n^{j_2}$ in L with $(i_1, j_1) \neq (i_2, j_2)$ and $m_2 = m_1 + kr$ for some integer k.

Now we find

$$\begin{aligned}
(X-a)^{m_2} &\equiv X^{m_2} - a && \text{since } m_2 \in L \\
&= X^{m_1+kr} - a && \text{since } m_2 = m_1 + kr \\
&\equiv X^{m_1} - a && \text{since } X^r \equiv 1 \bmod (X^r - 1) \\
&\equiv (X-a)^{m_1} \bmod (X^r-1, p) && \text{since } m_1 \in L
\end{aligned}$$

Thus we know that $(X-a)^{m_1} \equiv (X-a)^{m_2} \bmod (X^r-1, p)$ for all $1 \leq a \leq \ell$. We claim that this implies $m_1 = m_2$. Using this equality we can complete the proof as follows. We have $p^{i_1} n^{j_1} = p^{i_2} n^{j_2}$; cancelling gives $p^r = n^s$ for positive integers r, s, which implies that n is a power of p; since we assumed that n is composite, n is a nontrivial power of p: but then n would not have survived step 1.

Now let us prove

Lemma 4.14. *Assume that* $(X-a)^{m_1} \equiv (X-a)^{m_2} \bmod (X^r-1,p)$ *for all* $1 \le a \le \ell$. *Then* $m_1 = m_2$.

Proof. Assume without loss of generality that $m_2 \ge m_1$, and consider the polynomial $b(Z) = Z^{m_1} - Z^{m_2}$. We will show that b has more than $m_2 = \deg b$ roots in some field; this will imply that b is the zero polynomial, hence that $m_1 = m_2$.

Let η be a root of $X^r - 1$ in some extension field of $\mathbb{F}_p$. Then the congruence $(X-a)^{m_1} \equiv (X-a)^{m_2} \bmod (X^r-1,p)$ implies $(\eta-a)^{m_1} = (\eta-a)^{m_2}$ in $\mathbb{F}_q = \mathbb{F}_p(\eta)$. This in turn means that $\eta - a$ is a root of $b(Z)$ in $\mathbb{F}_q$.

Now observe that if α and β are roots of b, then so is $\alpha\beta$ (this is because α is a root of b if and only if $\alpha^{m_1} = \alpha^{m_2}$). Thus the elements $\prod_{a=1}^{\ell}(\eta-a)^e$ are roots of b for all $e \ge 0$.

Let $\ell' = \lfloor 2\sqrt{t}\log n\rfloor + 1$. Then $\ell' < \ell$ because $t = \#G \le \#(\mathbb{Z}/r\mathbb{Z})^\times \le r-1$ and $\ell = \lfloor 2\sqrt{r}\log n\rfloor + 1$. From the above we deduce that every element of

$$S = \Big\{ \prod_{a=1}^{\ell'} (\eta - a)^e : e \in \{0,1\} \Big\}$$

is a root of b. We now claim

Lemma 4.15. *If* η *is a primitive* r*-th root of unity, then* $\#S = 2^{\ell'}$.

Assuming this for the moment, we can complete the proof as follows. The polynomial $b(Z) = Z^{m_1} - Z^{m_2}$ has at least $2^{\ell'}$ roots in $\mathbb{F}_q$. Now

$$2^{\ell'} = 2^{\lfloor 2\sqrt{t}\log n\rfloor + 1} > 2^{2\sqrt{t}\log n} = n^{2\sqrt{t}} \ge m_2 = \deg b$$

shows that b has more than $\deg b = m_2$ roots, hence must be 0. $\square$

It remains to give a

Proof of Lemma 4.15. We have to show that the $2^{\ell'}$ products $\prod_{a=1}^{\ell'}(\eta-a)^e$ are pairwise distinct elements of $\mathbb{F}_q$. Note that these elements are values at $X = \eta$ of the polynomials $\prod_{a=1}^{\ell'}(X-a)^e \in \mathbb{F}_p[X]$; we first show that these polynomials are pairwise distinct. In fact, the linear polynomials $X-1$, $X-2$, …, $X-\ell'$ are distinct primes in $\mathbb{F}_p[X]$ since $\ell' < p$ since none of the numbers 2, 3, …, $\ell' < \ell$ divide n by step 3 of the algorithm. Since $\mathbb{F}_p[X]$ is factorial, the polynomials $\prod_{a=1}^{\ell'}(X-a)^e$ are pairwise distinct.

We need another little observation: if g has the form above, then $g(\eta)^m = g(\eta^m)$ for each $m = p^i n^j$. In fact, $g(X)^m \equiv g(X^m) \bmod (X^r-1,p)$; since η is a root of X^r-1, plugging in η gives $g(\eta)^m = g(\eta^m)$ in $\mathbb{F}_q = \mathbb{F}_p(\eta)$.

Now we need to show that for different polynomials $g_1 \ne g_2$ of the form above, we also have $g_1(\eta) \ne g_2(\eta)$. Assume therefore that $g_1(\eta) = g_2(\eta)$; then all the η^m for $m = p^i n^j$ are roots of the polynomial $g_1 - g_2 \in \mathbb{F}_p[X]$. Since η is a primitive r-th root of unity, the number of distinct η^m is equal to the number of distinct residue classes of the form $p^i n^j \bmod r$, which is just the order $t = \#G$ of G. But $\deg g_1, \deg g_2 \le \ell' < t$ (in fact, $\ell' \le \lfloor 2\sqrt{t}\log n\rfloor + 1$ and

$t \geq \text{ord}\,(n \bmod r) \geq 4(\log n)^2 + 2)$. This is only possible if $g_1 - g_2 = 0$, and this contradiction finally proves our claim. $\square$

It remains to show that the algorithm runs in polynomial time.

Step 1 can be done in polynomial time: for testing whether $n = a^b$ for $a, b \geq 2$ it is sufficient to test all possible $b \leq \log n$ (this follows from $\log n = b \log a \geq b$). Extracting roots is easy: just test whether the nearest integer a to $n^{1/b}$ satisfies $a^b = n$.

In step 2 we need to find some r with the property that the order of $n \bmod r$ is at least $4(\log n)^2 + 2$. Since the size of ℓ depends on r, we must now show that we can always find a sufficiently small r. To this end we prove

Lemma 4.16. *We have* $\text{lcm}\,(k+1, k+2, \ldots, 2k+1) \geq 2^{2k}$.

Proof. Since $x(1-x) \leq \frac{1}{4}$ in the interval $[0, 1]$, we have

$$\begin{aligned} 2^{-2k} &\geq \int_0^1 [x(1-x)]^k dx \\ &= \sum_{i=0}^k \binom{k}{i} \int_0^1 (-1)^i x^{k+i} dx \\ &= \sum_{i=0}^k \frac{M_i}{k+i+1} = \frac{M}{L}, \end{aligned}$$

where the M_i are integers, and where $L = \text{lcm}\,(k+1, k+2, \ldots, 2k+1)$. Since the integral on the left is clearly positive, it is $\geq \frac{1}{L}$. This implies $L \leq 2^{2k}$. $\square$

In particular, we have $\text{lcm}\,(1, 2, \ldots, 2k+1) \geq 2^{2k}$.

Now assume that all $r \leq R$ do not divide n and satisfy $\text{ord}\,(n \bmod r) \leq T := 4(\log n)^2 + 2$. Then $n^i \equiv 1 \bmod r$ for some $i \leq T$, hence each $r \leq R$ divides

$$\prod_{i=0}^T (n^i - 1) \leq n^{T^2}.$$

In particular, $\text{lcm}\,(1, 2, \ldots, R)$ divides this product, hence is $\leq n^{T^2}$. This shows that $R \leq T^2 \log n + 1$.

Thus if R is bigger than that, then not every $r \leq R$ can have order $\leq T$, i.e., there is an integer $r \leq T^2 \log n + 2 = O((\log n)^5)$ with $\text{ord}\,(n \bmod r) > T$.

If we test each r below this bound by computing the first $4(\log n)^2 + 2$ powers of $n \bmod r$, we need at most $O((\log n)^9)$ bit operations.

Computing the gcd of r numbers takes at most $O(r(\log n)^2) = O((\log n)^7)$ bit operations.

The proof that step 4 can be performed in polynomial time is left as an exercise.

Remark. If n is composite, the AKS test will declare n to be composite almost immediately; if n is prime, however, it takes so long that it is currently not practical as a primality test. The tests that are used in practice are the cyclotomic test (going back to Adleman et al., who used higher reciprocity laws, and was transformed into a test with Gauss and Jacobi sums; you need a fair amoung of algebraic number theory to understand this test) and ECPP (elliptic curve primality proving; here you need to understand the basics of the arithmetic of elliptic curves).

Exercises

4.1 Let p and q be distinct odd primes, and let $n = pq$. Let a be an integer such that $a \equiv 1 \bmod p$ and $a \equiv -1 \bmod q$. Which of the tests of Fermat, Euler, and Miller-Rabin does n pass with respect to the base a?

4.2 Show that $2^7 - 1 = 127$ using the Lucas-Lehmer test.

4.3 Apply the AKS test to $n = 31$ and $n = 143$.

4.4 Show that if n is composite and passes the Fermat test for base 2, then $N = 2^n - 1$ is composite and passes the Miller-Rabin test for base 2.

4.5 Write a `pari` program that determines the three smallest composite integers n that pass a Fermat test with base $a = 2$.

Chapter 5

Factorization Algorithms

In this chapter we will discuss various algorithms for finding factors of integers.

5.1 Factoring in the Dark Ages

The simplest factoring algorithm is

Trial Division

This is based on the fact that composite numbers n have at least one prime factor $\le \sqrt{n}$. Thus for finding a factor of n, compute $n = aq + r$ for $a = 2$, 3, 5, 7, 9, ... until you find a zero remainder. In the worst case (for example when $n = p^2$ is a square of a prime), this takes approximately $\frac{1}{2}\sqrt{n}$ divisions with remainder, so is far from being polynomial.

Trial Division is, however, an important technique for clearing integers n from small prime factors. Here is a simple trick for speeding it up: after the divisions by 2 and 3, do the following:

```
set s = 2; p = 5
test whether n is divisible by p;
put p = p+s; s=6-s; and repeat until p > sqrt(n)
```

This sieves out the multiples of 3. But even if you could sieve out all the composite numbers and divide only by the primes $\le \sqrt{n}$, you would need approximately $\sqrt{n}/\log\sqrt{n} = \frac{1}{2}\frac{n^{1/2}}{\log n}$ divisions with remainder. Since a division costs about $O(\log n)$ bit operations, we cannot go below $O(n^{1/2})$ bit operations.

In order to become familiar with programming in `pari`, here's how to translate the above algorithm into a pari program. We assume for simplicity that n is a composite number with $\gcd(n, 6) = 1$.

```
{t=0;s=2;p=5; while(1-t,r=Mod(n,p);
 if(r,,t=1;print(p));p=p+s;s=6-s)}
```

Here $t = 0$ until a factor is found, when $t = 1$. As long as $t = 0$, we trial divide n by the odd numbers ≥ 5 coprime to 3.

Fermat Factorization

Fermat's idea was to write the number n to be factored as a difference of squares: from $n = y^2 - y^2 = (x-y)(x+y)$ we can read off a factorization of n, and this will be a nontrivial factorization unless $x - y = 1$.

If $n = pq$ is odd, then $p = x - y$ and $q = x + y$ leads to $x = \frac{q+p}{2}$ and $y = \frac{q-p}{2}$; this explains why the method works. These equalities also show that $x \leq n$, and that $x \approx n$ if one of the prime factors p or q is rather small). So here's what you do:

1. `put` $w = \lfloor\sqrt{N}\rfloor$ `and` $x = w$
2. `compute` $y = \lfloor\sqrt{x^2 - n}\rfloor$
3. `if` $n = x^2 - y^2$`, print` $x - y$ `and` $x + y$
4. `if` $x < n$`, replace it by` $x + 1$ `and goto step 2.`

Since we assume that n has been shown composite using a primality test, Fermat's method will actually find a factor eventually. The number of steps that have to be performed is at worst $n - \sqrt{n} = O(n)$, which is even worse than trial division (using a few simple ideas, however, the test can be improved).

Nevertheless, Fermat's method is very good at finding the factorization of $n = pq$ if p and q are very close to each other. If some idle RSA programmer picks his primes with the commands

```
p = nextprime(random(10^150))
q = nextprime(p)
```

then the factorization of $N = pq$ will take only fractions of a second. Try it!

Also observe that Fermat's method factors n quickly if n can be factored as $n = ab$ with $a, b \approx \sqrt{n}$. If n does not have such a factorization, it might very well be that $3n$ can be written as $3n = 3ab$ with $3a, b \approx \sqrt{3n}$. Thus it is often a good idea to try the Fermat method for various small multipliers k; if it finds a factor, it will never be k.

Lehman's Method

By combining trial division (search for all prime factors up to $n^{1/3}$) and the Fermat method with multipliers you can come up with a factorization method that requires at most $O(n^{\frac{1}{3}+\varepsilon})$ steps for every $\varepsilon > 0$ (smaller ε will lead to bigger constants involved in O; thus you cannot simply take $\varepsilon = 0$). Lehman's method was published in 1974 and was the first factorization algorithm with complexity less than $O(\sqrt{n}\,)$.

Here's how it's done (we assume that $n \geq 14$ has been shown composite by a primality test):

1. Let $B = \lfloor n^{1/3} \rfloor$. Check whether $n \equiv 0 \bmod q$ for $q = 2, 3, 5, 7, \ldots$ up to B. If no factor is found, set $k = 0$.
2. [Loop on k] Set $k \longleftarrow k + 1$. Set $r = 1$ and $m = 2$ if k is even, and $r = k + n$ and $m = 4$ if k is odd.
3. [Loop on a] For all integers a satisfying $4kn \le a^2 \le 4kn + B^2$ and $a \equiv r \bmod m$, check whether $b = \sqrt{a^2 - 4kn}$ is an integer; if it is, then $\gcd(a + b, n)$ is a nontrivial factor of n. After all a have been tested, goto step 2.

For proving the correctness of this algorithm we need a result due to Dirichlet:

Proposition 5.1. *For any real $\theta \in \mathbb{R}$ and any positive integer C there exist integers r and $0 < s \le C$ such that $|s\theta - r| < \frac{1}{C}$.*

Proof. For a real number x, let $\{x\} = x - \lfloor x \rfloor$ denote its fractional part. Note that if $\{x\} > \{y\}$, then $\{x\} - \{y\} = \{x - y\}$.

Now consider the numbers $0, \theta, 2\theta, \ldots, C\theta$. These are $C + 1$ real numbers with $0 \le \{t\theta\} < 1$. Divide the interval $[0, 1)$ into C subintervals of length $\frac{1}{C}$. Since there are $C + 1$ numbers, by Dirichlet's box principle there exist integers $0 \le t, t' \le C$ such that $|\{t\theta\} - \{t'\theta\}| \le \frac{1}{C}$. Now put $s = |t' - t|$. $\square$

Applying this with $\theta = \frac{p}{q}$ gives

Corollary 5.2. *For any pair of integer $p, q \ge 1$ and any $C > 1$ there exist positive integers r, s with $s < C$ and $|\frac{r}{s} - \frac{p}{q}| < \frac{1}{sC}$.*

We now claim that Lehman's algorithm is correct, i.e., finds a prime factor of n. If step 1 does not find a prime factor, then all prime factors of n are $> n^{1/3}$, hence n has at most two prime factors. Assume therefore that $n = pq$ for primes $q \ge p > n^{1/3}$. This implies $1 \le \frac{q}{p} < n^{1/3}$ since $\frac{q}{p} = \frac{pq}{p^2} = \frac{n}{p^2} \le n^{1-\frac{2}{3}}$.

Applying Dirichlet's Lemma with $C = n^{1/6}\sqrt{q/p}$ shows that there exist integers r, s such that $|\frac{r}{s} - \frac{p}{q}| < \frac{1}{sC}$. Multiplying through by qs shows that

$$|qr - ps| < \frac{qs}{sC} = \frac{q}{n^{1/6}\sqrt{q/p}} = \frac{\sqrt{pq}}{n^{1/6}} = n^{\frac{1}{2} - \frac{1}{6}} = n^{\frac{1}{3}}.$$

Now put $k = rs$, $a = ps + qr$, and $b = |qr - ps|$; then $0 < b < n^{1/3}$, and

$$a^2 - 4kn = p^2s^2 + 2pqrs + q^2r^2 - 4pqrs = (qr - ps)^2 = b^2. \qquad (5.1)$$

We claim first that $k < n^{1/3}$: in fact we have

$$k = rs = \frac{r}{s}s^2 < \frac{p}{q}s^2 + \frac{s}{C} \le \frac{p}{q} \cdot \frac{q}{p}n^{1/3} + 1 = n^{1/3} + 1.$$

Next we show that $4kn \le a^2 \le 4kn + n^{2/3}$. The left inequality follows trivially from (5.1), the right one from $a^2 = 4kn + b^2 < 4kn + n^{2/3}$.

Thus if n survives the first step of Lehman's algorithm (trial division up to $n^{1/3}$), then there is a $k < n^{1/3}$ and an integer a satisfying $4kn \le a^2 \le 4kn+n^{2/3}$ such that $a^2 - 4kn = b^2$ is a square. This implies that a factor $\gcd(a+b,n)$ is found for some $k < n^{1/3}$.

It remains to show that $\gcd(a+b,n)$ is nontrivial. From $4kn = (a-b)(a+b)$ we see that if the gcd is trivial, then $n \mid (a-b)$ or $n \mid (a+b)$. This is not compatible with the fact that $a+b < n$. In fact, $a^2 \le 4kn+n^{2/3} < 4n^{4/3}+n^{2/3}$ implies $a \le 2n^{2/3}+1$, and since $b \le n^{1/3}$ we see that $a+b \ge n$ is possible only if $n \le 13$.

5.2 Pollard's $p-1$ and ρ methods

The $p-1$-method

Let us start with explaining the $p-1$-method. Let n be a composite integer, pick a base a with $\gcd(a,n) = 1$ and an integer B (for example $B = 10^5$; in practice, it can be as large as 10^{10}). Then set

$$k = \prod q^e, \qquad (5.2)$$

where the product is over all primes q, and e is chosen maximal with $q^e < B$. So for $B = 10$, we would have $k = 2^3 \cdot 3^2 \cdot 5 \cdot 7$.

Now compute $\gcd(a^k - 1, n)$ in the following way:

1. Fix a, B, and set $k = 1$ and $q = 2$.
2. Find e with $q^e \le B < q^{e+1}$.
3. Set $a \longleftarrow a^{q^e} \bmod n$ and find $\gcd(a-1,n)$.
4. If the gcd is 1, replace q by the next prime; if $q < B$, goto step 2, otherwise stop.

If no factor is found, one can increase B (multiply it by 10, say), and continue.

The idea behind these computations is easy to explain. Assume p is a prime factor of n. If the prime factors of $p-1$ are all less than B, then there's a good chance that $p-1$ is a divisor of the number k in (5.2). In this case, $k = (p-1)m$ for some integer m, hence $a^k = a^{(p-1)r} \equiv 1 \bmod p$, hence $\gcd(a^k-1,n)$ will be divisible by p. Thus the $p-1$-method finds prime factors of n with the propery that $p-1$ only has small prime factors. The running time obviously depends on the size of the largest prime factor of $p-1$, and *not* on n.

The largest prime factor ever found with the $p-1$-method is the 66-digit number

$$p = 672038771836751227845696565342450315062141551559473564642434674541$$

dividing $960^{119}-1$, with the factorization

$$p - 1 = 2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 17 \cdot 23 \cdot 31 \cdot 163 \cdot 401 \cdot 617 \cdot 4271 \cdot 13681 \cdot 22877 \cdot 43397 \\ \cdot 203459 \cdot 1396027 \cdot 6995393 \cdot 13456591 \cdot 2110402817.$$

It was found by T. Nohara on June 29, 2006. Actually, he was quite lucky; the next largest prime has "only" 58 digits.

The moral of the story is: when you create primes p, q for your RSA-key, make sure that $p-1$ and $q-1$ have at least one big prime factor ($> 10^{30}$, for example).

For sufficiently small exponents k, you can use the following one-line command in `pari` to find factors of integers n with the $p-1$-method:

```
n=13231;k=10;x=lift(Mod(2,n)^(k!));print(gcd(x-1,n))
```

To see how fast this works, try to factor $2^{67}-1$ with this method using $k = 1000, 2000, 3000, \ldots$; this factorization was presented by Cole in a lecture in 1903: he said that he had used "three years of Sundays" to find the factorization.

Pollard's ρ method

This method is based on the birthday paradox, which is about the surprising fact that the probability that among 23 randomly chosen people there are at least two with the same birthday is $> \frac{1}{2}$.

More generally, assume you pick n numbers from a set of cardinality q. What is the probability that you pick the same number twice? Let us compute the probability that this does not happen. The probability that the second number is different from the first is $1 - \frac{1}{q}$; the probability that the third number is different from the first and the second is $1 - \frac{2}{q}$; finally, the probability that the n-th number is different from the first $n-1$ numbers is $1 - \frac{n-1}{q}$. Thus the probability that all numbers are pairwise distinct is

$$\begin{aligned} P &= \Big(1 - \frac{1}{q}\Big)\Big(1 - \frac{2}{q}\Big)\cdots\Big(1 - \frac{n-1}{q}\Big) \\ &= \prod_{i=1}^{n-1}\Big(1 - \frac{i}{q}\Big) \\ &\le \prod_{i=1}^{n-1} e^{-i/q} && \text{since } 1 + x \le e^x \\ &= \exp\Big(-\frac{n(n-1)}{2q}\Big). \end{aligned}$$

Thus the probability that all n numbers are pairwise distinct is $\approx e^{-n^2/2q}$. If we want to make this probability $< \varepsilon$, then we have to choose $n > \sqrt{-2q \log \varepsilon}$.

Now consider a random map $f : S \longrightarrow S$ from a set S of cardinality q to itself. For a random $s \in S$, compute $f_0(s) = s$, $f_1(s) = f(s)$, $f_2(s) = f(f(s))$ etc. Since there are only finitely many elements in S, there must be $j < k$ such that $f_j(s) = f_k(s)$, and therefore $f_{j+1}(s) = f_{k+1}(s)$ etc. The analysis of the birthday paradox shows that this should happen with probability > 0.999, say, for some index $k = O(\sqrt{q})$.

Now assume you have a composite number n, and let p denote its smallest prime factor. Compute the numbers $f_j(s)$ for some randomly chosen $s \in (\mathbb{Z}/n\mathbb{Z})^\times$. Then we expect a cycle after $O(\sqrt{n}\,)$ steps. On the other hand, we expect that $f_j(s) \equiv f_k(s) \bmod p$ after about $O(\sqrt{p}\,)$ steps. Thus to find p we only have to compute the $f_i(s) \bmod n$ and check $\gcd(f_k(s) - f_j(s), n)$.

There is, however, a problem with the "only" in the last sentence. We know $0 \le i < j \le c\sqrt{p}$ for some constant c, hence there are about c^2p gcd's to check: for reasonable values of c, this is worse than trial division!

The way out of this problem is called Floyd's cycle-finding method. Let $l = j - i$ denote the length of the cycle; then $f_m(s) \equiv f_{m+l}(s) \equiv f_{m+2l}(s) \equiv \ldots \bmod p$ for all $m \ge i$. Now let $m = l\lceil i/l \rceil$ be the smallest multiple of l greater than the length of the tail i: then $f_m(s) \equiv f_{2m}(s) \bmod p$ since $2m - m = m > i$ is a multiple of l. Moreover, $m \le j = O(\sqrt{p},)$.

This shows that we can recover p by computing the gcd of the numbers $f_m(s)$ and $f_{2m}(s)$ for an index m of order $O(\sqrt{p}\,)$.

The last problem is finding a suitable random function $f : \mathbb{Z}/n\mathbb{Z} \longrightarrow \mathbb{Z}/n\mathbb{Z}$. Linear functions $f(s) = as + b$ for constants $a, b \in \mathbb{Z}/n\mathbb{Z}$ do not work: for most choices of a and b, the cycles mod p have length considerably greater than $\sqrt{p}$. It is an experimental fact that functions $f(s) = s^2 - a$ for $a \ne 0, 2$ seem to behave sufficiently random.

Here's a simple `pari` program that illustrates Pollard's rho method:

```
{s=2;a=1;x=Mod(s,n);y=x;f=1;i=0;
 while(f,i=i+1;x=x^2+a;y=y^2+a;y=y^2+a;
 p=gcd(lift(y-x),n);if(p-1,print(i,"   ",p);f=0,))}
```

5.3 Modern Methods

There are quite a few factorization algorithms we have not discussed:

1. The $p + 1$-method, which is an analog of Pollard's $p - 1$-method, but is based on the group $\mathbb{F}_{p^2}^\times/\mathbb{F}_p^\times$ of order $p+1$ instead of the group $\mathbb{F}_p^\times$ of order $p - 1$. This method does not produce factorizations as spectacular as the $p-1$-method since it's steps take about 3 times as many operations as the steps in the $p - 1$-method.

2. Lenstra's elliptic curve method ECM is another analog of the $p-1$-method, this time based on the group of points on an elliptic curve with coordinates in $\mathbb{F}_p$. ECM is more powerful than the $p - 1$-method since varying the elliptic curve will produce lots of different groups; in the $p-1$-method, we only have $(\mathbb{Z}/p\mathbb{Z})^\times$ to work with.

3. Shanks' SQUFOF (square form factorization) is based on the theory of quadratic forms; the underlying group is the class group of forms of discriminant $-n$ or $-4n$. It is extremely efficient for integers of about 20 digits, but becomes hopelessly slow for larger integers.

4. Brillhart's continued fraction method CFRAC. This method attemtps to find integers x, y with $x^2 - y^2 \equiv 0 \bmod n$ by computing the continued fraction of $\sqrt{n}$. If such a pair is found, $\gcd(x - y, n)$ will be a (possibly trivial) factor of n. Thus we can factor n if we can find sufficiently many such pairs.

5. The quadratic sieve by Pomerance: instead of constructing solutions of $x^2 - y^2 \equiv 0 \bmod n$ using continued fractions, Pomerance computes e.g. $x_a \equiv (r + a)^2 \bmod n$ for integers $r = \lfloor\sqrt{n}\rfloor$ and small $a > 0$; factoring the x_a and combining relations eventually produces enough relations $x^2 \equiv y^2 \bmod n$ to factor n.

6. The fastest factorization method known so far for large integers is Pollard's number field sieve. As CFRAC and the quadratic sieve, this method finds solutions of $x^2 - y^2 \equiv 0 \bmod n$, this time by doing calculations in rings of integers of algebraic number fields.

We will come back to a few of these later. For now we just observe that we are very far from having a polynomial time factorization method, which means that factorization probably will remain difficult for quite some time.

Exercises

5.1 Use Fermat's method to find the factorization of $n = 15251$

5.2 Some factorization methods can be improved if it is known that the factors of n have a special form. For example, if p is an odd prime, then each divisor of the Mersenne number $M_p = 2^p - 1$ has the form $2kp + 1$ for some k. Use this improvement to factor M_{11} with trial division.

5.3 Assume that $n = pq$ for integers $p = 2km + 1$ and $q = 2lm + 1$.

1. Show that $n + 1 = 4klm^2 + p + q$.
2. Show that this implies $\frac{p+q}{2} \equiv \frac{n+1}{2} \bmod 2m^2$.
3. Show that $n = x^2 - y^2$ for $x = \frac{p+q}{2}$ and $y = \frac{p-q}{2}$.
4. Show that $x \equiv \frac{n+1}{2} \bmod 2m^2$.

Explain how to use this observation to improve Fermat's method.

5.4 Use the results of the preceding exercise to factor $n = \frac{10^{22}+1}{89\cdot 101}$.

5.5 Show that, in Lehman's algorithm, we have

$$2\sqrt{kn} \le a \le 2\sqrt{kn} + \frac{n^{1/6}}{4\sqrt{k}}.$$

5.6 Show that, in Lehman's algorithm, the number of iterations in the loops on k and a is at most $\sum_{i=1}^{B} \frac{n^{1/6}}{4\sqrt{k}}$, and that this is $O(n^{1/3})$.

5.7 In Lehman's algorithm, prove the congruences $a \equiv 1 \bmod 2$ if k is even (you will need to assume that $\gcd(r, s) = 1$), and that $a \equiv k + n \bmod 4$ if k is odd.

5.8 Show that $n = 568971935269420243703269723 21$ is a strong pseudoprime (i.e., passes Miller-Rabin) for $a = p$ for all primes $p \leq 29$.

Show that the primality tests reveal different square roots of $-1 \bmod n$; show how you can use this information to factor n.

Also use Fermat's method with multiplier $k = 3$ to factor the number. What do you observe?

5.9 Consider the integer $n = 10007030021$. Write a little `pari` program and apply Pollard's rho method with various functions $f(x) = x^2 + a$ and starting values c, and count how many iterations it takes to find the factorization.

Chapter 6

The Security of RSA

We have seen that finding large primes p and q with, say, 150 to 200 digits is relatively easy, but that factoring their product $n = pq$ is extremely difficult. This shows that breaking RSA by factoring n is impossible.

In this chapter we will look at other possibilities of breaking RSA.

6.1 Breaking RSA

Assume that $n = pq$ is a product of two odd primes, and that e is an integer coprime to $\phi(n)$. It is not known whether breaking RSA is equivalent to factoring the integer n; what we do know, however, is that knowledge of the decryption exponent d allows you to factor n:

Proposition 6.1. *From n, d and e we can efficiently compute the factorization of n.*

Proof. We have $de - 1 = s(p-1)(q-1)$ for some integer s. Pick an integer r coprime to n; then $r^{ed-1} \equiv 1 \bmod n$. Now set $x \equiv r^{(de-1)/2} \bmod n$; then $x^2 \equiv 1 \bmod n$, and $\gcd(x-1, n)$ will be a nontrivial factor of n if $x \not\equiv \pm 1 \bmod n$. If $x \equiv -1 \bmod n$, we pick a different r and start all over. If $x \equiv 1 \bmod n$, we set $y \equiv r^{(de-1)/4} \bmod n$ and check whether $\gcd(y-1, n) = 1$ or not. If the gcd is trivial, then $y \equiv \pm 1 \bmod n$. If $y \equiv -1 \bmod n$, pick a new r; if $y \equiv 1 \bmod n$, set $z \equiv r^{(de-1)/8} \bmod n$ etc. $\square$

This algorithm is a Las-Vegas algorithm: it does not necessarily terminate, but when it does, it gives you a correct answer. What can we say about the probability that $gcd(x-1, n)$ is trivial? Since n is an RSA-key, we know that $n = pq$ for primes p and q. If $x^2 \equiv 1 \bmod n$, then $x^2 \equiv 1 \bmod p$ and $x^2 \equiv 1 \bmod q$; but p and q are primes, so this implies $x \equiv \pm 1 \bmod p$ and $x \equiv \pm 1 \bmod q$. Thus x is the solution of one out of four systems of linear congruences. Here are the possibilities:

$$\begin{array}{lll} x \equiv +1 \bmod p & x \equiv +1 \bmod q, & \gcd(x-1,n) = pq \\ x \equiv +1 \bmod p & x \equiv -1 \bmod q, & \gcd(x-1,n) = p \\ x \equiv -1 \bmod p & x \equiv +1 \bmod q, & \gcd(x-1,n) = q \\ x \equiv -1 \bmod p & x \equiv -1 \bmod q, & \gcd(x-1,n) = 1 \end{array}$$

Thus the gcd is trivial in 50% of all cases.

Here's an example. Take $p = 72643$, $q = 32603$, and $n = 2368379729$. We pick the encryption exponent $e = 1720260823$ and find $d = 1197334939$.

Now `pari` shows that $2^{(de-1)/2} \equiv 1 \bmod n$ and $2^{(de-1)/4} \equiv -1 \bmod n$. Thus we try $r = 3$ next and find $3^{(de-1)/2} \equiv 1 \bmod n$ and $3^{(de-1)/4} \equiv 1759486102 \bmod n$. Now we get $\gcd(1759486101, n) = 32603$.

Our next result shows that everybody who knows $\phi(n)$ can also factor n:

Proposition 6.2. *If n is a product of two odd primes, then p and q can be efficiently computed from $\phi(n)$.*

Proof. Let $\Phi = \phi(n) = (p-1)(q-1)$; then $\Phi = n - (p+q) + 1$. Thus we know $pq = n$ and $s = p + q = n - \Phi + 1$. Now consider the quadratic polynomial $X^2 - sX + n = (X-p)(X-q)$. Using the quadratic formula, we can compute its roots p and q via $p = \frac{1}{2}(s + \sqrt{s^2 - 4n}\,)$ and $q = \frac{1}{2}(s - \sqrt{s^2 - 4n}\,)$. □

6.2 Using a Shared Modulus

Assume that a list of users want to use RSA to set up a secure communication system. In such a situation one might be tempted to use one and the same modulus N for all users, but different public encryption exponents e_i. It turns out, however, that this is not a good idea.

Assume first that one of the internal users, say user number 1, is the bad guy. In this case, the user knows the decryption e xponent d_1, and therefore he can compute the factorization $N = pq$ and decrypt all messages sent to other users.

Now assume that the attacker Eve comes from the outside. If Alice sends the same message m to two users with public keys (N, e_1) and (N, e_2), then Eve knows $c_1 \equiv m^{e_1} \bmod N$ and $c_2 \equiv m^{e_2} \bmod N$ If $\gcd(e_1, e_2) = 1$, then Eve computes integers $t_1 \equiv e_1^{-1} \bmod e_2$ and sets $t_2 = \frac{t_1 e_1 - 1}{e_2}$. Then she can compute the message m via

$$\begin{aligned} c_1^{t_1} c_2^{-t_2} &\equiv m^{e_1 t_1} m^{-e_2 t_2} \\ &\equiv m^{1+e_2 t_2} m^{-e_2 t_2} = m \bmod N. \end{aligned}$$

6.3 Other Attacks

- Wiener has described an attack on RSA that works if $N = pq$ with $p < q < 2p$ and $d < \frac{1}{3} N^{1/4}$. His idea uses continued fractions. This is another reason why small decryption exponents should be avoided. Apparently there are also attacks against systems that use the Chinese Remainder Theorem to decrypt ciphertexts.

- Timing attacks. This is from wikipedia:

> Kocher described a new attack on RSA in 1995: if the attacker Eve knows Alice's hardware in sufficient detail and is able to measure the decryption times for several known ciphertexts, she can deduce the decryption key d quickly. This attack can also be applied against the RSA signature scheme. In 2003, Boneh and Brumley demonstrated a more practical attack capable of recovering RSA factorizations over a network connection (e.g., from a Secure Socket Layer (SSL)-enabled webserver). This attack takes advantage of information leaked by the Chinese remainder theorem optimization used by many RSA implementations.

There are lots of attacks on the RSA cryptosystem that have been studied in the past 30 years. So far, every possible attack can be eliminated by a *careful* implementation and a clever choice of public and private keys.

Exercises

6.1 How can Eve break the RSA cryptosystem with shared modulus if $d = \gcd(e_1, e_2) > 1$?

6.2 Assume that Alice uses the shared modulus $N = 18923$ and the encryption exponents $e_1 = 11$ and $e_2 = 5$. Suppose Alice encrypts the same message m twice, as $c_1 = 1514$ and $c_2 = 8189$. Show how to compute the plaintext m.

Chapter 7

Discrete Logarithms

7.1 Cryptographic Tasks

The purpose of cryptography is not just encrypting messages for secure transmission; for tasks such as online banking, a lot more problems will have to be solved.

- Encryption: this is what we have seen so far; an encryption algorithm transforms the plaintext into a ciphertext that only the receiver can decrypt.
- Authentication: there are two kinds of authentication protocols; first there should be methods that allow you to sign a message in order to guarantee that no one can modify it on its way to the receiver; in addition, it must be possible to add a signature to a message that makes clear that the message comes from you and not from someone else.
- key distribution: before communicating over an insecure channel, keys have to be distributed among the participants. In the simplest case, two users will have to agree on a common key (this task is called key exchange).
- zero knowledge protocols: here a user has to convince another that he knows some "secret" without revealing any information that would allow the other user to learn this secret. This is perhaps best explained by an example: assume Alice has a composite integer $n = pq$ as part of her public RSA key. Bob wants Alice to prove that she knows the factorization, but of course Alice cannot reveal the factors.

7.2 Diffie-Hellman Key Exchange

Let p be a large prime number. We know that the group $(\mathbb{Z}/p\mathbb{Z})^\times$ is cyclic (it is the multiplicative group of the finite field $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$). A generator g (more precisely, it is the residue class $g + p\mathbb{Z}$ that generates the group) of this group is called a primitive root modulo p; it is known that there are $\phi(p-1)$ primitive roots mod p.

In the Diffie-Hellman key exchange protocol, two users Alice and Bob agree on a large integer (to be used as a private key in a communication), a common key, and they do this in such a way that this number remains unknown to someone who watches their communication.

Alice	Eve	Bob
Alice and Bob agree upon a prime p and a primitive root g mod p		
Alice picks a random $a \in (\mathbb{Z}/p\mathbb{Z})^\times$		Bob picks a random $b \in (\mathbb{Z}/p\mathbb{Z})^\times$
Alice computes $A \equiv g^a \bmod p$		Bob computes $B \equiv g^b \bmod p$
Alice sends A to Bob	$\xrightarrow{A}$	
	$\xleftarrow{B}$	Bob sends B to Alice
Alice computes $K \equiv B^a \bmod p$		Bob computes $K \equiv A^b \bmod p$

Figure 7.1: The Diffie-Hellman key exchange

Here's what they do: they pick a large prime number p and a primitive root g modulo p (by definition, the powers of g mod p generate the whole group $(\mathbb{Z}/p\mathbb{Z})^\times$; for example, 3 is a primitive root modulo 7, but 2 is not). The numbers p and g are public. Then Alice picks a random number a from $\{0, 1, \ldots, p-2\}$ (of course, the choices $a = 0$ or $a = 1$ will be catastrophic, but this happens with probability $\approx \frac{1}{p}$, that is, never at all), and Bob similarly chooses some number b from the same interval; the numbers a and b are kept secret.

Now Alice computes $A \equiv g^a \bmod p$ and sends A to Bob; Bob computes $B \equiv g^b \bmod p$ and sends B to Alice. Then Alice computes $K \equiv B^a \bmod p$, and Bob $K \equiv A^b \bmod p$; the common key then is $K \equiv g^{ab} \bmod p$.

If Eve was eavesdropping, she knows p, g, $A \equiv g^a \bmod p$ and $B \equiv g^b \bmod p$. If she could solve the discrete logarithm problem (DLP), that is, compute the exponent a from the knowledge of p, g and $g^a \bmod p$, then she could easily compute K. As of today, no fast way of solving this problem is known, except in special cases, such as for primes p such that $p-1$ is divisible by small primes only. It is also unknown whether computing $g^{ab} \bmod p$ from g, $g^a \bmod p$ and $g^b \bmod p$ is as hard as solving the DLP, in other words, whether there is a faster method of computing $g^{ab} \bmod p$ from these data than solving the DLP.

Attacks on Diffie-Hellman

Assume that Alice and Bob have agreed on a prime p and a primitive root g mod p, and that Eve can intercept their messages. Eve writes $p = rq + 1$ for some small r; when Alice and Bob send their values of A and B to each other, Eve replaces them with $A^q \bmod p$ and $B^q \bmod p$. Then the shared key will be $k \equiv g^{abq} \bmod p$. But since $k^r \equiv 1 \bmod p$, the shared key is one out of r solutions of the congruence $k^r \equiv 1 \bmod p$. Thus Eve computes them all by testing g^q, g^{2q}, ..., $g^{(r-1)q} \bmod p$.

7.3 Solving the DLP

There are various methods for solving the DLP, just as there are various methods for factoring composite integers. Here we will briefly discuss a few of them.

In the following, assume that G is a finite cyclic group of order n, and with generator g. Given an element $a \in G$, the DLP is to compute an integer x (or rather, a residue class $x \bmod n$) such that

$$a = g^x. \tag{7.1}$$

Enumeration

The simplest method, comparable to trial division for factoring, is called "enumeration". We simply test whether $x = 0, 1, 2, \dots, n-1$ satisfies (7.1). This clearly requires $O(n)$ steps, and each step consists of a multiplication in G. This is clearly not efficient if $n = \#G$ is sufficiently large.

Consider e.g. $G = (\mathbb{Z}/17\mathbb{Z})^\times$, a cyclic group of order 16 with generator $g = 3$. Assume we have to solve $8 \equiv 3^x \bmod 17$. Then we compute

x	0	1	2	3	4	5	6	7	8	9	10
g^x	1	3	9	10	13	5	15	11	16	14	8

Thus $x = 10$ solves this DLP.

Shanks BSGS

Shanks developed his baby-step giant-step algorithm in connection with computing the class group of quadratic forms, but it turned out that this method can be applied to calculations in quite general groups.

Here's the basic idea: choose $m = \lceil \sqrt{n} \rceil$ and write $x = qm + r$; the bsgs algorithm computes q and r in the following way. From $g^{qm+r} = a$ we get $g^{qm} = ag^{-r}$. Now we compute the baby-steps

$$B = \{(ag^{-r}, r) : 0 \le r < m\}.$$

If B contains an element of the form $(1, r)$, then we put $x = r$; if not, compute $d = g^m$ and test for $q = 1, 2, 3, \dots$ whether the group element d^q (these are called the giant steps) is the first component of an element in B. If (δ^q, r) is such a pair, then $ag^{-r} = d^q = g^{qm}$, and we have solved the DLP since $a = g^{qm+r}$.

We now claim that there is a suitable $q < m$; this means that the bsgs algorithm requires $O(m) = O(\sqrt{\#G})$ steps, which is a lot better than in the enumeration algorithm (on the other hand, we have to store the elements in

B, and these are $O(\sqrt{\#G}\,)$ elements). In fact, this follows from the trivial observation that $x = qm + r < n$, hence $qm < n$ and thus finally $q < \frac{n}{m} < m$.

Here's an example; for solving $8 \equiv 3^x \bmod 17$, we put $m = 5$ and compute the baby steps

$$B = \{(8,0),(14,1),(16,2),(11,3),(15,4)\}.$$

Now put $d = g^5 \equiv 5 \bmod 17$ and compute the giant steps $5^1 \equiv 5 \bmod 17$, $5^2 \equiv 8 \bmod 17$; here we have a match: $8 \equiv 5^2 \equiv (3^5)^2 \equiv 3^{10} \bmod 17$. Of course, this baby example it not big enough to show the superiority of the bsgs algorithm.

Pollard's ρ algorithm

Pollard's idea for finding small prime factors of integers can also be adapted to solve the DLP $a = g^x$. To see how, we partition G into three disjoint subgroups $G = G_1 \cup G_2 \cup G_3$ and define a "random"function $f : G \longrightarrow G$ by

$$f(b) = \begin{cases} gb & \text{if } b \in G_1, \\ b^2 & \text{if } b \in G_2, \\ ab & \text{if } b \in G_3. \end{cases}$$

Then we choose a random element $x_0 \in \{1, 2, \ldots, n\}$ and compute a sequence of group elements $b_0 = g^{x_0}$, $b_1 = f(b_0)$, $b_2 = f(b_1)$ etc.

All these elements can be written in the form

$$b_i = g^{x_i} a^{y_i}.$$

This is clear for $i = 0$, where x_0 is given and $y_0 = 0$, and follows by induction on i; in fact, we easily see that

$$x_{i+1} \equiv \begin{cases} x_i + 1 \bmod n & \text{if } b_i \in G_1, \\ 2x_i \bmod n & \text{if } b_i \in G_2, \\ x_i \bmod n & \text{if } b_i \in G_3, \end{cases}$$

and

$$y_{i+1} \equiv \begin{cases} y_i \bmod n & \text{if } b_i \in G_1, \\ 2y_i \bmod n & \text{if } b_i \in G_2, \\ y_i + 1 \bmod n & \text{if } b_i \in G_3, \end{cases}$$

Since G has finitely many elements, we eventually must find a match $b_j = b_i$ for some $j = i + k$. Then

$$g^{x_i} a^{y_i} = g^{x_j} a^{y_j},$$

that is,

$$g^{x_i - x_j} = a^{y_j - y_i}.$$

Since every element can be written in the form g^x for some unique $x \bmod n$, and because $a = g^x$, this implies

$$x_i - x_j \equiv x(y_j - y_i) \bmod n.$$

If $d = \gcd(y_j - y_i, n) = 1$, this congruence allows us to compute x. If d is small, we can determine $x \bmod \frac{n}{d}$ and test the finitely many possibilities for x. If d is large, we rerun the algorithm with a different choice of x_0.

By the birthday paradox, we expect a match after about $O(\sqrt{\#G}\,)$ steps (how much the individual steps cost depends of course on the cost for multiplying two group elements). On the other hand, we must so far store all the triplets (b_i, x_i, y_i), and their cardinality is about $O(\sqrt{\#G}\,)$. By using Floyd's cycle finding method, however, we get an algorithm in which we only need to store one triple at a time: if (b_i, x_i, y_i) is stored, compute triples for $i+1, \ldots, j$ until a match with indices i, j is found; at $j = 2i$, store $(b_{2i}, x_{2i}, y_{2i}i)$ etc. There are also variants in which two, four, or eight triplets are stored.

Here are the calculations for solving $8 \equiv 3^x \bmod 17$. First we pick $G_1 = \{1, 2, \ldots, 6\}$, $G_2 = \{7, \ldots, 12\}$ and $G_3 = \{13, \ldots, 17\}$. Pick $x_0 = 2$; then we find

i	b_i	x_i	y_i
0	9	2	0
1	13	4	0
2	2	4	1
3	6	5	1
4	1	6	1
5	3	7	1
6	9	8	1

Here we find the match; solving $2 - 8 \equiv x(1 - 0) \bmod 16$ gives $x \equiv -6 \equiv 10 \bmod 16$. As an exercise, continue the calculation until Floyd's cycle finding method finds a match.

There is also a variant of Pollard's ρ method, namely Pollard's λ-method; see the literature for a description.

7.4 Pohlig-Hellman

The DLP in a finite group G is used in cryptography for a wide variety of groups. Not every group can be used, though: clearly the DLP in the cyclic group $G = \mathbb{Z}/n\mathbb{Z}$ is trivial, since a congruence $gx \equiv a \bmod n$, given g, n and a, can easily be solved using the Euclidean algorithm. There are, however, a lot of groups for which

1. the DLP is hard;
2. the group operation can be computed efficiently.

Examples are the multiplicative groups of finite fields (or subgroups of these), or the groups of points on elliptic curves over finite fields.

For understanding the security issues in these examples it is important to know about possible ways of solving the DLP. The method of Pohlig-Hellman shows how to reduce the computation of the DLP in a (cyclic subgroup of a) finite group to a similar problem where the group order is prime. We proceed in two steps.

Reduction to prime power order

Let G be a cyclic group of order n, let g denote a generator of G, and write

$$n = \#G = \prod p^{e(p)}.$$

We would like to solve the DLP

$$a = g^x$$

in G. For each prime $p \mid n$ we define

$$n_p = np^{-e(p)}, \quad g_p = g^{n_p}, \quad \text{and} \quad a_p = a^{n_p}.$$

Then g_p has order n_p, and we have

$$g_p^x = a_p.$$

Let $x(p)$ denote a solution of this DLP; then $x \equiv x(p) \bmod p^{e(p)}$, and by invoking the Chinese Remainder Theorem we can compute x from the individual $x(p)$:

Lemma 7.1. *In the situation above, let x be a solution of the linear system of congruences $x \equiv x(p) \bmod p^{e(p)}$. Then $a \equiv g^x \bmod n$.*

Proof. We have $(g^{-x}a)^{n_p} = g_p^{-x(p)} a_p = 1$ for all $p \mid n$. Thus the order of $g^{-x}a$ divides n_p for all such p. Since the n_p are coprime, this order must be 1. □

One possible way to compute the DLP in groups of prime power order would be the baby-step giant-step algorithm; if $\#H = p^{e(p)}$, this will take about $p^{e(p)/2}$ steps. If $n = \#G$ has at least two prime factors, this already a lot faster than applying the bsgs method to the full group G.

Reduction to prime order

Assume now that $\#G = p^e$ is a prime power. We want to solve $a = g^x$ for x; to this end, we write x in the form

$$x = x_0 + x_1 p + x_2 p^2 + \ldots + x_{e-1} p^{e-1}$$

for uniquely determined integers $0 \le x_i < p$. The goal is to show that the x_i can be determined by solving a suitable DLP in some group of order p.

To this end we raise the equation $a = g^x$ to the power p^{e-1}; then $g^{p^{e-1}x} = a^{p^{e-1}}$. Since $g^{p^{e-1}}$ generates a group of order p, we can solve this and find the residue class $x_0 \equiv x \bmod p$. This gives us x_0.

Now we proceed by induction. Assume that $x_0, \ldots, x_j$ have already been determined. Then we know that

$$g^{x_j p^j + \ldots + x_{e-1} p^{e-1}} = a g^{-(x_0 + x_1 p + \ldots + x_{j-1} p^{j-1})} =: a_i.$$

Raising this to the power p^{e-j-1} shows that

$$\Big(g^{p^{e-1}}\Big)^{x_i} = a_i^{p^{e-1}}.$$

Again, $g^{p^{e-1}}$ has order p, and solving the DLP in the group of order p generated by this element gives us $x_j \bmod p$.

Analysis

How long does it take to solve the DLP using Pohlig-Hellman? We assume that we know the prime factorization $n = \prod p^{e(p)}$. Then we have to compute the elements $g_p = g^{n_p}$ and $a_p = a^{n_p}$ for all primes dividing n. This requires $O(\log n)$ group operations. Then we compute the $x_j(p)$ using Pollard's ρ or Shanks' bsgs methods, which requires $O(\sqrt{p}\,)$ group operations for each p. Putting everything together with the Chinese Remainder Theorem is done within $\mathbb{Z}/n\mathbb{Z}$ and does not require any group operation; we know it requires $O((\log n)^2)$ bit operations. Thus the number of group operations required to solve the DLP is

$$O\Big(\sum_{p \mid n} (e(p)(\log n + \sqrt{p}\,))\Big).$$

This estimate is dominated by the terms $\sqrt{p}$. If $n = \#G$ is only divisible by small primes, the Pohlig-Hellman method solves the DLP very efficiently. Thus using special primes of the form $1 + k \cdot 2^n$ for small k and large n, for example, belongs to the "stupid things to do"-section in cryptographic schemes based on the difficulty of the DLP.

7.5 Index Calculus

The fastest known method for solving the DLP in the groups $(\mathbb{Z}/p\mathbb{Z})^\times$ (assuming $p-1$ has at least one large prime factor) is the Index Calculus. Assume we have to solve the DLP $g^x \equiv a \bmod p$. We pick a bound B and determine the "factor base" $F(B)$, which consists of all the primes $q \le B$. We say that an integer b is B-smooth if its prime factors are all $\le B$.

The Index Calculus proceeds in two steps:

1. Solve the DLP $g^{x(q)} \equiv q \bmod p$ for all $q \in F(B)$.

2. Find an exponent y such that $ag^y \bmod p$ is B-smooth.

Thus we can write

$$ag^y \equiv \prod_{q \in F(B)} q^{e(q)} \bmod p.$$

But then

$$ag^y \equiv \prod_{q \in F(B)} q^{e(q)} \equiv \prod_{q \in F(B)} g^{x(q)e(q)} \equiv g^{\sum x(q)e(q)} \bmod p,$$

which in turn implies

$$x \equiv -y + \sum_{q \in F(B)} x(q)e(q) \bmod (p-1).$$

It remains to explain how to perform the two steps above. The second step is easy to explain: just pick a random y and check whether ag^y factors over the factor base; if not, pick a new y.

Step 1 is performed similarly: we pick random exponents k and test whether $g^k \bmod p$ factors over the factor base. Assume that $g^k \equiv \prod_q q^{f(q,k)} \bmod p$; since we are trying to solve $g^{x(q)} \equiv q \bmod p$, we find $g^k \equiv \prod_q q^{x(q)f(q,k)} \bmod p$, hence $k \equiv \sum_q x(q)f(q,k) \bmod p-1$. Thus each relation gives us a congruence mod $p-1$ for $x(q)$. If we have more congruences than primes in the factor base, we can solve this system and compute the $x(q)$.

Here's an example: let us solve the DLP $57 \equiv 2^x \bmod 67$. We pick the factor base $F = \{2,3,5\}$ and compute a few random powers of 2; the following factor:

$$\begin{aligned} 2^{31} &\equiv 50 \equiv 2 \cdot 5^2 \bmod 67, \\ 2^{41} &\equiv 50 \equiv 2^2 \cdot 3 \bmod 67, \\ 2^{15} &\equiv 5 \equiv 5^1 \bmod 67, \\ 2^{12} &\equiv 9 \equiv 3^2 \bmod 67 \end{aligned}$$

These relations give us the congruences

$$\begin{aligned} x(2) + 2x(5) &\equiv 31 \bmod 66, \\ 2x(2) + x(3) &\equiv 41 \bmod 66, \\ x(5) &\equiv 15 \bmod 66, \\ 2x(3) &\equiv 12 \bmod 66 \end{aligned}$$

Here we are lucky in that we already have $x(5) \equiv 15 \bmod 66$; this shows that $x(2) \equiv 1 \bmod 66$ and $x(3) \equiv 39 \bmod 66$. (In less lucky circumstances we factor $66 = 2 \cdot 3 \cdot 11$ and solve each of the resulting systems of congruences mod 2, 3 and 11, and then apply the Chinese Remainder Theorem). Note that we now know $2^1 \equiv 2 \bmod p$, $2^{39} \equiv 3 \bmod p$, and $2^{15} \equiv 5 \bmod p$.

Now we find a 6-smooth number $57 \cdot 2^y \bmod 67$; we easily get $57 \cdot 2^2 \equiv 27 \equiv 3^3 \bmod p$. This shows $57 \cdot 2^2 \equiv (2^{39})^3 = 2^{117} \equiv 2^{51} \bmod p$ since $117 \equiv 51 \bmod 66$. But then $57 \equiv 2^{51-2} = 2^{49} \bmod p$.

Alice	Eve	Bob
Alice and Bob agree upon a prime p and a primitive root g mod p		
Alice picks $a \in (\mathbb{Z}/p\mathbb{Z})^\times$ at random, and then computes $A \equiv g^a \bmod p$.		
Alice sends A to Bob	$\xrightarrow{A}$	
		Bob picks $b \in (\mathbb{Z}/p\mathbb{Z})^\times$ at random, computes $B \equiv g^b \bmod p$ and encrypts his message m via $c \equiv mA^b \bmod p$
	$\xleftarrow{B, c}$	Bob sends B and c to Alice
Alice decrypts the message by computing $m \equiv cB^{p-1-a} \bmod p$		

Figure 7.2: The ElGamal public key cryptosystem

7.6 Cryptosystems based on DLP

ElGamal

There is also an encryption protocol based on DLP, called ElGamal. It works like this: Alice picks p, g, a as before, and computes $A \equiv g^a \bmod p$. Her public key is (p, g, A), her secret key is a. In order to encode messages $m < p$, Bob picks some b as before, computes $B \equiv g^b \bmod p$, and publishes (p, g, B); then he computes $c \equiv A^b m \bmod p$ and sends c to Alice.

At this point, Alice knows B and c, as well as her secret key a. She puts $f = p - 1 - a$ and computes

$$B^f c \equiv g^{b(p-1-a)} A^b m \equiv g^{-ab} g^{ab} m \equiv m \bmod p$$

since $g^{p-1} \equiv 1 \bmod p$.

Shamir's no-key protocol

The following cryptosystem was described in an unpublished mansucript of Shamir; it is also called Massey-Omura, and was first discovered by M. Williamson but not published since he was working at the GCHQ (a British Intelligence service) at the time.

Figure 7.3 explains the protocol, which is clearly based on the difficulty of the discrete log problem.

The advantage of Shamir's no-key protocol is the fact that no keys are involved; the main disadvantage is that for sending one message, Alice has to send two ciphertexts, and wait for Bob's ciphertext to arrive.

Alice	Eve	Bob
Alice and Bob agree upon a prime p		
Alice picks a pair of integers a, a^{-1} with $aa^{-1} \equiv 1 \bmod p-1$		Bob picks a pair $b, b^{-1} \in (\mathbb{Z}/(p-1)\mathbb{Z})^\times$ with $bb^{-1} \equiv 1 \bmod p-1$.
Alice computes $x \equiv m^a \bmod p$ and sends x to Bob.	$\xrightarrow{x}$	
	$\xleftarrow{y}$	Bob computes $y \equiv x^b \bmod p$ and sends y to Alice
Alice computes $z \equiv y^{a^{-1}} \bmod p$ and sends z to Bob	$\xrightarrow{z}$	
		Bob decrypts the message by computing $m \equiv z^{b^{-1}} \bmod p$

Figure 7.3: Shamir's no-key protocol

Another problem is the following: assume Eve can intercept the messages and change them. She intercepts Alice's $x \equiv m^a \bmod p$, picks integers c, c^{-1} with $cc^{-1} \equiv 1 \bmod p-1$, and sends $y_1 \equiv x^c \bmod p$ back to Alice, who will then compute $z_1 \equiv y_1^{a^{-1}} \bmod p$ and send it away; Eve intercepts this message, too, and computes $m \equiv z_1^{c^{-1}} \equiv \bmod p$.

Exercises

7.1 Solve the DLP $6 \equiv 2^x \bmod 101$ using enumeration, bsgs, Pollard's rho method, and Pohlig-Hellman.

7.2 Show that the sequence of b_i is periodic after a match has occurred.

7.3 This exercise explains why Floyd's cycle finding method works. Let s and $s+c$ denote the smallest indices with $b_s = b_{s+c}$; then the preperiod (the tail of the ρ) has length s, and the cycle has length c.

1. Let $i = 2^j$ be the smallest power of 2 with $2^j \geq s$. Show that b_i is inside the cycle.
2. If $i = 2^j \geq c$, show that one of the elements b_{i+1}, b_{i+2}, ..., b_{2i} is equal to b_i.

7.4 Solve the DLP $8 \equiv 3^x \bmod 17$ using Pohlig-Hellman.

Chapter 8

Basic Cryptographic Protocols

8.1 Authentication

RSA signatures

Assume that Alice and Bob each have their own RSA keys; the pairs (n_A, e_A) and (n_B, e_B) are public, the factorizations of n_A and n_B, as well as the decryption exponents d_A and d_B are secret. Then Alice and Bob can even sign their messages in such a way that Alice can verify whether a message she receives really did come from Bob or from someone else.

The basic idea is the following: assume that Alice and Bob have chosen her public keys (N_A, e_A) and (N_B, e_B) as well as her private keys d_A and d_B. Then instead of sending $c \equiv m^{e_A} \bmod N_A$ to Alice, Bob encrypts his messages twice by computing $c' \equiv c^{d_B} \bmod N_B$.

When Alice receives c', she first computes $c \equiv (c')^{e_B} \bmod N_B$ using Bob's public key, and then decrypts c with her own private key. Since only Bob knows d_B, and since this knowledge was necessary for the successful encryption, Alice concludes that the message must have come from Bob.

Unfortunately, this does not work in practice: although Alice can compute $c \equiv (c')^{e_B} \bmod N_B$, the smallest positive integer c representing this residue class is not necessarily the same as the one representing $c \equiv m^{e_A} \bmod N_A$; all we know is that they are congruent modulo N_B. If N_A has 302 digits and N_B only 300, then $m^{e_A} \bmod N_A$ will almost always be represented by an integer $> N_B$.

There are two solutions of this problem: instead of encrypting the messages m themselves, we can choose to encrypt only the values $h(m)$ for a suitably chosen hash function h with values $< \min\{N_A, N_B\}$. Another solution is the following: for signing the message m, Bob computes $s \equiv m^{d_B} \bmod N_B$ and then sends (m, s) to Alice. Alice then uses Bob's public key to check that

$m \equiv s^{e_B} \bmod N_B$. Since only Bob knows d_B, Alice is sure that the message must have come from Bob; in addition, Bob later cannot deny to have sent the message.

This protocol cannot be used for sending RSA-encrypted messages (e.g. by computing $c \equiv m^{e_A} \bmod N_A$ and sending (s, c)) since the message can be retrieved from the signature using Bob's public key.

Alice	Eve	Bob
Alice and Bob agree upon a hash function h		
picks two large primes p_A, q_A		picks two large primes p_B, q_B
computes $N_A = p_A q_A$		computes $N_B = p_B q_B$
picks random $e_A \in (\mathbb{Z}/\phi(N_A)\mathbb{Z})^\times$		picks random $e_B \in (\mathbb{Z}/\phi(N_B)\mathbb{Z})^\times$
solves $d_A e_A \equiv 1 \bmod \phi(N_A)$		solves $d_B e_B \equiv 1 \bmod \phi(N_B)$
publishes public key (N_A, e_A)	$\overset{(N,e)}{\longleftrightarrow}$	publishes public key (N_B, e_B)
		computes $c \equiv m^{e_A} \bmod N_A$
		computes $h \equiv h(m)^{d_B} \bmod N_B$
	$\overset{(c,h)}{\longleftarrow}$	sends (c, h) to Alice
computes $m \equiv c^{d_A} \bmod N_A$		
computes $H \equiv h^{e_B} \bmod N_B$		
verifies that $H = h(m)$		

Figure 8.1: The RSA signature protocol

Hash Functions

A hash function adds some sort of "signature" to a message; a very simple hash function would be the map sending a message m to its parity: add up the bits of m modulo 2. In practice, hash functions h are often required to have the following properties:

- a hash function should map a message of arbitrary lenght to a number with a fixed number of bits;
- for any message m, the value $h(m)$ is easy to compute;
- collision resistant: it is difficult to find two messages m, m' with $h(m) = h(m')$;
- preimage resistant: given a hash value y, it is difficult to find some m with $h(m) = y$.

In practice, hash functions map messages of arbitrary bit length to values of fixed bit length. Rivest constructed hash functions called MD4 and MD5, and the functions SHA-0 and SHA-1 constructed later, which were based on similar principles, produce a 160-bit digest from messages with at most 2^{64} bits.

DSA

The Digital Signature Algorithm (DSA) is based on the DLP. To set up the scheme, each user proceeds as follows:

1. Pick a random prime N of about 160 bits;
2. choose a random prime $p \equiv 1 \bmod N$ with about 1000 bits;
3. choose a generator g of the cyclic subgroup of $\mathbb{F}_p^\times$ of order N;
4. choose a random x with $0 < x < N$, and compute $y \equiv g^x \bmod p$.
5. The secret key is x, the public key is (p, N, g, y).

For finding a suitable g, pick a random integer h and compute $g = h^{(p-1)/N} \bmod p$; if this is $\neq 1$, the element g will have order N: clearly $g^N \equiv h^{p-1} \equiv 1 \bmod p$, so the order of $g \bmod p$ divides the prime N, and is > 1.

If Alice wants to sign a message m, she picks a hash function h with values $< N$ and computes $H = h(m)$. Then she chooses a random integer k with $0 < k < N$, computes the smallest positive remainder $\overline{g}$ of $g^k \bmod p$, and then sets $r \equiv \overline{g} \bmod N$. Finally, Alice finds an integer s such that $sk \equiv H + xr \bmod N$. She then signs the message m with the pair (r, s) of integers modulo N.

When Bob receives the encrypted message, he first decrypts it by computing m. Then he verifies Alice's signature as follows. Bob computes the hash value $H = h(m)$, $u \equiv s^{-1}H \bmod N$, $v \equiv s^{-1}r \bmod N$, and finally $\overline{g} \equiv g^u y^v \bmod p$. Since $g^u y^v \equiv g^{u+xv} \equiv g^k \bmod p$, he will find that $r \equiv \overline{g} \bmod N$.

Since the only known way of computing the signature (r, s) is by using x, which can be found by solving the DLP $y \equiv g^x \bmod p$, Bob concludes that the message must have been sent by Alice.

The advantage of DSA over the RSA signature scheme is that RSA-signatures that are about as secure as DSA-signatures are usually 3 times as long.

8.2 Zero Knowledge Proofs

FLIP stands for Fast Legendre Identification Protocol and was suggested recently by Banks, Lieman & Shparlinski. Assume Alice has an RSA-modulus $n = pq$ and wants to convince Bob that she knows p and q (without telling him what p and q are, of course). The protocol I will present in Fig. 8.2 is a radically simplified version of the original one, and very close to the Fiat-Shamir protocol that I will discuss next.

After running through the protocol, Bob will be convinced that Alice knows a prime factor p of her n, because there is no way known for computing the b_i except the one involving p.

Moreover, Eve cannot impersonate Alice: even if she knows n, a, and the vectors $(C_1, \ldots, C_r)$ and $(b_1, \ldots, b_r)$, this will not help her produce the bits $(b_1', \ldots, b_s')$ from a new set $(C_1', \ldots, C_r')$ that Bob sends to Alice, and which Eve intercepts.

In fact, we have

$$\Big(\frac{C_i}{n}\Big) = \Big(\frac{c_i^2 a^{b_i}}{n}\Big) = \Big(\frac{c_i}{n}\Big)^2 \Big(\frac{a}{n}\Big)^{b_i} = 1$$

since $(\frac{a}{n}) = (\frac{a}{p})(\frac{a}{q}) = (-1)^2 = +1$. Now for any integer C with $(\frac{C}{n}) = +1$, there are two possibilities:

1. We have $(\frac{C}{p}) = (\frac{C}{q}) = +1$; then C is a square mod p and mod q, and the Chinese Remainder Theorem shows that C is a square mod n and has four distinct roots.
2. We have $(\frac{C}{p}) = (\frac{C}{q}) = -1$; then C is a nonsquare mod p, hence a nonsquare mod n.

Determining the bit b with $(\frac{C}{p}) = (-1)^b$ is thus equivalent to finding out whether C is a square mod n or not. This is yet another fundamental difficult problem in number theory, just as factoring and solving the DLP, and there are numerous cryptographic protocols based on it.

Alice	Eve	Bob
Alice picks large random primes p, q, puts $n = pq$, and randomly chooses an integer a such that $(\frac{a}{p}) = (\frac{a}{q}) = -1$. She then publishes her public key (n, a)		
	$\xrightarrow{(n,a)}$	
		Bob picks $c_1, \ldots, c_r \in (\mathbb{Z}/n\mathbb{Z})^\times$ and some random vector $(b_1, \ldots, b_r) \in \mathbb{F}_2^r$. He computes $C_i \equiv c_i^2 a^{b_i} \bmod n$.
	$\xleftarrow{C_1, \ldots, C_r}$	Bob sends $C_1, \ldots, C_r$ to Alice.
Alice computes $(C_i/p) = (-1)^{b_i}$ and sends $(b_1, \ldots, b_r)$ to Bob.	$\xrightarrow{(b_1, \ldots, b_r)}$	Bob verifies that Alices vector $(b_1, \ldots, b_r)$ coincides with his.

Figure 8.2: FLIP

Finally, this is an *honest verifier* zero knowledge scheme: if Bob is honest, he learns nothing from the communication with Alice about the factorization of n that he did not know; after all, he only gets back the bits b_i that he sent to Alice in the first place (note, however, that at the beginning of the protocol he is given an integer a with $(\frac{a}{p}) = (\frac{b}{p}) = 1$, which contains a little bit of information on p and q; as long as this integer is part of the public key (and is not chosen anew in each protocol), this is not considered to be a leak of information during the communication).

If Bob is dishonest, however, he could simply send Alice, say, the integers 2, 3, 5, 7, $\ldots$; then he would learn from Alice's response the values $(\frac{2}{p})$, $(\frac{3}{p})$, $\ldots$, which he did not know before. So even if he cannot compute p from this knowledge, the protocol cannot be considered to be zero knowledge in this case.

In fact it is known that the first $O((\log p)^2)$ Legendre symbols $(\frac{a}{p})$ determine p, but no one so far has been able to come up with an efficient algorithm to compute p from these values.

Oblivious Transfer Protocol

Assume that Alice has a secret s. In an oblivious transfer protocol, Alice can convince Bob that she knows s, but in half of the cases, Bob will be able to learn the secret s. Fig. 8.3 explains how it works.

Alice	Eve	Bob
Alice picks two large primes p and q, generates the RSA public key (n, e), and computes $c \equiv s^e \bmod n$.		
	$\xrightarrow{(e,n,c)}$	
		Bob picks a random x, computes $a \equiv x^2 \bmod n$, and sends a to Alice.
	$\xleftarrow{a}$	
Alice computes the 4 square roots of $a \bmod n$, and sends one of them, say y, to Bob.		
	$\xrightarrow{y}$	
		Bob verifies that $y^2 \equiv a \bmod n$.

Figure 8.3: Oblivious transfer protocol

If Bob receives the square roots x or $n - x$ of $a \bmod n$, he learns nothing about s. If, however, $y^2 \equiv a \bmod n$ with $y \not\equiv \pm x \bmod n$, then $\gcd(y - x, n)$ will be a prime factor of n; from the prime factorization of n Bob is then able to compute the decryption exponent d as well as the secret $s \equiv c^d \bmod n$. Observe that Bob learns the secret with probability $\frac{1}{2}$, and that Alice does not know whether Bob knows about s or not. There are situations in which this outcome is desirable (of course not if keeping s secret is important for Alice).

We already know that computing square roots modulo $n = pq$ is at least as difficult as factoring n; in the next section we will see that computing square roots modulo primes p can be done efficiently. Since Alice knows the factorization of n, she can compute the square roots of $a \bmod p$ and $a \bmod q$, and then combine these with the Chinese Remainder Theorem to get the four square roots of $a \bmod n$.

In order to prevent Bob from learning the secret s, the above protocol has to be modified: see Fig. 8.4.

Alice	Eve	Bob
Alice picks two large primes p and q, generates the RSA public key (n, e), and computes $c \equiv s^e \bmod n$.	$\xrightarrow{(e,n,c)}$	
Alice picks a random integer a with $\sqrt{n} < a < n$ and computes $b \equiv a^2 \bmod n$.		Bob picks a random integer c with $\sqrt{n} < c < n$, computes $d \equiv c^2 \bmod n$.
Alice sends b to Bob.	$\xleftrightarrow{b,d}$	Bob sends d to Alice.
Alice solves $y^2 \equiv bd \bmod n$ and sends y to Bob.	$\xrightarrow{y}$	
		Bob verifies that $y^2 \equiv bd \bmod n$.

Figure 8.4: Modified Version of the OTP

In this protocol, Alice also picks a random square b, and she does not compute the square root of d (which sometimes allows Bob to factor n) but the square root of bd. Since Bob knows a square root of d, he will be able to find a square root of b as well. But as long as he does not know Alice's a, this will tell him nothing about the prime factors of n.

Note that in this protocol it is important that b and d are exchanged simultaneously. In fact, if Bob knows Alice's b beforehand, he could simply send $d = b$ back to Alice, who would then compute a square root of $bd = b^2$ modulo n; with probability $\frac{1}{2}$ this square root is $\neq \pm b$, and Bob will be able to factor n. (Of course Alice will get suspicious if Bob sends her the same b she sent to him; how can Bob prevent this?)

If, on the other hand, Alice knows Bob's d before she sends him b, then Malice can impersonate Alice in the following way: Malice selects a random integer r and sends Bob $b = dr^2$. Then Bob demands to see a square root y of $bd \bmod n$, but Malice can simply take $y \equiv dr \bmod n$.

The Fiat-Shamir Zero-knowledge Protocol

Consider the following problem: Alice knows a secret number s (some password, for example) and needs to convince Bob that she knows s without revealing any information about s. This is indeed possible; Fig. 8.5 explains how it works.

Alice	Eve	Bob
Alice picks two large primes p and q, and computes $n = pq$ and $v \equiv s^2 \bmod n$. She keeps p, q and s secret and publishes n and v.		
Now Alice picks a random $r \in (\mathbb{Z}/n\mathbb{Z})^\times$, computes $x \equiv r^2 \bmod n$, and sends x to Bob.	$\xrightarrow{x}$	
		Bob picks a bit $b \in \{0, 1\}$ at random.
	$\xleftarrow{b}$	
Alice computes $y \equiv r \cdot s^b \bmod n$.		
Alice sends y to Bob.	$\xrightarrow{y}$	
		Bob checks that $y^2 \equiv x{\cdot}v^b \bmod n$.

Figure 8.5: Fiat-Shamir zero-knowledge protocol

Note that $y^2 \equiv (rs^b)^2 \equiv r^2v^b \equiv x \cdot v^b \bmod n$. Now assume that Malice wants to cheat by pretending she is Alice. She does not know Alice's secret s, nor her primes p and q, but she knows the public $n = pq$ and $v \equiv s^2 \bmod n$. If Malice knew in advance that Bob would send the bit $b = 0$, she could send $x \equiv r^2 \bmod n$, and, after Bob sent back $b = 0$, send $y \equiv r \cdot s^0 \equiv r \bmod n$. If, however, she knew that Bob would send $b = 1$, Malice would send $x \equiv r^2v^{-b} \bmod n$ and, after Bob sent back $b = 1$, send $y \equiv r \bmod n$ since then $y^2 \equiv x \cdot v^b \bmod n$. This shows that, with probability $p = \frac{1}{2}$, Malice can successfully pretend she knows the secret s. Repeating the procedure with 20 random choices of r and b, however, will reduce the probability of cheating to 2^{-20}, which is less than 10^{-6}.

8.3 Secret Sharing

Assume that Alice wants to save a password she uses e.g. for online banking so that she can retrieve it in case she forgets it. She could give half of the bits of the password to Bob and the other half to Cathy; but this gives them a lot of information about the password. Here is a way of avoiding this leak of information: let w be the password; Alice picks a random integer $r < w$ and sends r to Bob and $w - r$ to Cathy. Now Alice has successfully split the information in such a way that each piece contains no information whatsoever about w.

It is clear how Alice has to proceed if she wants to split the password between three (or, more generally, any $n \geq 2$) people she trusts. If she has to retrieve her password, however, she will need to establish a contact with all n of them. Is there also a way of splitting the password between n trustees in such a way that she can retrieve the password as soon as she contacts m of them, where $2 \le m \le n$ is a fixed integer?

The answer is yes; the protocols that allow Alice to achieve this are called secret sharing protocols. Here are a few ideas of how to proceed.

The problem to be solved is the following: Alice has a secret w she would like to split between n trustees in such a way that

1. every group of m trustees can recover w;
2. fewer than m trustees cannot recover any information about w.

Linear Algebra

Let w be the password. Alice introduces variables $x_1, \ldots, x_m$ and produces a set of n linear equations with integral coefficients and the property that

1. all of them have $x_1 = w$ as a solution;
2. any set of m equations is linearly independent.

As soon as m trustees share their information, they solve their system of m linear equations and recover w.

Problem: if $m - 1$ of them get together, they know $m - 1$ linear equations for the x_i; if the parameters of this protocol are chosen too small, they might be ablo to compute a small list of possible (integral) solutions for x_1 and try them one after the other. To prevent this, Alice might pick a prime $p > w$ and work with equations over $\mathbb{F}_p$; then any $m-1$ trustees will not be able to deduce anything about w from their knowledge since any choice of x_1 can be extended to a solution of these $m - 1$ linearly independent equations over $\mathbb{F}_p$.

Chinese Remainder

Here Alice picks a prime $p > w$ and pairwise coprime moduli $M_1, \ldots, M_n > \sqrt[m]{p}$. Then she computes $c_i \equiv w \bmod M_i$ and distributes the c_i and p. As soon as m trustees get together, they can solve their system of linear congruences; since the product of their moduli is $> p$, they will be able to compute w as the smallest positive solution of this system using the Chinese Remainder Theorem.

Lagrange Interpolation

The basic idea is the following: assume Alice wants to solve the problem above with n and $m = 2$. She interprets w as the slope of a line $y = wx + b$ and gives each trustee a point P_j on this line. Then any two can recover the slope and determine w.

8.4 Online Poker

Alice and Bob want to play poker online, or on the telephone. Here we will explain how to begin with a fair deal. The idea is the following: Bob puts each of the 52 cards in a box, locks them, and sends them to Alice. Alice selects 10 boxes five for herself (on which she puts her own padlocks) and five for Bob. She returns the 10 boxes to Bob, who removes his locks from all 10 of them. He keeps the 5 unlocked boxes and takes out his cards, and sends the other 5 back to Alice, who unlocks them and takes out her cards.

Here's how to do this in practice. First, every card is encoded by some small number, for example

$2\spadesuit = 0201$	$3\spadesuit = 0301$	$\ldots$	$A\spadesuit = 1401$
$2\clubsuit = 0202$	$3\clubsuit = 0302$	$\ldots$	$A\clubsuit = 1402$
$2\diamondsuit = 0203$	$3\diamondsuit = 0303$	$\ldots$	$A\diamondsuit = 1403$
$2\heartsuit = 0204$	$3\heartsuit = 0304$	$\ldots$	$A\heartsuit = 1404$

Next, Alice and Bob agree on a large prime p. Alice selects a random integer a coprime to $p-1$ and computes $aa' \equiv 1 \bmod p-1$; similarly, Bob selects a random b and solves $bb' \equiv 1 \bmod p-1$.

In order to lock away the cards $m_1, \ldots, m_{52}$, Bob computes $c_j \equiv m_j^b \bmod p$ and sends the c_j to Alice, who selects 10 cards at random, locks away 5 of them by raising them to the a-th power mod p, and sends the 10 cards $d_1, \ldots, d_{10}$ back to Bob. Bob computes $C_j \equiv d_j^{b'} \bmod p$; five of these cards C_j (those not locked up by Alice) will be of the form m_j: this is his hand; the other five cards C_k will look like random numbers mod p. He sends them back to Alice, who will unlock them using her a'.

Question: where have you seen this protocol before?

8.5 Number Theoretic Background

Quadratic Residues

We first recall the basic properties of the Legendre symbol $(\frac{a}{p})$. It is defined for odd primes p and integers a coprime to p by

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a \equiv x^2 \bmod p, \\ -1 & \text{otherwise.} \end{cases}$$

Euler's criterium states that

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p.$$

It is now easy to check that $(\frac{a}{p}) = (\frac{b}{p})$ if $a \equiv b \bmod p$, and that $(\frac{a}{p})(\frac{b}{p}) = (\frac{ab}{p})$. The most important result about the Legendre symbol is the quadratic reciprocity law

$$\Big(\frac{p}{q}\Big)\Big(\frac{q}{p}\Big) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}}$$

for distinct odd primes p and q.

If $b = \prod p$ is a product of primes, and if $\gcd(a,b) = 1$, then we define the Jacobi symbol

$$\Big(\frac{a}{b}\Big) = \prod \Big(\frac{a}{p}\Big).$$

If $a \equiv x^2 \bmod b$, then $(\frac{a}{b}) = +1$; it is not true, however, that $(\frac{a}{b}) = +1$ always implies that a is a square modulo n. The reason for this formal definition of the Jacobi symbol is that it also satisfies the reciprocity law

$$\Big(\frac{a}{b}\Big)\Big(\frac{b}{a}\Big) = (-1)^{\frac{a-1}{2}\frac{b-1}{2}}.$$

Square Roots modulo p

There is an algorithm going back to Tonelli, which was improved by Shanks, that computes square roots mod p in polynomial time:

Proposition 8.1. *Let p be an odd prime, and assume that $(\frac{a}{p}) = +1$. Then the algorithm of Tonelli-Shanks solves $x^2 \equiv a \bmod p$ in $O((\log p)^4)$ bit operations.*

Proof. The complexity of the algorithm depends on the largest power of 2 dividing $p-1$, so the simplest case concerns primes $p \equiv 3 \bmod 4$.

Assume that $p = 4m-1$ and put $x = a^m$; then $x^2 \equiv a^{2m} = a^{\frac{p-1}{2}} a \equiv (\frac{a}{p})a = a \bmod p$, hence x is a square root of $a \bmod p$. The number of bit operations needed is clearly bounded by $O((\log p)^3)$ in this case.

The case $p \equiv 5 \bmod 8$ is slightly more tricky: assume as before that $(\frac{a}{p}) = +1$.

1. If $a^{(p-1)/4} \equiv 1 \bmod p$, then $x = a^{(p+3)/8}$ solves the congruence $x^2 \equiv a \bmod p$.
2. If $a^{(p-1)/4} \equiv -1 \bmod p$, then $x \equiv 2a(4a)^{(p-5)/8} \bmod p$ works.

This is easy to check.

Here's the actual algorithm.

1. Write $p - 1 = 2^e s$ for some odd integer s. Find a quadratic nonresidue n modulo p (pick one integer at random and test; with probability $\frac{1}{2}$ it will be a nonresidue mod p; if not, repeat).
2. Initialize:
   - $x \equiv a^{\frac{s+1}{2}} \bmod p$;
   - $b \equiv a^s \bmod p$;
   - $g \equiv n^s \bmod p$;

- $r = e$.

3. Find the minimal $m \in \mathbb{N}$ with $b^{2^m} \equiv 1 \bmod p$. If $m = 0$, return x and terminate. If $m > 0$, update the variables:

   - replace x by $xg^{2^{r-m-1}} \bmod p$;
   - replace b by $bg^{2^{r-m}} \bmod p$;
   - replace g by $g^{2^{r-m}} \bmod p$;
   - replace r by m.

   Now repeat.

Why does this work at all? First observe that the element b satisfies $b^{2^{r-1}} \equiv a^{2^{r-1}s} \equiv a^{\frac{p-1}{2}} \equiv 1 \bmod p$, hence the order of b is a power of 2, and an m as required in the last step does exist.

Next, the order of b decreases with each step, so eventually we must reach a point where $m = 0$, i.e., where $b = 1$. Since we always have $x^2 \equiv ba \bmod p$, we will have found a square root x of $a \bmod p$ then. Indeed, after initializing we have $x^2 \equiv a^{s+1} = a^s a \equiv ba \bmod p$. In the last step, assume we have $x^2 \equiv ba \bmod p$ before updating the values. Then $x_1^2 \equiv x^2 g^{2^{r-m}} = bag^{2^{r-m}} \equiv b_1 a \bmod p$, where x_1 and b_1 denote the values of x and b after the update.

Finally assume that b has order 2^m; we have to show that b_1 has order at most 2^{m-1}. In fact, observe that we have $b^{2^{m-1}} \equiv -1 \bmod p$; hence

$$b_1^{2^{m-1}} = (bg^{2^{r-m}})^{2^{m-1}} = b^{2^{m-1}} g^{2^{r-1}} \equiv (-1)^2 \equiv 1 \bmod p.$$

Counting the number of bit operations is easy; the only problem is finding a quadratic nonresidue n. The procedure above is probabilistic; trying values at random will quickly produce such an n with very high probability. □

This does not work if $N = p$ is not prime; in fact, we have already seen that if we had an algorithm that could extract square roots modulo N, then we could factor N quickly.

Exercises

8.1 Compute a square root of 2 mod 41 using Tonelli-Shanks.

8.2 Consider the simple-minded secret sharing protocol introduced at the beginning of Section 8.3.

1. How can Alice split the secret among three (or, more generally, n) trustees?
2. Assume that Alice picks a random r and sends each trustee i the numbers $(i, w - r \cdot i)$. How many trustees are needed to recover w?
3. Find a way to split the secret w among n trustees in such a way that 3 trustees are needed to recover m by generalizing the idea in part 2).

Alice	Eve	Bob
Alice picks two large primes p and q, generates the RSA public key (n, e), and computes $c \equiv s^e \bmod n$.	$\xrightarrow{(e,n,c)}$	
Alice picks a random integer a with $\sqrt{n} < a < n$ and computes $b \equiv a^2 \bmod n$.		Bob picks a random c with $\sqrt{n} < c < n$, computes $d \equiv c^2 \bmod n$.
Alice sends b to Bob.	$\overset{b,d}{\longleftrightarrow}$	Bob sends d to Alice.
		Bob tosses a fair coin to get a bit $B \in \{0, 1\}$, and sends B to Alice.
	$\xleftarrow{B}$	
Alice solves $y^2 \equiv bd^B \bmod n$ and sends y to Bob.		
	$\xrightarrow{y}$	
		Bob verifies that $y^2 \equiv bd^B \bmod n$.

Figure 8.6: Zero Knowledge Version of the OTP

8.3 The following version of a modified OTP can be found in some online lecture notes:

Examine the difference to the one we have presented, and discuss them. Is it zero knowledge? Can Malice impersonate Alice?

8.4 Explain how Alice can verify that Bob did not cheat in the online poker game, say by claiming that he has four aces.

Chapter 9

Finite Fields

Finite Fields are fields with finitely many elements. The most basic finite fields are the fields $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ with p elements, where p is a prime. In this chapter we will construct finite fields for $q = p^n$ elements, show that there are no others, and discuss their basic properties.

9.1 Polynomials

Since $\mathbb{F}_p$ is a field, the polynomial ring $\mathbb{F}_p[X]$ has unique factorization. It is easy to show (using Euclid's argument) that there are infinitely many primes. In fact, for any given degree $d \geq 1$ there is a monic irreducible polynomial of degree d, as can be seen by counting the number of composite polynomials.

Let $f \in \mathbb{F}_p[X]$ be a monic irreducible polynomial of degree d. Then (f) generates a maximal ideal (f) in $\mathbb{F}_p[X]$, hence the quotient ring $\mathbb{F}_q = \mathbb{F}_p[X]/(f)$ is a field. Its elements are residue classes modulo f, and they are represented by polynomials of degree $< d$: in fact, if $f(x) = x^d + ldots + a_1X + a_0$, then $x^d \equiv -a_{d-1}X^d - 1 - \ldots - a_1X - a_0 \bmod f$, hence we can reduce every polynomial to one of degree $< d$ by working mod f. On the other hand, the polynomials of degree $< d$ are pairwise distinct mod f: if not, then f divides their difference, which is a polynomial of degree $< d$. The residue classes $a \bmod f$ for $a \in \mathbb{F}_p$ obviously form a subfield of $\mathbb{F}_q$ isomorphic to $\mathbb{F}_p$.

Proposition 9.1. *For every integer $d \geq 1$ there is an irreducible monic polynomial $f \in \mathbb{F}_p[X]$; the quotient ring $\mathbb{F}_{p^d} = \mathbb{F}_p[X]/(f)$ is a finite field with exactly $q = p^d$ elements, containing $\mathbb{F}_p$ as a subfield.*

Example: to cinstruct a field with 4 elements, take the irreducible polynomial $f(X) = X^2 + X + 1 \in \mathbb{F}_2[X]$. The 4 elements of $\mathbb{F}_4$ then are the residue classes $0 = 0 + (f)$, $1 = 1 + (f)$, $\alpha = X + (f)$, and $1 + \alpha = 1 + X + (f)$. Addition is trivial; multiplication works like this: $\alpha^2 = \alpha + 1$ since $\alpha^2 = X^2 + (f) = X^2 - f + (f) = -X - 1 + (f) = 1 + X + (f)$, where we have used that $-1 = 1$ in $\mathbb{F}_2$.

Are there finite fields with 6 elements? The answer is no:

Proposition 9.2. *If $\mathbb{F}_q$ is a finite field with q elements, then $q = p^d$ is a power of a prime p (the characteristic of $\mathbb{F}_q$). Moreover, all fields with q elements are isomorphic.*

In order to be able to work with finite fields, we need algorithms for adding, multiplying and dividing elements. The only problem is division. Assume that $g(x) + (f)$ is a nonzero element in $\mathbb{F}_q = \mathbb{F}_p[X]/(f)$, where we may assume that $\deg g < d$. Clearly $f \nmid g$; since f is irreducible, we actually have $\gcd(f, g) = 1$, so by Bezout there are polynomials $r, s \in \mathbb{F}_p[X]$ (which can be computed using the Euclidean algorithm) such that $fr + gs = 1$. But then $g(x)s(x) \equiv 1 \bmod f$, hence the element $s + (f)$ is the inverse of $g + (f)$.

We leave it as an exercise to prove

Proposition 9.3. *Two elements of $\mathbb{F}_q$, where $q = p^d$, can be multiplied or divided in $O((\log q)^2)$ bit operations; an element can be raised to the n-th power in $O(\log n(\log q)^2)$ bit operations.*

We also know

Proposition 9.4. *The multiplicative group of a finite field is cyclic.*

This provides us with a wealth of groups in which the DLP is difficult to solve.

9.2 The Frobenius Automorphism

Consider the finite field $\mathbb{F}_p$; the map $a \longmapsto a^p$ is a ring homomorphism since $(a + b)^p = a^p + b^p$. In fact, the same argument shows that $\sigma : x \longmapsto x^p$ is an automorphism $\mathbb{F}_q \longrightarrow \mathbb{F}_q$; it is called the Frobenius automorphism.

Proposition 9.5. *Let $\mathbb{F}_q/\mathbb{F}_p$ be an extension of finite fields. Then the Frobenius automorphism σ fixes the elements of $\mathbb{F}_p$, and every element of $\mathbb{F}_q$ fixed by the Frobenius lies in $\mathbb{F}_p$.*

Proof. Since every element $a \in \mathbb{F}_p$ satisfies $a^p = a$, the Frobenius fixes every element in $\mathbb{F}_p$. Conversely, let $a \in \mathbb{F}_q$ be an element with $a^p = a$. Then a is a root of $X^p - X$; but this polynomial has at most p roots in any field, and the p elements of $\mathbb{F}_p$ are roots; thus there cannot be any other roots, and a must be one of them: $a \in \mathbb{F}_p$. □

This immediately implies

Proposition 9.6. *The extension $\mathbb{F}_q/\mathbb{F}_p$, where $q = p^d$, is a Galois extension; its Galois group is cyclic of order d and is generated by the Frobenius automorphism:* $\mathrm{Gal}\,(\mathbb{F}_q/\mathbb{F}_p) = \{\sigma^j : 0 \le j < d\}$. *In particular, the Galois group is cyclic and isomorphic to $\mathbb{Z}/d\mathbb{Z}$.*

Proof. For showing that $\mathbb{F}_q/\mathbb{F}_p$ is Galois, we only need to show that there are d distinct automorphisms of $\mathbb{F}_q/\mathbb{F}_p$. Clearly the σ^j are automorphisms. The elements fixed by σ^j are exactly the roots of $X^{p^j} - X$; thus σ^d fixes everything, and this means that $\sigma^d = 1$ is the identity map.

Next assume that $\sigma^r = \sigma^s$; then every $x \in \mathbb{F}_q$ is a root of $X^{p^r} - X^{p^s}$, and every nonzero $x \in \mathbb{F}_q$ is therefore a root of $X^{p^r - p^s} - 1$. For $r, s < d$ this is only possible if $r = s$. Thus the σ^j are indeed pairwise distinct, and $\mathbb{F}_q/\mathbb{F}_p$ is a cyclic extension of degree d as claimed. □

We now define two important maps from $\mathbb{F}_q$ to $\mathbb{F}_p$, namely the trace T and the norm N, by

$$\begin{aligned} T(x) &= \sum \sigma^j(x) = x + x^p + x^{p^2} + \cdots + x^{p^{d-1}}, \\ N(x) &= \prod \sigma^j(x) = x \cdot x^p \cdot x^{p^2} \cdots x^{p^{d-1}}. \end{aligned}$$

Proposition 9.7. *The trace T is a group homomorphism from the additive group $(\mathbb{F}_q, +)$ onto the additive group $(\mathbb{F}_p, +)$.*

Proof. We only show that the trace is in $\mathbb{F}_p$. To this end we have to show that $T(x)$ is fixed by the Frobenius. But

$$\begin{aligned} T(x)^p &= x + x^p + x^{p^2} + \cdots + (x^{p^{d-1}})^p \\ &= x^p + x^{p^2} + \ldots + x^{p^d} \\ &= x^p + x^{p^2} + \ldots + x = T(x). \end{aligned}$$

□

Proposition 9.8. *The norm N is a group homomorphism from the multiplicative group $(\mathbb{F}_q^\times, \cdot)$ onto the multiplicative group $(\mathbb{F}_p^\times, \cdot)$.*

Proof. As above, we find that $N(x)^p = N(x)$, hence $N(x) \in \mathbb{F}_p$. Moreover, $N(x) = 0$ if and only if $x = 0$, hence the norm is a map from $F_q^\times$ to $\mathbb{F}_p^\times$.

Let γ be a generator of $\mathbb{F}_q^\times$. We claim that $g = N(\gamma)$ generates $\mathbb{F}_p^\times$. This implies that the norm is surjective: for $a \in \mathbb{F}_p^\times$, write $a = g^t$; then $a = g^t = N(\gamma)^t = N(\gamma^t)$.

Since γ has order $q - 1$, we find that $N(\gamma) = \gamma^{1+p+p^2+\ldots+p^{d-1}} = \gamma^{\frac{q-1}{p-1}}$ has order $p - 1$ as claimed. □

In particular, the norm $\mathbb{F}_{p^2}^\times \longrightarrow \mathbb{F}_p^\times$ must have a kernel of the order $p + 1$; this subgroup $\ker N$ is cyclic (because $\mathbb{F}_{p^2}^\times$ is cyclic) and can be used to generalise lots of results in elementary number theory (primality proofs based on the factorization of $p + 1$; a $p + 1$-method of factoring; DLP in a group with $p + 1$ elements).

Exercises

9.1 Show that $f(X) = x^2 + x + 1$ is irreducible over $\mathbb{F}_2$ and use f to construct the field $\mathbb{F}_4$. Compute addition and multiplication tables, and determine the elements with norm 1.

9.2 Find an irreducible quadratic polynomial over $\mathbb{F}_3$, construct $\mathbb{F}_9$, find a generator γ of $\mathbb{F}_9^\times$, and determine all elements in the kernel of the norm map.

9.3 Find a normal basis for $\mathbb{F}_4$ and $\mathbb{F}_9$.

Chapter 10

Pell Conics

In the next chapter we will begin discussing the basic arithmetic of elliptic curves. In this chapter we will present similar results for a class of curves that is a lot simpler: Pell conics. These are curves described by an equation $\mathcal{P} : X^2 - dY^2 = 1$. If R is a domain, then

$$\mathcal{P}(R) = \{(x, y) \in R \times R : x^2 - dy^2 = 1\}$$

denotes the set of all points (x, y) on $\mathcal{P}$ with coordinates in R. Our first task will be the description of a group law on $\mathcal{P}(R)$. Then we will show how to use the groups $\mathcal{P}(\mathbb{F}_p)$ for primality tests, primality proofs, factoring, and for cryptography.

In the following, p will always denote an *odd* prime.

10.1 Group Law and Parametrization

We now use the technique of parametrization to count the number of points on $\mathcal{P}$ over $\mathbb{F}_p$:

Proposition 10.1. *Let p be an odd prime, consider the Pell conic $\mathcal{P} : X^2 - dY^2 = 1$, and assume that $p \nmid d$. Then*

$$\#\mathcal{P}(\mathbb{F}_p) = \begin{cases} p+1 & \text{if } (\frac{d}{p}) = -1, \\ p-1 & \text{if } (\frac{d}{p}) = +1. \end{cases}$$

Proof. The idea is to consider lines through $P = (1, 0)$; these will intersect the Pell conic in exactly two points, namely in P and another point P_m depending on the slope m of the line.

The lines through P are given by $L : Y = m(X - 1)$ (with one exception: the line $x = 1$). For computing the points of intersection of L and $\mathcal{P}$, plug $Y = m(X - 1)$ into the equation of $\mathcal{P}$; then we find

$$0 = X^2 - dY^2 - 1 = X^2 - dm^2(X - 1)^2 - 1 = (X - 1)(X + 1 - dm^2(X - 1)).$$

The first factor corresponds to the point P, the other gives $X = \frac{dm^2+1}{dm^2-1}$. Plugging this into the equation for L shows that the point of intersection is

$$P_m = \Big(\frac{dm^2+1}{dm^2-1}, \frac{2m}{dm^2-1}\Big).$$

Every point Q on $\mathcal{C} \setminus \{P\}$ with coordinates in $\mathbb{F}_p$ has such a representation: simply compute the slope m of the line PQ. Conversely, every value $m \in \mathbb{F}_p$ gives such a point, except when the denominator dm^2-1 vanishes. If $(\frac{d}{p}) = -1$, this does not happen, and then there are p points P_m plus P, hence $p+1$ points overall. If $(\frac{d}{p}) = +1$, however, then there are exactly two values of m for which $dm^2 - 1 = 0$, hence we find $p-2$ points P_m plus P, that is, $p-1$ points overall. □

Next we define a group law on $\mathcal{P}(\mathbb{F}_p)$ in the following way: fix the point $N = (1,0)$; in order to add two distinct points $P, Q \in \mathcal{C}(\mathbb{F}_p)$, draw a parallel to PQ through N, and let $P+Q$ be the second point of intersection (with $P+Q = N$ if the parallel is a tangent at N). In order to add P to itself, draw a parallel to the tangent to $\mathcal{P}$ at P through N, and let $2P$ denote the second point of intersection.

It is very easy to see that N is a neutral element with respect to this addition, that inverses exist, and that addition is commutative. Proving associativity is not so easy, but turns out to be a consequence of Pascal's Theorem.

We will use algebra to prove that this addition of points defines a group law. To this end, we first compute explicit formulas.

Theorem 10.2. *Let $P = (r,s)$ and $Q = (t,u)$ be points on the Pell conic $\mathcal{P}: X^2 - dY^2 = 1$ with neutral element $N = (1,0)$ over some field F. Then*

$$P + Q = (rt + dsu, ru + st).$$

We will prove that the geometric group law defined above leads to these formulas; note that we can use these formulas to *define* a group law on $\mathcal{P}$ for *any* ring R (not just fields as in the theorem).

Proof. Let $R = (rt + dsu, ru + st)$. It is sufficient to show that

1. R is on $\mathcal{P}$;

2. the slope of RN is equal to the slope of PQ.

These are rather simple calculations:

1. $(rt+dsu)^2 - d(ru+st)^2 = r^2(t^2-du^2) - ds^2(t^2-du^2) = r^2 - ds^2 = 1.$

2. Assume first that $r \neq t$. We have to show that

$$\frac{ru+st}{rt+dsu-1} = \frac{s-u}{r-t}.$$

Clearing denominators and cancelling equal terms shows that this is equivalent to

$$(r^2 - ds^2)u = s(t^2 - du^2) - s + u.$$

Plugging in $r^2 - ds^2 = t^2 - du^2 = 1$ then proves the claim.

Now assume that $r = t$. Then there are two cases:

(a) $P = -Q$; then $r = t$ and $s = -u$), hence $P + Q = N$ as well as $R = (r^2 - ds^2, 0) = (1, 0) = N$.

(b) $P = Q$; then $r = t$, $s = u$, and we have to show that the slope of RN equals the slope of the tangent at P. The slope of RN is $\frac{2rs}{r^2+ds^2-1} = \frac{2rs}{2ds^2} = \frac{r}{ds}$, where we have used $r^2 - 1 = ds^2$. In order to compute the slope of the tangent at P we take the derivative of $X^2 - dY^2 = 1$ with respect to X and find $2X - 2dYY' = 0$. Solving for Y' and evaluation at $(X, Y) = (r, s)$ shows that the slope of the tangent at P is $\frac{r}{ds}$.

□

In Algebraic Geometry you will learn that you may use classical formulas for derivatives even if you want to compute the tangent of an algebraic curve over, say, a finite field. In the case at hand, this can be checked directly. Call a line through P a tangent to some Pell conic $\mathcal{P}$ at P if the line intersects $\mathcal{P}$ only in P. It is then easily checked that the only slope for which this holds is the one predicted by calculus.

The Group Structure

We now will show that the groups $\mathcal{P}(\mathbb{F}_p)$ are cyclic. There are two cases to consider.

1. $d = a^2$ is a square in $\mathbb{F}_p^\times$. For a point $P = (r, s)$ on $\mathcal{P}(\mathbb{F}_p)$, the identity $1 = r^2 - ds^2 = (r - as)(r + as)$ suggest substituting $u = r - as$, $v = r + as$. Then $uv = 1$, hence $u, v \in \mathbb{F}_p^\times$. We now claim that the map $\phi : \mathcal{P}(\mathbb{F}_p) \longrightarrow \mathbb{F}_p^\times$ defined by $\phi(r, s) = r - as$ is a homomorphism. In fact, write $Q = (t, u)$; then $P + Q = (rt + dsu, ru + st)$, hence $\phi(P + Q) = rt + a^2su - a(ru + ts)$; on the other hand, $\phi(P)\phi(Q) = (r - as)(t - au) = rt + a^2su - a(ru + st) = \phi(P + Q)$.

   Thus ϕ is a group homomorphism. We now claim that ϕ is surjective. In fact, let $u \in \mathbb{F}_p^\times$ be given; then $uv = 1$ for $v = u^{-1}$. The equations $r - as = u$ and $r + as = v$ give us $r = \frac{u+v}{2}$ and $s = \frac{v-u}{2a}$; with $P = (r, s)$ we then find $\phi(P) = u$.

   Since ϕ is a surjective map between finite sets of the same cardinality, it must be bijective.

2. d is not a square in $\mathbb{F}_p^\times$: then $\mathbb{F}_{p^2} = \mathbb{F}_p(\sqrt{d}\,)$, and we can define a map $\phi : \mathcal{P}(\mathbb{F}_p) \longrightarrow \mathbb{F}_{p^2}^\times$ via $\phi(r,s) = r + s\sqrt{d}$.

   We now claim that ϕ is a homomorphism. In fact, write $Q = (t,u)$; then $P+Q = (rt+dsu, ru+st)$, as well as $\phi(P)\phi(Q) = (r+s\sqrt{d}\,)(t+u\sqrt{d}\,) = rt + dsu + (ru+st)\sqrt{d} = \phi(P+Q)$.

   This time, however, the map ϕ is not surjective since the pair (r,s) satisfies the condition $r^2 - ds^2 = 1$. We claim that $r^2 - ds^2 = N(r+s\sqrt{d}\,)$. In fact, let $\sigma : x \longmapsto x^p$ denote the Frobenius automorphism; then $N(x) = x \cdot x^p$ for $x \in \mathbb{F}_{p^2}$. We find $\sigma(r+s\sqrt{d}\,) = (r+s\sqrt{d}\,)^p = r^p + s^p\sqrt{d}^p = r - s\sqrt{d}$: in fact, $r^p = r$ and $s^p = s$ since r and s lie in the base field $\mathbb{F}_p$, and $\sqrt{d}^p = d^{(p-1)/2}\sqrt{d} = -\sqrt{d}$ by Euler's criterium: an element $d \in \mathbb{F}_p$ is a square if and only if $d^{(p-1)/2} = 1$.

   Thus $N(r+s\sqrt{d}\,) = (r+s\sqrt{d}\,)(r+s\sqrt{d}\,)^p = (r+s\sqrt{d}\,)(r-s\sqrt{d}\,) = r^2 - ds^2$. This shows that the image of ϕ lies in the kernel of the norm map $\mathbb{F}_{p^2}^\times[N]$.

   We now claim that $\phi : \mathcal{P}(\mathbb{F}_p) \longrightarrow \mathbb{F}_{p^2}^\times[N]$ is an isomorphism. Clearly $\ker\phi = \{(r,s) \in \mathcal{P}(\mathbb{F}_p) : r + s\sqrt{d} = 1\} = \{N\}$, so ϕ is injective. Since both groups have the same cardinality, the claim follows.

   Note that the results just proved can be summarized in the exact sequence

$$\begin{tikzcd} 1 \arrow[r] & \mathcal{P}(\mathbb{F}_p) \arrow[r] & \mathbb{F}_{p^2}^\times \arrow[r, "N"] & \mathbb{F}_p^\times \arrow[r] & 1 \end{tikzcd}$$

We have shown

Theorem 10.3. *Let p denote an odd prime and $\mathcal{P} : X^2 - dY^2 = 1$ a Pell conic over $\mathbb{F}_p$ with $d \neq 0$. Then*

$$\mathcal{P}(\mathbb{F}_p) \simeq \begin{cases} \mathbb{F}_p^\times & \text{if } d \text{ is a square in } \mathbb{F}_p^\times \\ \mathbb{F}_{p^2}^\times[N] & \text{if } d \text{ is not a square in } \mathbb{F}_p^\times. \end{cases}$$

In particular, the groups $\mathcal{P}(\mathbb{F}_p)$ are subgroups of the multiplicative groups of finite fields, hence are cyclic.

Functoriality

The formulas in Theorem 10.2 allow us to define group laws on Pell conics over arbitrary rings. In order to be able with $\mathcal{P}(\mathbb{Z}/n\mathbb{Z})$ we need more information on their structure.

Let R and S be commutative rings with unit; then so is their sum $R \oplus S = \{(r,s) : r \in R, s \in S\}$, where addition and multiplication are defined componentwise, and where $(0,0)$ and $(1,1)$ are the zero and the unit element, respectively. Note that even if R and S are domains, $R \oplus S$ has zero divisors since $(1,0) \cdot (0,1) = (0,0)$.

The Chinese Remainder Theorem provides us with lots of examples: for coprime integers $m, n \neq 0$ it states that $\mathbb{Z}/mn\mathbb{Z} \simeq \mathbb{Z}/m\mathbb{Z} \oplus \mathbb{Z}/n\mathbb{Z}$.

What can we say about $\mathcal{P}(R \oplus S)$? Every element in $\mathcal{P}(R \oplus S)$ has the form (x, y), where $x = (r_1, s_1)$ and $y = (r_2, s_2)$ with $r_1, r_2 \in R$ and $s_1, s_2 \in S$. The equation $x^2 - dy^2 = 1$ (where the 1 on the right hand side is nothing but the unit element $1 = (1, 1)$ in $R \oplus S$) is actually a system of two equations $r_1^2 - dr_2^2 = 1$ and $s_1^2 - ds_2^2 = 1$. Thus each point $(x, y) \in \mathcal{P}(R \oplus S)$ corresponds to two points $(r_1, r_2) \in \mathcal{P}(R)$ and $(s_1, s_2) \in \mathcal{P}(S)$. We leave it as an exercise to show that the map $\phi : \mathcal{P}(R \oplus S) \longrightarrow \mathcal{P}(R) \oplus \mathcal{P}(S)$ is a group homomorphism. It is also easy to see that ϕ is an isomorphism.

Thus we have

Theorem 10.4. *For rings R, S with units and a Pell conic $\mathcal{P} : X^2 - dY^2 = 1$ defined over $R \oplus S$ we have*

$$\mathcal{P}(R \oplus S) \simeq \mathcal{P}(R) \oplus \mathcal{P}(S).$$

In particular: if $n = pq$ for distinct odd primes p and q not dividing d, then $\mathcal{P}(\mathbb{Z}/n\mathbb{Z}) \simeq \mathcal{P}(\mathbb{Z}/p\mathbb{Z}) \oplus \mathcal{P}(\mathbb{Z}/q\mathbb{Z})$ is a cyclic group of order $(p - (\frac{d}{p}))(q - (\frac{d}{q}))$.

10.2 Factorization Methods

The $p + 1$-Method of Factoring

Let N be a composite integer, and assume that p is a prime factor of N. Assume moreover that we have found an integer d coprime to N with $(\frac{d}{p}) = -1$. Consider the Pell conic $\mathcal{P} : X^2 - dY^2 = 1$ and pick a random point $P \in \mathcal{P}(\mathbb{Z}/N\mathbb{Z})$ by picking a random integer m coprime to N and computing $P = (\frac{dm^2+1}{dm^2-1}, \frac{2m}{dm^2-1})$.

Note that in this calculation you need the fact that $dm^2 - 1$ is invertible mod N. If this fails, then $\gcd(dm^2 - 1, N)$ must be nontrivial. Since $p \nmid (dm^2 - 1)$ because of $(\frac{d}{p}) = -1$, we have $1 < \gcd(dm^2 - 1, N) < N$ in this case, and thus have found a factor of N (in practice, this does not occur, of course).

We know that the group $\mathcal{P}(\mathbb{Z}/p\mathbb{Z})$ is cyclic of order $p+1$; thus $(p+1)P = (1, 0)$ in $\mathcal{P}(\mathbb{Z}/p\mathbb{Z})$. This means that, as a point in $\mathcal{P}(\mathbb{Z}/N\mathbb{Z})$, the points $(p+1)P$ must have the form (x, y) with $x \equiv 1 \bmod p$ and $y \equiv 0 \bmod p$. This means that we have a good chance of recovering p by simply computing $\gcd(x - 1, N)$ or, equivalently, $\gcd(y, N)$. As in the $p - 1$-method, we cannot compute $(p + 1)P$ directly: we have to compute kP for some highly composite integer k; if k is a multiple of $p + 1$, then $kP = (1, 0)$ in $\mathcal{P}(\mathbb{Z}/p\mathbb{Z})$, and the gcd-calculation will very likely give us the prime factor p.

Here's the actual algorithm:

1. Choose d, a point $P \in \mathcal{P}(\mathbb{Z}/N\mathbb{Z})$, a bound B, and set $k = 1$ and $q = 2$.
2. Find e with $q^e \leq B < q^{e+1}$.
3. Set $P = (x, y) \longleftarrow q^e P$ and find $\gcd(x - 1, N)$.
4. If the gcd is 1, replace q by the next prime; if $q < B$, goto step 2, otherwise stop.

If no factor is found, one can increase B (multiply it by 10, say), and continue.

The most serious problem with the $p+1$-method is finding a value of d with $(\frac{d}{p}) = -1$. Since we do not know p in advance, we can only pick a random d; with probability $\frac{1}{2}$ we will have $(\frac{d}{p}) = -1$. If we do the calculations with three random d, then the probability that one of them satisfies $(\frac{d}{p}) = -1$ is $\frac{7}{8}$. Thus instead of running the $p+1$-method once we have to run it up to three times; since the $p+1$-method is about three times as slow as the $p-1$-method (assuming we have chosen the same bound B), we find that the overall running time is about 9 times that of the $p-1$-method.

The computation of q^eP is of course not done by adding P sufficiently often to itself but by the method of duplication and addition (squaring and multiplying in the multiplicative language).

Here's a simple example: take $N = 56^2+3 = 3139$, $d = -1$, $P = P_0 = (56, 2)$ and $B = 10$. Our first prime is $q = 2$, and $2^3 = 8$ is the smallest power < 10; we get $2P = (-7, 224)$, $4P = (97, 3)$ and $P_1 = 8P = (-17, 582)$, and since $(582, N) = 1$ we continue with $q = 3$. Here we have to compute $P_2 = 9P_1$, and this is done by doubling P_1 three times and adding P_1: $2P_1 = (577, -954)$, $4P_1 = (389, 873)$, $8P_1 = (1297, 1170)$, $9P_1 = (1520, 438)$, and now $\gcd(438, N) = 73$, hence $N = 73 \cdot 43$. Note that we cannot expect to find the second factor 43 with this method and $B = 10$ since $43 - (\frac{-1}{43}) = 43 + 1 = 4 \cdot 11$ has a prime factor larger than B. We did find 73 on the other hand since $73-(\frac{-1}{73}) = 73-1 = 2^3 3^2$ is a product of prime powers $< B$. In fact, P has order 9 on $\mathcal{C}(\mathbb{Z}/73\mathbb{Z})$, so we would have found it by simply computing $9P$. Check this!

Pollard's rho Method

We can also transfer Pollard's rho method to Pell conics. For factoring N, pick a Pell conic $X^2 - dY^2 = 1$ and random points $P, Q \in \mathcal{P}(\mathbb{Z}/N\mathbb{Z})$. Consider the function $f(P) = 2P + Q$; if this is a random function on $\mathcal{P}(\mathbb{Z}/N\mathbb{Z})$, then the following algorithm will find a prime factor p of N in $O(\sqrt{p}\,)$ steps:

1. Let $P_1 = P_2 = P$;
2. $P_1 \longleftarrow 2P_1 + Q$; $P_2 \longleftarrow 2P_2 + Q$; $P_2 \longleftarrow 2P_2 + Q$;
3. If $\gcd(x_1 - x_2, N) = 1$, where $P_j = (x_j, y_j)$, goto 2.
   If $p = \gcd(x_1 - x_2, N) > 1$, print p and terminate.

10.3 Primality Tests

Fermat's primality test was based on the observation that $a^{p-1} \equiv 1 \bmod p$ for primes p and integers a coprime to p. The analog here is

Proposition 10.5 (Fermat's Little Theorem)**.** *Let P be a point on the Pell conic $\mathcal{P} : X^2 - dY^2 = 1$ over $\mathbb{F}_p$, and assume that $(\frac{d}{p}) = -1$. Then $(p+1)P = (1, 0)$.*

Proof. The group $\mathcal{P}$ has $p - (\frac{d}{p}) = p + 1$ elements, so the claim follows from Lagrange's theorem that the order of an element divides the group order. $\square$

This can be used as a primality test: given an odd integer n, pick an integer d with $(\frac{d}{n}) = -1$ and a random point P on $\mathcal{P} : X^2 - dY^2 = 1$ over $\mathbb{Z}/n\mathbb{Z}$; then check whether $(n+1)P = (1,0)$.

Note that if n is prime, then the equation $2P = (1,0)$ has only two solutions, namely $(-1,0)$ and $(1,0)$. This means that we also have an analog of the Miller-Rabin test.

In elementary number theory we have proved various results about factors of Fermat and Mersenne numbers. These were based pn the following observation:

Proposition 10.6. *Let n be an odd integer and assume that $a^{n-1} \equiv 1 \bmod n$, but $\gcd(a^{\frac{n-1}{q}}, n) \neq 1$ for some prime $q \mid n-1$. Then every prime $p \mid n$ satisfies $p \equiv 1 \bmod n$.*

Proof. The congruences show that $q \mid \operatorname{ord}(a \bmod p)$: otherwise we would have $a^{\frac{n-1}{q}} \equiv 1 \bmod p$ and therefore $p \mid \gcd(a^{\frac{n-1}{q}}, n)$. On the other hand, Fermat's Little Theorem tells us that $\operatorname{ord}(a \bmod p) \mid (p-1)$. Thus we see that $q \mid (p-1)$, and this implies the claim. $\square$

This result has an analog in our situation:

Proposition 10.7. *Let n be an odd integer, P a point on the Pell conic $\mathcal{P} : X^2 - dY^2 = 1$ with $(\frac{d}{n}) = -1$, and assume that $(n+1)P = N$, but $\frac{n+1}{q}P = (x,y)$ with $\gcd(x-1, n) = 1$ for some prime $q \mid (n+1)$. Then every prime $p \mid n$ satisfies $p \equiv (\frac{d}{p}) \bmod q$.*

Proof. As above, the point P considered as a point on $\mathcal{P}(\mathbb{F}_p)$ has order divisible by q; on the other hand, the analog of Fermat's Little Theorem tells us that the order of P modulo p divides $p - (\frac{d}{p})$. Thus $q \mid p - (\frac{d}{p})$, and this proves our claim. $\square$

Corollary 10.8. *Let $\mathcal{P} : x^2 - dy^2 = 1$ be a Pell conic, and assume that n is an integer with $(\frac{d}{n}) = -1$. Assume moreover that $n + 1 = FR$ with $F > \sqrt{n} + 1$. Then n is prime if and only if there exists a point $P \in \mathcal{P}(\mathbb{Z}/n\mathbb{Z})$ such that*

i) $(n+1)P = (1,0)$;

ii) $\frac{n+1}{q} P \neq (1,0)$ *for each prime* $q \mid F$.

Proof. Repeat the proof above with F instead of some prime $q \mid n+1$. Then $p \equiv (\frac{d}{p}) \bmod F$ for all primes $p \mid n$. If n is composite, there must be a prime $p \mid n$ with $p < \sqrt{n}$; but then $p \not\equiv \pm 1 \bmod F$. $\square$

In the special case of Mersenne numbers $q = 2^p - 1$ (note that $q \equiv 7 \bmod 12$ for $p \geq 3$), we have $\frac{q+1}{2} = 2^{p-1}$, and if we choose $\mathcal{P} : x^2 - 3y^2 = 1$ and $P = (2,1)$, then the test above is nothing but the Lucas-Lehmer test.

In fact, we have $2P = P + P = (7,4)$, $4P = (97,56)$, ?? $2(x,y) = (x^2 + 3y^2, 2xy) = (2x^2 - 1, 2xy)$.

Consider $n = 31$; we find

i	$i * P$
1	$(2,1)$
2	$(7,4)$
4	$(4,25)$
8	$(0,14)$
16	$(-1,0)$
32	$(1,0)$

Since we only need the x-coordinates, we can simply work with the recursive sequence $x_0 = 2$, $x_{n+1} = 2x_n^2 - 1$.

The classical Lucas-Lehmer test works with $S_0 = 4$, $S_{n+1} = S_n^2 - 2$. In fact, we have $S_n = 2x_n$ since $S_{n+1} = S_n^2 - 2 = 4x_n^2 - 2 = 2(x_n^2 - 1) = 2x_{n+1}$.

10.4 Cryptography using Pell Conics

Let us now discuss a few cryptographic protocols based on Pell conics.

RSA

Alice picks and integer D and two primes $p \neq q$, computes $N = pq$ and $\Phi(N) = (p - (\frac{d}{p}))(q - (\frac{d}{q}))$, and picks integers d, e with $de \equiv 1 \bmod \Phi(N)$. Her public key is the triple (N, e, D).

Alice	Eve	Bob
picks two large primes p, q		
computes $N = pq$;		
picks random e coprime to $\Phi(N)$		
publishes public key (N, e, D)	$\xrightarrow{(N,e,D)}$	
		uses the parametrization of $\mathcal{P}$ to find the point P_m corresponding to the m.
		Bob computes $C = eP$
	$\xleftarrow{C}$	Bob sends c to Alice
Alice solves $de \equiv 1 \bmod \Phi(N)$		
she computes $P_m = dC$ and uses the parametrization of $\mathcal{P}$ to retrieve m		

Figure 10.1: The Pell analog of the RSA protocol

In order to encrypt a message $m < N$, Bob uses the parametrization of $\mathcal{P} : X^2 - DY^2 = 1$ over $\mathbb{Z}/N\mathbb{Z}$ to compute a point P corresponding to m. He computes the point $C = eP$ and sends it to Alice, who computes $P = dC$ (in fact: $dC = deP = (1 + k\Phi(N))P = P + k\Phi(N)P = P + N = P$) using her private key d.

Note that this idea is useless in practice: where RSA sends one encrypted message c about the size of N, the Pell version has to send twice as many bits per message, without having increased security.

Diffie-Hellman

The transfer to the Pell situation is straight forward: Alice and Bob agree on an integer d, a prime p with $(\frac{d}{p}) = -1$, and a point P that generates $\mathcal{P}(\mathbb{F}_p)$, where $\mathcal{P} : X^2 - dY^2 = 1$. Then they follow the protocol in Fig. 10.2.

Alice	Eve	Bob
Alice and Bob agree upon (d, p, P)		
Alice picks a random $a < p$		Bob picks a random $b < p$
Alice computes $A = aP$		Bob computes $B = bP$
Alice sends A to Bob	$\xrightarrow{A}$	
	$\xleftarrow{B}$	Bob sends B to Alice
Alice computes $K = aB$		Bob computes $K = bA$

Figure 10.2: The Pell analog of Diffie-Hellman key exchange

After having exchanged the necessary information, both Alice and Bob know the point $K = abP$; Eve knows d, p and P, as well as aP and bP. In order to break Diffie-Hellman, she has to solve the Diffie-Hellman problem: compute abP from P, $A = aP$ and $B = bP$. The only way known to do this is to solve the DLP $aP = A$ for a and then compute aB. The DLP in $\mathcal{P}(\mathbb{F}_p)$, on the other hand, is as hard as the DLP in $\mathbb{Z}/p\mathbb{Z}$, since both groups have about the same cardinality. One has to check, however, that $p+1$ (or $p-1$ in the second case) has at least one big prime factor: otherwise Pohlig-Hellman will solve the DLP.

Quadratic Residues and Nonresidues

Quite a few cryptographic protocols are based on the fact that computing square roots mod n is as difficult as factoring n.

The analog problem for Pell conics is the following: given a point $P \in \mathcal{P}(\mathbb{F}_p)$, determine whether $P = 2Q$ for some $Q \in \mathcal{P}(\mathbb{F}_p)$, and compute Q if it exists.

How can we tell whether $P = (x, y)$ has the form $P = 2Q$ for $Q = (r, s)$? We have $2(r, s) = (r^2 + ds^2, 2rs) = (2r^2 + 1, 2rs)$, hence we find $x = 2r^2 + 1$ as a necessary condition. This in turn implies that $2(x-1) = (2r)^2$ is a square, so $P = (x, y) = 2Q$ implies that $(\frac{2(x-1)}{p}) = +1$ if we work over $\mathbb{F}_p$. Conversely, if $(\frac{2(x-1)}{p}) = +1$, then we can solve $2(x-1) \equiv (2r)^2 \bmod p$ and $s^2 \equiv \frac{r^2-1}{d} \bmod p$. This shows that checking the solvability of $P = 2Q$ (and the computation of Q if it exists) can be performed as efficiently as taking square roots mod p.

For the same reason it is difficult to check whether $P = 2Q$ in $\mathcal{P}(\mathbb{Z}/n\mathbb{Z})$ for composite n when the prime factorization of n is unknown; of course $(\frac{2(x-1)}{n}) = 1$ is still a necessary condition, but it is not sufficient anymore. And even if we know that $P = 2Q$, computing Q from P is as difficult as factoring.

Exercises

10.1 Find all points on the unit circle $X^2 + Y^2 = 1$ over $\mathbb{F}_3$ and $\mathbb{F}_5$.

10.2 Let $P = (2, 2)$ be a point on the unit circle over $\mathbb{F}_7$. Compute $2P$, $3P$, $\ldots$; what is the order of P?

10.3 Show that the map $\phi : \mathcal{P}(R \oplus S) \longrightarrow \mathcal{P}(R) \oplus \mathcal{P}(S)$ is a group homomorphism. Also show that $\ker \phi = (1, 0)$, where $1 = (1, 1) \in R \oplus S$ and $0 = (0, 0)$, and that ϕ is surjective.

10.4 Consider the parabola $\mathcal{C} : Y = X^2$ over some field R. Show that the group law with neutral element $N = (0, 0)$ is given by $(r, r^2) + (s, s^2) = (r + s, (r + s)^2)$, and that we have $\mathcal{C}(R) \simeq (R, +)$, the additive group of R.

10.5 Construct a Pell analog of ElGamal.

10.6 Construct a Pell analog of Shamir's no-key protocol.

10.7 Assume you have an algorithm that, given a point $P \in \mathcal{P}(\mathbb{Z}/n\mathbb{Z})$ of the form $P = 2Q$, can compute Q efficiently. Show how to use this algorithm for factoring n.

10.8 Construct a Pell analog of FLIP.

10.9 The discriminant of the Pell conic $C^2 - dY^2 = 1$ is $\Delta = 4d$. If we want to construct Pell conics of discriminant $\Delta \equiv 1 \bmod 4$, we have to look at curves

$$\mathcal{P} : X^2 + XY - mY^2 = 1.$$

These have discriminant $\Delta = 1 + 4m$. Define a group law on $\mathcal{P}$ with neutral element $N = (1, 0)$, and show that the addition formulas can be given the form

$$(r, s) + (t, u) = (rt + su, ru + st + su).$$

Show that if $(\frac{\Delta}{p}) = -1$, then the map $\mathcal{P}(\mathbb{F}_p) \longrightarrow \mathbb{F}_{p^2}^\times$ sending $(a, b) \in \mathcal{P}(\mathbb{F}_p)$ to $a + b\omega \in \mathbb{F}_{p^2}^\times$, where $\omega = \frac{1+\sqrt{\Delta}}{2}$, is an homomorphism. More exactly, show that $\mathcal{P}(\mathbb{F}_p) \simeq \mathbb{F}_{p^2}^\times[N]$.

10.10 Consider the Pell conic $\mathcal{P} : X^2 + XY - Y^2 = 1$ with neutral element $N = (1, 0)$. Compute the multiples of $P = (1, 1)$ and show that $2^k P = (F_{k-1}, F_k)$, where F_k is the k-th term in the Fibonacci sequence $F_0 = F_1 = 1$, $F_{n+1} = F_n + F_{n-1}$.

10.11 Let $p \neq 5$ be a prime. Show that $p \mid F_n$ for $n = p - (\frac{5}{p})$.

Chapter 11

Elliptic Curves

Let F be a field (in number theory, $F = \mathbb{Q}$, an algebraic number field, or a local field; in cryptography, we usually may safely assume that F is a finite field, or even $F = \mathbb{F}_p$). A curve

$$\mathcal{E} : Y^2 = X^3 + aX + b$$

is called an elliptic curve if the polynomial $f(X) = X^3 + aX + b$ does not have multiple roots over an algebraic closure of F. For example, the curve $Y^2 = X^3 + 5$ is an elliptic curve over $\mathbb{F}_7$, but not over $\mathbb{F}_5$ (where $Y^2 = X^3$ has a triple root). the set $\mathcal{E}(F) = \{(x, y) \in F \times F : y^2 = x^3 + ax + b\}$ is called the set of (affine) F-rational points on E.

11.1 The Group Law

The group law on conics was defined in the affine plane; for getting a group law on elliptic curves, it is necessary to work in the projective plane.

The Projective Plane

Hasse's Theorem

Now in order to find the best strategy for factoring numbers using the elliptic curve method ECM, one needs to know a lot about the possible orders of elliptic curves over finite fields. It is clear that $E(\mathbb{F}_p)$ is finite since there are only $p^2 + p + 1$ points in the projective plane $\mathbb{P}^2\mathbb{F}_p$. Heuristically, we would expect a group order in the vicinity of $p + 1$: for approximately half the elements of $\mathbb{F}_p$ the values of $f(x) = x^3 + ax + b$ should be squares, and those squares correspond to two points $(x, \pm y)$ on the elliptic curve.

The simplest method to improve the trivial bound $\#E(\mathbb{F}_p) \le p^2 + p + 1$ is due to Postnikov.

Let $\mathbb{F}_q$ be a finite field of characteristic $p > 2$, and $f \in \mathbb{F}_q[X]$ a nonzero polynomial. In order to determine how many values of f are squares, we consider the equation

$$f(X)^{(q-1)/2} - 1 = 0 \tag{11.1}$$

and the polynomial

$$R(X) = 2f(X)(1 - f(X)^{(q-1)/2}) + f'(X)(X^q - X). \tag{11.2}$$

Clearly any root of (11.1) is a root of (11.2). Since

$$\begin{aligned} R'(X) = {} & 2f'(X)(1 - f(X)^{(q-1)/2}) \\ & + f'(X)(f(X)^{(q-1)/2} - 1) + f''(X)(X^q - X), \end{aligned}$$

any root of (11.1) is a double root of (11.2).

Now $\deg R = \frac{q+1}{2}\deg f$, hence the number N_f of solutions of (11.1) satisfies $2N_f \le \frac{q+1}{2}\deg f$.

Now R is a polynomial of degree $\deg R = \frac{p+1}{2}\deg f$ with at least N_f roots of multiplicity ≥ 2 and at least $\delta \le \deg f$ roots of multiplicity ≥ 1, where δ is the number of roots of f in $\mathbb{F}_p$. Thus the number N_f of solutions of (11.1) satisfies $2N_f + \delta \le \frac{p+1}{2}\deg f$.

Similarly, any root of

$$f(X)^{(p-1)/2} + 1 = 0 \tag{11.3}$$

is at least a double root of

$$R(X) = 2f(X)(1 + f(X)^{(p-1)/2}) + f'(X)(X^p - X). \tag{11.4}$$

If we denote the number of roots of (11.3) by M_f, then $2M_f + \delta \le \frac{p+1}{2}\deg f$. Moreover, $N_f + M_f + \delta = p$.

In the special case of a cubic polynomial f, we get $2N_f + \delta - p \le \frac{p+3}{2}$ and $2M_f + \delta - p \le \frac{p+3}{2}$. This implies $|2N_f + \delta - p| \le \frac{p+3}{2}$.

Since the number of solutions of $y^2 = f(x)$ over $\mathbb{F}_p$ is given by $2N_f + \delta$, we find

Proposition 11.1. *The number N of $\mathbb{F}_p$-rational points (including the point $\mathcal{O}$ at infinity) on a Weierstrass elliptic curve satisfies $|N - (p+1)| \le \frac{p+3}{2}$.*

This takes care of the existence of $\mathbb{F}_p$-rational points as well as of a bound for the number of $\mathbb{F}_p$-rational points. Already in the 1930s, a much sharper bound had been given by Hasse:

Theorem 11.2. *The number N of $\mathbb{F}_p$-rational points on an elliptic curve $Y^2 = X^3 + aX + b$ over $\mathbb{F}_p$ (including the point $\mathcal{O}$ at infinity) satisfies $|N - (p+1)| \le 2\sqrt{p}$.*

11.2 Addition Formulas

In fields F of characteristic $\ne 2, 3$, every nonsingular cubic with an F-rational point can be transformed into a curve described by a short Weierstrass equation

$y^2 = x^3 + ax + b$; for working with elliptic curves over fields with characteristic 2 or 3, these short Weierstrass forms are not suitable, and we will have to work with equations in long Weierstrass form

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{11.5}$$

for $a_i \in F$. If F has characteristic $\neq 2$, completing the square on the left hand side allows you to get rid of the xy-term; if the characteristic is $ne3$, you similarly can complete the cube on the right hand side to get rid of the x^2-term.

Let E be an elliptic curve in long Weierstrass form (11.5) defined over some field K. For $P \in E(K)$ we define $-P$ as the third point of intersection of the line through P and $\mathcal{O} = [0 : 1 : 0]$ with E. If $P = [x_1 : y_1 : 1]$, the line $P\mathcal{O}$ is given by $x = x_1$; intersection gives

$$y^2 + (a_1x_1 + a_3)y - (x_1^3 + a_2x_1^2 + a_4x_1 + a_6) = 0,$$

that is, the sum of the two roots of this quadratic equation is $-(a_1x_1 + a_3)$; note that this equation describes the affine points only. Since one root is given by $y = y_1$, the other one must be $-(a_1x_1 + a_3) - y_1$.

Given two points $P_1 = (x_1, y_1)$ and $P_1 = (x_2, y_2)$ in $E(K)$ with $x_1 \neq x_2$, let $-P_1 - P_2 = P_3 = (x_3, y_3)$ be the third point of intersection of the line P_1P_2 with E. The line P_1P_2 has the equation $y = y_1 + m(x - x_1)$ with $m = (y_2 - y_1)/(x_2 - x_1)$; plugging this into (11.5) yields a cubic equation in x with the roots x_1, x_2 and x_3. The sum of these roots is the coefficient of x^2 (observe that the coefficient of x^3 is -1), hence $x_3 = -x_1 - x_2 - a_2 + m(a_1 + m)$. Plugging this into the line equation we find the y-coordinate of P_3, hence

$$x_3 = -x_1 - x_2 + m(a_1 + m), \quad y_3 = -[y_1 + m(x_3 - x_1) + a_1x_3 + a_3].$$

If $x_1 = x_2$, then $y_1 = \pm y_2$. If $y_1 = -y_2$, then we put $P_1 + P_2 = \mathcal{O}$; if $y_1 = y_2$, that is, $P_1 = P_2$, then we let $-2P_1$ be the third point of intersection of the tangent to E in P_1 with E; a simple calculation then gives

Theorem 11.3. *Let E/K be an elliptic curve given in long Weierstrass form (11.5). The chord-tangent method defines an addition on the set $E(K)$ of K-rational points on E; the addition formulas are given by*

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3),$$

where

$$\begin{aligned} x_3 &= -x_1 - x_2 - a_2 + a_1m + m^2 \\ y_3 &= -y_1 - (x_3 - x_1)m - a_1x_3 - a_3 \end{aligned}$$

and

$$m = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } x_1 \neq x_2 \\ \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } x_1 = x_2 \end{cases}$$

In particular, for $P = (x, y)$ the x-coordinate of $2P$ is given by

$$x_{2P} = \frac{x^4 - b_4x^2 - 2b_6x - b_8}{4x^3 + b_2x^2 + 2b_4x + b_6}. \tag{11.6}$$

Specializing this to curves in short Weierstrass form, we get

$$x_3 = -x_1 - x_2 + m^2, \ y_3 = -y_1 - m(x_3 - x_1),$$

where

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } x_2 \neq x_1, \\ \frac{3x^2 + a}{2y} & \text{if } x_2 = x_1, y \neq 0. \end{cases}$$

The cases not covered by these formulas are

a) $x_1 = x_2$ and $y_1 = -y_2$: here $P_1 = -P_2$, hence $P_1 + P_2 = \mathcal{O}$;

b) $2(x, y)$ with $y = 0$: here $(x, 0)$ is a point of order 2, that is, $2(x, y) = \mathcal{O}$.

In order to see these formulas in action take the curve $E : y^2 + xy = x^3 - 18x + 27$. Here $a_1 = 1$, $a_2 = a_3 = 0$, $a_4 = -18$ and $a_6 = 27$. We find $b_2 = 1$, $b_4 = -36$, $b_6 = 108$, $b_8 = -324$, hence $c_4 = 865$, $c_6 = -24625$ and $\Delta = 23625 = 3^3 \cdot 5^3 \cdot 7$. Thus E is an elliptic curve over $\mathbb{F}_p$ for all $p \neq 3, 5, 7$. The point $P = (1, 1)$ is in $E(\mathbb{F}_2) : y^2 + xy = x^3 + 1$; let us compute a few multiples of P.

Bu the addition law we have

$$m = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} = \frac{x_1^2 - y_1}{x_1} = 0,$$

hence $x_{2P} = -2x_P + m + m^2 = 0$ and $y_{2P} = -y_P - (x_{2P} - x_P)m - x_{2P} = 1$, i.e., $2P = (0, 1)$.

We get $3P$ by adding P and $2P$; here $m = (y_2 - y_1)/(x_2 - x_1) = 0$, hence $x_{3P} = -x_P - x_{2P} = 1$, as well as $y_{3P} = -y_P - x_{3P} = 0$, hence $3P = (1, 0)$.

Finally $4P = P + 3P$, but here m is not defined because $x_P = x_{3P}$. This implies (since $P \neq 3P$) that $4P = \mathcal{O}$. In fact we have seen that $-(x, y) = (x, -a_1x - a_3 - y)$, so in our case we have $-(x, y) = (x, -x - y)$, in particular $-(1, 1) = (1, 0)$.

Thus P generates a group of order 4. Listing all points in $E(\mathbb{F}_2)$ we find that these are all, and we have proved that $E(\mathbb{F}_2) \simeq \mathbb{Z}/4\mathbb{Z}$.

11.3 ECM

ECM is the elliptic curve method for factoring integers, invented by Hendrik Lenstra. It is the elliptic curve analog of the $p \pm 1$-methods we have already discussed.

As above, pick random numbers x, y modulo N and then pick a, b such that $y^2 \equiv x^3 + ax + b \bmod N$. Now the points on this curve do not form a group with respect to the geometric group law: the problem is that when you compute $P + Q$, the denominator in the formula for x_{P+Q} might be divisible by a prime dividing N. But if the goal is to find such factors, such a failure of the addition formulas is exactly what you want. Thus take $P = (x, y)$ and compute $kP = (x_k, y_k)$ on $E(\mathbb{Z}/N\mathbb{Z})$ for a highly composite number k. For quite a while nothing exciting will happen. But if k becomes so large that the order of the group $E(\mathbb{Z}/p\mathbb{Z})$ divides k, then kP will be the point at infinity on $E(\mathbb{Z}/p\mathbb{Z})$, which means that the denominator of both x_k and y_k must be divisible by p. Thus if we compute the gcd of N and the denominators of x_k during the calculations, eventually we will find that this gcd becomes divisible by p. Except in the rare case when this gcd becomes N, we will have found a nontrivial factor.

In practice one does not work with a single curve, but with several curves simultaneously; if no factor is found, more elliptic curves are used, and the bound for k is increased.

Example

Let $M_n = 2^n - 1$ denote the n-th Mersenne number. On April 25, 1998, the complete factorization of the Mersenne number M_{589} was found. Known factors at the time were $M_{19} = 524287$, $M_{31} = 2147483647$, as well as 18083479 and 36064471. The factorization

$$\begin{aligned} p_{46} &= 2023706519999643990585239115064336980154410119 \\ p_{103} &= 1363513392978191135736018344773125784835722102 2 \\ &\quad 1191396363935505105689670585273510338697541273201602 7769 \end{aligned}$$

of the remaining factor was found using the elliptic curve

$$E : y^2 \equiv x^3 + Ax^2 + x \bmod p_{46},$$

where $A \equiv 7801204199434046494328977903655178242683 03676 \bmod p_{46}$. Its group order is

$$\begin{aligned} \#E(\mathbb{Z}/p_{46}\mathbb{Z}) &= 2023706519999643990585250126089270445504437408 \\ &= 2^5 \cdot 3 \cdot 7^2 \cdot 223 \cdot 661 \cdot 2141 \cdot 2621 \cdot 847031 \cdot 5965699 \cdot 6047191 \cdot 17020639711 \end{aligned}$$

The factorizations of $p_{46} \pm 1$ are

$$\begin{aligned} p_{46} - 1 &= 2 \cdot 7 \cdot 19 \cdot 31 \cdot 53 \cdot 181 \cdot 641 \cdot 39910918849486318887656194928323841, \\ p_{46} + 1 &= 2^3 \cdot 3^3 \cdot 5 \cdot 17 \cdot 97 \cdot 113632646048089975438831565430470598351 1, \end{aligned}$$

hence p_{46} could not have been discovered by the $p - 1$ or $p + 1$ method.

The current record factor found by ECM is the 67-digit prime factor

$$4444349792156709907895752551798631908946180608768737946280238078881$$

of $10^{381}+1$ found by B. Dodson in August 2006. The group order $\#E(\mathbb{F}_p)$ of the elliptic curve that turned out to be successful was

$$\begin{aligned} N = {} & 2^2.3.131.124847.1244459.1785599.3000931.4032877.27225659. \\ & 29985143.87373729.11805290281 \end{aligned}$$

11.4 ECPP

ECPP is the abbreviation of elliptic curve primality proofs. The basis for these proofs is the following

Theorem 11.4 (Goldwasser-Killian). *Let n be an integer coprime to 6, E a smooth Weierstrass cubic defined mod n (so E is an elliptic curve over $\mathbb{F}_n$ if n is prime), and let $N = FR$ be natural numbers.*

Assume that there exists a point $P \in E(\mathbb{Z}/n\mathbb{Z})$ such that

i) $NP = \mathcal{O}$;

ii) $\frac{N}{q}P \neq \mathcal{O}$ *for each prime* $q \mid F$;

then for every prime $p \mid n$ we have $F \mid \#E(\mathbb{F}_p)$. Moreover, if $F > (n^{1/4}+1)^2$, then n is prime.

Proof. Modulo some obvious changes this is the same proof as for conics. We have $NP = \mathcal{O}$ in $E(\mathbb{F}_p)$, but $\frac{N}{q}P \neq \mathcal{O}$ for all $q \mid F$, and this implies that F divides the order of the eliptic curve $E(\mathbb{F}_p)$. If $F > (n^{1/4}+1)^2$, then $\#E(\mathbb{F}_p) > (n^{1/4}+1)^2$; from Hasse's theorem we know that $\#E(\mathbb{F}_p) \le p+1+2\sqrt{p} = (\sqrt{p}+1)^2$. This gives a contradiction of $p \le \sqrt{n}$, hence all prime factors of n are $> \sqrt{n}$, and this means that n is prime. $\square$

The actual primality proof based on this criterium works as follows:

1. Pick a random pair $a, b \in \mathbb{N}$ with $\gcd(4a^3+27b^2, n) = 1$ (If n is prime, this guarantees that the polynomial $f(X) = X^3 + aX + b$ has distinct roots in $\mathbb{F}_p$), and consider $E : Y^2 = f(X)$.

2. Compute the order N of the group $E(\mathbb{Z}/n\mathbb{Z})$. There is an ingenious polynomial time algorithm due to Schoof that does this if n is prime; if Schoof's algorithm fails, return "n is composite".

3. Try to find a probable prime factor $q > (n^{1/4}+1)^2$ of N with $q < N$. If this fails, go back to step 1.

4. Find a point P on E (pick a random value of x, test whether $(\frac{f(x)}{n} = +1$; if yes, compute a square root y of $f(x) \bmod n$ (our algorithm works if n is prime; if the square root algorithm fails, return "n is composite"), and set $P = (x, y)$.

5. Compute $Q = \frac{N}{q}P$; if this calculation fails, return "n is composite"; if $Q = \mathcal{O}$, go back to step 4. Now check whether $qQ = \mathcal{O}$; if this calculation fails, or if $qQ \neq \mathcal{O}$, return "n is composite". Otherwise return "If q is prime, then so is n".

It remains to verify that q is prime. This is done via ECPP; since the size of q decreases approximately by at least a factor of 2 in every step, the size of q will eventually be small enough so that the $q-1$-test (or even trial division) will prove that q is prime.

The bottleneck of this version of ECPP is the application of Schoof's point counting algorithm. Atkin and Morain have suggested a variant of ECPP in which curves with a *given order* are constructed. This is currently one of the fastest general primality proof algorithms available (the other one is based on cyclotomic number fields, and objects like Gauss and Jacobi sums).

11.5 Point Counting

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve defined over $\mathbb{F}_p$. For applications to primality tests we have to find the group order $\#E(\mathbb{F}_p)$.

The Naive Method

The naive method proceeds by computing the Legendre symbols $(\frac{f(x)}{p})$ for all $x = 0, 1, \ldots, p-1$; if this symbol is -1, there is no point (x, y) on E, if it is 0, we have 1 point $(x, 0)$, and if it is 2, there are two of them, namely (x, y) and $(x, -y)$ for some y with $y^2 = f(x)$. Thus we find

$$\#E(\mathbb{F}_p) = \sum_{x=0}^{p-1} \Big(1 + \Big(\frac{f(x)}{p}\Big)\Big).$$

This algorithm needs clearly $O(p)$ steps and is therefore useless for large primes p.

Shanks' Baby-step Giant-step method

A slightly better method for moderate values of p is provided by the baby-step giant-step method of Shanks:

1. Pick a point $P \in \mathrm{E}(\mathbb{F}_p)$ and compute $Q = (p+1)P$.

2. Choose an integer $m > p^{1/4}$. Compute and store the points $\mathcal{O}$, P, $2P$, $\ldots$, mP (baby steps).

3. Compute the points $Q + k(2mP)$ for $k = -m, -m+1, \ldots, m$ (giant steps) until there is a match $Q + 2mkP = \pm jP$ for some $j \leq m$.

4. Put $M = p + 1 + 2mk \pm j$; we know that $MP = \mathcal{O}$. Factor M and let $p_1, \ldots, p_r$ denote its prime factors.

5. If $\frac{M}{p_i} P = \mathcal{O}$ for some i, replace M with M/p_i and go back to step 4. If $\frac{M}{p_i} P \neq \mathcal{O}$ for all i, the order of P is M.

If we want to compute the group order of $E(\mathbb{F}_p)$, we find as many points P and their orders until the lcm of their orders divides exactly one integer in the interval $[p - 2\sqrt{p}, p + 2\sqrt{p}]$. The analysis of the bsgs method (why is there a match?) is straightforward and left as an exercise.

Schoof's Algorithm

Schoof's algorithm uses more advanced concepts like the Frobenius automorphism of an elliptic curve, or division polynomials. This is unfortunately beyond the scope of these lectures. There is a huge activity in the area of point counting on elliptic curves, and in recent years quite advanced methods have been proposed to solve this problem effectively for large values of p.

Exercises

11.1 List all points on $E(\mathbb{F}_3)$ and $E(\mathbb{F}_5)$ for $E : y^2 = x^3 + x$. Compute an "addition table" for these groups, and determine their structure.

11.2 Show that $\#E(\mathbb{F}_p) + \#E'(\mathbb{F}_p) = 2p + 2$, where $E : y^2 = x^3 + ax + b$ and $E' : y^2 = x^3 + g^2ax + g^3b$ are elliptic curves defined over $\mathbb{F}_p$, and $(\frac{g}{p}) = -1$.

Chapter 12

Elliptic Curve Cryptography

In this chapter we will discuss cryptographic protocols based on the groups $E(\mathbb{F}_p)$. In the following, we will always work with an elliptic curve $E : Y^2 = X^3 + aX + b$ over $\mathbb{F}_p$; G will denote a point on $E(\mathbb{F}_p)$ with order n (quite often a prime); moreover we put $\#E(\mathbb{F}_p) = nh$.

12.1 Key Exchange

The elliptic analog of Diffie-Hellman works like this: Alice and Bob agree on the parameters (p, a, b, G, n, h).

Alice	Eve	Bob
Alice and Bob agree upon (p, a, b, G, n, h)		
Alice picks a random $a \leq n$		Bob picks a random $b \leq n$
Alice computes $A = aG$		Bob computes $B = bG$
Alice sends A to Bob	$\xrightarrow{A}$	
	$\xleftarrow{B}$	Bob sends B to Alice
Alice computes $K = aB$		Bob computes $K = bA$

Figure 12.1: The Diffie-Hellman key exchange

EC-DH is based on the assumption that it is difficult to solve the elliptic Diffie-Hellman problem: computing abG from the knowledge of G, aG, and bG. Clearly anyone who can solve the DLP on elliptic curves (compute a from G

and aG) can solve the EDH problem. It is not known whether there is a more efficient way of solving EDH than via discrete logs.

12.2 Signatures

Here we discuss the elliptic analog of DSA.

EC-DSA

The elliptic analog of DSA proceeds as follows.

1. Alice picks an elliptic curve E over some finite field $\mathbb{F}_q$ such that $\#E(\mathbb{F}_q) = fr$, where r is a large prime and f is very small ($f = 1, 2, 4$ are typical values).

2. Alice chooses a point G of order r on $E(\mathbb{F}_q)$.

3. Alice chooses a random a with $0 < a < r$ and computes $Q = aG$.

4. The secret key is a, the public key is (q, E, r, G, Q).

If Alice wants to sign a message m, she picks a hash function h with values $< r$ and computes $h = h(m)$. Then she chooses a random integer k with $0 < k < r$, computes $R = kG = (x, y)$, and then sets $s \equiv k^{-1}(h + ax) \bmod r$. She then signs the message m with the pair (R, s).

When Bob receives the encrypted message, he first decrypts it by computing m. Then he verifies Alice's signature as follows. Bob computes the hash value $h = h(m)$, $u \equiv s^{-1}h \bmod r$, $v \equiv s^{-1}x \bmod r$, and finally $U = uG + vQ$. He declares the signature valid if $U = R$.

In fact, if the message is signed correctly, then

$$U = uG + vQ = s^{-1}hG + s^{-1}xQ = s^{-1}(h + ax)G = kG = R.$$

12.3 Message Encryption via Elliptic Curves

We will briefly discuss elliptic variants of Massey-Omura, as well as two elliptic analogs of RSA, namely KMOV and Demytko's scheme. A typical problem is that of embedding a message m into a given elliptic curve E. One solution is the following: add 10 bits to the message m in such a way that the enlarged message m' is the x-coordinate of a point on E. This is possible with high probability and seems to work well in practice.

ElGamal

If Alice wants to send Bob a message M encrypted with ElGamal, Bob has to set up a public key as follows. He picks an elliptic curve E defined over a finite field $\mathbb{F}_q$ such that the DLP is hard for the group $E(\mathbb{F}_q)$. He chooses a point

P on E such that the order of P is (divisible by) a large prime. He picks a random integer s and computes $B = sP$. Bob's public key then is $(\mathbb{F}_q, E, P, B)$, his private key is the integer s.

Alice	Eve	Bob
Alice picks a random integer k and computes $M_1 = kP$ as well as $M_2 = M + kB$, and sends M_1, M_2 to Bob.	$\xrightarrow{M_1, M_2}$	
		Bob decrypts the message by computing $M = M_2 - sM_1$

Figure 12.2: The ElGamal public key cryptosystem

This works because $M_2 - sM_1 = (M + kB) - s(kP) = M + k(sP) - skP = M$.

If Alice uses the same "random" integer k for two messages M and M', and if Eve knows the plaintext M, then she can compute M'. How?

Massey-Omura

Here Alice and Bob agree on an elliptic curve E defined over a finite field $\mathbb{F}_q$, where E is chosen in such a way that the DLP in $E(\mathbb{F}_q)$ is hard (just as for the standard DLP, there are some bad choices for which the DLP can be solved efficiently). They compute the group order $N = \#E(\mathbb{F}_q)$ and use a method for embedding messages as points M on the elliptic curve E. Then they basically follow the classical protocol of Shamir:

Alice	Eve	Bob
Alice and Bob agree upon (E, q)		
Alice picks m_A, m_A^{-1} with $m_A m_A^{-1} \equiv 1 \bmod N$		Bob picks a pair m_B, m_B^{-1} with $m_B m_B^{-1} \equiv 1 \bmod N$.
Alice computes $M_1 = m_A M$ and sends M_1 to Bob.	$\xrightarrow{M_1}$	
	$\xleftarrow{M_2}$	Bob computes $M_2 = m_B M$ and sends M_2 to Alice
Alice computes $M_3 = m_A^{-1} M_2$ and sends M_3 to Bob	$\xrightarrow{M_3}$	
		Bob decrypts the message by computing $M = m_B^{-1} M_3$

Figure 12.3: Massey-Omura

In fact, $m_B^{-1}M_3 = m_B^{-1}m_A^{-1}m_Bm_AM = M$ since $m_Am_A^{-1}M = (kN+1)M = M$ etc.

KMOV

This protocol, which was suggested by Koyama, Maurer, Okamoto & Vanstone, works as follows. Alice picks primes $p, q \equiv 2 \bmod 3$ amd forms the RSA-key $n = pq$. It is easy to see that $\#E(\mathbb{F}_p) = p + 1$ for all primes $p \equiv 2 \bmod 3$ and elliptic curves $E : Y^2 = X^3 + b$ over $\mathbb{F}_p$. Her public encryption key is a random integer e chosen coprime to $N = \operatorname{lcm}(p + 1, q + 1)$, and her private decryption key d is computed via $de \equiv 1 \bmod N$. Although $E(\mathbb{Z}/n\mathbb{Z})$ is not a group, it is still true that $NP = \mathcal{O}$ for (almost) all points on $E(\mathbb{Z}/n\mathbb{Z})$.

For sending a message $M = (m_1, m_2) \in E(\mathbb{Z}/n\mathbb{Z})$ to Alice, Bob computes $b \equiv m_2^2 - m_1^3 \bmod n$ and computes $C = eM$ on the elliptic curve $E : y^2 = x^3 + b$; then he sends the pair (b, C) to Alice, who decrypts it by computing $M = dC = deM$ on $E : y^2 = x^3 + b$ and recovers the original message by dropping the last 10 bits.

Demytko

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_p$, and let d denote a quadratic nonresidue mod p. Then the cubic $E_d : dy^2 = x^3 + ax + b$ can be brought into Weierstrass form by multiplying through with d^3 ands setting $d^2y = Y$, $dx = X$: this gives $Y^2 = X^3 + d^2aX + d^3b$. The elliptic curve E_d is called a quadratic twist of E.

It is an easy exercise to show that if $\#E(\mathbb{F}_p) = p + 1 - a_p$, then $\#E_d(\mathbb{F}_p) = p + 1 + a_p$. Moreover, if x is not the x-coordinate of a point on E, then it is the x-coordinate of a point on E_d. For a point $P = (x, y)$, let $k * x$ denote the x-coordinate of the point kP.

Now Alice chooses an RSA-modulus $n = pq$ and an elliptic curve $E : y^2 = x^3 + ax + b$ defined over $\mathbb{Z}/n\mathbb{Z}$. She also finds quadratic nonresidues $u \bmod p$ and $v \bmod q$, and defines the quadratic twists $E^{+-} = E_u(\mathbb{F}_p) \oplus E(\mathbb{F}_q)$, $E^{-+} = E(\mathbb{F}_p) \oplus E_v(\mathbb{F}_q)$, and $E^{--} = E_u(\mathbb{F}_p \oplus E_v(\mathbb{F}_q)$. Let e be an integer coprime to the groups orders of these four curves (i.e., coprime to $p+1\pm a_p$ and $q+1\pm a_q$). In order to encrypt a message m, she first computes $(\frac{f(m)}{p})$ and $(\frac{f(m)}{q})$; if both symbols are positive, then m is the x-coordinate of some point $M \in E(\mathbb{Z}/n\mathbb{Z})$. If $(\frac{f(m)}{p}) = +1$ and $(\frac{f(m)}{q}) = -1$, then m is the x-coordinate of some point $M \in E^{+-}$ etc. Bob now computes $C = e * M$ and sends C to Alice. For decrypting the message, Alice sets

$$\begin{aligned}
N_1 &= \operatorname{lcm}(p+1-a_p, p+1-a_q) && \text{if } (\tfrac{w}{p}) = +1, (\tfrac{w}{q}) = +1,\\
N_2 &= \operatorname{lcm}(p+1-a_p, p+1+a_q) && \text{if } (\tfrac{w}{p}) = +1, (\tfrac{w}{q}) = -1,\\
N_3 &= \operatorname{lcm}(p+1+a_p, p+1-a_q) && \text{if } (\tfrac{w}{p}) = -1, (\tfrac{w}{q}) = +1,\\
N_4 &= \operatorname{lcm}(p+1+a_p, p+1+a_q) && \text{if } (\tfrac{w}{p}) = -1, (\tfrac{w}{q}) = -1,
\end{aligned}$$

where $w \equiv c^3 + ac + b \bmod n$. Then she computes $d_i \equiv e^{-1} \bmod N_i$ and decrypts the message via $d_i * c = d_i e * m = m$.

Bibliography

[1] J. Buchmann, *Introduction to cryptography*, Springer-Verlag 2001

Gives an undergrad introduction to cryptography, and also provides the necessary background from algebra and elementary number theory.

[2] H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag 1993.

This has become the bible of computational number theorists.

[3] R. Crandall, C. Pomerance, *Prime numbers. A computational perspective*, Springer-Verlag 2000

An excellent account of primality tests and factorization methods, along with the necessary background from number theory. Contains exercises and lots of problems suitable for research at all levels.

[4] N. Koblitz, *Algebraic aspects of cryptography*, Springer-Verlag 1999

A more advanced account of certain topics in cryptography. Mostly discusses complexity questions and finite fields, and gives a readable introductions to elliptic and hyperelliptic curves, and their use in cryptography.

[5] H. Riesel, *Prime numbers and computer methods for factorization*, Birkhäuser 1985.

This book is out of date nowadays, but it contains a very detailed description of the primality tests and factorization methods that had been used prior to the 1980s.

[6] L. Washington, *Elliptic Curves. Number Theory and Cryptography*, Chapman & Hall, 2003

Maybe the best mix between an elementary approach to elliptic curves and one that is serious enough to prove the basic results needed for cryptographic applications.