Mathematical Cryptology Hand-ins

March 2, 2010

Chapter 1

Review exercises

1.1 What are the axioms of a group, ring and field?

A group $(G, \cdot)$ is a set G closed under a binary operation $\cdot$, such that the following axioms are satisfied:

1. $\cdot$ is associative.
2. There is an identity element.
3. Every element has an inverse.

A Ring $(R, +, \cdot)$ is a set R together with two binary operations $+$ and $\cdot$, such that the following axioms are satisfied:

1. $(R, +)$ is an abelian group.
2. Multiplication is associative.
3. The left and right distributive laws hold.

A Field $(F, +, \cdot)$ is a ring with the additional properties

1. Multiplication is commutative.
2. All non-zero elements have a multiplicative inverse.

1.2 Why are the nonzero integers not a group under multiplication?

Not all elements have inverses.

1.3 Why do we say nonzero real numbers when we look at the reals as a group under multiplication?

Zero is the only element that does not have an inverse.

1.4 Why are the integers not a field?

All non-zero elements must have multiplicative inverses in a field, but this is not the case.

1.5 Why is $\mathbb{F}_p[x]/(f(x))$ always a ring?

Checking the ring axioms we see that

1. Addition is associative.
2. Zero is an additive unit.
3. Every element has an additive inverse.
4. Addition is commutative.
5. Multiplication is associative.
6. The left and right distributive laws hold.

1.6 What is the value of the Euler function at $N = p$ and $N = pq$, where p and q are primes.

$\phi(p) = p - 1$.
$\phi(pq) = (p-1)(q-1)$.

1.7 Define the Legendre and explain how this is used to test for squares modulo a prime p.

The Legendre symbol is defined for prime p as

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p \text{ divides } a, \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p. \end{cases}$$

If $\left(\frac{a}{p}\right) = 1$, then the equation

$$x^2 = a \pmod{p}$$

has (two) solutions, and a therefore has square roots modulo p.

1.8 If an integer a modulo a composite number N has Jacobi symbol 1 with respect to N, is it a square modulo N?

Not necessarily.

1.9 If there are 60 different coloured balls in a bag, how many do I have to take out on average (with replacement) before I obtain a repeated colour.

This number is given by the approximation formula (end of Chapter 2 in [1])

$$\sqrt{\frac{60\pi}{2}} \approx 9,7$$

.

Standard exercises

1.10 Show that if n is odd and factors into k distinct prime factors, then the number of solutions to $x^2 = 1 \pmod{n}$ is equal to 2^k.

The equation sytem

$$\begin{cases} x^2 = 1 \pmod{p_1} \\ x^2 = 1 \pmod{p_2} \\ \dots \\ x^2 = 1 \pmod{p_k} \end{cases}$$

corresponds to the 2^k equation systems

$$\begin{cases} x = u_1 \pmod{p_1} \\ x = u_2 \pmod{p_2} \\ \dots \\ x = u_k \pmod{p_k} \end{cases}$$

where the u_i can be either $+1$ or -1. The first equation system above is satisfied for those values of x that satisfy *any one* of the 2^k equation systems below. The equation systems below all have a uniquely determined solutions modulo n by the Chinese Remainder Theorem. This shows that there are *at most* 2^k solutions to $x^2 = 1 \pmod{n}$. But all solution must also be unique as any two of the 2^k equation systems cannot produce the same solution. Otherwise we would end up with a contradicting pair of equations

$$\begin{cases} x = 1 \pmod{p_l} \\ x = -1 \pmod{p_l} \end{cases}$$

for some $1 \leq l \leq k$.

1.11 Show that g is a generator of $\mathbb{F}_p^*$ if and only if $g^{p-1} = 1 \pmod{p}$ and $g^{(p-1)/q} \neq 1 \pmod{p}$ for all prime divisors q of $p-1$.

g is a generator of $\mathbb{F}_p^*$ if and only if $g^n = 1 \pmod{p}$ for $n = \left|\mathbb{F}_p^*\right|$ but for no smaller value for n. Now, $n = \left|\mathbb{F}_p^*\right| = p - 1$, so it only remains to see that $g^{(p-1)/q} = 1 \pmod{p}$ for some prime divisor q when g is not a generator. But that is clear since the order of g must divide $p - 1$.

1.12 Given a composite integer N (which we cannot factor) and the three integers a, b and $c = a^{\frac{1}{3}} b^{\frac{1}{5}} \pmod{N}$, show to compute $a^{\frac{1}{3}} \pmod{N}$ and $b^{\frac{1}{5}} \pmod{N}$ with low complexity.

Note that a has a unique inverse in $\mathbb{Z}_N$ iff $gcd(a, N) = 1$, so non-existence of a^{-1} implies that we can find a factor of N, which contradicts the purpose of this exercise. We therefore assume that a and b are invertible.
Compute a^{-1}, b^{-1} and $c^{10} = a^{\frac{10}{3}} b^{\frac{10}{5}} = a^2 a^{\frac{1}{3}} b^2$. c^{10} can be calculated using repeated squaring of c for a total of four multiplications. Apply inverses to isolate $a^{\frac{1}{3}}$. Inverses and multiplications modulo N take $O((\lg N)^2)$ bit operations

(see [2]), so a total of (at most) 10 such operations are needed. Correspondingly, to isolate $b^{\frac{1}{5}}$ compute $c^6 = a^{\frac{6}{3}} b^{\frac{6}{5}} = a^2 b b^{\frac{1}{5}}$ (three multiplications) and apply the inverses a^{-1} and b^{-1} for a total of (at most) eight operations.

1.13 By hand or by calculator, compute the Jacobi symbols

$$\left(\frac{311}{653}\right) \text{ and } \left(\frac{666}{777}\right).$$

$\left(\frac{311}{653}\right) = \left(\frac{653}{311}\right) = \left(\frac{31}{311}\right) = -\left(\frac{311}{31}\right) = -\left(\frac{1}{31}\right) = -1.$
$\left(\frac{666}{777}\right) = 0$ since $gcd(666, 777) > 0$.

Programming exercise

1.14 Implement an algorithm to compute Legendre symbols which uses the law of quadratic reciprocity.

```
#include <stdlib.h> /* atoi */
#include <stdio.h> /* printf */

#define ODD(n) ((n) & 1)

int legendre(int q, int p) {
  if (q < 2) return q;
  if (q == 2) return ODD((p*p - 1) / 8) ? -1 : 1;
  if (q > p) return legendre(q % p, p);
  return ODD((p-1)*(q-1)/4) ? -legendre(p, q) : legendre(p, q);
}

void main(int argc, char *argv[]) {
  int q = atoi(argv[1]), p = atoi(argv[2]); /* no input check */
  printf("( %d / %d ) = %d\n", q, p, legendre(q, p));
}
```

Chapter 2

Review exercises

2.1 When is an elliptic curve singular?

An elliptic curve

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

with

$$\begin{cases} b_2 = a_1^2 + 4a_2 \\ b_4 = a_1a_3 + 2a_4 \\ b_6 = a_3^2 + 4a_6 \\ b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\ \Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6 \end{cases}$$

is singular if and only if $\Delta = 0$.

2.2 If two curves have the same j-invariant, what does it mean?

Two elliptic curves with the same j-invariant are isomorphic over $\overline{K}$ (the algebraic closure of K). (Conversely, two curves that are isomorphic over K have the same j-invariant.)

2.3 Describe the chord-tangent process and how it relates to the elliptic curve group law.

The K-rational points on the curve, plus the origin, form an abelian group under the addition operation defined by the chord process.

> **Chord Process:** Let L denote the line that intersects the two given K-rational points P and Q. L intersects the curve at a third point denoted R. Define $P + Q$ as R with a negated y-coordinate.

The tangent process is simply a degenerated chord process in which P and Q overlap. In this case the tangent line at $P = Q$ is used. If the line is vertical, there will be only two intersection points. In this case the origin is taken as the third point.

2.4 What is the trace of Frobenius, and what inequality does it satisfy?

Over a finite field $\mathbb{F}_q$, the number of rational points on a curve is finite and denoted $|E(\mathbb{F}_q)|$. The expected number of points on the curve is around $q+1$ and if we set

$$|E(\mathbb{F}_q)| = q + 1 - t,$$

then t is the trace of Frobenius at q and the following inequality holds (H. Hasse, 1933):

$$|t| \leq 2\sqrt{q}.$$

2.5 When is an elliptic curve anomalous?

When the trace of Frobenius is one.

2.6 Why does one use projective coordinates?

To avoid the need for costly division.

2.7 By roughly what precentage does point compression reduce the amount of bandwith needed to transmit an elliptic curve point?

Almost 50%.

Standard exercises

2.8 From the geometric definition of the group law derive the formulae in Lemma 2.2.

For an elliptic curve

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6,$$

let $P_i = (x_i, y_i)$ denote points on the curve. Then $-P_i = (x_i, -y_i - a_1x_i - a_3)$. Let $L : y = \lambda x + \mu$ be the line passing through P_1 and P_2. Denote the left and right hand side of the equation for E by $A(X, Y)$ and $B(X)$, respectively, and consider the polynomial $f(x) = B(x) - A(x, \lambda x + \mu)$ which is zero for the three x-coordinates where E and L intersect. We know that the three roots are x_1, x_2 and x_3, so $f(x) = (x - x_1)(x - x_2)(x - x_3)$. By comparing the coefficients of the x^2-terms on each side we obtain

$$\begin{aligned} &a_2 - \lambda^2 - a_1\lambda = -x_1 - x_2 - x_3 \Leftrightarrow \\ &x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \end{aligned}$$

as desired. The equation for y_3 is obtained by combining L with the inversion formula for $-P_3$ above. This gives us

$$\begin{cases} x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 = -(\lambda + a_1)x_3 - \mu - a_3 \end{cases}$$

as in Lemma 2.2 in Smart's book [1]. The explicit formulas for λ and μ are derived from a slope computation and an algebraic manipulation of $L(P_1)$ in the case $x_1 \neq x_2$. For the point doubling case one can compute the line tangent to E at P_1 (Let $f(x,y) = B(x) - A(x,y) = 0$ define the curve, compute derivatives, and so on...).

2.9 Let E denote the elliptic curve

$$Y^2 = X^3 + aX + b$$

over a field of characteristic greater that three. Describe the possible points of order three on the curve.

A point on this curve is a point of order 3, if and only if its x-coordinate is a solution to

$$3x^4 + 6ax^2 + 12bx - a^2 = 0.$$

To see this, we note that we must have

$$2P = -P.$$

Hence, the x-coordinate of $2P$ must be equal to the x-coordinate of P. This gives an equation to solve, which can be simplified to the above equation.

2.10 Show that the curves

E_1**:** $Y^2 = X^3 + aX + b$**,**

E_2**:** $Y^2 = X^3 + ad^2X + bd^3$**,**

defined over the field K are isomorphic over the algebraic closure of K. When are they isomorphic over K?

We need to verify that $j(E_1) = j(E_2)$ (j-invariant) to show isomorphism over the algebraic closure. We have

$$j(E) = \frac{c_4^3}{\Delta} = \frac{1728c_4^3}{c_4^3 - c_6^2}.$$

It is sufficient to note that the number of factors d in c_4^3 and c_6^2 are equal so that all d's in the nominator and denominator may be removed. Using the more complex Δ-formula for $char(E_1) \in \{2,3\} \ni char(E_2)$ does not change anything.

To investigate isomorphism over K we seek a change of basis. We try to find variables $r, s, t \in K$ and $u \in K^*$ such that the change of variables

$$\begin{cases} X = u^2X' + r \\ Y = u^3Y' + su^2X' + t \end{cases}$$

transforms E_1 into E_2. We get

$$\begin{aligned}
&Y^2 = X^3 + aX + b \Leftrightarrow \\
&(u^3Y' + su^2X' + t)^2 = (u^2X' + r)^3 + a(u^2X' + r) + b \Leftrightarrow \\
&(u^3Y')^2 = (u^2X')^3 + a(u^2X') + b \Leftrightarrow && [r = s = t = 0] \\
&Y'^2 = X'^3 + av^4X' + bv^6, && \left[v = u^{-1}\right]
\end{aligned}$$

which holds when $v^2 = d$ for some $v \in K^*$. Isomorphism over K is therefore true if and only if d is a square root in K.

2.11 Write down an elliptic curve version of the ElGamal encryption scheme.

Let E be an elliptic curve with generator p, and let the message m be represented as a point on E. Denoting multiplication-by-a by $[a]P = \underbrace{P + \ldots + P}_{a}$ according to Smart [1], and we obviously have $[a]([b]P) = [ab]P = [ab]P$. Now choose a random number r as the private key. Select a random ephemeral key number k and compute the ciphertext

$$(c_1, c_2) = ([k]p, m + [kr]p).$$

Decryption can then be performed according to

$$c_2 - [r]c_1 = (m + [kr]p) - [r]([k]p) = m.$$

Programming exercise

2.12 Implement a software library to perform elliptic curve addition and doubling over the integers modulo p. Implement ElGamal as in 2.11.

Chapter 3

Review exercises

3.1 What are the three most common letters in English? What are the most common bigrams and trigrams in English?

Letters: E, T, A
Bigrams: TH, HE, AN
Trigrams: THE, ING, AND

3.2 Explain how the Vigenére cipher is related to the shift cipher.

The Vigenére cipher uses a p-character key to define p *different* shift amounts, which are repeated in a cyclic fashion. This is essentially a way of using p different shift ciphers.

3.3 Explain relations between the substitution cipher and the Enigma cipher.

The Enigma cipher combines several substitution ciphers in one machine. A set of rotors are at the core of the Enigma, and each rotor specifiec one substitution cipher. The rotors move to produce a new substitution for every letter that is encrypted.

3.4 Describe the difference and similarities between permutation and substitution as cipher components.

Permutations move data around but leaves it intact in some sense, while substitutions change the data but leaves it in its place. Similarities...(can't think of any :)...they both scramble data.

Standard exercise

3.5 In a chosen plaintext attack the attacker is allowed to ask for the encryption of a plaintext of her choosing. Show that the Caesar, Vigenère and substitution ciphers can be broken instantly under a chosen plaintext attack. Determine the smallest amount of plaintext needed.

Assuming 26-letter alphabet A-Z below.
The Caesar cipher, when defined as a shift cipher with key three, requires no plaintext to be broken as the key is already known. When the key is unknown, a single character plaintext 'A' is sufficient.
The key of a substitution cipher can be recovered by looking at the encyption of the plaintext 'ABC...Y' of length 25.
If the period p of the Vigenère cipher is known, then the plaintext consisting of p consecutive A's will suffice to reveal all the shifts. It is not necessarily possible to determine p when unknown, which is realized by considering the case with ∞ alphabets and shifts determined by the decimals of π. Under more relaxed conditions we can determine p, but then we need multiple queries. If we are allowed to keep asking for the next character encryption of 'A' incrementally, then we will have found the key after precisely p characters. We may not be able to verify this, however, unless we have a separate plaintext/ciphertext pair to check against (case A). Under the assumption that the second occurrence of the first character reveals the maximal period, as in 'AAAAAAAA' → 'KEYWORDK', $p + 1$ characters are sufficient (case B). Many other arrangements (cases) seem possible. If we are *not* allowed to encrypt incrementally, then we can use the doubling strategy. Starting with the two-character sequence 'AA' and doubling the length until we find the period, we use at most $4p-6$ or $4p-2$ plaintext characters for the two cases A and B, respectively.
For a polyalphabetic substitution cipher with a known period p (the number of alphabets), the plaintext

$$\underbrace{A \ldots A}_{p}\underbrace{B \ldots B}_{p} \ldots \underbrace{Y \ldots Y}_{p}$$

with p repetitions of all but the last character will reveal all subkeys for a total of $25p$ plaintext characters. With p unknown it gets trickier. The worst case scenario will have us try the above string for all values of $p \geq 2$, which implies a need for

$$25\,(2 + 3 + 4 + \ldots + p) = 25(p + 2)(p - 1)/2$$

characters to recover all keys for a total of $(4p-6)+25(p+2)(p-1)/2$ plaintext characters when incremental encryption is not allowed.

Chapter 5

Review exercises

5.1 Is an attacker more interested in $p(C = c|P = m)$ or $p(P = m|C = c)$? Explain.

$p(P = m|C = c)$ should be more interesting to an attacker in general, because this entity says something about an unknown plaintext when only the ciphertext is available.

5.2 How is entropy defined?

Let X be a random variable which takes on a finite set of values x_i with $1 \leq i \leq n$, and has probability distribution $p_i = P(X = x_i)$. The entropy of X is defined to be

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

5.3 If a random variable X takes at most t values, what are the maximum and minimum values possible for $H(X)$?

Minimum value: 0
Maximum value: $\log_2 t$

5.4 Are $H(P|K, C) = 0$ and/or $H(K, P) = H(K)+H(P)$ always true for any cipher? Explain why/why not.

$H(P|K, C) = 0$: Yes, if we know both the key K and ciphertext C, then we also implicitly know the plaintext P.
$H(K, P) = H(K) + H(P)$: No, the equation is true only when K and P are independent. If the key is derived from the plaintext in some simple way, for example, or if the key is communicated in the message (pointless, but possible), then the equality is not valid.

5.5 Define the terms spurious keys and unicity distance.

Spurious keys (from [1]):
Let $c \in \mathbb{C}^n$ be a ciphertext of length n, and let $\mathbb{K}(c)$ be the set of keys which produce a "meaningful" decryption of c. The number of spurious keys, given c,

is defined to be $|\mathbb{K}(c)| - 1$.
The average number of spurious keys for a ciphertext of length n is given by $\bar{s}_n = \sum_{c \in \mathbb{C}^n} p(C = c)(|\mathbb{K}(c)| - 1) = \left(\sum_{c \in \mathbb{C}^n} p(C = c)\,|\mathbb{K}(c)|\right) - 1$.
Unicity distance (from [2]):
The unicity distance of a cipher is the minimum amount of ciphertext required to allow a computationally unlimited adversary to recover the unique encryption key.

Chapter 7

Review exercises

7.1 What is an LFSR? Why can they not be used directly to create a stream cipher?

A Linear Feedback Shift Register is a linear bit container whose content is shifted (moved) from left to right one bit at discrete time instances. The bit position at the far left is filled with a linear combination of the content of selected bit positions in the bit container. The bit that "falls out" at the right end is called the output bit. LFSRs cannot be used directly to create a stream cipher because they are inherently linear, which would make breaking it quite easy.

Chapter 8

Review exercises

8.1 What is Kerchhoff's principle?

The encryption and decryption algorithm should be publicly known, and the secrecy of the message, given the ciphertext, should depend entirely on the secrecy of the key.

8.2 Describe the operation of a Feistel cipher.

The plaintext block is divided into two halves L_0 and R_0, after which r rounds proceed according to

$$\begin{cases} L_i = R_{i-1}, \\ R_i = R_{i-1} \oplus F(K_i, R_{i-1}), \end{cases}$$

for $1 \leq i \leq r$ where the K_i's are the subkeys derived from a keyschedule, and F is a complicated "inner" function. The ciphertext is given by L_r and R_r.

8.3 What needs to change in a Feistel cipher between encryption and decryption?

The order in which the subkeys are applied.

8.4 Describe the operation of triple DES.

Single DES is applied three times with different keys. Cipherttext c is produced from plaintext m according to

$$c = DES_{k3}(DES_{k2}^{-1}(DES_{k1}(m)))$$

.

8.6 Describe the role of permutations and substitutions in DES.

The expansion permutation expands and permutes the right half of 32 bits to 48 bits. The expansion permutation is such that one input bit affects two substitutions in the output via the S-boxes below. This spreads dependencies and creates an avalanche effect. The S-boxes provide non-linearity, and are the main contributors to the security of DES.

8.7 Describe the role of permutations and substitutions in AES.

The substitution S-boxes (SubBytes) provide non-linearity, while the permutations (ShiftRows) allows different parts of the state to be mixed (MixColumns) for increased diffusion.

8.8 What problems are associated with ECB mode, and how does CBC mode solve these issues?

Identical plaintext blocks are encrypted into identical ciphertext blocks in ECB mode. This does not happen in CBC mode as the preceeding cipherblock in this case is added to the next plaintext block before encyption.

8.9 What is a MAC? Give example where one could use a MAC.

A Message Authentication Code is a keyed hash function that can be used to ensure the integrity of a message. A message and an (unkeyed) hash of it may at first seem adequate for this purpose, but an adversary may manipulate both the message *and* its hash. A MAC solves this problem.

Standard exercises

8.10 Let $\bar{a}$ denote the the bitwise complement of a. Show that if

$$c = DES_k(m)$$

then

$$\bar{c} = DES_{\bar{k}}(m).$$

In the inner function F (see chapter 8, figure 5 in Smart's book [1]), the inverted plaintext and key cancel out at the XOR after the expansion but before the S-boxes. This is clear after realizing that the key schedule simply produces complemented subkeys (bit selection and circular shifts do not affect). Comparing to the uninverted case below we have $F(\bar{R_0}, \bar{K_1}) = F(R_0, K_1)$. We then get (see chapter 8, figure 4 in Smart's book [1]) $R_1 = \bar{L_0} + F(R_0, K_1)$ (compared to $R_1 = L_0 + F(R_0, K_1)$ uninverted). The contributions of F are added in the rounds below, just as in the uninverted case, and the inverted L's and R's cancel out the inverted subkeys in F. This effect will continue through the rounds to finally produce the inverted ciphertext.

8.11 Consider the following composition of a block cipher with an n-bit key size

$$c = E_{k_1}(E_{k_2}(m)),$$

to produce a cipher with a key size of $2n$ bits. Show that there is a chosen ciphertext attack on this composition that requires $O(2^n)$ memory and $O(2^n)$ encryption/decryptions using E.

Fix a ciphertext c and obtain the corresponding plaintext m. For all 2^n possible keys i for k_1, store $m_i = E_i^{-1}(c)$ with the corresponding key in a hash table (searchable in constant time) sorted by m_i. Now run through all 2^n keys j for k_2. Compute $c_j = E_j(m)$ and check if c_j exists in the hash table. If $c_j = m_l$ for some l, output $k_1 = l$ and $k_2 = j$, and we are done. False alarms are possible, but these can be identified by verifying against an additional plaintext/ciphertext-pair. Note that a *known* (rather than chosen) ciphertext is sufficient.

8.12 Consider triple DES but with two keys instead of three, i.e.,

$$c = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_1}(m))).$$

Describe a chosen plaintext attack on this two-key version of triple DES which requires roughly 2^{56} steps and storage of 2^{56} encryptions under single DES.

Choose a fixed plaintext a. Then generate a table containing $m_i = DES_i^{-1}(a)$ for all 2^n keys i for k_1. The m_i's are the potential plaintexts that, when encrypted with key k_1, produce a. For every m_i we ask for the corresponding ciphertext c_i and compute $b_i = DES_i^{-1}(c_i)$. Store i and b_i in a hash table sorted on the b_i's. This is the off-line part of the job. Now run through all 2^n keys j for k_2. Compute $\bar{b}_j = DES_j^{-1}(a)$ and check if $\bar{b}_j$ exists in the hash table. If $\bar{b}_j = b_l$ for some l, output $k_1 = l$ and $k_2 = j$, and we are done. False alarms are possible, but these can be identified by verifying against an additional plaintext/ciphertext-pair.

Chapter 9

Review exercises

9.1 Describe the key distribution problem.

This is a key agreement problem. How do Alice and Bob safely and efficiently decide on which key to use for communication? How do we get that key to Alice and Bob without having Eve pick it up along the way? This problem is part of the more general key management problem.

9.2 How are nonces used in the protocols in this chapter?

They make sure that messages are "fresh".

Standard exercise

9.3 Alice and Bob uses the following protocol for message distribution. Alice chooses her message M, a random value a and sends $A_1 = M \oplus a$. Bob then selects a random b and sends $B_1 = A_1 \oplus b$. Finally, Alice returns $A_2 = B_1 \oplus a$ to Bob. Bob can now compute the message as $A_2 \oplus b$. Show that this protocol is insecure as Eve can recover the message.

An adversary sees messages A_1, B_1 and A_2, which are composed of

1. $M \oplus a$,
2. $M \oplus a \oplus b$,
3. $M \oplus a \oplus b \oplus a$,

so adding (xoring) all messages produces M.

Chapter 10

Review exercises

10.1 What is a MAC? Give examples where one could use a MAC.

See Exercise 8.9.

10.2 What is a suitable output size for a hash function?

Large enough for it to be computationally infeasible to generate preimages, second preimages and collisions. If we decide that, say, 2^{80} is beyond the capability of all attackers then, due to the birthday attack, $2 \times 80 = 160$ bits is a suitable output length.

10.3 What is the relation between the three different properties a hash function should meet?

Preimage Problem: Given $h(x)$, find a y such that $h(y) = h(x)$.

2^{nd} Preimage Problem: Given x, find a y such that $h(y) = h(x)$.

Collision Problem: Find x and y such that $h(x) = h(y)$.

Preimage Resistance is weaker than both 2^{nd} Preimage Resistance and Collision Resistance. 2^{nd} Preimage Resistance is weaker than Collision Resistance.

10.4 Describe briefly how hash functions in the MD4 family are designed.

MD4, MD5, SHA-0, SHA-1, SHA-256, SHA-384 and SHA-512 are all iterated hash functions of Davies-Meyer form. The output H_t of a message $M = M_1 \cdots M_t$ divided into t blocks (after padding) is defined by

$$\begin{cases} H_0 = \text{constant}, \\ H_i = f(H_{i-1}, M_i) \oplus H_{i-1}, 1 \leq i \leq t, \end{cases}$$

where f is a compression function.

Standard exercise

10.5 Give an example of a hash function that is preimage resistant but not second preimage resistant, assuming the existence of a preimage resistant hash function.

Let h be a preimage resistant hash function. Define the new hash function h_2 as applying h to all except the first character of the message m of length l, such that $h_2(m) = h(m_{2\ldots l})$. It is now infeasible to compute a preimage, but given the preimage, say, 'JESUS', of a hash, then all of

AESUS,

BESUS,

CESUS,

...

JESUS,

...

are also preimages of the same hash value.

Note: Comedian Eddie Izzard used the listing above to argue that there had been many disciples before Jesus. If he was right, you'd might want to look twice if you meet Kesus.

Chapter 11

Review exercises

11.1 What is the current fastest factoring algorithm?

The Number Field Sieve is fastest asymptotically (complexity $L_N(\frac{1}{3}, 1.923)$).
The Number Field Sieve is fastest in practice for numbers with more than 100 decimal digits.
The Quadratic Sieve is fastest for factoring integers of between 80 and 100 decimal digits (complexity $L_N(\frac{1}{2}, 1)$).
The Elliptic Curve Method is good for factoring integers $p < 2^{50}$ (complexity $L_p(\frac{1}{2}, c)$).
Trial Division is exponential but works well for very small numbers (complexity $L_N(1, 1)$).

11.2 What is meant by the problems FACTORING, RSA, QUADRES? Put the problems in order of difficulty.

Given $N = pq$ but not p or q, the problems are

> **FACTORING:** Find p and q.
>
> **RSA:** Given $c \in \mathbb{Z}_N$ and an integer e with $gcd(e, (p-1)(q-1)) = 1$, find m such that $m^e \equiv c \pmod{N}$.
>
> **QUADRES:** Given a, determine if $a \equiv x^2 \pmod{N}$ for some x.

$RSA \leq_P FACTORING$
$QUADRES \leq_P FACTORING$
$QUADRES \leq_P RSA$ (?)

11.3 What is meant by the problems DLP, DHP, DDH? Put the problems in order of difficulty.

Given an abelian group $(G, \times)$ and $g \in G$, the problems are

> **DLP:** Given $h \in G$ such that $h = g^x$ find x.
>
> **DHP:** Given $a = g^x$ and $b = g^y$ find $c = g^{xy}$.
>
> **DDH:** Given $a = g^x, b = g^y$ and $c = g^z$, determine if $z = xy$.

$DDH \leq_P DHP \leq_P DLP$

11.4 Describe RSA encryption.

Choose two large distinct primes p and q, and let $N = pq$.
Choose public exponent $e < n$ such that $gcd(e, \phi(N)) = 1$.
Calculate private exponent $d = e^{-1} \pmod{\phi(N)}$.
Encrypt plaintext m to produce ciphertext c according to $c = m^e \pmod{N}$.

11.5 Could one ever have an RSA encryption exponent which was even?

That would contradict the requirement $gcd(e, \phi(N)) = 1$.

11.6 Comment the statements:

a) Knowing the RSA decryption exponent is equivalent to knowing the factors of the RSA modulus.

b) Breaking the RSA encryption algorithm is equivalent to factoring the RSA modulus.

a) Yes, see Section 3.2 in [1].

b) This equivalency has not been proven, and there are some indications to the contrary.

11.7 ElGamal is a randomized encryption algorithms in that encrypting the same plaintext twice will produce different ciphertexts. Is this a good or bad property?

RSA must employ more or less elaborate padding schemes, which is not necessary in ElGamal. Repeated messages can only be detected *after* decryption in ElGamal.

11.8 Describe advantages and disadvantages of Rabin encryption compared to RSA.

The problem of breaking Rabin is proven to be equivalent to factoring, which is not the case for RSA. Encryption is very much faster in Rabin than RSA. RSA beats Rabin when it comes to decryption. Rabin is used much less than RSA.

Standard exercises

11.9 Let N denote an RSA exponent and let $\lambda(N) = lcm(p-1, q-1)$. If e is the encryption exponent, show that the decryption exponent d can be chosen so that $e \cdot d = 1 \pmod{\lambda(N)}$.
Hint: Show that $\lambda(N)$ is the largest order of an element in the group $\mathbb{Z}_N$.

This is a special case of the Carmichael function (see [3]).
Following the hint, first note that $\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ (CRT). $\mathbb{Z}_p^*$ is cyclic of order of $p-1$ and $\mathbb{Z}_q^*$ is cyclic of order of $q-1$. Choose a generator a from $\mathbb{Z}_p^*$ and b from $\mathbb{Z}_q^*$. Then $ord(a) = p-1$ and $ord(b) = q-1$, and the element ab generates a cyclic group of order $ord(ab) = lcm(ord(a), ord(b)) = lcm(p-1, q-1) = \lambda(N)$. Choose d as above. For all plaintexts m in this group with corresponding ciphertext $c = m^e$ we then have $c^d = m^{ed} = m^{1+i\lambda(N)} \equiv m \pmod{N}$ (for some i) as desired.

11.10 Let N denote an RSA exponent and let $\lambda(N) = lcm(p-1, q-1)$. Suppose that the RSA encryption exponent e has order k in $\mathbb{Z}_{\lambda(N)}^*$. Show that

$$m^{e^k} = m \pmod{N}.$$

Hence conclude that the order of e modulo $p-1$ or $q-1$ should be large.

In this case we have $m^{e^k} = m^{1+i\lambda(N)} \equiv m \pmod{N}$ for some i and, if k is small, we can compute e^j in $\mathbb{Z}_{\lambda(N)}^*$ for increasing values of j until we find $k = ord(e)$. Then simply use $d = e^{k-1}$ for decryption.

11.11 Show that if a user is stupid, and chooses a prime N as the modulus in the RSA scheme, it can be trivially broken.

Compute d as the inverse of e modulo N, so that $e \cdot d = 1 \pmod{N}$. Then decrypt as usual.

11.12 Explain why, for large plaintexts, it is better to use public key encryption to transport a symmetric key, and then to use a symmetric encryption scheme to encrypt the data.

Symmetric encryption is generally much faster.

11.13 Show that breaking the Rabin encryption algorithm is equivalent to factoring.

$RABIN \leq_P FACTORING$: If we can factor $N = pq$, then we have the private key (p, q).
$SQUAREROOT \leq_P RABIN$: By choosing $B = 0$, decryption degenerates into $m = \sqrt{c} \pmod N$, which implies ability to compute square roots.
$FACTORING \equiv_P SQUAREROOT \leq_P RABIN \leq_P FACTORING \Rightarrow RABIN \equiv_P FACTORING$.

Chapter 12

Review exercises

12.1 What is the Fermat primality test and what is the main problem associated with it?

The Fermat test checks if

$$a^{n-1} = 1 \pmod{n}$$

for any one of several randomly selected bases $a \in \{2, \ldots, n-1\}$ and if so, outputs "Definitely Composite". If not, the algorithm outputs "Probable Prime". All Carmichael numbers incorrectly pass as (likely) prime numbers.

12.2 What is the main difference between a primality testing algorithm like Miller-Rabin, and a primality proving algorithm like ECPP?

ECPP *proves* primality while Miller-Rabin merely makes it unlikely.

12.3 What is meant by a smooth number and how can one use smooth numbers in factoring algorithms.

Smoothness is another way of saying "Easy to factor using trial division". A number N is B-smooth if every prime factor of N is strictly smaller than B. All modern factoring methods use relations between smooth numbers to enable computing $|\mathbb{Z}_N^*|$, which equates to knowing the factors of N.

12.4 Describe the $P-1$ factoring algorithm.

Given $N = pq$, assume that $p-1$ is B-smooth but $q-1$ is not. Then $p-1$ will divide $B!$, but $q-1$ will most likely not. Computing

$$a = 2^{B!} \pmod{N},$$

we get

$$a = 1 \pmod{p},$$

since $p-1$ divides $B!$ and $a^{p-1} = 1 \pmod{p}$. If

$$a \neq 1 \pmod{q},$$

then p will divide $a-1$ but q will not, and p can be computed as $gcd(a-1, N)$.

12.5 Why does finding two numbers x and y with $x^2 = y^2 \pmod N$ allow us to factor N with a probability around $\frac{1}{2}$?

$x^2 = y^2 \pmod N$ implies $x^2 - y^2 = (x+y)(x-y) = 0 \pmod N$, so p and q each divide either $x+y$ or $x-y$ with equal probability. The chance that p and q are not both in the same factor $x \pm y$ is about $\frac{1}{2}$.

12.6 What is meant by a sieve and why are they used in modern factoring algorithms?

A sieve can be seen as a filtering device, and modern factoring algorithms filter out smooth numbers when looking for linear relations between such numbers.

Standard exercises

12.7 Show that if a composite number n passes the Fermat test to base a, but fails the Miller-Rabin test for the same base, then we can factor n.

The Miller-Rabin test to base a for $n-1 = 2^s m$ with m odd, uses the following set of equations.

$$\begin{cases} a^m = 1 \pmod n, \\ a^{2^j m} = -1 \pmod n, 0 \leq j < s. \end{cases}$$

n is declared "Probable Prime" if any one of the equalities are found to be valid. If none of the above equalities are valid, then $x = a^{2^{s-1}m}$ is neither $+1$ nor -1. But $x^2 = 1 \pmod n$ by the Fermat test, so $(x+1)(x-1) = 0 \pmod n$ and we can use $gcd(x+1, n) > 1$ *and* $gcd(x-1, n) > 1$ to factor n.

The next three questions discuss Pollard's Rho method of factoring. Define the sequence

$$x_0 = 2, x_{i+1} = f(x_i) = x_i^2 + 1 \pmod N.$$

12.8 Show that if we find two values of the sequence with $x_i = x_j \pmod p$ then we can factor N with high probability by computing $gcd(x_i - x_j, N)$.

$x_i = x_j \pmod p$ implies $x_i - x_j = kp$ for some k, so

$$gcd(x_i - x_j, N) = gcd(kp, pq) = p$$

unless k has a factor q, which is not very likely.

12.9 Argue that we must eventually find two indices $i \neq j$ such that $x_i = x_j \pmod p$.

The range of f is finite, so the sequence $x_i, i \geq 0$, must eventually fall into a cycle.

12.10 Show how to find two such indices using a small amount of storage.

Use Floyd's cycle finding algorithm.

Chapter 13

Review exercises

13.1 What cryptographic conclusion do we draw from the Pohlig-Hellman algorithm?

A hard discrete logarithm problem should be set in a group whose order has a large prime factor.

13.2 How does the Baby-Step/Giant-Step method work?

Divide and conquer, write $h = g^x = g^{i+j\lceil\sqrt{p}\rceil}$ and note that $x \leq p$ implies existence of $0 \leq i, j < \lceil\sqrt{p}\rceil$ for all x. Compute the baby steps g^i for $0 \leq i < \lceil\sqrt{p}\rceil$ and store the resulting pairs (g^i, i) in a hash table sorted on the g^i's. Then compute the giant steps one by one according to $h_j = hg^{j\lceil\sqrt{p}\rceil}$ for $0 \leq j < \lceil\sqrt{p}\rceil$ and try to find a match in the baby-step table. A match produces the equality $g^i = hg^{j\lceil\sqrt{p}\rceil}$, which says $h = g^{i+j\lceil\sqrt{p}\rceil}$ for some specific i and j.

13.3 What are the practical problems associated with the Baby-Step/Giant-Step method and how are these overcome when using Pollard's Rho method?

The storage requirement is $O(\sqrt{p})$, which is highly impractical. By noting that a deterministic walk on a finite set must necessarily revisit some node(s) one can see that the walk must ultimately end up in a cycle. Pollard's Rho method caches in on this idea by utilizing Floyd's cycle finding algorithm to find a collision in the "walk function" such that $x_i = x_j$ for some $i \neq j$. The resulting equality can then be used to deduce the exponent x.

13.4 What cryptographic conclusions can we draw from the existence of the Pollard's Rho method?

A group of order 2^n provides at most $\frac{n}{2}$ bits of security.

13.5 What cryptographic conclusions can we draw from the existence of index-calculus algorithms in finite fields?

The group size p for a discrete logarithm based system needs to be of the same order of magnitude as an RSA modulus ($p \geq 2^{1024}$).

13.6 Discuss the statement that elliptic curves offer higher strength per bit than finite fields.

The only practical general algorithm to solve the discrete logarithm problem on an elliptic curve is the parallel Pollard's Rho method. However, some (or most) elliptic curve instances may actually be vulnerable to efficient group-specific algorihms.

Standard exercises

13.7 Use a calculator to compute x such that

$$3^x = 5 \pmod{p},$$

where $p - 1 = 2 \cdot 3 \cdot 101 \cdot 103 \cdot 107^2$.

Our oracle (Maple) tells us that $g = 3$ is a generator of $\mathbb{Z}_p^*$, so we have

$$h = 5 = 3^x \pmod{p}$$

in the standard setting. Reduce to the five subgroups by raising the equation to the power $\frac{p-1}{p^e}$. We obtain the five discrete logarithm problems

$$\begin{cases} -1 &= h^{\frac{p-1}{2}} &= g^{\frac{p-1}{2}x_2} &= (-1)^{x_2} \pmod{n}, \\ 51005556 &= h^{\frac{p-1}{3}} &= g^{\frac{p-1}{3}x_3} &= 51005556^{x_3} \pmod{n}, \\ 636048942 &= h^{\frac{p-1}{101}} &= g^{\frac{p-1}{101}x_{101}} &= 669701651^{x_{101}} \pmod{n}, \\ 688984597 &= h^{\frac{p-1}{103}} &= g^{\frac{p-1}{103}x_{103}} &= 673495024^{x_{103}} \pmod{n}, \\ 79653202 &= h^{\frac{p-1}{107^2}} &= g^{\frac{p-1}{107^2}x_{107^2}} &= 136102507^{x_{107^2}} \pmod{n}, \end{cases}$$

where the value of x_i is $x \pmod{i}$ to follow the notation of Smart [1]. It is clear that $x_2 = x_3 = 1$, and our oracle tells us that $x_{101} = 78$ and $x_{103} = 60$. For x_{107^2} we set

$$x_{107^2} = x_{107^2,0} + 107 \cdot x_{107^2,1},$$

where $x_{107^2,0}, x_{107^2,1} \in \{0, \ldots, 106\}$. We are now solving

$$h' = 79653202 = 136102507^{x_{107^2}} = g'^{x_{107^2}}$$

in a group of order 107^2. Raise this equation to the power 107 by setting $h'' = h'^{107}$ and $g'' = g'^{107}$ and solve the discrete logarithm problem

$$h'' = 340829531 = 527927400^{x_{107^2,0}} = g''^{x_{107^2,0}},$$

in the cyclic group of order 107 by having our oracle tells us that $x_{107^2,0} = 85$. So now we have

$$\frac{h'}{g'^{85}} = g''^{x_{107^2,1}},$$

which is another discrete logarithm problem in the cyclic group of order 107. We find $x_{107^2,1} = 54$ and conclude that $x_{107^2} = 85 + 107 \cdot 54 = 5863$ to produce

the system

$$\begin{cases} x = 1 \pmod 2 \\ x = 1 \pmod 3 \\ x = 78 \pmod{101} \\ x = 60 \pmod{103} \\ x = 5863 \pmod{107^2}, \end{cases}$$

which we can solve by CRT to obtain $x = 332896987$.

Chapter 14

Review exercises

14.1 Explain the man in the middle attack on the Diffie-Hellman key exchange algorithm.

Eve can be the man if she intercepts the commumication between Alice and Bob and exchanges her own keys with both of them.

14.2 Why do digital signature algorithms solve the problem raised by the man in the middle attack?

Signatures provide an identity link.

14.3 Why does a MAC not provide non-repudiation?

MACs say nothing about who has sent the message, there is no link to any identity.

14.4 Describe the DSA signature algorithm, and describe where the security comes from.

Choose a 160-bit prime q and a large prime p such that

- p has between 512 and 2048 bits,
- p divides $q - 1$.

Uniformly at random pick an integer $h < p$ with

$$h^{\frac{p-1}{q}} = g \neq 1.$$

g is then an element of order q in $\mathbb{F}_p^*$. This determines the public parameters (p, q, g). Each user generates their own private signing key x with $0 < x < q$ and associated public key $y = g^x \pmod{p}$.

A signature of a message m is the pair (r, s) obtained from the following recipe:

- Compute the hash $h = H(m)$.
- Choose a random ephemeral key, $0 < k < q$.

- Compute $r = (g^k \pmod p) \pmod q$.
- Compute $s = \frac{(h+xr)}{k} \pmod q$.

To verify a signature (r, s) on the message m, perform the following steps.

- Compute the hash $h = H(m)$.
- Compute $a = \frac{h}{s} \pmod q$.
- Compute $b = \frac{r}{s} \pmod q$.
- Compute $v = (g^a y^b \pmod q) \pmod q$ (y public key of sender).
- Accept signature if and only if $v = r$.

Security comes from the difficulty of computing x given y (discrete logarithm).

14.5 What are the advantages of Schnorr signatures over DSA?

Schnorr signatures are provably secure under the assumption that the discrete logarithm problem is hard. No such proof is known for DSA.

Schnorr is more efficient than DSA for response-type applications in constrained devices, as the response part s of the signature is easy to compute.

14.6 Explain the similarities and differences between the Diffie-Hellman protocol and the MQV protocol.

The message sending part looks identical, but in MQV one computes a common session key that will depend on the long term public keys of Alice and Bob.

Standard exercises

14.7 Show that if a user uses the same ephemeral key k to sign two different messages in the DSA algorithm then an attacker can recover the long term private key x.

A signature of a message m_1 is the pair (r_1, s_1) obtained from the following recipe:

- Compute the hash $h_1 = H(m_1)$.
- Choose a random ephemeral key, $0 < k < q$.
- Compute $r_1 = (g^k \pmod p) \pmod q$.
- Compute $s_1 = \frac{h_1+xr_1}{k} \pmod q$.

Computing the signature $(r_2, s_2))$ of a second message m_2 then yields

- $h_2 = H(m_2)$.

- We use the same ephemeral key k.
- Compute $r_2 = (g^k \pmod p) \pmod q = r_1$.
- Compute $s_2 = \frac{h_2 + xr_2}{k} \pmod q$.

We therefore have

$$s_2 = \frac{(h_2 + xr_2)}{k} = \frac{(h_2 + xr_1)}{k} \pmod q,$$

and

$$s_2 - s_1 = \frac{h_2 - h_1}{k} \pmod q.$$

Since h_1, h_1, s_1 and s_2 are known we can now compute k as $\frac{h_2 - h_1}{s_2 - s_1} \pmod q$. After that we can compute the long term key x according to

$$x = \frac{ks_2 - h_2}{r_1} \pmod q.$$

Chapter 15

Review exercises

15.1 What is the maximum and average number of multiplications in the basic binary exponentiation algorithm?

Average: $\frac{3t}{2} - 1$
Maximum: $2t - 1$

15.2 Why is the smallest possible Hamming weight of an RSA public exponent equal to two?

Primes of the form $2^n + 1$ have Hamming weight two, and no large prime p can have a smaller Hamming weight as p must be both large (which sets a high bit) and odd (sets the least significant bit).

15.3 What stops one from using signed exponentiation technique in an RSA implementation?

Negation in the elliptic curves case is essentially free as we simply negate the x-coordinate. We can therefore choose the coefficients d_i in our multiplicand

$$d = \sum_{i=0}^{l} d_i 2^{e_i}$$

so that $d_i \in \{\pm 1, \pm 3, \ldots, \pm(2^{w-1} - 1)\}$. This is not possible to do in the RSA case.

15.4 Why is Karatsuba multiplication rarely used in an RSA implementation?

Because division must be used in combination, but division is very costly and can be avoided by using Montgomery representation instead.

15.5 What is meant by Montgomery representation and why does it offer advantages in systems implementing algorithms such as RSA?

It means that we multiply numbers by a factor $R = 2^W$ (W=word size) and use the word size property to transform division into a shift operation.

15.6 In algorithms for finite fields of characteristic two, why does one use an analogue of Karastsuba multiplication and not Montgomery multiplication?

Multiplication of polynomials can be efficiently performed by a divide-and-conquer strategy, which is precisely what Karatsuba is, a divide-and-conquer strategy for (ordinary) multiplication.

Standard exercises

15.7 Assume modular multiplication of a k-bit number requires k^2 operations. How much faster is two 512-bit modular exponentiations by 512-bit exponents, compared to a single 1024-bit modular exponentiation by a 1024-bit exponent? Conclude that the CRT method for RSA decryption and signing is more efficient.

A modular exponentiation of an n-bit number with an n-bit exponent requires roughly $f(n) = \frac{3n}{2}$ n-bit modular multiplications, if we employ repeated squaring, for a total workload of $w(k) = f(k) \cdot k^2 = \frac{3k^3}{2}$ operations for k-bit numbers. Two 512-bit modular exponentiations by 512-bit exponents are therefore a factor

$$\frac{w(2k)}{2w(k)} = \frac{\frac{3(2k)^3}{2}}{2 \cdot \frac{3k^3}{2}} = \frac{(2k)^3}{2k^3} = 4$$

faster than one corresponding 1024-bit operation. Combining the two smaller exponentiations requires only two modular multiplications, so the CRT method for RSA decryption is therefore roughly 4 times faster according to this estimate.

15.8 Suppose a device uses CRT to speed up RSA decryption or signatures, i.e. it computes

$$m_p = c^{d \pmod{p-1}} \pmod{p},$$

$$m_q = c^{d \pmod{q-1}} \pmod{q},$$

and then computes m from m_p and m_q via CRT. However, an attacker manages to get the device to compute m_p incorrectly. Show that if m_q is still computed correctly then the attacker can use this broken device to recover the private key.

Suppose we have two devices, one that computes m correctly and one that computes the erroneous value m' using the correspondingly incorrect value for m'_p. We have

$$m = m_p \cdot q \cdot (q^{-1} \pmod{p}) + m_q \cdot p \cdot (p^{-1} \pmod{q})$$

according to CRT, so

$$m' - m = (\underbrace{m'_p - m_p}_{<p}) \cdot q \cdot (\underbrace{q^{-1} \pmod p}_{<p}).$$

Note that this expression lacks a factor p, so $GCD(N, m' - m) = q$.

15.9 Show that the RSA encryption and decryption algorithms can be implemented in $O(n^3)$-bit operations, where n is the bit length of the modulus N.

Exponentiation is what takes time, and repeated squaring solves this in time $O(n^3)$. We have $O(n)$ modular squarings, each requiring $O(n^2)$ operations, which amounts to $O(n^3)$-bit operations, as desired.

15.10 Show that ElGamal encryption requires about $2 \log p$ multiplications modulo p. Hence deduce its bit complexity $O((\log p)^3)$.

$\log p$ should be read as the number of bits in the numbers we are dealing with here. ElGamal encryption uses two exponentiations, which translates to $2 \log p$ modular multiplications if we employ repeated squaring. The $O(\log p)$ modular multiplications each take $O((\log p)^2)$ operations for a total bit complexity of $O((\log p)^3)$.

Chapter 17

Review exercises

17.1 How does one compute the continued fraction expansion of a real number α?

Define the following sequences starting with $\alpha_0 = \alpha, p_0 = a_0$ and $q_0 = 1, p_1 = a_0 a_1 + 1$ and $q_1 = a_1$,

$a_i = \lfloor \alpha_i \rfloor$,

$\alpha_{i+1} = \frac{1}{\alpha_i - a_i}$,

$p_i = a_i p_{i-1} + p_{i-2}$ for $i \geq 2$,

$q_i = a_i q_{i-1} + q_{i-2}$ for $i \geq 2$.

The integers $a_0, a_1, a_2, \ldots$ are the continued fraction expansion of α and the fractions

$$\frac{p_i}{q_i}$$

are called the convergents.

17.2 In Wiener's attack on small private exponent RSA, what number does one compute the continued fraction of?

$\frac{e}{N}$.

17.3 What is a lattice?

A discrete version of a vector (sub)space.

17.4 What do you think is the most important property of an LLL reduced lattice basis, from the point of view of a cryptographer?

The shortest vector often appears as the first basis vector.

17.5 Suppose I have the equation $f(x_0) = 0 \pmod N$ where f is a polynomial of degree d. How small must x_0 be before Coppersmith's method will find it in polynomial time?

$|x_0| < N^{\frac{1}{d}}$.

17.6 Explain Håstad's attack on RSA.

Håstad shows how to break a system protected against the small-public-exponent attack on RSA by padding messages with user-specific data. With pad i for user i we encipher according to

$$c_i = (i \cdot 2^h + m)^e \pmod{N_i}.$$

For k users set

$$g_i(x) = (i \cdot 2^h + x)^e - c_i, 1 \leq i \leq k.$$

Then the goal is to recover m such that

$$g_i(m) = 0 \pmod{N_i}.$$

Assuming that $m < N_i$ for all i, we set $N = N_1 \cdots N_k$ and use the Chinese Remainder Theorem to compute

$$g(x) = \sum_{i=1}^{k} t_i g_i(x)$$

and

$$g(m) = 0 \pmod N.$$

We can then use Coppersmith's theorem, since g has degree e and is monic, to recover m in polynomial time as long as we have as many cipkertexts as the encryption exponent is large ($k > e$).

Standard exercises

17.7 Show that for an LLL reduced basis of an n-dimensional lattice L, if x is a nonzero vector in L then

$$||b_1|| \leq 2^{(n-1)/2}||x||.$$

We want to show that $||b_1||^2 \leq 2^{n-1}||x||^2$. From the second LLL condition we get

$$||b_i^*||^2 \geq (\frac{3}{4} - \underbrace{\mu_{i,i-1}^2}_{\leq \frac{1}{4}})||b_{i-1}^*||^2 \geq \frac{1}{2}||b_{i-1}^*||^2,$$

so that $||b_{i-1}^*||^2 \leq 2||b_i^*||^2$. Since $b_1 = b_1^*$, we therefore have

$$||b_1||^2 \leq 2^{j-1}||b_j^*||^2, 1 \leq j \leq n.$$

If $x = \sum_{j=1}^{n} x_j^* b_j^*$, let k be the smallest (or any) index for which $x_k^* \neq 0$. Then

$$||x|| \geq ||b_k^*||$$

so that

$$2^{n-1}||x||^2 \geq 2^{n-1}||b_k^*||^2 \geq ||b_1||^2$$

as desired.

Chapter 18

Review exercises

18.1 What is meant by semantic security, and how does this relate to perfect security?

A scheme has *perfect security*, or information theoretical security, when an adversary with infinite computing power can learn nothing about the plaintext given the ciphertext.
Semantic Security is like perfect security, except that we are dealing with a computationally bounded adversary.

18.2 Give the definition of polynomial security.

A system is said to have polymonial security (or has indistinguishable encryptions) if no adversary can win the following game with probability greater than $\frac{1}{2}$. The adversary A is given a public key encryption function f_y corresponding to some public key y. The adversary now runs in two stages:

- **Find:** The adversary produces two plaintexts m_0 and m_1 of equal lengths.
- **Guess:** The adversary is given the ciphertext c_b of one of the plaintexts $m_b, b \in \{0, 1\}$. The goal of A is to guess the value of b.

18.3 What restriction is put on the decryption oracle in an adaptive chosen ciphertext attack?

The oracle will not decrypt the target ciphertext.

18.4 Give two problems with the RSA algorithm as described in earlier chapters.

RSA is neither polynomially nor CCA2 secure.

Chapter 19

Review exercises

19.1 Describe the complexity classes $\mathcal{P}$, $\mathcal{NP}$, co–$\mathcal{NP}$.

A decision problem $\mathcal{DP}$ is in complexity class $\mathcal{P}$ if there is an algorithm that takes any instance I for which the answer is *yes* and delivers that answer in polynomial time. If the answer is *no*, then the algorithm is not required to terminate in polynomial time (or at all), but if it answers it should do so correctly.

A decision problem $\mathcal{DP}$ is in complexity class $\mathcal{NP}$ if for every instance I for which the answer is *yes* there is a witness which can be verified in polynomial time. If the answer is *no*, then the algorithm is not required to terminate, but if it does answer it should do so correctly.

co–$\mathcal{NP}$ is defined by reversing the roles of *yes* and *no* in the definition of $\mathcal{NP}$.

19.2 What is a super-increasing knapsack?

A knapsack problem with a restriction on the sizes of the elements. Every element must be larger than the sum of all smaller elements.

19.3 How does the Merkle-Hellman cryptosystem translate a super-increasing knapsack into a hard knapsack problem?

By randomly selecting two integers N and M and transforming the set members according to

$$x \longmapsto x \cdot N \pmod{M}.$$

19.4 What is meant by a hard predicate for a function and how is this concept related to the concept of semantic security?

Let $f : S \to T$ denote a one-way function where S and T are finite sets, and let $B : S \to \{0, 1\}$ denote a binary functions (a predicate). A *hard predicate* B for f is a function for which $B(x)$ is easy to compute given $x \in S$ and for which it is hard to compute $B(x)$ given only $f(x)$.

19.5 Explain the difference between a Monte-Carlo, an Atlantic City and a Las Vegas algorithm.

- **Monte-Carlo Algorithm:**
  Always outputs *false* if the answer is *false*.
  Answers *true* with probability $\geq \frac{1}{2}$.
  Otherwise answers *false* (sometimes incorrectly).
- **Atlantic City Algorithm:**
  Outputs *true* with probability $\geq \frac{3}{4}$ of being correct.
  Outputs *false* with probability $\geq \frac{3}{4}$ of being correct.
- **Las Vegas Algorithm:**
  Will terminate with the correct answer with probability $\geq \frac{1}{2}$.
  Otherwise will not terminate.

Gambling styles in Monte Carlo and Las Vegas differ. When you go to Monte Carlo you stay for a fixed amount of time and leave with whatever you have left. In Las Vegas you stay until you are broke, so the result is known, but the time you will spend is not.

Standard exercises

19.6 Show that in our definition of $\mathcal{BPP}$ one can replace the constant $\frac{2}{3}$ by any proportion in the range $(\frac{1}{2}, 1)$.

With a probability in the given range, running the test several times will guarantee any desired accuracy < 1.

19.7 Show that if one-way functions exist then $\mathcal{P} \neq \mathcal{NP}$.

If a one-way function f exists, then x is a witness of $f(x)$ and one cannot compute x from $f(x)$ in polynomial time. In other words, $\mathcal{P} \neq \mathcal{NP}$.

Chapter 20

Review exercises

20.1 What is the random oracle model and why does a proof in the random oracle model not apply to the real world.

The random oracle model replaces the hash function with a mathematical model of a perfect idealized hash function. All proofs in the random oracle model automatically fail for real life applications as the conditions are changed once we replace the idealized hash function with an actual one that we cannot prove to be (and isn't) idealized.

20.2 What is the forking lemma?

This is a proof technique used in the random oracle model. Assuming that all possible outputs of the random oracle in algorithm A are stored, we define algorithm B_A by running algorithm A with an oracle producing the same output except for one randomly chosen entry. The differences in output for these entries are examined and used to break the underlying problem.

20.3 How does RSA-OAEP work?

RSA-OAEP is a RSA with a padding scheme. With the pre-requisites

f is a k-bit to k-bit one-way trapdoor function,

k_0 and k_1 are numbers large enough for a computational effort of 2^{k_0} and 2^{k_1} to be infeasible,

$n = k - k_0 - k_1$,

$G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}$ is a hash function,

$H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}$ is a hash function,

R is a random bit string of length k_0,

m is an n-bit message,

encryption is defined according to

$$f(\{m||0^{k_1} \oplus G(R)\}||\{R \oplus H(m0^{k_1} \oplus G(R))\}).$$

Chapters 24-25

Review exercises

24.1 What is a commitment scheme?

A commitment scheme is a procedure that forces Alice to make and commit to a choice $c \in \{1, \ldots, n\}$. Alice is also required to be able to provide a proof of commitment $p(c)$ for her choice without revealing c. At some point in time when Alice discloses her choice, Bob must be able to verify Alice's prior commitment (by calculating $p(c)$ from c).

24.2 What does it mean for a committment scheme to be binding or concealing?

A commitment scheme is said to be information theoretically (resp. comutaionally) *binding* if no infinitely powerful (resp. computationally bounded) adversary can win the following game.

- The adversary outputs a value c, plus values x and r which produce this commitment.
- The adversary must then output a value $x' \neq x$ and a value r' such that

$$C(x, r) = C(x', r').$$

A commitment scheme is said to be information theoretically (resp. comutaionally) *concealing* if no infinitely powerful (resp. computationally bounded) adversary can win the following game.

- The adversary outputs two messages x_0 and x_1 of equal length.
- The challenger generates r at random and a random bit $b \in \{0, 1\}$.
- The challenger computes $c = C(x_b, r)$ and passses c to the adversary.
- The adversarial goal is to now guess the bit b.

25.3 How can a zero-knowledge protocol for proof of knowledge be used as an identification scheme?

By proving posession of a secret entity that is linked to identity.

25.4 In the protocol for graph isomorphism, show that if Peggy produces the same graph H in two different runs of the protocol, then Victor can determine the secret graph isomorphism.

If Victor asks for $\phi \cdot \psi$ one time and ψ the next, he can simply calculate ϕ.

25.5 How do you turn an interactive zero-knowledge identification scheme into a digital signature scheme?

See Section 3.1 in Smart[1], where this is accomplished by hashing the commitment and a message to produce the challenge.

Standard exercises

24.6 Given a group G and four elements g_1, g_2, h_1, h_2, suppose Peggy knows that

$$\log_{g_1}(h_1) = \log_{g_2}(h_2) = x$$

and that Peggy knows x. Give a protocol for Peggy to convince Victor that not only she knows the discrete logarithm but that the two discrete logarithms are equal.

1. Peggy chooses a random number $r \in \{1, \ldots, n-1\}$ and sends g_1^r and g_2^r to Victor.
2. Victor responds with a random number $s \in \{1, \ldots, n-1\}$.
3. Peggy computes $q = r + sx \pmod{n}$ and returns the result to Victor.
4. Victor accepts Peggy's assertion of equality if and only if $g_1^q = g_1^r h_1^s$ and $g_2^q = g_2^r h_2^s$.

25.7 Show that the zero-knowledge proof of validity for the committment $B_a(x)$ given in the text for when $x \in \{-1, 1\}$ satisfies the three requirements.

Completeness follows from checking that a valid response from Peggy will result in the protocol run giving Victor a positive response.

Soundness follows from algebraically checking that if Peggy tries to cheat then she needs to solve a hard problem or be really lucky.

A simulation argument shows zero-knowledge. Suppose c was chosen first, rather than after Peggy's commitment to α_1 and α_2. Then the simulation would compute, in order, assuming B is the item being proved correct,

$$\begin{aligned} d_1 &= \text{random number} \\ d_2 &= c - d_1 \\ r_1 &= \text{random number} \\ r_2 &= \text{random number} \\ \alpha_1 &= g^{r_1}(B \cdot h)^{-d_1} \\ \alpha_2 &= g^{r_2}(B \cdot h^{-1})^{-d_2} \end{aligned}$$

Bibliography

[1] N. Smart.
Cryptography: An Introduction.
3rd Edition, 2009.
http://www.cs.bris.ac.uk/~nigel/Crypto_Book.

[2] A. J. Menezes, Paul. C. van Oorschot and Scott. A. Vanstone.
Handbook of Applied Cryptography.
Fourth printing, CRC Press, 1997.
http://www.cacr.math.uwaterloo.ca/hac.

[3] Wikipedia.
The Carmichael function.
February 3rd, 2010.
http://en.wikipedia.org/wiki/Carmichael_function