

3.9.3 The Rabin problem — individual bits

Let $n = pq$, where p and q are distinct primes each congruent to 3 modulo 4.

3.88 Fact Define the predicate $B : Q_n \rightarrow \{0, 1\}$ by $B(x) = x \bmod 2$; that is, $B(x)$ is the least significant bit of the quadratic residue x . Then B is a hard predicate for the Rabin function (see page 115).

3.89 Fact Let $k = O(\lg \lg n)$ be an integer. Define the k -bit predicate $B^{(k)} : Q_n \rightarrow \{0, 1\}^k$ by $B^{(k)}(x) = x \bmod 2^k$. That is, $B^{(k)}(x)$ consists of the k least significant bits of the quadratic residue x . Then $B^{(k)}$ is a hard k -bit predicate for the Rabin function.

Thus the Rabin function has $\lg \lg n$ simultaneously secure bits.

3.10 The subset sum problem

The difficulty of the subset sum problem was the basis for the (presumed) security of the first public-key encryption scheme, called the Merkle-Hellman knapsack scheme (§8.6.1).

3.90 Definition The *subset sum problem* (SUBSET-SUM) is the following: given a set $\{a_1, a_2, \dots, a_n\}$ of positive integers, called a *knapsack set*, and a positive integer s , determine whether or not there is a subset of the a_j that sum to s . Equivalently, determine whether or not there exist $x_i \in \{0, 1\}$, $1 \leq i \leq n$, such that $\sum_{i=1}^n a_i x_i = s$.

The subset sum problem above is stated as a decision problem. It can be shown that the problem is computationally equivalent to its computational version which is to actually determine the x_i such that $\sum_{i=1}^n a_i x_i = s$, provided that such x_i exist. Fact 3.91 provides evidence of the intractability of the subset sum problem.

3.91 Fact The subset sum problem is **NP**-complete. The computational version of the subset sum problem is **NP**-hard (see Example 2.74).

Algorithms 3.92 and 3.94 give two methods for solving the computational version of the subset sum problem; both are exponential-time algorithms. Algorithm 3.94 is the fastest method known for the general subset sum problem.

3.92 Algorithm Naive algorithm for subset sum problem

INPUT: a set of positive integers $\{a_1, a_2, \dots, a_n\}$ and a positive integer s .

OUTPUT: $x_i \in \{0, 1\}$, $1 \leq i \leq n$, such that $\sum_{i=1}^n a_i x_i = s$, provided such x_i exist.

1. For each possible vector $(x_1, x_2, \dots, x_n) \in (\mathbb{Z}_2)^n$ do the following:
 - 1.1 Compute $l = \sum_{i=1}^n a_i x_i$.
 - 1.2 If $l = s$ then return(a solution is (x_1, x_2, \dots, x_n)).
2. Return(no solution exists).

3.93 Fact Algorithm 3.92 takes $O(2^n)$ steps and, hence, is inefficient.

3.94 Algorithm Meet-in-the-middle algorithm for subset sum problem

INPUT: a set of positive integers $\{a_1, a_2, \dots, a_n\}$ and a positive integer s .

OUTPUT: $x_i \in \{0, 1\}$, $1 \leq i \leq n$, such that $\sum_{i=1}^n a_i x_i = s$, provided such x_i exist.

1. Set $t \leftarrow \lfloor n/2 \rfloor$.
2. Construct a table with entries $(\sum_{i=1}^t a_i x_i, (x_1, x_2, \dots, x_t))$ for $(x_1, x_2, \dots, x_t) \in (\mathbb{Z}_2)^t$. Sort this table by first component.
3. For each $(x_{t+1}, x_{t+2}, \dots, x_n) \in (\mathbb{Z}_2)^{n-t}$, do the following:
 - 3.1 Compute $l = s - \sum_{i=t+1}^n a_i x_i$ and check, using a binary search, whether l is the first component of some entry in the table.
 - 3.2 If $l = \sum_{i=1}^t a_i x_i$ then return(a solution is (x_1, x_2, \dots, x_n)).
4. Return(no solution exists).

3.95 Fact Algorithm 3.94 takes $O(n2^{n/2})$ steps and, hence, is inefficient.

3.10.1 The L^3 -lattice basis reduction algorithm

The L^3 -lattice basis reduction algorithm is a crucial component in many number-theoretic algorithms. It is useful for solving certain subset sum problems, and has been used for cryptanalyzing public-key encryption schemes which are based on the subset sum problem.

3.96 Definition Let $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ be two vectors in \mathbb{R}^n . The *inner product* of x and y is the real number

$$\langle x, y \rangle = x_1 y_1 + x_2 y_2 + \dots + x_n y_n.$$

3.97 Definition Let $y = (y_1, y_2, \dots, y_n)$ be a vector in \mathbb{R}^n . The *length* of y is the real number

$$\|y\| = \sqrt{\langle y, y \rangle} = \sqrt{y_1^2 + y_2^2 + \dots + y_n^2}.$$

3.98 Definition Let $B = \{b_1, b_2, \dots, b_m\}$ be a set of linearly independent vectors in \mathbb{R}^n (so that $m \leq n$). The set L of all integer linear combinations of b_1, b_2, \dots, b_m is called a *lattice* of dimension m ; that is, $L = \mathbb{Z}b_1 + \mathbb{Z}b_2 + \dots + \mathbb{Z}b_m$. The set B is called a *basis* for the lattice L .

A lattice can have many different bases. A basis consisting of vectors of relatively small lengths is called *reduced*. The following definition provides a useful notion of a reduced basis, and is based on the Gram-Schmidt orthogonalization process.

3.99 Definition Let $B = \{b_1, b_2, \dots, b_n\}$ be a basis for a lattice $L \subset \mathbb{R}^n$. Define the vectors b_i^* ($1 \leq i \leq n$) and the real numbers $\mu_{i,j}$ ($1 \leq j < i \leq n$) inductively by

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}, \quad 1 \leq j < i \leq n, \quad (3.8)$$

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \quad 1 \leq i \leq n. \quad (3.9)$$

The basis B is said to be *reduced* (more precisely, *Lovász-reduced*) if

$$|\mu_{i,j}| \leq \frac{1}{2}, \quad \text{for } 1 \leq j < i \leq n$$

(where $|\mu_{i,j}|$ denotes the absolute value of $\mu_{i,j}$), and

$$\|b_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \|b_{i-1}^*\|^2, \quad \text{for } 1 < i \leq n. \quad (3.10)$$

Fact 3.100 explains the sense in which the vectors in a reduced basis are relatively short.

3.100 Fact Let $L \subset \mathbb{R}^n$ be a lattice with a reduced basis $\{b_1, b_2, \dots, b_n\}$.

- (i) For every non-zero $x \in L$, $\|b_1\| \leq 2^{(n-1)/2} \|x\|$.
- (ii) More generally, for any set $\{a_1, a_2, \dots, a_t\}$ of linearly independent vectors in L ,

$$\|b_j\| \leq 2^{(n-1)/2} \max(\|a_1\|, \|a_2\|, \dots, \|a_t\|), \quad \text{for } 1 \leq j \leq t.$$

The L^3 -lattice basis reduction algorithm (Algorithm 3.101) is a polynomial-time algorithm (Fact 3.103) for finding a reduced basis, given a basis for a lattice.

3.101 Algorithm L^3 -lattice basis reduction algorithm

INPUT: a basis (b_1, b_2, \dots, b_n) for a lattice L in \mathbb{R}^m , $m \geq n$.

OUTPUT: a reduced basis for L .

1. $b_1^* \leftarrow b_1$, $B_1 \leftarrow \langle b_1^*, b_1^* \rangle$.
2. For i from 2 to n do the following:
 - 2.1 $b_i^* \leftarrow b_i$.
 - 2.2 For j from 1 to $i-1$, set $\mu_{i,j} \leftarrow \langle b_i, b_j^* \rangle / B_j$ and $b_i^* \leftarrow b_i^* - \mu_{i,j} b_j^*$.
 - 2.3 $B_i \leftarrow \langle b_i^*, b_i^* \rangle$.
3. $k \leftarrow 2$.
4. Execute subroutine RED($k, k-1$) to possibly update some $\mu_{i,j}$.
5. If $B_k < (\frac{3}{4} - \mu_{k,k-1}^2) B_{k-1}$ then do the following:
 - 5.1 Set $\mu \leftarrow \mu_{k,k-1}$, $B \leftarrow B_k + \mu^2 B_{k-1}$, $\mu_{k,k-1} \leftarrow \mu B_{k-1} / B$, $B_k \leftarrow B_{k-1} B_k / B$, and $B_{k-1} \leftarrow B$.
 - 5.2 Exchange b_k and b_{k-1} .
 - 5.3 If $k > 2$ then exchange $\mu_{k,j}$ and $\mu_{k-1,j}$ for $j = 1, 2, \dots, k-2$.
 - 5.4 For $i = k+1, k+2, \dots, n$:

Set $t \leftarrow \mu_{i,k}$, $\mu_{i,k} \leftarrow \mu_{i,k-1} - \mu t$, and $\mu_{i,k-1} \leftarrow t + \mu_{k,k-1} \mu_{i,k}$.
 - 5.5 $k \leftarrow \max(2, k-1)$.
 - 5.6 Go to step 4.
- Otherwise, for $l = k-2, k-3, \dots, 1$, execute RED(k, l), and finally set $k \leftarrow k+1$.
6. If $k \leq n$ then go to step 4. Otherwise, return (b_1, b_2, \dots, b_n) .

RED(k, l) If $|\mu_{k,l}| > \frac{1}{2}$ then do the following:

1. $r \leftarrow \lfloor 0.5 + \mu_{k,l} \rfloor$, $b_k \leftarrow b_k - r b_l$.
 2. For j from 1 to $l-1$, set $\mu_{k,j} \leftarrow \mu_{k,j} - r \mu_{l,j}$.
 3. $\mu_{k,l} \leftarrow \mu_{k,l} - r$.
-

3.102 Note (explanation of selected steps of Algorithm 3.101)

- (i) Steps 1 and 2 initialize the algorithm by computing b_i^* ($1 \leq i \leq n$) and $\mu_{i,j}$ ($1 \leq j < i \leq n$) as defined in equations (3.9) and (3.8), and also $B_i = \langle b_i^*, b_i^* \rangle$ ($1 \leq i \leq n$).
- (ii) k is a variable such that the vectors b_1, b_2, \dots, b_{k-1} are reduced (initially $k = 2$ in step 3). The algorithm then attempts to modify b_k , so that b_1, b_2, \dots, b_k are reduced.

- (iii) In step 4, the vector b_k is modified appropriately so that $|\mu_{k,k-1}| \leq \frac{1}{2}$, and the $\mu_{k,j}$ are updated for $1 \leq j < k-1$.
- (iv) In step 5, if the condition of equation (3.10) is violated for $i = k$, then vectors b_k and b_{k-1} are exchanged and their corresponding parameters are updated. Also, k is decremented by 1 since then it is only guaranteed that b_1, b_2, \dots, b_{k-2} are reduced. Otherwise, b_k is modified appropriately so that $|\mu_{k,j}| \leq \frac{1}{2}$ for $j = 1, 2, \dots, k-2$, while keeping (3.10) satisfied. k is then incremented because now b_1, b_2, \dots, b_k are reduced.

It can be proven that the L^3 -algorithm terminates after a finite number of iterations. Note that if L is an integer lattice, i.e. $L \subset \mathbb{Z}^n$, then the L^3 -algorithm only operates on rational numbers. The precise running time is given next.

3.103 Fact Let $L \subset \mathbb{Z}^n$ be a lattice with basis $\{b_1, b_2, \dots, b_n\}$, and let $C \in \mathbb{R}$, $C \geq 2$, be such that $\|b_i\|^2 \leq C$ for $i = 1, 2, \dots, n$. Then the number of arithmetic operations needed by Algorithm 3.101 is $O(n^4 \log C)$, on integers of size $O(n \log C)$ bits.

3.10.2 Solving subset sum problems of low density

The density of a knapsack set, as defined below, provides a measure of the size of the knapsack elements.

3.104 Definition Let $S = \{a_1, a_2, \dots, a_n\}$ be a knapsack set. The *density* of S is defined to be

$$d = \frac{n}{\max\{\lg a_i \mid 1 \leq i \leq n\}}.$$

Algorithm 3.105 reduces the subset sum problem to one of finding a particular short vector in a lattice. By Fact 3.100, the reduced basis produced by the L^3 -algorithm includes a vector of length which is guaranteed to be within a factor of $2^{(n-1)/2}$ of the shortest non-zero vector of the lattice. In practice, however, the L^3 -algorithm usually finds a vector which is much shorter than what is guaranteed by Fact 3.100. Hence, the L^3 -algorithm can be expected to find the short vector which yields a solution to the subset sum problem, provided that this vector is shorter than most of the non-zero vectors in the lattice.

3.105 Algorithm Solving subset sum problems using L^3 -algorithm

INPUT: a set of positive integers $\{a_1, a_2, \dots, a_n\}$ and an integer s .

OUTPUT: $x_i \in \{0, 1\}$, $1 \leq i \leq n$, such that $\sum_{i=1}^n a_i x_i = s$, provided such x_i exist.

1. Let $m = \lceil \frac{1}{2} \sqrt{n} \rceil$.
2. Form an $(n+1)$ -dimensional lattice L with basis consisting of the rows of the matrix

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & ma_1 \\ 0 & 1 & 0 & \cdots & 0 & ma_2 \\ 0 & 0 & 1 & \cdots & 0 & ma_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & ma_n \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & ms \end{pmatrix}$$

3. Find a reduced basis B of L (use Algorithm 3.101).
4. For each vector $y = (y_1, y_2, \dots, y_{n+1})$ in B , do the following:

4.1 If $y_{n+1} = 0$ and $y_i \in \{-\frac{1}{2}, \frac{1}{2}\}$ for all $i = 1, 2, \dots, n$, then do the following:

For $i = 1, 2, \dots, n$, set $x_i \leftarrow y_i + \frac{1}{2}$.

If $\sum_{i=1}^n a_i x_i = s$, then return(a solution is (x_1, x_2, \dots, x_n)).

For $i = 1, 2, \dots, n$, set $x_i \leftarrow -y_i + \frac{1}{2}$.

If $\sum_{i=1}^n a_i x_i = s$, then return(a solution is (x_1, x_2, \dots, x_n)).

5. Return(FAILURE). (Either no solution exists, or the algorithm has failed to find one.)

Justification. Let the rows of the matrix A be b_1, b_2, \dots, b_{n+1} , and let L be the $(n+1)$ -dimensional lattice generated by these vectors. If (x_1, x_2, \dots, x_n) is a solution to the subset sum problem, the vector $y = \sum_{i=1}^n x_i b_i - b_{n+1}$ is in L . Note that $y_i \in \{-\frac{1}{2}, \frac{1}{2}\}$ for $i = 1, 2, \dots, n$ and $y_{n+1} = 0$. Since $\|y\| = \sqrt{y_1^2 + y_2^2 + \dots + y_{n+1}^2}$ the vector y is a vector of short length in L . If the density of the knapsack set is small, i.e. the a_i are large, then most vectors in L will have relatively large lengths, and hence y may be the unique shortest non-zero vector in L . If this is indeed the case, then there is good possibility of the L^3 -algorithm finding a basis which includes this vector.

Algorithm 3.105 is not guaranteed to succeed. Assuming that the L^3 -algorithm always produces a basis which includes the shortest non-zero lattice vector, Algorithm 3.105 succeeds with high probability if the density of the knapsack set is less than 0.9408.

3.10.3 Simultaneous diophantine approximation

Simultaneous diophantine approximation is concerned with approximating a vector $(\frac{q_1}{q}, \frac{q_2}{q}, \dots, \frac{q_n}{q})$ of rational numbers (more generally, a vector $(\alpha_1, \alpha_2, \dots, \alpha_n)$ of real numbers) by a vector $(\frac{p_1}{p}, \frac{p_2}{p}, \dots, \frac{p_n}{p})$ of rational numbers with a smaller denominator p . Algorithms for finding simultaneous diophantine approximation have been used to break some knapsack public-key encryption schemes (§8.6).

3.106 Definition Let δ be a real number. The vector $(\frac{p_1}{p}, \frac{p_2}{p}, \dots, \frac{p_n}{p})$ of rational numbers is said to be a *simultaneous diophantine approximation of δ -quality* to the vector $(\frac{q_1}{q}, \frac{q_2}{q}, \dots, \frac{q_n}{q})$ of rational numbers if $p < q$ and

$$\left| p \frac{q_i}{q} - p_i \right| \leq q^{-\delta} \text{ for } i = 1, 2, \dots, n.$$

(The larger δ is, the better is the approximation.) Furthermore, it is an *unusually good simultaneous diophantine approximation* (UGSDA) if $\delta > \frac{1}{n}$.

Fact 3.107 shows that an UGSDA is indeed unusual.

3.107 Fact For $n \geq 2$, the set

$$S_n(q) = \left\{ \left(\frac{q_1}{q}, \frac{q_2}{q}, \dots, \frac{q_n}{q} \right) \mid 0 \leq q_i < q, \text{ gcd}(q_1, q_2, \dots, q_n, q) = 1 \right\}$$

has at least $\frac{1}{2}q^n$ members. Of these, at most $O(q^{n(1-\delta)+1})$ members have at least one δ -quality simultaneous diophantine approximation. Hence, for any fixed $\delta > \frac{1}{n}$, the fraction of members of $S_n(q)$ having at least one UGSDA approaches 0 as $q \rightarrow \infty$.

Algorithm 3.108 reduces the problem of finding a δ -quality simultaneous diophantine approximation, and hence also a UGSDA, to the problem of finding a short vector in a lattice. The latter problem can (usually) be solved using the L^3 -lattice basis reduction.

Handbook of Applied Cryptography by A. Menezes, P. van Oorschot and S. Vanstone.

3.108 Algorithm Finding a δ -quality simultaneous diophantine approximation

INPUT: a vector $w = (\frac{q_1}{q}, \frac{q_2}{q}, \dots, \frac{q_n}{q})$ of rational numbers, and a rational number $\delta > 0$.

OUTPUT: a δ -quality simultaneous diophantine approximation $(\frac{p_1}{p}, \frac{p_2}{p}, \dots, \frac{p_n}{p})$ of w .

1. Choose an integer $\lambda \approx q^\delta$.
2. Use Algorithm 3.101 to find a reduced basis B for the $(n+1)$ -dimensional lattice L which is generated by the rows of the matrix

$$A = \begin{pmatrix} \lambda q & 0 & 0 & \cdots & 0 & 0 \\ 0 & \lambda q & 0 & \cdots & 0 & 0 \\ 0 & 0 & \lambda q & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda q & 0 \\ -\lambda q_1 & -\lambda q_2 & -\lambda q_3 & \cdots & -\lambda q_n & 1 \end{pmatrix}$$

3. For each $v = (v_1, v_2, \dots, v_n, v_{n+1})$ in B such that $v_{n+1} \neq q$, do the following:
 - 3.1 $p \leftarrow v_{n+1}$.
 - 3.2 For i from 1 to n , set $p_i \leftarrow \frac{1}{q} \left(\frac{v_i}{\lambda} + pq_i \right)$.
 - 3.3 If $|p \frac{q_i}{q} - p_i| \leq q^{-\delta}$ for each i , $1 \leq i \leq n$, then return $(\frac{p_1}{p}, \frac{p_2}{p}, \dots, \frac{p_n}{p})$.
4. Return(FAILURE). (Either no δ -quality simultaneous diophantine approximation exists, or the algorithm has failed to find one.)

Justification. Let the rows of the matrix A be denoted by b_1, b_2, \dots, b_{n+1} . Suppose that $(\frac{q_1}{q}, \frac{q_2}{q}, \dots, \frac{q_n}{q})$ has a δ -quality approximation $(\frac{p_1}{p}, \frac{p_2}{p}, \dots, \frac{p_n}{p})$. Then the vector

$$\begin{aligned} x &= p_1 b_1 + p_2 b_2 + \cdots + p_n b_n + p b_{n+1} \\ &= (\lambda(p_1 q - p q_1), \lambda(p_2 q - p q_2), \dots, \lambda(p_n q - p q_n), p) \end{aligned}$$

is in L and has length less than approximately $(\sqrt{n+1})q$. Thus x is short compared to the original basis vectors, which are of length roughly $q^{1+\delta}$. Also, if $v = (v_1, v_2, \dots, v_{n+1})$ is a vector in L of length less than q , then the vector $(\frac{p_1}{p}, \frac{p_2}{p}, \dots, \frac{p_n}{p})$ defined in step 3 is a δ -quality approximation. Hence there is a good possibility that the L^3 -algorithm will produce a reduced basis which includes a vector v that corresponds to a δ -quality approximation.

3.11 Factoring polynomials over finite fields

The problem considered in this section is the following: given a polynomial $f(x) \in \mathbb{F}_q[x]$, with $q = p^m$, find its factorization $f(x) = f_1(x)^{e_1} f_2(x)^{e_2} \cdots f_t(x)^{e_t}$, where each $f_i(x)$ is an irreducible polynomial in $\mathbb{F}_q[x]$ and each $e_i \geq 1$. (e_i is called the *multiplicity* of the factor $f_i(x)$.) Several situations call for the factoring of polynomials over finite fields, such as index-calculus algorithms in $\mathbb{F}_{2^m}^*$ (Example 3.70) and Chor-Rivest public-key encryption (§8.6.2). This section presents an algorithm for square-free factorization, and Berlekamp's classical deterministic algorithm for factoring polynomials which is efficient if the underlying field is small. Efficient randomized algorithms are known for the case of large q ; references are provided on page 132.