Mathematical Introduction to Coding Theory and Cryptography

Yi Ouyang

School of Mathematical Sciences, University of Science and Technology of China

Email address: yiouyang@ustc.edu.cn

Contents

CHAPTER 1

Preliminaries

1. Theory of Integers

1.1. Size of an integer. Let $\mathbb{Z}$ be the ring of integers and $\mathbb{Z}_+$ the subset of positive integers. For $N \in \mathbb{Z}_+$, suppose the binary expansion of N is

$$N = a_0 + \cdots + a_{n-1} \cdot 2^{n-1} + 2^n,\ a_i = 0 \text{ or } 1.$$

Then $n = [\log_2 N] + 1$ is called the size or length of N, as it takes n bits to store the number N in a computer.

1.2. Division with remainder.

Theorem 1.1. *For $a, b \in \mathbb{Z}$ and $b \neq 0$, there exist unique integers q (called quotient) and r (called remainder) in $\mathbb{Z}$, such that*

$$a = bq + r,\ 0 \leq r < |b|.$$

Recall that an ideal I of a commutative ring R is a nonempty subset of R such that for any $a, b \in I$ and $r, s \in R$, $ra + sb \in I$. Certainly any subset $(a) = aR$ for $a \in R$ is an ideal of R, which is called a principal ideal of R. R is called a principal ideal domain (PID) if every ideal of R is principal

Theorem 1.2. *The ring of integers $\mathbb{Z}$ is a PID.*

Proof. Let I be any non-zero ideal of $\mathbb{Z}$, let $0 \neq a \in I$, then $\pm a \in I$ and hence $I \cap \mathbb{Z}_+ \neq \emptyset$. Let d be the smallest positive integer in I. Then on one hand, by definition $d\mathbb{Z} \subseteq I$. On the other hand, for any $b \in I$, write $b = dq + r$, $0 \leq r < d$. Then $r = b - dq \in I$. By the minimality of d, $r = 0$, $b \in d\mathbb{Z}$ and hence $I \subseteq d\mathbb{Z}$. □

Theorem 1.3 (Bezout). *For $a, b \in \mathbb{Z}$ not all zero, there exist $u, v \in \mathbb{Z}$ such that $ua + vb = \gcd(a, b)$. Moreover, $\gcd(a, b) = 1$ if and only if there exist u, $v \in \mathbb{Z}$ such that $ua + vb = 1$.*

Proof. Let I be the ideal generated by a and b. Then $I = d\mathbb{Z}$ for some $d > 0$ by Theorem 1.2. It suffices to show $d = \gcd(a, b)$. On one hand, d is a $\mathbb{Z}$-linear combination of a and b, hence a multiple of $\gcd(a, b)$ and $d \geq \gcd(a, b)$. On the other hand, Let $a = qd + r$, $0 \leq r < d$, then $r = a - qd \in I$ and by the minimality of d, one must have $r = 0$ and $d \mid a$. Similarly $d \mid b$. Hence d is a common divisor of a and b, and $d \leq \gcd(a, b)$. Thus $d = \gcd(a, b)$. □

Problem 1.1. Find all possible u's (and v's) such that the Bezout identity $ua + vb = \gcd(a, b)$ is satisfied.

1.3. Euclidean Algorithm. The Euclidean Algorithm is an algorithm finding the greatest common divisor of two integers. The algorithm was first described in *Euclid's Elements* 2300 years ago but is still effective and widely used in practice nowadays.

Algerithm 1.1 (Euclidean Algorithm). *Input: $a, b \in \mathbb{Z}$, not simultaneously zero.*

Output: $\gcd(a, b) \in \mathbb{Z}_+$, *$u$ and v in $\mathbb{Z}$ such that $ua + vb = \gcd(a, b)$.*

- *Let $q_0 = a$ and $r_0 = |b| > 0$. Perform the division algorithm: $q_0 = r_0 q_1 + r_1$, $0 \le r_1 < r_0$.*
- *If $r_1 = 0$, then $\gcd(a, b) = r_0$; if $r_1 > 0$, repeat the division algorithm: $r_0 = r_1 q_2 + r_2$.*
- *Continue until $r_{n-1} = r_n q_{n+1}$ (i.e., $r_{n+1} = 0$), then $r_n = \gcd(a, b)$.*
- *Moreover,*

$$r_n = r_{n-2} - r_{n-1} q_n = \cdots = ua + vb.$$

Example 1.1. Compute $\gcd(33, 111)$.

Problem 1.2. Decide the complexity of Euclidean Algorithm.

1.4. Congruent theory. For $n \in \mathbb{Z}_+$. The integer a is congruent to b modulo n, denoted by $a \equiv b \bmod n$, if n divides $a - b$, i.e., a and b have the same remainder if divided by n. The congruent relation is an equivalence relation:

(1) (reflexive) $a \equiv a \bmod n$ for any $a \in \mathbb{Z}$;
(2) (symmetric) If $a \equiv b \bmod n$, then $b \equiv a \bmod n$;
(3) (transitive) If $a \equiv b \bmod n$ and $b \equiv c \bmod n$, then $a \equiv c \bmod n$.

Moreover, if $a \equiv b \bmod n$ and $c \equiv d \bmod n$, then

$$a \pm c \equiv b \pm d \bmod n, \qquad ac \equiv bd \bmod n.$$

For $i \in \mathbb{Z}$, the congruent class $\bar{i}$ of i modulo n is the subset $i + n\mathbb{Z} = \{i + nk \mid k \in n\mathbb{Z}\}$ of $\mathbb{Z}$. Then

$$\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z} := \{\bar{0}, \bar{1}, \cdots, n \bar{-} 1\}$$

is the set of all congruent classes of $\mathbb{Z}$ modulo n.

Proposition 1.1. *The set $\mathbb{Z}/n\mathbb{Z}$ is a commutative ring under the following addition and multiplication:*

$$\bar{a} + \bar{b} = \overline{a + b}, \quad \bar{a} \cdot \bar{b} = \overline{ab}.$$

Moreover,

(0) The reduction map $\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, $i \mapsto \bar{i}$ is a surjective homomorphism of commutative rings.

(1) The multiplicative group $(\mathbb{Z}/n\mathbb{Z})^\times$, i.e. the set of invertible elements in $\mathbb{Z}/n\mathbb{Z}$, is the set $\{\bar{a} \mid 0 < a < n,\ \gcd(a, n) = 1\}$.

(2) As a cyclic group, the set of generators of $\mathbb{Z}/n\mathbb{Z}$ is $(\mathbb{Z}/n\mathbb{Z})^\times$. Consequently, the set of generators of a cyclic group of order n with a generator g is $\{g^a \mid 0 < a < n,\ \gcd(a,n) = 1\}$.

(3) $\mathbb{Z}/n\mathbb{Z}$ is a domain if and only if $n = p$ is a prime. When this is the case, $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ is a field of p elements.

Proof. We prove (1). On one hand, if $\gcd(a,n) = 1$, then there exist u and v such that $au + vn = 1$, hence $\bar{u}$ is the inverse of $\bar{a}$. On the other hand, if $\bar{a}$ is invertible in $\mathbb{Z}/n\mathbb{Z}$, let $\bar{b}$ be its inverse. Then $\overline{ab} = \bar{1}$ and hence $ab \equiv 1 \bmod n$. Then $ab = 1 + kn$ and $\gcd(a,n) = 1$. □

Remark 1.1. From now on, we drop $\bar{}$ and write $\mathbb{Z}/n\mathbb{Z} = \{0, 1, \cdots, n-1\}$, and call it the residue ring modulo n.

Let $\varphi(n)$ be the order of the group $(\mathbb{Z}/n\mathbb{Z})^\times$. The function $n \mapsto \varphi(n)$ is called Euler's totient function. By Proposition 1.1(2), $\varphi(n)$ is the number of generators of a cyclic group of order n. One also has

Corollary 1.1. *The following identities hold:*

$$n = \sum_{d|n} \varphi(d), \tag{1.1}$$

$$\varphi(n) = \sum_{d|n} \mu(d) \frac{n}{d} \tag{1.2}$$

where $\mu(d)$ is the Möbius function: $\mu(d) = (-1)^s$ if d is squarefree of s prime factors and 0 if d is not squarefree.

Proof. The set $X = \{0, 1, \cdots, n-1\}$ is the disjoint union of $X_d = \{a \in X \mid \gcd(a,n) = d\}$ for $1 \le d \mid n$. But $X_d = \{cd \mid 0 \le c < \frac{n}{d},\ \gcd(c, \frac{n}{d}) = 1\}$ is of order $\varphi(n/d)$, hence $n = \sum_{d|n} \varphi(n/d) = \sum_{d|n} \varphi(d)$. The second formula follows from Möbius inversion formula. □

Suppose $m \mid n$. Then the natural reduction map $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$, $a \mapsto a$ is again a surjective homomorphism.

Theorem 1.4 (Chinese Remainder Theorem). *If $\gcd(m,n) = 1$, then the reduction map induces an isomorphism of rings:*

$$\mathbb{Z}/mn\mathbb{Z} \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}.$$

Equivalently, if n has a factorization

$$n = p_1^{\alpha_1} \cdots p_s^{\alpha_s},$$

then the reduction map induces

$$\mathbb{Z}/n\mathbb{Z} \cong \prod_{i=1}^{s} \mathbb{Z}/p_i^{\alpha_i}\mathbb{Z}.$$

Theorem 1.5 (Equivalent form of Chinese Remainder Theorem). *If m and n are coprime, then the congruent equations*

$$\begin{cases} x \equiv a \bmod m \\ x \equiv b \bmod n \end{cases}$$

is solvable for any integers a and b, and all solutions are in the same congruent class modulo mn.

Chinese Remainder Theorem has an immediate consequence:

Corollary 1.2. *(1) If $\gcd(m,n)=1$, then $\varphi(mn)=\varphi(m)\varphi(n)$.*

(2) If n has a factorization $n=p_1^{\alpha_1}\cdots p_s^{\alpha_s}$, then $\varphi(n)=\prod_{i=1}^{s}\varphi(p_i^{\alpha_i})$.

(3) If p is a prime and $\alpha\in\mathbb{Z}_+$, then $\varphi(p^\alpha)=p^{\alpha-1}(p-1)$.

Applying Lagrange's Theorem in group theory that the order of an element in a group is a factor of the group order, then one has the famous theorems of Euler and Fermat:

Theorem 1.6. *(Euler) If $a\in\mathbb{Z}$ and $\gcd(a,n)=1$, then $a^{\varphi(n)}\equiv 1 \bmod n$. Equivalently, $a^{\varphi(n)}=1$ if $a\in(\mathbb{Z}/n\mathbb{Z})^\times$.*

(Fermat's Little Theorem) In particular, if p is a prime and $a\in\mathbb{Z}$, then $a^p\equiv a \bmod p$. Equivalently $a^p=a$ if $a\in\mathbb{F}_p$.

1.5. Quotient ring and Chinese Remainder Theorem in general setting. Let R be a commutative ring and I be an ideal of R. For $a,b\in R$, a is congruent to b modulo I, denoted by $a\equiv b \bmod I$, if $a-b\in I$. This congruent relation is again an equivalence relation in R. For $a\in R$, let $\bar{a}$ be the congruent class of a modulo I. Let R/I be the set of congruent classes modulo I. Define

$$\bar{a}+\bar{b}:=\overline{a+b}, \qquad \bar{a}\cdot\bar{b}:=\overline{ab}.$$

Then R/I becomes a commutative ring, called the quotient ring of R modulo I, and the natural map

$$\pi: R\to R/I, \quad a\mapsto \bar{a}$$

is a surjective ring homomorphism.

Theorem 1.7 (Chinese Remainder Theorem). *Suppose $I_1,\cdots,I_n$ are ideals of R such that $I_i+I_j=R$ if $i\neq j$. Then one has*

$$R/I_1\cap\cdots\cap I_n\cong\prod_{i=1}^{n}R/I_i.$$

2. Polynomials over a field

In this section, we assume F is a field and a polynomial is of one variable over F.

2.1. Degree of a polynomial. The set $F[x]$ of polynomials of one variable over F is a commutative ring. For $0 \neq f(x) = a_0 + a_1 x + \cdots + a_n x^n \in F[x]$ and $a_n \neq 0$, set $\deg(f) := n$ and $\deg 0 := -\infty$. F is a natural subring of $F[x]$ by inclusion. A polynomial $f(x)$ is called non-constant if $f(x) \notin F$, i.e. $\deg(f) \geq 1$.

PROPOSITION 1.2. *The degree function $f \mapsto \deg(f)$ satisfies the following two properties:*

(1) $\deg(f+g) \leq \max\{\deg f, \deg g\}$*;*
(2) $\deg(fg) = \deg(f) + \deg(g)$*.*

In other words, let $\gamma > 1$ be a constant. Then the function $f \mapsto |f| := \gamma^{\deg f}$ is a metric on $F[x]$ satisfying

(1) $|f+g| \leq \max\{|f|, |g|\}$*;*
(2) $|fg| = |f| \cdot |g|$*.*

2.2. Division with remainder.

THEOREM 1.8. *Suppose $f(x)$, $g(x) \in F[x]$ and $f(x) \neq 0$. Then there exist unique polynomials $q(x)$ and $r(x)$ such that*

$$g(x) = q(x)f(x) + r(x), \ \deg(r) < \deg(f).$$

In particular, for $a \in F$ and $g(x) \in F[x]$, there exists a unique polynomial $q(x) \in F[x]$ such that

$$g(x) = q(x)(x-a) + g(a).$$

Consequently $x - a \mid g(x)$ if and only if a is a root of $g(x)$, i.e., $g(a) = 0$.

The following results follow from the above Theorem.

THEOREM 1.9. *The ring $F[x]$ is a PID.*

THEOREM 1.10 (Bezout). *For $f(x)$, $g(x) \in F[x]$, there exist polynomials $u(x)$ and $v(x)$, such that $uf + vg = \gcd(f,g)$. Moreover, $\gcd(f,g) = 1$ if and only if there exist polynomials $u(x)$ and $v(x)$, such that $uf + vg = 1$.*

We leave it to the readers to formulate the Euclidean algorithm for polynomials.

REMARK 1.2. The greatest common divisor is unique up to a multiple of some element in $F^\times$. However, if $F_1 \subseteq F_2$ are two fields, f and g are polynomials in $F_1[x]$, then their gcd in $F_1[x]$ and $F_2[x]$ are the same polynomial in $F_1[x]$ if we assume the gcd is monic.

THEOREM 1.11 (Lagrange). *A non-constant polynomial $f(x)$ over a field F has at most $\deg(f)$ roots (counting multiplicities) in F.*

PROOF. By induction to the degree of f. $\square$

2.3. Congruent theory. Suppose $m(x)$ is a non-constant polynomial of degree n. For $f(x) \in F[x]$, let $\overline{f(x)} := f(x) + m(x)F[x]$ be the congruent class of $f(x)$ modulo $m(x)$.

PROPOSITION 1.3. *The quotient ring $F[x]/(m(x))$ of $F[x]$ modulo the ideal $(m(x)) = m(x)F[x]$ is $\{\overline{a(x)} \mid \deg(a) < n\}$.*

(1) It is an F-vector space of dimension n with a basis $\{\bar{1}, \bar{x}, \cdots, \overline{x^{n-1}}\}$.

(2) Its group of units $(F[x]/(m(x)))^{\times} = \{\overline{a(x)} \mid \deg(a) < n,\ \gcd(a, m) = 1\}$.

(3) It is a domain if and only if $m(x)$ is irreducible. When this is the case, $F[x]/(m(x))$ is a field extension of F of degree n, isomorphic to $F(\alpha)$ where α is a root of $m(x)$ in an algebraic closure of F via the evaluation map at α: $a(x) \mapsto a(\alpha)$.

REMARK 1.3. For simplicity, we drop $^{-}$ and write $F[x]/(m(x)) = \{a(x) \mid \deg(a) < n\}$.

3. Finite fields

Suppose p is a prime, then

$$\mathbb{F}_p := \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$$

is a field of p elements, which is also called a prime field. Fermat's Little Theorem tells us that $a^p = a$ for any $a \in \mathbb{F}_p$.

3.1. Main theorem of finite fields. Now suppose F is a finite field of characteristic p, then F is an $\mathbb{F}_p$-vector space of finite dimension, and hence F must be of order $p^{\dim_{\mathbb{F}_p} F}$.

LEMMA 1.1. *If F is a field of characteristic $p > 0$, then $f(x^p) = f(x)^p$ for any $f(x) \in F[x]$.*

PROOF. This is because the binomial coefficient $\binom{p}{i}$ is a multiple of p for $1 \le i \le p - 1$. □

THEOREM 1.12. *Let p be a prime and $\overline{\mathbb{F}}_p$ be a fixed algebraic closure of $\mathbb{F}_p$.*

(1) For each $r \in \mathbb{Z}_+$, there exists a unique field, denoted by $\mathbb{F}_{p^r}$, of order p^r inside $\overline{\mathbb{F}}_p$ given by $\mathbb{F}_{p^r} = \{x \in \overline{\mathbb{F}}_p,\ x^{p^r} = x\}$.

(2) The multiplicative group $\mathbb{F}_{p^r}^{\times} = \mathbb{F}_{p^r} - \{0\}$ is a cyclic group of order $p^r - 1$.

(3) Let $f(x) \in \mathbb{F}_p[x]$ be any irreducible polynomial of degree r over $\mathbb{F}_p$ and $\alpha \in \overline{\mathbb{F}}_p$ be a root of $f(x)$. Then

$$\mathbb{F}_{p^r} = \mathbb{F}_p(\alpha) = \{a_0 + a_1\alpha + \cdots + a_{r-1}\alpha^{r-1} \mid 0 \le a_i \le p - 1\}.$$

Now suppose q is a p-power.

(4) $\mathbb{F}_{q^{r_1}} \supseteq \mathbb{F}_{q^{r_2}}$ if and only if $r_2 \mid r_1$, i.e., q^{r_1} is a power of q^{r_2}.

(5) In $\mathbb{F}_q[x]$, one has the factorization

$$x^{q^r} - x = \prod_{d|r} \prod_{\substack{f(x) \text{ monic irred.} \\ \deg f = d}} f(x).$$

In particular, let $N_{q,d} = \#\{f(x) \in \mathbb{F}_q[x]$ monic irreducible of degree $d\}$, then

$$q^r = \sum_{d|r} N_{q,d}, \qquad N_{q,d} = \sum_{d|r} \mu(d) q^{r/d}.$$

(6) For any $r \in \mathbb{Z}_+$, $\mathbb{F}_{q^r}/\mathbb{F}_q$ is a cyclic Galois extension of degree r with the q-Frobenius

$$\sigma_q : x \mapsto x^q$$

a generator of $\mathrm{Gal}(\mathbb{F}_{q^r}/\mathbb{F}_q)$.

Proof. (1) Let $X_{p^r} = \{x \in \overline{\mathbb{F}}_p \mid x^{p^r} = x\}$ be the root set of $x^{p^r} - x$. By Lagrange's Theorem, X_{p^r} is of order p^r. If F is a subfield of order p^r in $\overline{\mathbb{F}}_p$, then the group $F^\times = F \backslash \{0\}$ is of order $p^r - 1$ and every $a \in F^\times$ must satisfy $a^{p^r - 1} = 1$ and hence $a \in X_{p^r}$. Certainly $0 \in X_{p^r}$ and thus $F \subseteq X_{p^r}$. They must be equal since both have p^r elements. Now it is easy to check that X_{p^r} is indeed a field.

(2) This is a special case of the following fact: any finite multiplicative group in a field must be cyclic, which is a consequence of Lagrange's Theorem and the fact that $n = \sum_{d|n} \varphi(d)$.

(3) This follows from the isomorphism $\mathbb{F}_p[x]/(f(x)) \to \mathbb{F}_p(\alpha)$, $a(x) \mapsto a(\alpha)$.

(4) On one hand if $r_2 \mid r_1$, certainly $\mathbb{F}_{q^{r_1}} = X_{q^{r_1}} \supseteq \mathbb{F}_{q^{r_2}} = X_{q^{r_2}}$. On the other hand, if $\mathbb{F}_{q^{r_1}} \supseteq \mathbb{F}_{q^{r_2}}$, then $\mathbb{F}_{q^{r_2}}^\times$ is a subgroup of $\mathbb{F}_{q^{r_1}}^\times$, hence $q^{r_2} - 1 \mid q^{r_1} - 1$. However, in general $\gcd(q^m - 1, q^n - 1) = q^{\gcd(m,n)} - 1$.

(5) For any monic irreducible $f(x)$ of degree $d \mid r$ over $\mathbb{F}_q$ with a root $\alpha \in \overline{\mathbb{F}}_p$, $\alpha \in \mathbb{F}_{q^r}$ and $x^{q^r} - x$ is a zero polynomial of α, hence $f(x) \mid x^{q^r} - x$. On the other hand, if $g(x)$ is an irreducible factor of $x^{q^r} - x$ of degree d and β is a root, then $\beta \in X_{q^r} = \mathbb{F}_{q^r}$ and $\mathbb{F}_q(\beta) = \mathbb{F}_{q^d} \subseteq \mathbb{F}_{q^r}$, hence $d \mid r$. Since $x^{q^r} - x$ is separable (has no multiple roots), we get the identity.

(6) It is easy to check $\sigma_q \in \mathrm{Gal}(\mathbb{F}_{q^r}/\mathbb{F}_q)$. Let α be an generator of $\mathbb{F}_{q^r}^\times$. Then $\sigma_q^d(\alpha) \neq \alpha$ for $d < r$ and hence σ_q is of order $\geq r$. Since $\mathrm{Gal}(\mathbb{F}_{q^r}/\mathbb{F}_q)$ is of order $\leq r$, it must be of order r and have σ_q as its generator. □

Remark 1.4. Both $\mathbb{Z}_{p^r}$ and $\mathbb{F}_{p^r}$ are finite rings of order p^r, and $\mathbb{Z}_p = \mathbb{F}_p$. However, $\mathbb{F}_{p^r}$ is a field but $\mathbb{Z}_{p^r}$ is not even a domain if $r > 1$.

Proposition 1.4. *Suppose q is p-power and $f(x) \in \mathbb{F}_q[x]$ is monic irreducible of degree r. Let $\alpha \in \overline{\mathbb{F}}_p$ be a root of $f(x)$, then*

$$f(x) = (x - \alpha)(x - \alpha^q) \cdots (x - \alpha^{q^{r-1}}) = \prod_{i=0}^{r-1} (x - \sigma_q^i(\alpha)).$$

PROOF. It suffices to check the root set of $f(x)$ is $\{\alpha, \cdots, \alpha^{q^{r-1}}\}$. $\square$

DEFINITION 1.1. Suppose q is p-power. An element $\alpha \in \mathbb{F}_q$ is called a primitive element in $\mathbb{F}_q$ if α is a generator of the cyclic group $\mathbb{F}_q^\times$.

A monic irreducible polynomial $f(x) \in \mathbb{F}_q[x]$ of degree r is called a primitive polynomial over $\mathbb{F}_q$ if a (any) root of f is a primitive element in $\mathbb{F}_{q^r}$.

PROPOSITION 1.5. *There are $\varphi(q^r-1)/r$ primitive polynomials of degree r in $\mathbb{F}_q[x]$.*

PROOF. By Proposition 1.1(2), there are $\varphi(q^r-1)$ primitive elements in $\mathbb{F}_{q^r}$. Now every primitive polynomial of degree r has r distinct roots which are primitive elements in $\mathbb{F}_{q^r}$. $\square$

EXAMPLE 1.2. The polynomial x^2+x+1 is irreducible of degree 2 in $\mathbb{F}_2[x]$. Let α be a root of it. Then $\mathbb{F}_4 = \{0, 1, \alpha, 1+\alpha\}$. Since $\alpha(\alpha+1) = \alpha^2+\alpha = 1$, $\alpha^{-1} = 1+\alpha$. α is a generator of $\mathbb{F}_4^\times$, hence x^2+x+1 is primitive.

EXAMPLE 1.3. The polynomial x^3+x^2+2 is irreducible of degree 3 in $\mathbb{F}_3[x]$. Let α be a root of it. Then $\mathbb{F}_{27} = \{c_0 + c_1\alpha + c_2\alpha^2 \mid 0 \leq c_i \leq 2\}$. We can check $\alpha^{-1} = \alpha^2+1$ and $\alpha^{13} = 1$, hence x^3+x^2+2 is not a primitive polynomial.

Notations

If not stated otherwise, from now on, we shall use the following notations/conventions.

- p is a prime.
- $q > 1$ is a power of p and $\mathbb{F}_q$ is the finite field of order q.
- $\mathbb{F}_q^{m\times n}$ is the $\mathbb{F}_q$-vector space of all $m \times n$ matrices over $\mathbb{F}_q$.
- $\mathbb{F}_q^n$ is identified with the row vector space $\mathbb{F}_q^{1\times n}$.
- A row vector of dimension n is a $1 \times n$ matrix, a column vector of dimension n is an $n \times 1$ matrix.
- $0^n := (0, 0, \cdots, 0)$, $1^n := (1, 1, \cdots, 1) \in \mathbb{F}_q^n$. If there is no need to specify n, we also use 0 to represent the zero vector.
- e_i is the row vector of dimension n whose i-th entry is 1 and other entries are 0.
- I_n is the identity matrix of size n.
- For a vector $c = (c_1, c_2, \cdots, c_n)$, the cyclic matrix generated by c is the square matrix

$$P(c) = \begin{pmatrix} c_1 & c_2 & \cdots & c_{n-1} & c_n \\ c_2 & c_3 & \cdots & c_n & c_1 \\ & \iddots & \iddots & \iddots & \\ c_{n-1} & c_n & & c_{n-3} & c_{n-2} \\ c_n & c_1 & \cdots & c_{n-2} & c_{n-1} \end{pmatrix}_{n\times n}.$$

- For a matrix A, $\operatorname{rank}(A)$ and A^T are the rank and transpose of A.
- The row (resp. column) vector space of a matrix A is the vector space generated by all row (resp. column) vectors of A, both spaces are of dimension $\operatorname{rank}(A)$.

Part 1

Coding Theory

CHAPTER 2

Introduction to Coding Theory

1. Background

Transmitting messages through an unreliable and sometimes noisy channel often results in errors. Coding theory is the basic tool to detect and correct the errors to ensure safe and sound communication. It has many practical applications, used not only in network communication, USB channels and satellite communication, but also in hard disks, CDs and DVDs etc.

We give several examples in application.

Example 2.1 (Parity check code). For a (binary) message $x = x_1 \cdots x_n$ in bits string, define the parity check bit $p(x) = \sum_{i=1}^{s} x_i$. The code $C(x) = (x, p(x))$ can detect one single error because the error will change the sum of components of $C(x)$ from 0 to 1, but it cannot correct the error or detect two errors.

Example 2.2 (Repetition code). In a noisy channel, it is often wise to send a message repeatedly many times. Following this idea we can construct the repetition code. Let x be a message and let r be the number of errors that we wish to correct, define $C(x) = x \parallel x \parallel \cdots \parallel x$ where x is repeated $2r+1$ times. This code can correct r errors. In fact, for the $2r+1$ values of x, at most r of them are changed and thus at least $r+1$ values remain unchanged, so the original value of x must be the value with majority.

Example 2.3 (Resident Identity Card Number). The Identity Card Number of a Chinese resident consists of 18 digits. For a ID number $a_1a_2 \cdots a_{18}$, a_1a_2 is the code for province/municipality/autonomous region, a_3a_4 is the city/prefecture code, a_5a_6 is the county/district code, $a_7a_8a_9a_{10}$ is the birth year, $a_{11}a_{12}$ is the birth month, $a_{13}a_{14}$ is the birth date, $a_{15}a_{16}a_{17}$ is the code assigned by local police station with a_{17} being odd for male and even for female. The last digit a_{18} is the check digit, given by the formula

$$a_{18} = 1 - \sum_{i=1}^{17} 2^{18-i} a_i \bmod 11 \in \mathbb{F}_{11} = \{0, 1, \cdots, 10\}, \tag{2.1}$$

with the letter X representing 10.

For example, suppose someone has the resident Identity Card number $53010219200508011X$. By computing, $\sum_{i=1}^{17} 2^{18-i} a_i = 189 = 2$ in $\mathbb{F}_{11}$ and hence (2.1) holds, so this number is a legal identity card number. Based on this number, one knows that this person is a male born in May 8, 1920 and was living in Wuhua District, Kunming City, Yunnan Province when the card was issued.

Example 2.4 (International Standard Book Number). The International Standard Book Number (ISBN), administered by International ISBN Agency and National ISBN Agencies, is an identifier for public available books (text-based monographic publications) used by publishers, booksellers, libraries, internet retailers and other supply chain participants for ordering, listing, sales records and stock control purposes. The ISBN identifies the registrant as well as the specific title, edition and format.

The old version of ISBN, used until the end of December 2006, is also a code in the finite field $\mathbb{F}_{11}$. The ISBN of a book consists of 10 digits. The first nine digits are grouped in three sections: the registration group element of length up to 5 digits which is assigned by International ISBN Agency and identifies the particular country, geographical region, or language area; the registrant element of length up to 5 digits which is assigned by National ISBN Agencies and identifies the particular publisher or imprint; and the publication element which is assigned by the publisher and identifies the particular edition and format of a specific title. The last digit a_{10} is the check digit, calculated via the formula

$$a_{10} = \sum_{i=1}^{9} i a_i \in F_{11}. \tag{2.2}$$

Again here 10 is represented by the letter X.

Current version of ISBN is in use since January 1, 2007. It consists of 13 digits and is divided into 5 sections. The prefix element, either 978 or 979, represents the European Article Number. The next three sections of 9 digits are the same as in the old version: the registration group element is of length up to 5 digits, the registrant element is of length up to 7 digits and the publication element is of length up to 6 digits. The check digit, the final single digit validating the number, is the weighted sum of the previous 12 digits with alternate weights of 3 and 1 in a modulus 10 system, i.e.,

$$a_{13} = 3\sum_{i=1}^{6} a_{2i-1} + \sum_{i=1}^{6} a_{2i} \bmod 10. \tag{2.3}$$

For example, the author has a book whose ISBN is 978-7-04-052753-5. Here the prefix element is 978; the registration group element is 7, representing China; the registrant element is 04, representing Higher Education Press; the publication element is 052753. The check digit $5 = (9+8+0+0+2+5)\times 3 + (7+7+4+5+7+3)\times 1 \mod 10$, validating this number.

Problem 2.1 (Main problems of coding theory).

(1) Construct codes that can correct a maximal number of errors while using a minimal amount of redundancy.
(2) Construct codes with efficient encoding and decoding procedures.

2. Basic Definitions

We first give the definition of a code.

DEFINITION 2.1. Let $A = \{a_1, \cdots, a_q\}$ be an alphabet of size q. A block code C of length n over A is a nonempty subset of A^n. An element $c \in C$ is called a codeword. The number of elements in C, denoted $|C|$, is called the size of this code. A code of length n and size M is called an (n, M)-code over A (or an $(n, M)_q$-code if one wants to specify the size q of the alphabet).

REMARK 2.1. It seems that the definition of a code depends on the actual alphabet, in fact its size is more essential. Let A and A' be two alphabets of the sane size, let $\sigma : A \to A'$ be a bijection of sets. Then σ induces bijection of $C \subseteq A^n$ to $\sigma(C) \subseteq A'^n$. C and $\sigma(C)$ can be regarded as the same code. Hence there is no difference to choose whatever alphabet of the same size. However, some alphabet is easier to describe mathematically than others. For example, if q is a prime power, the alphabet of size q is often chosen to be the finite field $\mathbb{F}_q$.

If A is of size q, A^n is of order q^n, i.e. n-tuples of A can label q^n objects. Hence to label M objects, $([\log_q M] + 1)$-tuples are sufficient. From this the rate of a code is introduced to evaluate its efficiency and redundancy.

DEFINITION 2.2. Let C be an (n, M)-code over an alphabet of size q. The rate of C is defined to be

$$\mathrm{rate}(C) := \frac{\log_q M}{n}.$$

By definition, one certainly has

$$0 < \mathrm{rate}(C) \leq 1. \tag{2.4}$$

One can see that the closer rate(C) is to 0, the more redundant and the less efficient is the code.

EXAMPLE 2.5. The rate of the trivial code $C = A^n$ is 1. This code cannot correct any error.

EXAMPLE 2.6. Let C be the repetition code (of any set X of length n over an alphabet of size q) which can correct r errors. Then $0 < \mathrm{rate}(C) \leq \frac{n}{(2r+1)n} = \frac{1}{2r+1}$ and $\lim\limits_{r\to\infty} \mathrm{rate}(C) = 0$.

DEFINITION 2.3 (Hamming distance). For two elements $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_n$ in A^n, the Hamming distance of x and y is defined to be $d(x, y) = \#\{i \mid x_i \neq y_i\}$.

The following proposition is immediate.

PROPOSITION 2.1. *The function d is a metric, then for every $x, y, z \in A^n$, we have*

1. $0 \leq d(x, y) \leq n$.
2. $d(x, y) = 0$ *if and only if* $x = y$.
3. $d(x, y) = d(y, x)$.
4. *(triangle inequality)* $d(x, z) \leq d(x, y) + d(y, z)$.

DEFINITION 2.4 (Nearest neighbor decoding rule). Let C be a code of length n over an alphabet A. The nearest neighbor decoding rule states that every $x \in A^n$ is decoded to $c_x \in C$ which is closest to x, i.e., $D(x) = c_x$ where c_x is the unique codeword such that $d(x, c_x) = \min_{c \in C}\{d(x, c)\}$. If there exist at least two codewords with minimal distance to x, then $\perp$ is returned (and the decoding is failed).

DEFINITION 2.5. The distance of a code C, denoted by $d(C)$, is the minimal distance of all pairs of different codewords, i.e.,

$$d(C) := \min\{d(c_1, c_2) \mid c_1, c_2 \in C, c_1 \neq c_2\}.$$

An (n, M)-code of distance d is called an (n, M, d)-code, n, M and d are called its parameters.

DEFINITION 2.6. The relative distance of a code C of length n and distance d is $\delta(C) = \frac{d-1}{n}$.

EXAMPLE 2.7. (1) The trivial code A^n: $d = 1$ and $\delta = 0$.

(2) The repetition code $C = \{(a, \cdots, a) \mid a \in \mathbb{F}_q\} \subset \mathbb{F}_q^n$ is a (n, q, n)-code over $\mathbb{F}_q$. If $n \to +\infty$, $\delta(C) = \frac{n-1}{n} \to 1$ and $\text{rate}(C) = \frac{1}{n} \to 0$.

DEFINITION 2.7. Let C be a code of length n over an alphabet A.

1. C detects μ errors if for every codeword $c \in C$ and every $x \in A^n$, if $0 < d(c, x) \leq \mu$, then $x \notin C$.
2. C corrects ν errors if for every codeword $c \in C$ and every $x \in A^n$, if $d(x, y) \leq \nu$, then the nearest neighbor decoding of x outputs c.

THEOREM 2.1.

1. *A code C can detect μ errors if and only if $d(C) > \mu$.*
2. *A code C can correct ν errors if and only if $d(C) \geq 2\nu + 1$.*

PROOF. (1) The condition $d(C) > \mu$ is equivalent to that $d(c, c') > \mu$ for any two codewords $c \neq c' \in C$. This in turn is equivalent to that for any $c \in C$, any $x \in A^n$ satisfying $0 < d(c, x) \leq \mu$ must not be in C, i.e., that C can detect μ errors.

(2) Suppose $d(C) \geq 2\nu + 1$. Let $x \in A^n$ and $c \in C$ such that $d(x, c) \leq \nu$. Then for any $c' \in C \backslash \{c\}\}$, $d(x, c') \geq d(c, c') - d(x, c) \geq \nu + 1$ and hence c is the nearest neighbor of x.

Conversely, for $c \in C$, let $D(c, \nu) = \{x \in A^n \mid d(c, x) \leq \nu\}$. That C can correct ν errors means nothing but the sets $D(c, \mu)$ for $c \in C$ are pairwise disjoint. Assume $c \neq c'$ and $d(c, c') < 2\nu + 1$. Then the set

$I = \{1 \le i \le n \mid c_i \ne c'_i\}$ is of order $\le 2\nu$. We can divide it into two disjoint subsets I_1 and I_2, each of order $\le \nu$. Let $x \in A^n$ such that $x_i = c_i$ for $i \notin I$ or $i \in I_1$ and $x_i = c'_i$ for $i \in I_2$, then $x \in D(c,\mu) \cap D(c',\mu)$, impossible. Hence $d(c,c') \ge 2\nu + 1$ for $c \ne c'$. This means $d(C) \ge 2\nu + 1$. □

Definition 2.8 (Most likely decoding rule). Let C be a code of length n over an alphabet A. The most likely decoding rule states that every $x \in A^n$ is decoded to $c_x \in C$ satisfying

$$\Pr(x \text{ is received} \mid c_x \text{ was sent}) = \max_{c \in C} \Pr(x \text{ is received} \mid c \text{ was sent}).$$

If there exist more than one c with maximal probability, then $\perp$ is returned.

A code with the alphabet $A = \mathbb{F}_2 = \{0,1\}$ is called a binary code. A binary symmetric channel is a binary channel such that

$$\Pr(1 \text{ is received} \mid 0 \text{ was sent}) = \Pr(0 \text{ is received} \mid 1 \text{ was sent}) = p,$$

$$\Pr(1 \text{ is received} \mid 1 \text{ was sent}) = \Pr(0 \text{ is received} \mid 0 \text{ was sent}) = 1 - p.$$

The probability p is called the crossover probability of this binary symmetric channel.

Theorem 2.2. *In a binary symmetric channel with $p < \frac{1}{2}$, the most likely decoding rule is equivalent to the nearest neighbor decoding rule.*

Proof. Suppose C is a binary code of length n. Let x be the received word in $\mathbb{F}_2^n$. For every $c \in C$ and for every $0 \le i \le n$, if $d(x,c) = i$, then

$$\Pr(x \text{ is received} \mid c \text{ was sent}) = p^i(1-p)^{n-i}.$$

Since $p < \frac{1}{2}$, we have that $\frac{1-p}{p} > 1$ and

$$p^i(1-p)^{n-i} = p^{i+1}(1-p)^{n-i-1} \cdot \frac{1-p}{p} > p^{i+1}(1-p)^{n-i-1}.$$

Thus the sequence $p^i(1-p)^{n-i}$ $(0 \le i \le n)$ is a decreasing sequence. This implies that the codeword c closest to x is the one with maximal probability that this codeword was sent and x is received. □

CHAPTER 3

Linear Codes

1. Basic definitions

1.1. Linear code and its dual code.

Definition 3.1. A linear code of length n over $\mathbb{F}_q$ is a vector subspace of $\mathbb{F}_q^n$. The dimension of a linear code is its dimension as an $\mathbb{F}_q$-vector space.

A linear code is called an $[n,k,d]_q$-code or an $[n,k,d]$-code over $\mathbb{F}_q$ if it is of length n, dimension k (that is, of size q^k) and distance d over the alphabet $\mathbb{F}_q$.

For $x = (x_1, \cdots, x_n)$ and $y = (y_1, \cdots, y_n)$ in $\mathbb{F}_q^n$, the inner product $x \cdot y$ is

$$x \cdot y = (x, y) := xy^T = x_1y_1 + \cdots + x_ny_n.$$

The inner product is a non-degenerate symmetric bilinear form on $\mathbb{F}_q^n$ which induces an isomorphism of $\mathbb{F}_q^n$ to its dual via the map $x \mapsto (y \mapsto x \cdot y)$. Assume W is a subspace of $\mathbb{F}_q^n$, the orthogonal complement of W in $\mathbb{F}_q^n$ is the subspace given by

$$W^\perp := \{y \in \mathbb{F}_q^n \mid x \cdot y = 0 \text{ for all } x \in W\}.$$

Then $\mathbb{F}_q^n = W \oplus W^\perp$ and in particular

$$\dim W^\perp = \dim \mathbb{F}_q^n - \dim W = n - \dim W. \tag{3.1}$$

By the fact $(W^\perp)^\perp = W$, we can switch the role of W and $W^\perp$.

Definition 3.2. Let C be a linear code over $\mathbb{F}_q$ of length n. The orthogonal complement $C^\perp$ of C in $\mathbb{F}_q^n$ is called the dual code of C.

A linear code C is called self-orthogonal if $C \subseteq C^\perp$. A linear code C is called self-dual if $C = C^\perp$.

Proposition 3.1. *The dual code of $C^\perp$ is C, and*

(1) *if C is a $[n,k,d]$-code, then $\dim C^\perp = n - k$;*
(2) *if C is a self-orthogonal $[n,k,d]$-code, then $k \leq n/2$;*
(3) *if C is a self-dual $[n,k,d]$-code, then n must be even and $k = n/2$.*

Proof. Clear, just use the identity $\dim C + \dim C^\perp = n$. □

1.2. Hamming weight.

Definition 3.3. The Hamming weight of a vector x in $\mathbb{F}_q^n$ is

$$\mathrm{wt}(x) := \#\{i \mid x_i \neq 0\} = d(x, 0),$$

where d is the Hamming distance in $\mathbb{F}_q^n$.

Certainly for $n = 1$, $\mathrm{wt}(x) = 1 - \delta_{x,0}$ where δ is the Kronecker symbol (i.e. $\delta_{ij} = 0$ if $i \neq j$ and 1 if $i = j$). In general, one has

$$\mathrm{wt}(x) = \sum_{i=1}^{n} \mathrm{wt}(x_i).$$

Lemma 3.1. *On $\mathbb{F}_2^n$, set $x * y := (x_1y_1, \cdots, x_ny_n)$. Then*

$$\mathrm{wt}(x + y) = \mathrm{wt}(x) + \mathrm{wt}(y) - \mathrm{wt}(x * y).$$

Hence $\mathrm{wt}(x) + \mathrm{wt}(y) \geq \mathrm{wt}(x + y)$.

Proof. It suffices to check the $n = 1$ case. □

Lemma 3.2. *On $\mathbb{F}_q^n$ for general q, one has*

$$\mathrm{wt}(x) + \mathrm{wt}(y) \geq \mathrm{wt}(x + y) \geq \mathrm{wt}(x) - \mathrm{wt}(y).$$

Proof. Again it suffices to check the $n = 1$ case. □

Definition 3.4. The weight of a linear code C, denote by $\mathrm{wt}(C)$, is the minimal Hamming weight of nonzero codewords in C, i.e.

$$\mathrm{wt}(C) = \min_{0 \neq c \in C} \mathrm{wt}(c).$$

Theorem 3.1. *For a linear code C, one always has $d(C) = \mathrm{wt}(C)$.*

Proof. Apply the fact $d(c, c') = d(c-c', 0) = \mathrm{wt}(c-c')$ to the definitions of $d(C)$ and $\mathrm{wt}(C)$. □

1.3. Advantages of Linear Codes. There are some advantages for linear codes:

(1) Linear codes are vector spaces over finite fields, so tools and theories from linear algebra can be used to study linear codes. In particular, a linear code can be described by a basis.

(2) By the identity $d(C) = \mathrm{wt}(C)$, the distance of a linear code is the smallest Hamming weight of a nonzero codeword. This fact makes it is easier to find the distance of a linear code.

(3) (Linear) mapping is usually simple to describe.

1.4. Generator and parity check matrices.

Definition 3.5. Let C be a linear code over $\mathbb{F}_q$.

(1) A generator matrix G for C is a matrix whose rows form a basis of C as an $\mathbb{F}_q$-vector space;

(2) A parity check matrix H for C is a generator matrix for $C^{\perp}$.

Remark 3.1. If C is a linear $[n, k]$-code over $\mathbb{F}_q$, then a generator matrix G of C is a $k \times n$ matrix of full row rank and is a parity check matrix of $C^{\perp}$, a parity check matrix H of C is a $(n-k) \times n$ matrix of full row rank and is a generator matrix of $C^{\perp}$.

On the other hand, for a matrix A of full row rank, the row vector space W of A and its orthogonal complement $W^{\perp}$ are both linear codes, A is a generator matrix of W and a parity check matrix of $W^{\perp}$. Because of this, a matrix of full row rank is called a legal generator or parity check matrix.

We recall the following facts from linear algebra.

Lemma 3.3. *Suppose A and A' are matrices of the same size over a field F. Then A' can be obtained from A through finite steps of elementary row transformations if and only if $A' = MA$ for some invertible matrix M. In this case, the row vector spaces of A and A' are the same vector space.*

Lemma 3.4. *Let A be an $n \times m$ matrix over a field F. Then the solutions of the homogeneous linear equations $(x_1, \cdots, x_n)A = 0$ form a subspace of F^n of dimension $n - \operatorname{rank}(A)$.*

Theorem 3.2. *Suppose C is an $[n, k]$-code over $\mathbb{F}_q$, G is a generator matrix and H a parity check matrix of C. Then a matrix G' is a generator matrix of C if and only if $G' = MG$ for some invertible matrix M, a matrix H' is a parity check matrix of C if and only if $H' = NH$ for some invertible matrix N.*

Proof. This follows from Lemma 3.3. □

Theorem 3.3. *Let C be an $[n, k]$-code over $\mathbb{F}_q$, G a generator matrix and H a parity check matrix of C respectively.*

(1) *A vector $v \in \mathbb{F}_q^n$ is a codeword in C if and only if $vH^T = 0$. In other words, $C = \{x \in \mathbb{F}_q^n \mid xH^T = 0\}$.*
(2) *A vector $v \in \mathbb{F}_q^n$ is a codeword in $C^{\perp}$ if and only if $vG^T = 0$. In other words, $C^{\perp} = \{x \in \mathbb{F}_q^n \mid xG^T = 0\}$.*
(3) *$HG^T = GH^T = 0$.*
(4) *A $k \times n$ matrix A is a generator matrix of C if and only if $\operatorname{rank}(A) = k$ and $AH^T = 0$.*
(5) *An $(n-k) \times n$ matrix B is a parity check matrix of C if and only if $\operatorname{rank}(B) = n - k$ and $BG^T = 0$.*

Proof. (2) follows from (1) via the duality of C and $C^{\perp}$. (3) follows from (1) and (2). (5) follows from (4) by duality. It suffices to show (1) and (4).

(1) By definition $\dim C = \operatorname{rank}(G) = k$. Let $W = \{x \in \mathbb{F}_q^n \mid xH^T = 0\}$. By Lemma 3.4, $\dim(W) = n - \operatorname{rank}(H^T) = k$. C and W are of the same dimension. We just need to show $C \subseteq W$.

Let the set of row vectors of G be $\{\alpha_1, \cdots, \alpha_k\}$ and the set of row vectors of H be $\{\beta_1, \cdots, \beta_{n-k}\}$. For any i and for any $\beta \in C^\perp$, $\alpha_i \beta^T = 0$, thus

$$\alpha_i H^T = \alpha_i(\beta_1^T, \cdots, \beta_{n-k}^T) = 0, \ \textit{i.e.}\ \alpha_i \in W.$$

Hence C, as the space generated by α_i $(i = 1, \cdots, k)$, is contained in W.

(4) If A is a generator matrix of C, then $\operatorname{rank}(A) = k$ by definition and $AH^T = 0$ by (1). On the other hand, the row vector space U of A has dimension $= \operatorname{rank}(A) = k$ and the row vectors of A form a basis of U. Since $AH^T = 0$, by (1), C contains all row vectors of A. This means $C \supseteq U$. Then $C = U$ since both are of dimension k. In other words, A is a generator matrix for C. □

Definition 3.6. A generator matrix (resp. a parity check matrix) of the form $(I_k \mid X)$ (resp. $(Y \mid I_{n-k})$) is called a standard form.

Corollary 3.1. *If $G = (I_k, X)$ is a generator matrix in standard form for C, then $H = (-X^T, I_{n-k})$ is a parity check matrix in standard form for C, and vice versa.*

Proof. Just need to check $GH^T = 0$. □

Theorem 3.4. *Let C be a linear code with a parity check matrix H. Then*

1. *$d(C) \geq d$ if and only if every $d-1$ columns of H are linearly independent;*
2. *$d(C) \leq d-1$ if and only if there exist $d-1$ columns of H which are linearly dependent.*

Proof. First note that it suffices to show the $\Rightarrow$ parts of (1) and (2).

Let $H = (\beta_1^T, \cdots, \beta_n^T)$ with β_j^T the j-th column vector of H. Then for a vector $v = (\lambda_1, \cdots, \lambda_n) \in \mathbb{F}_q^n$, $vH^T = \sum_{i=1}^n \lambda_i \beta_i$. Theorem 3.3 tells us that $v \in C$ if and only if v is a solution of the equations $\sum_{i=1}^n \lambda_i \beta_i = 0$.

(1) If $d(C) \geq d$, then any non-zero codeword $c \in C$ has weight at least d. So any solution x of the equations $\sum_{i=1}^n x_i \beta_i$ is either 0 or has weight at least d. Hence for any $i_1 < \cdots < i_k$ with $k < d$, the vector equation $\sum_{j=1}^k x_{i_j} \beta_{i_j}$ has only zero solution, which means that $\{\beta_{i_1}, \cdots, \beta_{i_k}\}$ are linearly independent, hence every $d-1$ columns of H are linearly independent.

(2) If $d(C) \leq d-1$, one can find $0 \neq c = (c_1, \cdots, c_n) \in C$ such that $\operatorname{wt}(c) \leq d-1$. This means the set $\mathcal{I} = \{i \mid c_i \neq 0\}$ is of order $\leq d-1$. Then $c \in C$ implies that the vectors $\beta_i (i \in \mathcal{I})$ are linearly dependent. □

Corollary 3.2. *The distance $d(C) = d$ if and only if any $d-1$ columns of H are linearly independent and there exists d columns of H which are linearly dependent.*

2. Hamming codes

2.1. Binary Hamming code. Let m be a fixed positive integer. There are $n = 2^m - 1$ non-zero vectors in the column vector space $\mathbb{F}_2^{m\times 1}$. We write $\mathbb{F}_2^{1\times m}\backslash\{0\} = \{u_1, \cdots, u_n\}$. The matrix $H := (u_1, \cdots, u_n)$ is a $m \times n$ matrix of full row rank m, hence it is a legal parity check matrix.

Definition 3.7. The binary Hamming code $H(2, m)$ is the linear code over $\mathbb{F}_2$ with H a parity check matrix, i.e., $H(2, m) = \{c \in \mathbb{F}_2^n \mid cH^T = 0\}$.

Proposition 3.2. *The binary Hamming code $H(2, m)$ is a linear $[2^m - 1, 2^m - m - 1, 3]$-code over $\mathbb{F}_2$.*

Proof. The length and dimension are clear. For the distance, note that

- Every 2 columns of H are linearly independent;
- For two columns u_i and u_j, then $u_i, u_j, u_i + u_j = u_k$ are linearly dependent.

By Corollary 3.2, we know $d(H(2, m)) = 3$. □

2.2. Hamming code. Again let m be a positive integer. There are $q^m - 1$ non-zero column vectors in $\mathbb{F}_q^{1\times m}$. We say that two vectors $\alpha \sim \beta$ if there exists $\lambda \in \mathbb{F}_q^\times$ such that $\alpha = \lambda\beta$. The relation $\sim$ is an equivalence relation in $\mathbb{F}_q^{1\times m}\backslash\{0\}$. The set of equivalent classes is actually the protective space $\mathbb{P}_{\mathbb{F}_q}^{m-1}$ of dimension $m - 1$ over $\mathbb{F}_q$.

Pick one representative in each equivalent class, say $v_1, \cdots, v_{\frac{q^m-1}{q-1}}$ and let $H = (v_1, \cdots, v_{\frac{q^m-1}{q-1}})$. Then H is an $m \times \frac{q^m-1}{q-1}$ matrix over $\mathbb{F}_q$ of full row rank m and hence a legal parity check matrix.

Definition 3.8. The (extended) Hamming code $H(q, m)$ is the linear code over $\mathbb{F}_q$ with H a parity check matrix, i.e., $H(q, m) = \{c \in \mathbb{F}_q^n \mid cH^T = 0\}$, where $n = \frac{q^m-1}{q-1}$.

Again we have

- Every 2 columns of $H(q, m)$ are linearly independent;
- There exist 3 columns of $H(q, m)$ which are linearly dependent.

Hence

Proposition 3.3. *The Hamming code $H(q, m)$ is a linear $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m, 3]$-code over $\mathbb{F}_q$.*

3. Equivalence of codes

Definition 3.9. Two codes C and C' of the same alphabet and length are called equivalent if there exists a distance-preserving bijection $\sigma : C \to C'$, i.e. $d(c_1, c_2) = d(\sigma(c_1), \sigma(c_2))$ for any $c_1, c_2 \in C$.

By definition, if C and C' are equivalent, they must have the same parameters.

Example 3.1. We give three examples of distance-preserving bijections.

(1) Permutation of the coordinates. Let C be a code of length n over A. Let σ be a permutation of $\{1, \cdots, n\}$. Then

$$C' = \{(c_{\sigma(1)}, \cdots, c_{\sigma(n)}) \mid (c_1, \cdots, c_n) \in C\}$$

is an equivalent code of C. If C is a linear code, then the effect of this permutation on a generator matrix or a parity check matrix of C is nothing but rearranging columns.

(2) Permutation of the alphabet A. Let τ be a permutation of the alphabet A. Then for a code C over A, the code

$$C' = \{(\tau(c_1), \cdots, \tau(c_n)) \mid (c_1, \cdots, c_n) \in C\}$$

is an equivalent code of C.

(3) Translation. Suppose C is a linear code of length n over $\mathbb{F}_q$. Then for any vector $v \in \mathbb{F}_q^n$, the code $C + v = \{c + v \mid c \in C\}$ is an equivalent code of C. Note that if $v \notin C$, then $C + v$ is not a linear code.

Example 3.2. It seems that the Hamming code $H(q, m)$ depends on H, which in turn depends on the choice of representatives and the order of the vectors, however, all of them are equivalent.

Theorem 3.5. *Every linear code is equivalent to a linear code with a generator matrix (resp. a parity check matrix) in standard form.*

Proof. Let G be a generator matrix of the linear code C. By Gaussian Elimination Method, after row transformation, we may assume $G = (g_{ij})_{k\times n}$ is in row echelon form satisfying

$$\begin{aligned} &g_{1j_1} = \cdots = g_{kj_k} = 1,\ j_1 < j_2 < \cdots < j_k, \\ &g_{ij} = 0 \text{ if either } j < j_i \text{ or } i < s \text{ and } j = j_s \text{ for some } s. \end{aligned}$$

Rearranging the columns of G, we get a matrix in standard form. Thus C is equivalent to a linear code with a generator matrix in standard form. The parity check case is the same. □

Remark 3.2. In the world of linear algebra, any generator matrix is equivalent to the matrix $(I_k \mid 0)$, however, a linear code with a generator matrix $(I_k \mid X)$ with $X \neq 0$ is usually not equivalent to the linear code with a generator matrix $(I_k \mid 0)$.

From now on, we regard equivalent codes as the same code.

4. Encoding and decoding Algorithms

4.1. Encoding messages into a linear code. Usually this is easy. Suppose the messages are stored in the space $\mathbb{F}_q^k$. Suppose C is a linear $[n, k]$-code over $\mathbb{F}_q$ and G is a generator matrix of C. Let α_i be the i-th row vector of G. Then the linear map

$$E:\ \lambda = (\lambda_1, \cdots, \lambda_k) \mapsto c = \lambda G = \sum_{i=1}^{k} \lambda_i \alpha_i$$

encodes the message space $\mathbb{F}_q^k$ into the code C.

Moreover, if $G = (I_k \mid X)$ is in standard form, then $E(\lambda) = c = (\lambda, \lambda X)$, thus one can recover the original message λ easily by just taking the first k coordinates of the codeword c.

4.2. Decoding. For a linear $[n, k, d]$ code C over $\mathbb{F}_q$, let

$$\ell = \ell(C) := [\frac{d-1}{2}].$$

Through a noisy communication channel, a codeword $v \in C$ was sent and a vector $w \in \mathbb{F}_q^n$ is received. The decoding problem is to recover v from w, or equivalently, find the error $\varepsilon = v - w$ from w. As we know by Theorem 2.1, up to ℓ errors can be corrected, we shall assume $\mathrm{wt}(\varepsilon) \leq \ell$.

One ideal for decoding is that ε and w are in the same coset $C + w$ and the weight of ε is small. If some unique element of weight $\leq \ell$ can be found in this coset, then it must be the error vector ε. So the decoding problem is reduced to find the element in the coset $C + w$ with the smallest Hamming weight.

Definition 3.10. In a coset of C, the codeword with the smallest Hamming weight is called a leader of this coset.

Lemma 3.5. *Let C be a linear $[n, k, d]$-code. If the smallest Hamming weight of a coset $C + v$ is $\leq \ell(C)$, then the leader of this coset is unique.*

Proof. Suppose v_1 and v_2 are two leaders of this coset. Then $v_1 - v_2 \in C$ and $\mathrm{wt}(v_1 - v_2) \leq \mathrm{wt}(v_1) + \mathrm{wt}(v_2) < d$, hence $v_1 = v_2$. □

Example 3.3. The linear code $C = \{0000, 1011, 0101, 1110\}$ over $\mathbb{F}_2$ is a $[4, 2, 2]$-code. There are four cosets for C with respect to $\mathbb{F}_2^4$:

- $C + 0000 = \{0000, 1011, 0101, 1110\}$, leader is 0000;
- $C + 0001 = \{0001, 1010, 0100, 1111\}$, leader is 0001 or 0100;
- $C + 0010 = \{0010, 1001, 0111, 1100\}$, leader is 0010;
- $C + 1000 = \{1000, 0011, 1101, 0110\}$, leader is 1000.

One can see that the two leaders in the coset $C + 0001$ are of weight 1, so C cannot correct 1 error. Certainly from $d(C) = 2$, we know C can correct $\ell = [\frac{d-1}{2}] = 0$ errors.

The syndrome decoding is a generic method to decode a general linear code if its distance is bounded.

Definition 3.11. Let C be an $[n, k, d]$-code over $\mathbb{F}_q$. Suppose H is a parity check matrix of C. For $w \in \mathbb{F}_q^n$, the syndrome of w by H is

$$S(w) = S_H(w) = wH^T \in \mathbb{F}_q^{n-k}.$$

It is easy to check that

Lemma 3.6.

(1) *$S(u + v) = S(u) + S(v)$;*

(2) *$S(u) = 0$ if and only if $u \in C$;*
(3) *$S(u) = S(v)$ if and only if u and v are in the same coset.*

Definition 3.12. A syndrome decoding array (SDA) for C, or a syndrome lookup table, is the table containing the pairs $(\varepsilon, S(\varepsilon))$ where ε is the leader of some coset of C in $\mathbb{F}_q^n$.

Algerithm 3.1 (SDA decoding algorithm). *Suppose an SDA for C is already constructed.*
(1) *Compute $S(w)$;*
(2) *Look at the SDA, find $S(\varepsilon) = S(w)$;*
(3) *Decoding $v = w - \varepsilon$.*

To make the SDA decoding algorithm effective, we need to find an efficient way to construct SDA. Note that

- There are too many cosets: in fact, there are q^{n-k} cosets of C in $\mathbb{F}_q^n$. A SDA should not contain the leaders of all cosets.
- Not all cosets have a unique leader, as can be seen from Example 3.3, but every coset has at most 1 leader of weight $\le \ell = [\frac{d-1}{2}]$ by Lemma 3.5.
- Only up to ℓ errors can be corrected, i.e., if a coset has a leader of weight $> \ell$, the number of errors cannot be corrected.

This leads to the following construction:

Algerithm 3.2 (Constructing SDA). *For each $\varepsilon \in \mathbb{F}_q^n$ of* wt $\le \ell$, *compute and store $(\varepsilon, S(\varepsilon))$ in SDA.*

Proposition 3.4. *The number of elements ε of weight $\le \ell$ is*

$$\sum_{i=0}^{\ell} \binom{n}{i} (q-1)^i \le \binom{n}{\ell} q^\ell.$$

Hence the SDA decoding algorithm is a polynomial time algorithm if d is bounded.

Proof. The number of ε with $\mathrm{wt}(\varepsilon) = i$ is $\binom{n}{i}(q-1)^i$, hence follows the first formula. To show the inequality, just expand $q^\ell = (q-1+1)^\ell$ by Newton's Binomial Theorem and use the identity $\binom{n}{\ell}\binom{\ell}{i} = \binom{n}{i}\binom{n-i}{n-\ell}$. □

If d is small (for example $d \le 5$ and hence $\ell \le 2$), one can also use the following algorithm.

Algerithm 3.3. *Suppose C is an $[n, k, d]$-code over $\mathbb{F}_q$, $\ell = [\frac{d-1}{2}]$ and $H = (u_1^T, \cdots, u_n^T)$ is a parity check matrix of C. For the received vector w, compute its syndrome $S_H(w) = wH^T$.*
(1) *If $S_H(w) = 0$, then $\varepsilon = 0$ and $v = w$.*
(2) *If $S_H(w) \neq 0$, then one must have $S_H(w) = a_{i_1}u_{i_1} + \cdots + a_{i_t}u_{i_t}$ for some $1 \le t \le \ell$. Let $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)$, where $\varepsilon_{i_j} = a_{i_j}$ $(1 \le j \le t)$ and $\varepsilon_k = 0$ for all other k. Then $v = w - \varepsilon$.*

Correctness of Algorithm 3.3. Let $\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)$ be the error vector, then $S_H(\varepsilon) = \varepsilon_1 u_1 + \cdots + \varepsilon_n u_n = S_H(w)$. If $S_H(w) = 0$, there is nothing to prove. If $S_H(w) \neq 0$, then $\varepsilon \neq 0$ and $1 \leq t := \mathrm{wt}(\varepsilon) = \#\{i \mid \varepsilon_i \neq 0\} \leq \ell$. Thus $S_H(w)$ must be of the form $\varepsilon_{i_1} u_{i_1} + \cdots + \varepsilon_{i_t} u_{i_t}$ for some $1 \leq t \leq \ell$.

Now suppose $a = (a_1, \cdots, a_n)$ and $b = (b_1, \cdots, b_n) \in \mathbb{F}_q^n$, both of weight $\leq \ell$ satisfying $S_H(a) = a_1 u_1 + \cdots + a_n u_n = b_1 u_1 + \cdots + b_n u_n = S_H(b)$. Then $S_H(a - b) = 0$ and $a - b \in C$. Since $\mathrm{wt}(a - b) \leq \mathrm{wt}(a) + \mathrm{wt}(b) < d - 1$, one must have $a - b = 0$ and $a = b$. This implies that if $S_H(w)$ is of the form $a_1 u_1 + \cdots + a_n u_n$ with $a = (a_1, \cdots, a_n)$ of weight $\leq \ell$, one must have $a = \varepsilon$. □

Algorithm 3.3 can be used to decode the Hamming Code $H(q, m)$. Note that $H(q, m)$ is a linear $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m, 3]$-code over $\mathbb{F}_q$. Then $d = 3$ and $[\frac{d-1}{2}] = 1$. So it can only correct 1 error. We then have the following algorithm.

Algerithm 3.4 (Decoding Hamming Codes). *Given a parity check matrix $H = (u_1^T, \cdots, u_n^T)$ of $H(q, m)$. Suppose a codeword v was sent and a vector $w \in \mathbb{F}_q^n$ is received. Then $\varepsilon = w - v$ is of weight ≤ 1.*

(1) *Compute the syndrome $S_H(w) = wH^T$.*
(2) *If $S_H(w) = 0$, then $\varepsilon = 0$ and $v = w$.*
(3) *If $S_H(w) \neq 0$, then $S_H(w) = \delta u_i^T$ for some $i \in \{1, \cdots, n\}$ and $\delta \in \mathbb{F}_q^\times$. Then $\varepsilon = \delta e_i$ and $v = w - \varepsilon = w - \delta e_i$, where $e_i \in \mathbb{F}_q^n$ whose i-th component is 1 and other components 0.*

CHAPTER 4

Bounds of codes and codes with good bounds

1. Bounds of codes

1.1. Basic setup. As mentioned in Problem 2.1, the construction of codes with good properties is the main problem of coding theory. Naturally we want to know relations between the parameters of the codes.

Definition 4.1. Fix integers $n \geq 1$, $1 \leq d \leq n$ and $q > 1$ which is a prime-power in the linear code case. Define

$$A_q(n,d) := \max\{M \mid \exists \text{ an } (n,M,d)\text{-code of alphabet size } q\},$$
$$B_q(n,d) := \max\{q^k \mid \exists \text{ an } [n,k,d]\text{-code over } \mathbb{F}_q\}.$$

Moreover, the code (resp. linear code) with maximal size $A_q(n,d)$ (resp. $B_q(n,d)$ for linear code) is called an optimal code.

Upper and lower bounds provide us important guidance for the construction of codes: there is no need to find codes of size exceeding the upper bound and a code within the lower bound should have good properties. This leads to the following fundamental problem in coding theory.

Problem 4.1. Find upper/lower bounds of $A_q(n,d)$ and $B_q(n,d)$.

Proposition 4.1. *Fix $n \geq 1$ and $q > 1$ a prime power.*

1. *$B_q(n,d) \leq A_q(n,d) \leq q^n$ for all possible d;*
2. *$B_q(n,1) = A_q(n,1) = q^n$;*
3. *$B_q(n,n) = A_q(n,n) = q$.*

Here for $A_q(n,d)$, $q > 1$ is enough.

Proof. (1) By definition, a code is a subset of A^n, so $M \leq q^n$.

(2) Both A^n and $\mathbb{F}_q^n$ are of length n and distance $d = 1$.

(3) The repetition code $\{(a,\cdots,a) \mid a \in \mathbb{F}_q\}$ is a linear $[n,1,n]$-code over $\mathbb{F}_q$, so $q \leq B_q(n,n) = A_q(n,n)$. It suffices to prove that $A_q(n,n) \leq q$. Suppose not, there exists a code C of size $q+1$, length n and distance n. Since C is of size $q+1$, by Pigeonhole principle, there must exist two codewords $c_1 \neq c_2$ with the same value at one coordinate. So their distance $d(c_1,c_2) < n = d(C)$, not possible. ☐

1.2. Sphere covering lower bound and sphere packing upper bound.

Definition 4.2. Fix an alphabet A of size $q > 1$. For $u \in A^n$ and integer $r \geq 0$, $S_A(u,r) := \{v \in A^n \mid d(u,v) \leq r\}$ is the ball with center u and radius r, and its volume $V_q^n(r) := |S_A(u,r)|$.

Lemma 4.1. *One has*

$$V_q^n(r) = \begin{cases} \sum\limits_{i=0}^{r} \binom{n}{i}(q-1)^i, & \text{if } 0 \leq r \leq n; \\ q^n, & \text{if } r \geq n. \end{cases} \tag{4.1}$$

Proof. If $r \geq n$, then for any $v \in A^n$, $d(u,v) \leq r$, so $S_A(u,r) = A^n$ and $V_q^n(r) = q^n$.

Now suppose $r < n$. Let the surface $S_A(u,r)^\circ = \{v \in A^n \mid d(u,v) = r\}$. Then $S_A(u,r) = \bigcup\limits_{i=0}^{r} S_A(u,i)^\circ$ is a disjoint union. It suffices to show

$$|S_A(u,i)^\circ| = \binom{n}{i}(q-1)^i.$$

But $v = (v_1, \cdots, v_n) \in S_A(u,i)^\circ$ if and only if there are exactly i components among $\{1, \cdots, n\}$ where u and v have different coordinates. There are $\binom{n}{i}$ ways to choose the i components and $q-1$ ways to choose each coordinate of v in these i components. $\square$

Theorem 4.1. $A_q(n,d) \geq \frac{q^n}{V_q^n(d-1)}$.

Proof. Let $M = A_q(n,d)$ and C be an optimal (n,M,d) code. We claim that for every $u \in A^n$, there exists a codeword $c \in C$ such that $d(u,c) \leq d-1$. Indeed, if not, there exists $u \in A^n$ such that $d(c,u) \geq d$ for all $c \in C$. Add u to C and we obtain an $(n, M+1, d)$ code, contrary to the maximality of M. Hence $A^n = \bigcup\limits_{c \in C} S_A(c, d-1)$. This implies $q^n \leq MV_q^n(d-1)$ and $M = A_q(n,d) \geq \frac{q^n}{V_q^n(d-1)}$. $\square$

Theorem 4.2 (Hamming bound). $A_q(n,d) \leq \frac{q^n}{V_q^n([\frac{d-1}{2}])}$.

Proof. Let C be an arbitrary (n,M,d)-code. It suffices to show $M \leq \frac{q^n}{V_q^n([\frac{d-1}{2}])}$. For any two codewords $c \neq c' \in C$, if $u \in S_A(c, [\frac{d-1}{2}]) \cap S_A(c', [\frac{d-1}{2}])$, then $d \leq d(c,c') \leq d(c,u) + d(u,c') \leq 2[\frac{d-1}{2}] < d$, impossible. Hence $S_A(c, [\frac{d-1}{2}])$ and $S_A(c', [\frac{d-1}{2}])$ are disjoint. By the inclusion $\bigcup_{c \in C} S_A(c, [\frac{d-1}{2}]) \subset A^n$, then $MV_q^n([\frac{d-1}{2}]) \leq q^n$ and $M \leq \frac{q^n}{V_q^n([\frac{d-1}{2}])}$. $\square$

Definition 4.3. An (n,M,d)-code of alphabet size q is called a perfect code if the Hamming bound is achieved, i.e., $M = \frac{q^n}{V_q^n([\frac{d-1}{2}])}$.

By definition, a perfect code is optimal.

Example 4.1. The Hamming code $H(q,m)$ is a $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m, 3]$ code over $\mathbb{F}_q$. Then $[\frac{d-1}{2}] = 1$, $V_q^n(1) = 1 + \binom{n}{1}(q-1) = q^m$, and $M = q^{n-m} = \frac{q^n}{V_q^n(1)}$. Hence $H(q,m)$ is a perfect code.

1.3. Singleton bound and MDS codes.

THEOREM 4.3 (Singleton Bound). *Fix n, d and q, then $A_q(n,d) \leq q^{n-d+1}$. In particular, if C is a linear $[n,k,d]$-code over $\mathbb{F}_q$, then $k \leq n-d+1$.*

PROOF. Let C be an arbitrary (n,M,d)-code. We delete the last $d-1$ coordinates of codewords in C and obtain a new code C' of length $n-d+1$. Assume $c_1, c_2 \in C$ are different codewords and the corresponding codeword in C' are c_1' and c_2' respectively. By $d(c_1,c_2) \geq d$, we have $d(c_1',c_2') \geq 1$ and hence $c_1' \neq c_2'$. This means that C' is again of size M. Thus $M \leq q^{n-d+1}$. □

DEFINITION 4.4. A linear $[n,k,d]$-code is called a maximal distance separable code (or MDS code in short) if $k = n-d+1$.

THEOREM 4.4. *Let C be an $[n,k,d]$-code, G a generator matrix and H a parity check matrix of C. Then the followings are equivalent:*

(1) *C is an MDS code;*
(2) *every $n-k$ columns of H are linearly independent;*
(3) *every k columns of G are linearly independent;*
(4) *$C^{\perp}$ is an MDS code.*

PROOF. (1) $\Rightarrow$ (2): since $d = n-k+1$ and every $d-1$ columns of H are linearly independent (Theorem 3.4).

(2) $\Rightarrow$ (1): (2) means that $d \geq n-k+1$ by Theorem 3.4, and the Singleton bound indicates that $d \leq n-k+1$. Hence $d = n-k+1$.

(3) $\Leftrightarrow$ (4): same as (1) $\Leftrightarrow$ (2).

(1) $\Rightarrow$ (4) (then naturally (4) $\Rightarrow$ (1)): we know $C^{\perp}$ is an $[n, n-k]$-code, so it suffices to prove that $d(C^{\perp}) = k+1$. Suppose not, the Singleton bound indicates that $d(C^{\perp}) \leq k$, hence there exists a nonzero codeword $c \in C^{\perp}$ of weight at most k. That is, at least $n-k$ coordinates of c is 0. Without loss of generality, we may assume the last $n-k$ coordinates of c are zeros. We write $H = (A, H')$, where $A \in \mathbb{F}_q^{(n-k)\times k}$ and $H' \in \mathbb{F}_q^{(n-k)\times(n-k)}$. Then c is a linear combination of row vectors of H, thus the same linear combination of the row vectors of H' is 0. By the equivalence of (1) and (2), H' is invertible and hence its row vectors are linearly independent. If a linear combination of the row vectors of H' equals zero, all coefficients must be zero. Hence $c = 0$, contradiction! □

EXAMPLE 4.2. (1) The repetition code is a $[n,1,n]$-code, hence is an MDS code.

(2) Hamming code $H(m,q)$ is not an MDS code if $m \geq 3$.

1.4. Plotkin bound. This is an upper bound for binary codes.

THEOREM 4.5 (Plotkin bound). *For a binary (n,M,d)-code, if $2d > n$, then:*

$$M \leq \begin{cases} 2[\frac{d}{2d-n}], & \textit{if } M \textit{ is even;} \\ 2[\frac{d}{2d-n}] - 1, & \textit{if } M \textit{ is odd.} \end{cases} \tag{4.2}$$

PROOF. Write $C = \{c_i \mid 1 \leq i \leq M\} \subseteq \mathbb{F}_2^n$. The matrix $A = ((c_1 + c_2)^T, \cdots, (c_{M-1}+c_M)^T)^T$ is a $\binom{M}{2} \times n$ matrix over $\mathbb{F}_2$. Let N be the number of 1's in the entries of A. We compute N in two ways. On one hand, since $\mathrm{wt}(c_i + c_j) \geq d$, we have $N \geq d\binom{M}{2}$. On the other hand, suppose N_i is the number of 1's in the i-th coordinate of all codewords of C. Then the number of 1's in the i-th coordinate among all vectors $c_s + c_t$ $(s \neq t)$ is $N_i(M - N_i)$, so $N = \sum_{i=1}^n N_i(M - N_i)$. If M is even, then $N_i(M - N_i) \leq M^2/4$ and

$$d\binom{M}{2} \leq N \leq \sum_{i=1}^{n} (\frac{M}{2})^2 = \frac{nM^2}{4},$$

which means $M \leq \frac{2d}{2d-n}$ and $\frac{M}{2} \leq [\frac{d}{2d-n}]$. If M is odd, then $N_i(M - N_i) \leq (M+1)(M-1)/4$ and

$$d\binom{M}{2} \leq N \leq \sum_{i=1}^{n} \frac{(M+1)(M-1)}{4},$$

hence $\frac{M+1}{2} \leq [\frac{d}{2d-n}]$. □

1.5. Gilbert-Varshamov Bound (GV Bound). In 1952. Gilbert and Varshamov independently proved the following lower bound for $B_q(n, d)$.

THEOREM 4.6 (Gilbert-Varshamov Bound). *Assume that $2 \leq d \leq n$, $1 \leq k \leq n$ and $V_q^{n-1}(d-2) < q^{n-k}$. Then there exists a linear $[n, k]$-code over $\mathbb{F}_q$ with distance at least d.*

PROOF. It suffices to construct a legal parity check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ such that any $d-1$ columns of H are linearly independent. Write $H = (c_1, \cdots, c_n)$. We need to construct the column vectors $c_i \in \mathbb{F}_q^{(n-k)\times 1}$ for $1 \leq i \leq n$. This is done as follows:

(1) For $1 \leq i \leq n-k$, let $c_i = e_i$ be the i-th standard column vector. This implies that H is of full row rank and hence legal.
(2) For $n-k \geq j \geq n-1$, after c_j is constructed, find a vector $c_{j+1} \notin \mathrm{Span}\{c_i \mid i \in I\}$ where I is any subset of $\{1, \cdots, j\}$ of order $d-2$.

We are left to show the induction step in (2) can always continue. For any $j < n$, the number N of vectors which are linear combinations of $\leq d-2$ vectors (with terms of 0 coefficient removed) in $\{c_i \mid 1 \leq i \leq j\}$ satisfies

$$N \leq \sum_{i=0}^{d-2} \binom{j}{i}(q-1)^i \leq \sum_{i=0}^{d-2} \binom{n-1}{i}(q-1)^i = V_q^{n-1}(d-2) < q^{n-k}.$$

This means we can always choose c_{j+1} for $n-k \leq j < n$. □

COROLLARY 4.1 (Gilbert-Varshamov Bound). *If $q > 1$ is a prime-power and $2 \leq d \leq n$, then:*

$$B_q(n,d) \geq \frac{q^{n-1}}{V_q^{n-1}(d-2)}. \tag{4.3}$$

Proof. Let $k = n - \lceil \log_q(1 + V_q^{n-1}(d-2)) \rceil$. Then

$$q^{n-k} \geq 1 + V_q^{n-1}(d-2) > V_q^{n-1}(d-2).$$

By Theorem 4.6, there exists an $[n, k, d']$-code C over $\mathbb{F}_q$ with distance $d' \geq d$. Then $C' = \{(c, 0^{d'-d}) \mid c \in C\}$ is a linear $[n + d' - d, k, d']$ code (so called the extension operation). Suppose $c, c' \in C'$ such that $d(c, c') = d'$, then the set $I = \{i \mid c_i \neq c'_i\}$ is of order d'. Take $J \subseteq I$ and $|J| = d' - d$. Deleting all j-th components ($j \in J$) of codewords of C' (so called the puncturing operation), then the new code obtained is an $[n, k, d]$-code. So

$$B_q(n, d) \geq q^k = q^{n - \lceil \log_q(1+V_q^{n-1}(d-2)) \rceil} \geq \frac{q^{n-1}}{1 + V_q^{n-1}(d-2)} \geq \frac{q^{n-1}}{V_q^{n-1}(d-2)}.$$

This gives the Gilbert-Varshamov bound. □

Example 4.3. For the Hamming code $H(q, m)$, $n = \frac{q^m - 1}{q-1}$, $k = n - m$ and $d = 3$. Then $V_q^{n-1}(d-2) = q^m - q + 1$ and $q^k = q^{n-m} \geq \frac{q^{n-1}}{q^m - q + 1}$. Hence $H(q, m)$ beats the GV bound.

2. Golay Codes

2.1. The codes G_{24} and G_{23}.

Definition 4.5. The extended Golay Code G_{24} is a linear $[24, 12]$-code over $\mathbb{F}_2$ with a generator matrix $G = (I_{12} \mid A)$ in standard form, where

$$A = \begin{pmatrix} 0 & 1^{11} \\ (1^{11})^T & P \end{pmatrix}, \quad P = \begin{pmatrix} 1&1&0&1&1&1&0&0&0&1&0 \\ 0&1&1&0&1&1&1&0&0&0&1 \\ 1&0&1&1&0&1&1&1&0&0&0 \\ 0&1&0&1&1&0&1&1&1&0&0 \\ 0&0&0&1&0&1&1&0&1&1&1 \\ 1&0&0&0&1&0&1&1&0&1&1 \\ 1&1&0&0&0&1&0&1&1&0&1 \\ 1&1&1&0&0&0&1&0&1&1&0 \\ 0&1&1&1&0&0&0&1&0&1&1 \\ 1&0&1&1&1&0&0&0&1&0&1 \end{pmatrix}.$$

Here $1^{11} = (1, \cdots, 1) \in \mathbb{F}_2^{11}$ whose entries are all 1, and $P = P(\alpha)$ is the cyclic matrix whose first row is $\alpha = 11011100010$.

Proposition 4.2. *G_{24} satisfies the following conditions:*

(1) *$n = 24$ and $k = 12$.*
(2) *It has a parity check matrix $H = (A \mid I_{12})$.*
(3) *It is self-dual: $G_{24} = G_{24}^{\perp}$.*
(4) *For all codewords $c \in G_{24}$, $\mathrm{wt}(c)$ is divisible by 4.*
(5) *There exists no codeword of weight 4 in G_{24}, hence $d(G_{24}) = 8$.*

Proof. The first two claims are easy, we omit the proof here.

For (3), for any 2 rows r_i, r_j of G, their inner product $r_i \cdot r_j = r_i r_j^T = 0$, hence $G_{24} \subseteq G_{24}^{\perp}$. Also $\dim(G_{24}) = \dim(G_{24}^{\perp}) = 12$, so $G_{24} = G_{24}^{\perp}$.

For (4), let $v \in G_{24}$. Suppose v is the sum of k different rows of G. We show by induction to k that $4 \mid \mathrm{wt}(v)$. If $k = 1$, i.e. $v = r_i$, then $\mathrm{wt}(v) = 8$ or 12. If $k = 2$, i.e., $v = r_i + r_j$, then by Lemma 3.1,

$$\mathrm{wt}(v) = \mathrm{wt}(r_i) + \mathrm{wt}(r_j) - 2\mathrm{wt}(r_i * r_j).$$

Since $r_i \cdot r_j = \sum_{\ell=1}^{24} r_{i,\ell} r_{j,\ell} = 0 \in \mathbb{F}_2$, the vector $r_i * r_j = (r_{i,\ell} r_{j,\ell})$ is of even weight. So $4 \mid \mathrm{wt}(v)$. This argument is applicable to general k.

For (5), suppose on the contrary that there exists $v \in G_{24}$ such that $\mathrm{wt}(v) = 4$. Write $v = (v_1, v_2)$ with both $v_i \in \mathbb{F}_2^{12}$. Note that G_{24} is self-dual, so $H = (A \mid I_{12})$ is also a generator matrix for G_{24}. Now v_1 (resp. v_2) is uniquely a sum of standard vectors $e_i \in \mathbb{F}_2^{12}$, v must be the sum of the corresponding row vectors r_i of G (resp. row vectore r'_i of H). By symmetry of v_1 and v_2, we may assume $\mathrm{wt}(v_1) \le 2$:

- If $v_1 = 0$, then $v = 0$, not possible;
- If $\mathrm{wt}(v_1) = 1$, then $v_1 = e_i$ and $v = r_i$ must be one of the rows, impossible by simple observation;
- If $\mathrm{wt}(v_1) = 2$, then $v_1 = e_i + e_j$ and $v = r_i + r_j$, also impossible by observation.

This finishes the proof of (5). □

Definition 4.6. The binary Golay code G_{23} is the linear code over $\mathbb{F}_2$ with a generator matrix $\hat{G} = (I_{12} \mid \hat{A})$, where $\hat{A}$ is obtained by deleting the last column of A.

Theorem 4.7. *G_{23} is a binary $[23, 12, 7]$-code, and it is perfect.*

Proof. Obviously G_{23} is a binary $[23, 12]$-code, and $d(G_{23}) \ge 7$ since $d(G_{24}) = 8$. The last row of $\hat{G}$ is of weight 7, so $d(G_{23}) = 7$.

Now $n = 23$, $k = 12$ and $[\frac{d-1}{2}] = 3$, then

$$V_2^n(3) = \sum_{i=0}^{3} \binom{23}{i} = 2048 = 2^{11}.$$

Thus $\frac{2^{23}}{V_2^n(3)} = 2^{12}$ and G_{23} is perfect. □

2.2. The codes G_{12} and G_{11}.

Definition 4.7. The extended ternary Golay code G_{12} is the linear $[12, 6]$-code over $\mathbb{F}_3$ with a generator matrix $G = (I_6 \mid A)$ where

$$A = \begin{pmatrix} 0 & 1^5 \\ (1^5)^T & P \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 & 0 \\ 2 & 2 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 2 & 2 \end{pmatrix}.$$

Here $1^5 \in \mathbb{F}_3^5$ is the vector whose entries are all 1; P is the cyclic matrix whose first row is 01221.

The ternary Golay code G_{11} is the linear $[11, 6]$-code over $\mathbb{F}_3$ with a generator matrix $\hat{G} = (I_6 \mid \hat{A})$ where $\hat{A}$ is the matrix obtained by deleting the last columns of A.

Theorem 4.8.

(1) *G_{12} is a $[12, 6, 6]$-self dual code.*

(2) G_{11} *is a* $[11, 6, 5]$*-ternary perfect code.*

Proof. For (1): it is easy to show G_{12} is a self-dual $[12, 6]$-ternary code, so it suffices to show that $d(G_{12}) = 6$, equivalently, that no non-zero codeword has weight ≤ 6. Assume $c = (c_1, c_2)$ is a nonzero codeword of weight ≤ 6, where $c_i \in \mathbb{F}_3^6$. Since G_{12} is self-dual, $H = (-A^T \mid I_6) = (-A \mid I_6)$ is also a generator matrix of G_{12}. By symmetry, one may assume $\mathrm{wt}(c_1) \leq 2$:

- $\mathrm{wt}(c_1) = 0$ indicates $c = 0$, contradiction to $c \neq 0$;
- $\mathrm{wt}(c_1) = 1$, then $c = \pm r_i$, this is impossible;
- $\mathrm{wt}(c_1) = 2$, then $c = \pm r_i \pm r_j$, again it is impossible.

For (2), it is easy to show that G_{11} is a $[11, 6, 5]$-ternary code. Now $d = 5$, $[\frac{d-1}{2}] = 2$, and $V_3^n(2) = \sum_{i=0}^{2} 2^i \binom{11}{i} = 243 = 3^5 = 3^{n-k}$, hence G_{11} is perfect. $\square$

2.3. Decoding of G_{24} and G_{23}. The basic fact is that $G = (I_{12} \mid A)$ and $H = (A \mid I_{12})$ are both generator and parity check matrices of G_{24}. Let $\alpha_i \in \mathbb{F}_2^{12}$ $(i = 1, \cdots, 12)$ be the i-th row vector of A. We know $\mathrm{wt}(\alpha_i) \geq 7$ and $\mathrm{wt}(\alpha_i + \alpha_j) \geq 6$ if $i \neq j$. By $GH^T = 0$ we know $A^2 = I$.

Suppose v was sent and w is received, ε is the error. Let $S_1 = S_G(w) = wG^T = \varepsilon G^T$ and $S_2 = S_H(w) = wH^T = \varepsilon H^T$. Write $\varepsilon = (\varepsilon_1, \varepsilon_2)$ with each $\varepsilon_i \in \mathbb{F}_2^{12}$. Then $S_1 = \varepsilon G^T = \varepsilon_1 + \varepsilon_2 A$ and $S_2 = \varepsilon H^T = \varepsilon_1 A + \varepsilon_2 = S_1 A$.

Note that $d = 8$, $[\frac{d-1}{2}] = 3$, so we need to assume $\mathrm{wt}(\varepsilon) \leq 3$. We have the following analysis.

(1) If $\varepsilon_2 = 0$, then $S_1 = \varepsilon_1$, $\varepsilon = (S_1, 0)$ and $\mathrm{wt}(S_1) \leq 3$. Similarly, if $\varepsilon_1 = 0$, then $S_2 = \varepsilon_2$, $\varepsilon = (0, S_2)$ and $\mathrm{wt}(S_2) \leq 3$.

(2) If ε_1 and ε_2 are both not zero, then they have weight at least 1.
   - If $\mathrm{wt}(\varepsilon_2) = 1$, then $\varepsilon_2 = e_i \in \mathbb{F}_2^{12}$ for some i. Hence $S_1 = \varepsilon_1 + e_i A = \varepsilon_1 + \alpha_i$. Then $\mathrm{wt}(S_1) \geq \mathrm{wt}(\alpha_i) - \mathrm{wt}(\varepsilon_1) \geq 5$. Similarly, If $\mathrm{wt}(\varepsilon_1) = 1$, then $\varepsilon_1 = e_j \in \mathbb{F}_2^{12}$ for some j. Hence $S_2 = \varepsilon_2 + \alpha_j$ and $\mathrm{wt}(S_2) \geq 5$.
   - If $\mathrm{wt}(\varepsilon_1) = 1$ and $\mathrm{wt}(\varepsilon_2) = 2$, then $S_1 = \varepsilon_1 + \alpha_i + \alpha_j$ and $\mathrm{wt}(S_1) \geq 5$; if $wt(\varepsilon_1) = 2$ and $\mathrm{wt}(\varepsilon_2) = 1$, then similarly we have $\mathrm{wt}(S_2) \geq 5$.

   So anyway we have $\mathrm{wt}(S_1) > 3$ and $\mathrm{wt}(S_2) > 3$ in this case.

This gives the following algorithm:

Algerithm 4.1 (Decoding G_{24}).

(1) *Compute* $S_1 = S_G(w)$ *and* $S_2 = S_H(w)$.
(2) *If either* $S_1 = 0$ *or* $S_2 = 0$, *then* $v = w$.
(3) *If* $\mathrm{wt}(S_1) \leq 3$, *then* $\varepsilon = (S_1, 0)$; *if* $\mathrm{wt}(S_2) \leq 3$, *then* $\varepsilon = (0, S_2)$.
(4) *If there exists* i *such that* $\mathrm{wt}(S_1 + \alpha_i) \leq 2$, *then* $\varepsilon = (S_1 + \alpha_i, e_i)$; *if there exists* i *such that* $\mathrm{wt}(S_2 + \alpha_j) \leq 2$, *then* $\varepsilon = (e_j, S_2 + \alpha_j)$.

Now the decoding for G_{23} is easy.

Algerithm 4.2 (Decoding G_{23}).

(1) *For* $w = (w_1, \cdots, w_{23})$ *received, let* $w' = (w_1, \cdots, w_{23}, 1 + w_1 + \cdots + w_{23})$.
(2) *By Algorithm 4.1, find* ε' *such that* $\mathrm{wt}(\varepsilon') \leq 3$ *and* $c' = w' - \varepsilon' \in G_{24}$.
(3) c *is the vector by removing the last component of* c'.

Here we note that if $c \in G_{23}$ was the sent and $d(w, c) \leq 3$, let $c' \in G_{24}$ be a codeword extending c, then $d(w', c') \leq 4$. Since $\mathrm{wt}(c')$ is even and $\mathrm{wt}(w')$ is odd, we must have $d(w', c') \leq 3$ and c' is unique. Hence Algorithm 4.1 is applicable.

2.4. No more perfect codes. van Lint and Tietäväinen in 1973 showed that Hamming codes and Golay codes are the only perfect codes.

3. Other examples of optimal codes

3.1. Reed-Solomon Codes. Suppose $k \leq n \leq q$ and q is a prime power.

Definition 4.8. Pick distinct elements $\alpha_1, \cdots, \alpha_n \in \mathbb{F}_q$. For any polynomial $f(x) \in \mathbb{F}_q[x]$, set

$$c_f := (f(\alpha_1), \cdots, f(\alpha_n)) \in \mathbb{F}_q^n.$$

Then the Reed-Solomon code $\mathrm{RS}_{q,n,k}$, also called the polynomial code, is the code $\{c_f \mid \deg(f) \leq k - 1\} \subset \mathbb{F}_q^n$.

Theorem 4.9. $\mathrm{RS}_{q,n,k}$ *is a linear* $[n, k, n - k + 1]$*-code, hence an MDS code.*

Proof. Let $V_k := \{f \in \mathbb{F}_q[x] \mid \deg(f) \leq k - 1\}$. Then V_f is an $\mathbb{F}_q$-vector space of dimension k. Define the map

$$\varphi : V_k \to \mathbb{F}_q^n, \quad \varphi(f) = c_f.$$

It is easy to see that φ is $\mathbb{F}_q$-linear and $\varphi(V_k) = \mathrm{RS}_{q,n,k}$. Moreover, if $\varphi(f) = 0$, then α_i $(i = 1, \cdots k)$ are k different roots of f whose degree $\leq k - 1$, hence f must be the zero polynomial. In other words, φ is injective and $\dim_{\mathbb{F}_q} \mathrm{RS}_{q,n,k} = \dim_{\mathbb{F}_q} V_k = k$.

Next we consider the distance: if $f \in V_k$ is not the zero polynomial, then f has at most $k - 1$ roots in $\mathbb{F}_q$, so $\mathrm{wt}(c_f) \geq n - k + 1$. Hence $d \geq n - k + 1$. Moreover, for the polynomial $g(x) = \prod_{i=1}^{k-1}(x - \alpha_i)$, $\mathrm{wt}(c_g) = n - k + 1$, hence $d = n - k + 1$. The last fact can also be deduced from the Singleton bound. □

By the isomorphism $\varphi : V_k \to \mathrm{RS}_{q,n,k}$, the basis $\{x^i \mid 0 \leq i \leq k - 1\}$ of V_k corresponds to the basis. $\{c_{x^i} \mid 0 \leq i \leq k - 1\}$ of $\mathrm{RS}_{q,n,k}$. So we have:

Proposition 4.3. $\mathrm{RS}_{q,n,k}$ *has a generator matrix*

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix}.$$

Remark 4.1. Later on we shall study Generalized Reed-Solomon code, in particular,

(1) Find a parity check matrix of $\mathrm{RS}_{q,n,k}$.
(2) Decode the Reed-Solomon code.

Example 4.4. Suppose $q = n$ and $\mathbb{F}_q = \{\alpha_i \mid 1 \le i \le n\}$. Then

(1) $C_k := \mathrm{RS}(q, q, k)$ is a $[q, k, q-k+1]$-code over $\mathbb{F}_q$;
(2) $C_k^{\perp} = C_{q-k}$.

Proof. We only prove (2), since (1) is trivial. Note that C_k has a basis: $\{c_{x^i} \mid 0 \le i \le k-1\}$ and C_{q-k} has a basis: $\{c_{x^i} \mid 0 \le i \le q-k-1\}$.

(a) If $i = j = 0$, then $(c_1, c_1) = \sum_{\ell=1}^{q} 1 = q = 0 \in \mathbb{F}_q$;
(b) If otherwise, then $1 \le i + j \le q - 2$ and $(c_i, c_j) = \sum_{\ell=1}^{q} \alpha_\ell^{i+j} = \sum_{\alpha \in \mathbb{F}_q^*} \alpha^{i+j} = 0 \in \mathbb{F}_q$, where α is a primitive root of $\mathbb{F}_q$.

Combining (a) and (b), we have $C_k \perp C_{q-k}$. Finally we check the dimensions: $\dim C_k + \dim C_{q-k} = q$, so $C_k^{\perp} = C_{q-k}$. □

Example 4.5. Suppose $n = q - 1$ and $\mathbb{F}_q^{\times} = \{\alpha_i \mid 1 \le i \le q-1\}$. Let $C'_k := \mathrm{RS}(q, q-1, k)$ and $C''_k := \{c_f \mid \deg(f) \le k-1,\ f(0) = 0\}$. Then

(1) C'_k is a $[q-1, k, q-k]$-MDS code over $\mathbb{F}_q$;
(2) C''_k is a $[q-1, k-1, q-k+1]$-MDS code over $\mathbb{F}_q$, and $(C''_k)^{\perp} = C'_{q-k}$.

Proof. Exercise. □

3.2. Hadamard Codes.

Definition 4.9. A square matrix H_n of size n is called a Hadamard matrix if all its entries are in $\{-1, 1\}$ and $H_n H_n^T = nI_n$.

The existence and construction of Hadamard matrices are still active research problems. First one has

Proposition 4.4 (Sylvester). *If n is a 2-power, then one can construct a Hadamard matrix of size n inductively:*
(1) $H_1 = 1$;
(2) $H_{2n} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}$.

PROOF. Only need to check $H_{2n}H_{2n}^T = 2nH_{2n}$:

$$H_{2n}H_{2n}^T = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix} \begin{pmatrix} H_n^T & H_n^T \\ H_n^T & -H_n^T \end{pmatrix}$$
$$= \begin{pmatrix} H_nH_n^T + H_nH_n^T & H_nH_n^T - H_nH_n^T \\ H_nH_n^T - H_nH_n^T & H_nH_n^T + H_nH_n^T \end{pmatrix} = 2nI_{2n}. \qquad \square$$

DEFINITION 4.10. Suppose $H_n = (h_{ij})$ is a Hadamard matrix and $h_i = (h_{ij})$ $(i = 1, \cdots, n)$ is the i-th row vectors of H_n. The Hadamard code Had_n derived from H_n is the binary code

$$\mathrm{Had}_n := \{\pm h_i \mid 1 \le i \le n\} \subseteq \{\pm 1\}^n.$$

LEMMA 4.2. *For any $i \neq j$, $d(h_i, h_j) = \frac{n}{2}$. That is, n must be even.*

PROOF. We has $h_ih_j^T = \sum_{k=1}^n h_{ik}h_{jk} = n\delta_{ij} = 0$ if $i \neq j$. So exactly half of $h_{ik}h_{jk} = 1$, i.e., $h_{ik} = h_{jk}$. Thus $d(h_i, h_j) = \frac{n}{2}$. $\square$

PROPOSITION 4.5. Had_n *is a binary $(n, 2n, \frac{n}{2})$-code.*

PROOF. Obviously Had_n is a binary $(n, 2n)$-code. And we notice that for any $i \neq j$, by the above lemma, $d(-h_i, -h_j) = d(-h_i, h_j) = d(h_i, h_j) = \frac{n}{2}$. So the distance of Had_n is $\frac{n}{2}$. $\square$

REMARK 4.2. Another form of Plotkin Bound states that: if there exist codewords $c_1, \cdots, c_M$ in $\{\pm 1\}^n$ such that $d(c_i, c_j) \ge \frac{n}{2}$ for every $i \neq i$, then $M \le 2n$. In this sense, the Hadamard Code Had_n is optimal.

3.3. Walsh-Hadamard Code. Let m be a fixed positive integer and $n = 2^m - 1$. Write $\mathbb{F}_2^{m\times 1}\backslash\{0\} = \{u_1, \cdots, u_n\}$. The matrix $G = (u_1, \cdots, u_n)$ is a $m \times n$ matrix of full row rank m. Recall G (denoted as G there) is used as a parity check matrix to define the binary Hamming code $H(2, m)$ in last Chapter.

DEFINITION 4.11. The binary Walsh-Hadamard Code WH_m is the linear code over $\mathbb{F}_2$ with G as a generator matrix, i.e., the dual code of $H(2, m)$.

PROPOSITION 4.6. *The Walsh-Hadamard Code* WH_m *is a* $[2^m-1, m, 2^{m-1}]$*-code, any non-zero codeword of* WH_m *has weight* 2^{m-1}*, and* WH_m *is optimal to the Plotkin bound.*

PROOF. For $x = (x_1, \cdots, x_m) \in \mathbb{F}_2^m$, then $x \mapsto c(x) = xG$ is a bijection of $\mathbb{F}_2^m$ to WH_m. Moreover, $\mathrm{wt}(c(x)) = \#\{z \in \mathbb{F}_2^{m\times 1} \mid xz = 1\}$. If $c(x)$ is a non-zero codeword, suppose $x_i \neq 0$. Note that for any $z \in \mathbb{F}_2^{m\times 1}$, $xz = 0$ if and only if $x(z + e_i^T) = 1$, so exactly half of the z's in $\mathbb{F}_2^{m\times 1}$ satisfy $xz = 1$. Hence $\mathrm{wt}(c(x)$ and $d(\mathrm{WH}_m) = 2^{m-1}$.

In this case $2d - n = 1$ and the Plotkin bound is that $M \le 2d = 2^m$. So WH_m is optimal. $\square$

CHAPTER 5

Constructing Codes from Other Codes

1. General Rules for Construction

THEOREM 5.1. *Suppose C is a linear $[n, k, d]$-code over $\mathbb{F}_q$.*

(1) *Extension: for $r \geq 1$, there exists an $[n+r, k, d]$-code.*
(2) *Puncturing: for $1 \leq r \leq d-1$, there exists an $[n-r, k, d-r]$-code.*
(3) *For $1 \leq r \leq d-1$, there exists an $[n, k, d-r]$-code.*
(4) *Subcode: for $1 \leq r \leq k-1$, there exists an $[n, k-r, d]$-code;*
(5) *For $1 \leq r \leq k-1$, there exists an $[n-r, k-r, d]$-code.*

PROOF. (1) $C' = \{(c, 0^r) \mid c \in C\}$ is the code we want, where 0^r is a r-dimensional vector whose coordinates are all zero.

(2) Let $c_0 \in C$ be a codeword with $\mathrm{wt}(c_0) = d$. Let $I_0 = \{i \mid c_{0,i} \neq 0\}$, then $|I_0| = d$. We fix $I \subset I_0$ with $|I| = r$. For $c \in C$, let $\hat{c}$ be the codeword obtained by erasing coordinates of I of c. And we obtain a new code $\hat{C} = \{\hat{c} \mid c \in C\}$. Then:

(a) $\hat{C}$ is of length $n-r$;
(b) $\mathrm{wt}(\hat{c_0}) = d-r$ and $\mathrm{wt}(\hat{c}) \geq d-r$ for all $\hat{c}$, hence $\hat{C}$ is of distance $d-r$;
(c) The linear map $C \to \hat{C}$, $c \mapsto \hat{c}$ is injective, as $d(\hat{c}_1, \hat{c}_2) \geq d(c_1, c_2) - r > 0$ for $c_1 \neq c_2$.

Combining (a)-(c), $\hat{C}$ is an $[n-r, k, d-r]$-code.

(3) By (1) there exists an $[n+r, k, d]$-code, and by (2) we obtain an $[n, k, d-r]$-code.

(4) Let $c_1 \in C$ such that $\mathrm{wt}(c_1) = d$. Expand $\{c_1\}$ We can find a set of vectors $\{c_i \mid 1 \leq i \leq k-r\}$ in C which is linearly independent, the span of these vectors is an $[n, k-r, d]$-code.

(5) If $n = k$, then $d = 1$ and $\mathbb{F}_q^{n-r}$ is the code we want. If $n > k$, we may assume C has a parity check matrix H in the form $H = (I_{n-k} \mid X)$. Erase the last r columns of H, we get a legal parity check matrix $H_1 = (I_{n-k} \mid X_1)$ of size $(n-k) \times (n-r)$, which gives a linear $[n-r, k-r]$-code C'. Since every $d-1$ columns in H_1 are linearly independent, $d(C') \geq d$. Apply (3), we obtain an $[n-r, k-r, d]$-code. $\square$

PROPOSITION 5.1. *Suppose C_i $(i = 1, 2)$ is a linear $[n_i, k_i, d_i]_q$ code. The direct sum of C_1 and C_2 is $C_1 \oplus C_2 := \{(c_1, c_2) \mid c_1 \in C_1, c_2 \in C_2\}$ has the following properties:*

- *it is a linear $[n_1+n_2, k_1+k_2, \min(d_1, d_2)]_q$-code;*

- *It has a generator matrix $G = \operatorname{diag}(G_1, G_2)$ and a parity check matrix $H = \operatorname{diag}(H_1, H_2)$.*

THEOREM 5.2. *Suppose C_i $(i = 1, 2)$ is a linear $[n_i, k_i, d_i]_q$ code. Then the code*

$$C = \{(u, u + v) \mid u \in C_1, v \in C_2\}$$

is a linear $[2n, k_1 + k_2, \min(2d_1, d_2)]$-code.

PROOF. We can construct a bijection from $C_1 \oplus C_2$ to C, hence the dimension of C is k_1+k_2. It suffices to prove the distance of C is $\min(2d_1, d_2)$.

Let $0 \neq c = (u, u + v) \in C$, then $\mathrm{wt}(c) = \mathrm{wt}(u) + \mathrm{wt}(u + v)$. Then

(a) if $u = 0$, then $\mathrm{wt}(c) = \mathrm{wt}(v) \geq d_2 \geq \min(2d_1, d_2)$;
(b) if $v = 0$, then $\mathrm{wt}(c) = 2\mathrm{wt}(u) \geq 2d_1 \geq \min(2d_1, d_2)$;
(c) if $u \neq$ and $v \neq 0$, then $\mathrm{wt}(c) = \mathrm{wt}(u)+\mathrm{wt}(u+v) \geq \mathrm{wt}(u)+\mathrm{wt}(v)-\mathrm{wt}(u) = \mathrm{wt}(v) \geq d_2 \geq \min(2d_1, d_2)$.

Thus $d(C) \geq \min(2d_1, d_2)$. Now let $c_i \in C_i$ $(i = 1, 2)$ be the codeword with $\mathrm{wt}(c_i) = d_i$. Then $\mathrm{wt}((c_1, c_1)) = 2d_1$ and $\mathrm{wt}((0, c_2)) = d_2$, one of them has weight $\min(2d_1, d_2)$. Hence $d(C) = \min(2d_1, d_2)$. □

For $x \in \mathbb{F}_2^n$, the bitwise complement of x is the vector

$$\bar{x} := x + 1^n = (x_1 + 1, \cdots, x_n + 1) \in \mathbb{F}_2^n.$$

COROLLARY 5.1. *If C is a binary $[n, k, d]$-code, then $\tilde{C} = \{(c, c), (c, \bar{c}) \mid c \in C\}$ is a binary $[2n, k + 1, \min(2d, n)]$-code.*

PROOF. Let $C_1 = C$, $C_2 = \{0^n, 1^n\}$ (which is a binary $[n, 1, n]$-code), and apply the theorem above. □

THEOREM 5.3. *Suppose $d \geq 1$ is odd.*

(1) The existence of an $(n, M, d)_2$-code is equivalent to the existence of an $(n + 1, M, d + 1)_2$-code.

(2) The existence of an $[n, k, d]_2$-code is equivalent to the existence of an $[n + 1, k, d + 1]_2$-code.

PROOF. (2) follows from the proof of (1).

On one hand suppose C is an (n, M, d)-binary code. Let $C' = \{c' = (c_1, \cdots, c_n, c_1 + \cdots + c_n) \mid c = (c_1, \cdots, c_n) \in C\}$. Then $d(x', y') \geq d(x, y) \geq d + 1$ for any $x \neq y \in C$. Moreover, if $d(x, y) = d$, then x and y differ in d positions, so the sum of all coordinates of x and y are not equal, hence $d(x', y') = d + 1$. Thus C' is an $(n + 1, M, d + 1)_2$-code.

On the other hand, let C' be an $(n+1, M, d+1)$-code. Suppose $x', y' \in C'$ and $d(x', y') = d + 1$. Suppose $x'_i \neq y'_i$. Deleting the i-th coordinate of all codewords of C, the elements obtained form a new code C. It is easy to check C is an $(n, M, d)_2$ code. □

2. Reed Muller Codes

2.1. Reed-Muller Code of first degree.

Definition 5.1. For $m \geq 1$, the Reed-Muller Code $R(1, m)$ of the first degree is the binary linear code defined inductively as follows:

(1) $R(1,1) := \mathbb{F}_2^2 = \{00, 01, 10, 11\}$;
(2) once $R(1, m)$ is defined, then $R(1, m+1) := \{(c, c), (c, \bar{c}) \mid c \in R(1, m)\}$.

Proposition 5.2. *The Reed-Muller Code $R(1, m)$ is a linear $[2^m, m+1, 2^{m-1}]_2$-code, in which every codeword except 0^n and 1^n has weight 2^{m-1}.*

Proof. For the parameters, we prove by induction on m. The case $m = 1$ is trivial. The induction step just follows from Corollary 5.1.

For the weights of non-trivial codewords, we also use induction on m. The case $m = 1$ is trivial. Now assume every codeword except 0^n and 1^n in $R(1, m)$ has weight 2^{m-1}. Let $c \in R(1, m+1)$, $c \neq 0^{2^{m+1}}$ or $1^{2^{m+1}}$. Then

(a) either $c = (u, u)$ with $u \in R(1, m)$. In this case, $u \neq 0^n$ or 1^n, so $\mathrm{wt}(u) = 2^{m-1}$ and $\mathrm{wt}(c) = 2\mathrm{wt}(u) = 2^m$.
(b) or $c = (u, \bar{u})$ with $u \in R(1, m)$. In this case, note that $\bar{0}^n = 1^n$, then we always have $\mathrm{wt}(c) = \mathrm{wt}(u) + \mathrm{wt}(\bar{u}) = 2^m$.

This completes the proof. □

Proposition 5.3. *The following facts hold:*

(1) *The matrix $G_1 = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ is a generator matrix for $R(1, 1)$.*
(2) *If G_m is a generator matrix for $R(1, m)$, then $G_{m+1} = \left(\begin{smallmatrix} G_m & G_m \\ 0^n & 1^n \end{smallmatrix}\right)$ is a generator matrix for $R(1, m+1)$.*

Proof. Easy, just use the fact $\bar{x} = x + 1^n$. □

2.2. Boolean functions of m variables.

Definition 5.2. Let $m \geq 1$. A Boolean function is a map

$$f : \mathbb{F}_2^m \to \mathbb{F}_2,\ (a_1, \cdots, a_m) \mapsto f(a_1, \cdots, a_m).$$

Let B_m be the set of all Boolean functions of m-variables.

We know $|B_m| = 2^{2^m}$. Moreover,

Theorem 5.4. *Every $f \in B_m$ can be written uniquely as*

$$f(x_1, \cdots, x_m) = \sum_I c_I x_I, \tag{5.1}$$

where I runs over all subsets of $[m] = \{1, \cdots, m\}$, $c_I \in \mathbb{F}_2$, and

$$x_I = \begin{cases} 1, & \text{if } I = \emptyset, \\ \prod_{i \in I} x_i, & \text{if } I \neq \emptyset. \end{cases}$$

This form is called the algebraic normal form of f. Equivalently B_m is an $\mathbb{F}_2$-vector space with a basis $\{x_I \mid I \subseteq [m]\}$.

PROOF. We first show that

$$f(x_1,\cdots,x_m)=\sum_{(a_1,\cdots,a_m)\in\mathbb{F}_2^m} f(a_1,\cdots,a_m)\prod_{i=1}^m(x_i+a_i+1). \tag{5.2}$$

We know that

$$\prod_{i=1}^m(x_i+a_i+1)=\begin{cases}1, & \text{if } x_i=a_i \text{ for all } i;\\ 0, & \text{if otherwise.}\end{cases}$$

Hence the two Boolean functions in both sides of (5.2) take the same value for any $(a_1,\cdots,a_m)\in\mathbb{F}_2^m$, hence they are the same function. Expand the left hand side of (5.2), then f has the form of (5.1).

Since $|B_m|=2^{2^m}$ and the number of functions in the right hand side of (5.1) is at most 2^{2^m}, the form in (5.1) must be unique. □

DEFINITION 5.3. For a Boolean function f, its algebraic normal form (ANF) is the representation given by (5.1), and its (algebraic) degree is the number of variables in the longest item of the algebraic normal form, i.e.,

$$\deg(f):=\max\{|I|\mid c_I=1\}.$$

2.3. General Binary Reed-Muller Code $\mathrm{RM}(r,m)$**.** Fix m and let $n=2^m$. For $u=(u_1,\cdots,u_m)^T\in\mathbb{F}_2^{m\times1}$ and $f\in B_m$, we let $f(u)=f(u_1,\cdots,u_m)$. Then

$$\begin{pmatrix}x_1(u)\\ \vdots\\ x_m(u)\end{pmatrix}=\begin{pmatrix}u_1\\ \vdots\\ u_m\end{pmatrix}=u. \tag{5.3}$$

Write $\mathbb{F}_2^{m\times1}=\{v_1,\cdots,v_n\}$. For $f\in B_m$, set

$$c_f:=(f(v_1),\cdots,f(v_n))\in\mathbb{F}_2^n.$$

LEMMA 5.1. *The map $B_m\to\mathbb{F}_2^n$, $f\mapsto c_f$ is an isomorphism of $\mathbb{F}_2$-vector spaces. Moreover, if we define the multiplication in $\mathbb{F}_2^n$ as the component-wise multiplication, then*

$$c_{fg}=(fg(v_1),\cdots,fg(v_n))=(f(v_1)g(v_1),\cdots,f(v_n)g(v_n))=c_fc_g.$$

DEFINITION 5.4. The binary Reed-Muller Code of r-th degree is the code

$$\mathrm{RM}(r,m):=\{c_f\in\mathbb{F}_2^n\mid f\in B_m,\deg f\le r\}.$$

We first have some quick facts about $\mathrm{RM}(r,m)$:

- its size $n=2^m$;
- its dimension $k(r,m)=\sum_{i=0}^r\binom{m}{i}$. In particular, $k(1,m)=m+1$.

For $f\in B_m$, $a=0$ or 1, set

$$N_m(f=a):=\#\{u\in\mathbb{F}_2^{m\times1}\mid f(u)=a\}.$$

Then we have

$$N_m(f=0)+N_m(f=1)=2^m=n, \tag{5.4}$$

$$\mathrm{wt}(c_f)=N_m(f=1). \tag{5.5}$$

Lemma 5.2. *For* $\deg f = r$, *then* $N_m(f=1)\geq 2^{m-r}$. *And if* $r\leq m-1$, *then* $N_m(f=1)$ *is even.*

Proof. If $|I|=r\leq m-1$, then $x_I(u)=1$ for $u=(u_1,\cdots,u_m)^T$ if and only if $u_i=1$ for all $i\in I$, thus $N_m(x_I=1)=2^{m-r}=\mathrm{wt}(c_{x_I})$ is even, i.e., the sum of all components of c_{x_I} is 0. By (5.1), then $\mathrm{wt}(c_f)=N_m(f=1)$ is even if $\deg f\leq m-1$.

Now we show $N_m(f=1)\geq 2^{m-r}$ by induction on m. The base case $m=1$ is trivial. For the inductive step, suppose this is true for $m-1$ and $m\geq 2$. We discuss the value $N_m(f=1)$ according to r:

(a) If $r=0$, then $f=1$, so $N_m(f=1)=2^m$.

(b) If $r=m$, then f is not the zero polynomial, so $N_m(f=1)\geq 1$.

(c) If $1\leq r\leq m-1$, then f can be uniquely written as:

$$f(x_1,\cdots,x_m)=x_mh(x_1,\cdots,x_{m-1})+g(x_1,\cdots,x_{m-1})$$

where $g,h\in B_{m-1}$, $\deg g\leq r$ and $\deg h\leq r-1$. By induction, if $h=0$, then

$$N_m(f=1)=2N_{m-1}(g=1)\geq 2^{m-1-r+1}=2^{m-r};$$

if $h\neq 0$, then

$$\begin{aligned}N_m(f=1)=&N_{m-1}(h=1)+2N_{m-1}(g=1)N_{m-1}(h=0)\\ \geq&N_{m-1}(h=1)\geq 2^{(m-1)-(r-1)}=2^{m-r}.\end{aligned}$$

In conclusion, $N_m(f=1)\geq 2^{m-r}$. □

Theorem 5.5. *Let* $m\geq 1$ *and* $0\leq r\leq m$.

(1) $\mathrm{RM}(r,m)$ *is a* $[2^m,k(r,m),2^{m-r}]$*-code.*

(2) *If* $0\leq r\leq m-1$, *then* $\mathrm{RM}(r,m)^\perp=\mathrm{RM}(m-r-1,m)$.

Proof. (1): we have shown that $n=2^m$, $k=k(r,m)$ and $d\geq 2^{m-r}$. For $f=\prod_{i=1}^r x_i$, $\mathrm{wt}(c_f)=2^{m-r}$, hence $d=2^{m-r}$.

(2): we know $\mathrm{RM}(r,m)=\{c_f\mid f\in B_m,\deg f\leq r\}$ and $\mathrm{RM}(m-r-1,m)=\{c_g\mid g\in B_m,\deg g\leq m-r-1\}$. If $\deg(f)\leq r$ and $\deg(g)\leq m-r-1$, then $\deg fg\leq m-1$ and $N_m(fg=1)$ is even by the above Lemma. Hence

$$(c_f,c_g)=\sum_{a\in\mathbb{F}_2^{m\times 1}} f(a)g(a)=N_m(fg=1)=0\in\mathbb{F}_2 \ \Rightarrow\ c_f\perp c_g.$$

Hence $\mathrm{RM}(r,m)\perp\mathrm{RM}(m-r-1,m)$. Check the dimensions we have

$$k(r,m)+k(m-r-1,m)=2^m=n.$$

So $\mathrm{RM}(r,m)^\perp=\mathrm{RM}(m-r-1,m)$. □

EXAMPLE 5.1. The Reed-Muller code $\mathrm{RM}(0,m)=\{0^n,1^n\}$ is a $[2^m,1,2^m]$-code with a generator matrix $1^n=(1,\cdots,1)$, hence 1^n is a parity check matrix for the dual code $\mathrm{RM}(m-1,m)$. That is,

$$\mathrm{RM}(m-1,m)=\{(c_1,\cdots,c_n)\mid\sum_{i=1}^n c_i=0\in\mathbb{F}_2\}$$

is a $[2^m,2^m-1,2]$-code.

2.4. The code $\mathrm{RM}(1,m)$**.** We now study the case $r=1$. By definition and Theorem 5.5, $\mathrm{RM}(1,m)$ has a basis $\{c_1,\ c_{x_i}\mid 1\le i\le m\}$ and is a $[2^m,m+1,2^{m-1}]_2$-code. By (5.3), $\mathrm{RM}(1,m)$ has a generator matrix of size $(m+1)\times n$:

$$G_m=\begin{pmatrix}c_1\\c_{x_1}\\\vdots\\c_{x_m}\end{pmatrix}=\begin{pmatrix}1&1&\cdots&1\\v_1&v_2&\cdots&v_n\end{pmatrix}.$$

Up to equivalence, the order of $v_i\in\mathbb{F}_2^{m\times1}$ is not essential. For all m, we take the lexicographic order for the vectors in $\mathbb{F}_2^{m\times1}$. If $1\le a\le 2^m=n$, suppose the binary expansion of $a-1$ is

$$a-1=\sum_{i=1}^m a_i2^{i-1},\ a_i\in\{0,1\}\subseteq\mathbb{Z}.$$

Then let $v_a=(a_1,a_2,\cdots,a_m)^T\in\mathbb{F}_2^{m\times1}$. In particular, $v_1=(0,\cdots,0)^T$, $v_n=(1,\cdots,1)^T$ and $v_{a+2^{m-1}}-v_a=(0,\cdots,0,1)^T$ if $1\le a\le 2^{m-1}=\frac{n}{2}$. By abuse of notation, we let $\mathrm{RM}(1,m)$ be the code defined by this order for all m. Then $G_1=\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)$ and for $m\ge2$,

$$G_m=\begin{pmatrix}G_{m-1}&G_{m-1}\\0^{\frac{n}{2}}&1^{\frac{n}{2}}\end{pmatrix}.$$

G_m is also a generator matrix for $R(1,m)$ by Proposition 5.3. Thus

PROPOSITION 5.4. $R(1,m)=\mathrm{RM}(1,m)$.

Next, we divide G_m as follows:

$$G_m=\begin{pmatrix}1&1^{n-1}\\(0^{n-1})^T&G'_m\end{pmatrix}.$$

Then $G'_m=(v_2,\cdots v_n)$ is an $m\times(2^m-1)$ matrix and $\mathbb{F}_2^{m\times1}\setminus\{0\}=\{v_2,\cdots v_n\}$. Thus G'_m is a generator matrix for the linear code

$$\mathrm{RM}(1,m)'=\{(c_2,\cdots,c_n)\in\mathbb{F}_2^{n-1}\mid(0,c_2,\cdots,c_n)\in\mathrm{RM}(1,m)\}.$$

But it is also a generator matrix for the Walsh-Hadamard Code $\mathrm{WH}_m=H(2,m)^\perp$, hence

PROPOSITION 5.5. $\mathrm{RM}(1,m)'=\mathrm{WH}_m=H(2,m)^\perp$ *is a linear* $[2^m-1,m,2^{m-1}]$*-code over* $\mathbb{F}_2$ *whose nonzero codewords are all of weight* 2^{m-1}..

We can also compute the distance of $\mathrm{RM}(1,m)'$ directly. For any codeword $c_f \in \mathrm{RM}(1,m)$, $f = \sum_{i=0}^{m} b_i x_i$. If $f = 0$, then $\mathrm{wt}(c_f) = 0$; if $f = 1$, then $\mathrm{wt}(c_f) = 2^m$. Otherwise suppose $b_i = 1$, then $f(v) + f(v + e_i) = 1$ for any $v \in \mathbb{F}_2^{m\times 1}$, this implies that $\mathrm{wt}(c_f) = 2^{m-1}$. Now $c_f = (0, c_2, \cdots, c_n)$ if and only if $b_0 = 0$. For this c_f, if not 0, then $\mathrm{wt}(c_f) = 2^{m-1}$ and $\mathrm{wt}(c_2, \cdots, c_n) = 2^{m-1}$. Hence $\mathrm{RM}(1,m)'$ is a $[2^m - 1, m, 2^{m-1}]$-code whose nonzero codewords are all of weight 2^{m-1}.

CHAPTER 6

Weight Enumerators and the MacWilliams Theorem

1. MacWilliams Identity

Definition 6.1. For a linear $[n, k, d]$-code C over $\mathbb{F}_q$, if $0 \le i \le n$, let

$$A_i = A_i(C) := \#\{c \in C \mid \mathrm{wt}(c) = i\}.$$

The weight enumerator of C is the polynomial

$$W_C(x, y) := \sum_{c \in C} x^{n-\mathrm{wt}(c)} y^{\mathrm{wt}(c)} = \sum_{i=0}^{n} A_i x^{n-i} y^i \in \mathbb{Z}[x, y].$$

Lemma 6.1. *We have the following facts:*

1. $A_0 = 1$, $A_i = 0$ *if* $0 < i < d$ *and* $A_d > 0$, *i.e.,* $d = \min\{i \mid i > 0,\ A_i > 0\}$.
2. $\sum_{i=0}^{n} A_i = |C| = q^k$.
3. $W_C(x, y)$ *is homogeneous of degree* n, $W_C(1, 0) = A_0 = 1$ *and* $W_C(1, 1) = q^k$.

As a consequence, the weight enumerator determines the parameters of a linear code.

By definition, equivalent codes have the same weight enumerator.

Theorem 6.1 (MacWilliams Identity). *Let* C *be a linear* $[n, k, d]_q$*-code and* $C^\perp$ *be its dual. Then*

$$W_{C^\perp}(x, y) = q^{-k} W_C(x + (q-1)y, x - y). \tag{6.1}$$

Lemma 6.2. *For* $q = p^m$, Tr *is the trace map*

$$\mathrm{Tr} = \mathrm{Tr}_{q/p} : \mathbb{F}_q \to \mathbb{F}_p,\ a \mapsto a + a^p + \cdots + a^{p^{m-1}}$$

is an $\mathbb{F}_p$*-linear surjective map such that*

$$\sum_{x \in \mathbb{F}_q} \zeta^{\mathrm{Tr}(ax)} = \begin{cases} q & \text{if } a = 0; \\ 0 & \text{if } a \in \mathbb{F}_q^\times, \end{cases}$$

where $\zeta = \zeta_p = \exp(\frac{2\pi i}{p})$ *is a primitive* p*-th root of unity. Note that* ζ^b *is well-defined if* $b \in \mathbb{F}_p$.

Proof. The linearity is clear. Now if Tr is not surjective, it must be the zero map, but the equation $x + x^p + \cdots + x^{p^{m-1}} = 0$ has at most p^{m-1} roots, contradiction. Thus the kernel of Tr is $(m-1)$-dimensional, and for each $b \in \mathbb{F}_p$, the number of $x \in \mathbb{F}_q$ such that $\mathrm{Tr}(x) = b$ is p^{m-1}.

For the last identity, the case $a = 0$ is trivial. If $a \neq 0$, one can replace a by 1, and the left hand side of the identity is $p^{m-1} \sum_{b\in\mathbb{F}_p} \zeta^b = 0$. □

Proof. Let $A_i^\perp = A_i(C^\perp) = |\{c \in C^\perp \mid \mathrm{wt}(c) = i\}|$. Then $W_C(x, y) = \sum_{i=0}^n A_i x^{n-i} y^i$ and $W_{C^\perp}(x, y) = \sum_{i=0}^n A_i^\perp x^{n-i} y^i$.

For $u = (u_1, \cdots, u_n) \in \mathbb{F}_q^n$, set

$$\begin{aligned} g_u(z) :=& \sum_{(v_1,\cdots,v_n)\in\mathbb{F}_q} \zeta^{\mathrm{Tr}(u_1v_1+\cdots+u_nv_n)} z^{\mathrm{wt}(v_1)+\cdots+\mathrm{wt}(v_n)} \\ =& \prod_{i=1}^n \sum_{v_i\in\mathbb{F}_q} \zeta^{\mathrm{Tr}(u_iv_i)} z^{\mathrm{wt}(v_i)}. \end{aligned}$$

We compute $\sum_{u\in C} g_u(z)$ in two ways. On one hand, if $u_i = 0$, i.e. $\mathrm{wt}(u_i) = 0$, then

$$\sum_{v_i\in\mathbb{F}_q} \zeta^{\mathrm{Tr}(u_iv_i)} z^{\mathrm{wt}(v_i)} = 1 + (q-1)z;$$

if $u_i \neq 0$, i.e. $\mathrm{wt}(u_i) = 1$, then

$$\sum_{v_i\in\mathbb{F}_q} \zeta^{\mathrm{Tr}(u_iv_i)} z^{\mathrm{wt}(v_i)} = 1 - z.$$

Hence

$$g_u(z) = (1-z)^{\mathrm{wt}(u)}(1 + (q-1)z)^{n-\mathrm{wt}(u)}$$

and

$$\begin{aligned} \sum_{u\in C} g_u(z) =& \sum_{u\in C} (1-z)^{\mathrm{wt}(u)}(1+(q-1)z)^{n-\mathrm{wt}(u)} \\ =& \sum_{i=0}^n A_i(1+(q-1)z)^{n-i}(1-z)^i. \end{aligned} \tag{6.2}$$

On the other hand, $u \cdot v = u_1v_1 + \cdots + u_nv_n$ is the inner product of u and v. We have

$$\sum_{u\in C} g_u(z) = \sum_{u\in C}\sum_{v\in\mathbb{F}_q^n} \zeta^{\mathrm{Tr}(u\cdot v)} z^{\mathrm{wt}(v)} = \sum_{v\in\mathbb{F}_q^n} z^{\mathrm{wt}(v)} \sum_{u\in C} \zeta^{\mathrm{Tr}(u\cdot v)}.$$

We claim that:

$$\sum_{u\in C} \zeta^{\mathrm{Tr}(u\cdot v)} = \begin{cases} q^k & \text{if } v \in C^\perp; \\ 0 & \text{otherwise.} \end{cases}$$

For $v \in C^\perp$, this is clear. If $v \notin C^\perp$, there exists $u' \in C$ such that $u' \cdot v = \alpha \neq 0$. Let $\beta \in \mathbb{F}_q$ such that $\mathrm{Tr}(\beta) = 1$. Then $\tilde{u} = \alpha^{-1}\beta u' \in C$ satisfies

$\tilde{u} \cdot v = \beta$. Hence

$$\sum_{u\in C} \zeta^{\mathrm{Tr}(u\cdot v)} = \sum_{u\in C} \zeta^{\mathrm{Tr}((u+\tilde{u})\cdot v)} = \zeta \sum_{u\in C} \zeta^{\mathrm{Tr}(u\cdot v)},$$

sand $\sum_{u\in C} \zeta^{\mathrm{Tr}(u\cdot v)} = 0$. The claim is proved. Thus

$$\sum_{u\in C} g_u(z) = q^k \sum_{i=0}^{n} A_i^{\perp} z^i. \tag{6.3}$$

By (6.2) and (6.3),

$$\sum_{i=0}^{n} A_i (1+(q-1)z)^{n-i}(1-z)^i = q^k \sum_{i=0}^{n} A_i^{\perp} z^i. \tag{6.4}$$

Take $z = \frac{y}{x}$ in (6.4), we get the MacWilliams Identity. □

1.1. Application of the MacWilliams Identity.

Example 6.1. The repetition code $C = \{(c, c, \cdots, c) \mid c \in \mathbb{F}_q\}$ is an $[n, 1, n]_q$-code, its weight enumerator $W_c(x, y) = x^n + (q-1)y^n$.

The dual code $C^\perp = \{c = (c_1, \cdots, c_n) \mid c_1 + \cdots + c_n = 0\}$, a code of dimension $n-1$, as it has a parity check matrix $(1, \cdots, 1) = 1^n$. By MacWilliams Identity,

$$W_{C^\perp}(x, y) = \frac{1}{q}[(x + (q-1)y)^n + (q-1)(x-y)^n].$$

Since $A_1^\perp = 0$ and $A_2^\perp = (q-1)\binom{n}{2} \neq 0$, $C^\perp$ is $[n, n-1, 2]_q$-code.

Example 6.2. The Reed-Muller Code RM$(1, m)$ is $[2^m, m+1, 2^{m-1}]_q$-code, in which every codeword except 0^{2^m} and 1^{2^m} has weight 2^{m-1}. Let $n = 2^m$. Then

$$W_C(x, y) = x^n + (2^{m+1} - 2)x^{n/2}y^{n/2} + y^n.$$

Then the weight enumerator of its dual code $C^\perp = \mathrm{RM}(m-2, m)$ is

$$W_{C^\perp}(x, y) = \frac{1}{2^{m+1}}(x+y)^n + (2^{m+1} - 2)(x^2 - y^2)^{n/2} + (x-y)^n.$$

For $m \geq 2$, we again obtain the fact that RM$(m-2, m)$ is $[2^m, 2^m - m - 1, 4]_2$-code.

Example 6.3. We know the code $C' = \mathrm{RM}(1, m)' = \mathrm{WH}_m = H(2, m)^\perp$ is a linear $[2^m - 1, m, 2^{m-1}]_2$-code whose nonzero codewords are all of weight 2^{m-1}. Let $n = 2^m$. Then

$$W_{C'}(x, y) = x^{n-1} + (2^m - 1)x^{n/2-1}y^{n/2}.$$

Thus for the binary Hamming Code $H = H(2, m)$, we have:

$$W_H(x, y) = 2^{-m}[(x+y)^{n-1} + (2^m - 1)(x^2 - y^2)^{n/2-1}(x-y)].$$

By computation, $A_1(H) = A_2(H) = 0$ and $A_3(H) = \frac{1}{6}(n-1)(n-2)$, hence $d(H) = 3$ and we recover the fact that $H(2,m)$ is a linear $[2^m-1, 2^m-1-m, 3]_2$-code.

THEOREM 6.2. *Suppose C is a linear MDS $[n,k,d]_q$-code, i.e. $d = n-k+1$. Then for all w that $d \leq w \leq n$,*

$$A_w = \binom{n}{w}(q-1)\sum_{j=0}^{w-1}(-1)^j\binom{w-1}{j}q^{w-1-j}. \tag{6.5}$$

Thus the weight enumerator of an MDS code is uniquely determined by its parameters n, k and q.

PROOF. Note that $C^\perp$ is $[n, n-k, k+1]$-code. Let $y = 1$ in the MacWilliams Identity, then $W_C(x,1) = q^{k-n}W_{C^\perp}(x+q-1, x-1)$ and

$$\sum_{i=0}^{n} A_i x^{n-i} = q^{k-n}\sum_{i=0}^{n} A_i^\perp (x+q-1)^{n-i}(x-1)^i. \tag{6.6}$$

The left hand side of (6.6) equals

$$\begin{aligned}\sum_{i=0}^{n} A_i((x-1)+1)^{n-i} &= \sum_{i=0}^{n} A_i \sum_{r=0}^{n-i}(x-1)^r\binom{n-i}{r}\\ &= \sum_{r=0}^{n}(x-1)^r\sum_{i=0}^{n-r}A_i\binom{n-i}{r}.\end{aligned}$$

The right hand side of (6.6) equals

$$\begin{aligned}&\frac{1}{q^{n-k}}\sum_{i=0}^{n}A_i^\perp\sum_{\ell=0}^{n-i}\binom{n-i}{\ell}(x-1)^{\ell+i}q^{n-\ell-i}\\ &= \frac{1}{q^{n-k}}\sum_{i=0}^{n}A_i^\perp\sum_{r=i}^{n}\binom{n-i}{r-i}(x-1)^r q^{n-r}\\ &= \sum_{r=0}^{n}(x-1)^r q^{n-r}\sum_{i=0}^{r}\binom{n-i}{r-i}A_i^\perp.\end{aligned}$$

Then for $0 \leq r \leq n$, we have

$$\frac{1}{q^k}\sum_{i=0}^{n-r}\binom{n-i}{r}A_i = \frac{1}{q^r}\sum_{i=0}^{r}\binom{n-i}{r-i}A_i^\perp. \tag{6.7}$$

Note that $A_i = 0$ if $1 \leq i \leq n-k$ and $A_i^\perp = 0$ if $1 \leq i \leq k$, so if $r \leq k$, (6.7) becomes

$$\frac{1}{q^k}\binom{n}{r} + \frac{1}{q^k}\sum_{i=n-k+1}^{n-r}\binom{n-i}{r}A_i = q^{-r}\binom{n}{r}. \tag{6.8}$$

Take $r = k - 1$ in (6.8), we have $A_d = A_{n-k+1} = \binom{n}{d}(q-1)$, (6.5) is satisfied in this case. Let $w = d + \ell$, this means that the theorem is true for $\ell = 0$.

For general ℓ, take $r = k - \ell - 2$ in (6.8) where $w = d + \ell$. For $w \geq n-k+1$, we have

$$A_w = (q^{k+w-n} - 1)\binom{n}{w} - \sum_{i=n-k+1}^{w-1} \binom{n-i}{w-i} A_i.$$

Then the general case follows by induction. □

CHAPTER 7

Sequences over finite fields

1. Sequences and Power Series over Finite Fields

1.1. Periodic sequences. In this section we shall study sequences and power series over finite fields. First observe that a sequence $\mathbf{a} = (a_0, a_1, \cdots)$ over $\mathbb{F}_q$ decides and is determined by a power series $\mathbf{a}(x) = \sum_{n\geq 0} a_n x^n$. By abuse of notations we regard them as the same object.

Let $\mathbb{F}_q[[x]]$ be the set of power series over $\mathbb{F}_q$. It is a commutative ring: if $a(x) = \sum_{n\geq 0} a_n x^n$ and $b(x) = \sum_{n\geq 0} b_n x^n$, then $a(x)+b(x) = \sum_n (a_n+b_n)x^n$ and $a(x)\cdot b(x) = \sum_n m_n x^n$ where $m_n = \sum_{i+j=n} a_i b_j$.

Lemma 7.1. *A power series $a(x) \in \mathbb{F}_q[[x]]$ is invertible if and only if $a_0 \in \mathbb{F}_q^\times$, i.e., a_0 is invertible.*

Proof. If $a(x)b(x) = 1$, then $a_0 b_0 = 1$ and hence $a_0 \in \mathbb{F}_q^\times$. On the other hand, supoose $a(x) = \sum_{n\geq 0} a_n x^n$ and a_0 is invertible. Set $b_0 = a_0^{-1}$, and for $n \geq 1$, set b_n inductively by

$$b_n = -a_0^{-1}(a_1 b_{n-1} + \cdots + a_n b_0).$$

Then $b(x) = \sum_n b_n x^n$ is the inverse of $a(x)$. $\square$

Definition 7.1. A sequence $\mathbf{a} = (a_0, a_1, \cdots, a_n, \cdots)$ is called periodic (resp. purely periodic) if there exists an integer $T > 0$ such that $a_{i+T} = a_i$ for $i \geq N$ for some $N \geq 0$ (resp. for all $i \geq 0$), and T is call a period of a. In this case, we also write $\mathbf{a} = (a_0, \cdots, a_{N-1}, \dot{a}_N, \cdots, \dot{a}_{N+T-1})$.

The smallest period of a periodic sequence a is denoted as $\operatorname{per}(a)$.

Clearly every period of a periodic sequence a is a multiple of $\operatorname{per}(a)$.

Definition 7.2. For a non-constant polynomial $f(x) \in \mathbb{F}_q[x]$ such that $f(0) \neq 0$, if there exists $k > 0$ such that $f(x) \mid x^k - 1$, then $\operatorname{per}(f)$ is defined to be the smallest such k.

Of course if $f(0) = 0$, then $x \mid f(x)$ and $\gcd(x, x^n - 1) = 1$, so there exists no k such that $f(x) \mid x^k - 1$.

Lemma 7.2. *Suppose $m, n > 0$ are integers. Suppose F is a field. Then in the ring $F[x]$, one has*

$$\gcd(x^m - 1, x^n - 1) = x^{\gcd(m,n)} - 1.$$

In particular, for $f(x) \in \mathbb{F}_q[x]$, if $\operatorname{per}(f)$ exists, then for any $n > 0$ that $f(x) \mid x^n - 1$, one must have $\operatorname{per}(f) \mid n$.

Proof. Suppose $m > n$ and $m = qn + r$ such that $0 \le r < n$. Then

$$x^m - 1 = x^{qn+r} - x^r + x^r - 1 = x^r(x^{qn} - 1) + (x^r - 1).$$

Then if $d(x) \mid x^m - 1$ and $d(x) \mid x^n - 1 \mid x^{qn} - 1$, one must have $d(x) \mid x^r - 1$. This implies $d(x) \mid x^{\gcd(m,n)} - 1$. On the other hand, $x^{\gcd(m,n)} - 1$ is clearly a common divisor of $x^m - 1$ and $x^n - 1$. □

Proposition 7.1. *Suppose $f(x) \in \mathbb{F}_q[x]$ is non-constant and $f(0) \neq 0$.*

(1) *If $f(x) = p(x) \neq x$ is irreducible of degree n and α is a root of $p(x)$, then $\mathrm{per}(p(x))$ is the order of α in the group $\mathbb{F}_{q^n}^\times$ and hence is a factor of $q^n - 1$. In particular, $\mathrm{per}(p(x)) = q^n - 1$ if and only if $p(x)$ is a primitive polynomial.*
(2) *If $f(x) = p(x)^m$ with $p^{t-1} < m \le p^t$, then $\mathrm{per}(f(x)) = p^t \mathrm{per}(p(x))$.*
(3) *If $f(x) = f_1(x)f_2(x)$ and $\gcd(f_1(x), f_2(x)) = 1$, then $\mathrm{per}(f) = [\mathrm{per}(f_1), \mathrm{per}(f_2)]$.*

In all, $\mathrm{per}(f)$ is well defined.

Proof. (1) In this case, $p(x) \mid x^k - 1$ if and only if $\alpha^k = 1$, hence $\mathrm{per}(p(x))$ is nothing but the order of α.

(2) Let $\mathrm{per}(p(x)) = s$. Then $p(x) \| x^s - 1$ as $s \mid q^n - 1$ and $x^s - 1$ is separable. Then $p(x)^{p^t} \| (x^s - 1)^{p^t} = x^{p^t s} - 1$ for every integer $t \ge 0$. Hence $p(x)^m \nmid x^{p^{t-1}s} - 1$ if $p^{t-1} < m \le p^t$.

We now prove the statement by induction on t. If $p^{t-1} < m \le p^t$, then $p(x)^m \mid p(x)^{p^t} \mid (x^s - 1)^{p^t} = x^{p^s t} - 1$ and $\mathrm{per}(p(x)^m) \mid p^t s$. On the other hand, by induction $p^{t-1}s = \mathrm{per}(p(x)^{p^{t-1}}) \mid \mathrm{per}(p(x)^m)$. But $\mathrm{per}(p(x)^m) \neq p^{t-1}s$ by the argument in the last paragraph, hence $\mathrm{per}(p(x)^m) = p^t s$.

(3) Let $N_1 = \mathrm{per}(f_1)$, $N_2 = \mathrm{per}(f_2)$ and $N = [N_1, N_2]$. Then $f_i(x) \mid x^{N_i} - 1 \mid x^N - 1$ and hence $f(x) \mid x^N - 1$. Then $\mathrm{per}(f)$ exists and is a factor of N. On the other hand, $f_i(x) \mid f(x) \mid x^{\mathrm{per}(f)} - 1$, hence $N_i \mid \mathrm{per}(f)$ by Lemma 7.2 and $N \mid \mathrm{per}(f)$. Hence $N = \mathrm{per}(f)$. □

Theorem 7.1. *A sequence $\mathbf{a}$ is purely periodic if and only if the power series $\mathbf{a}(x)$ is a proper fraction, that is,*

$$\mathbf{a}(x) = \sum_n a_n x^n = \frac{g(x)}{f(x)}, \quad f(x), g(x) \in \mathbb{F}_q[x], \ \deg g < \deg f, \ f(0) \neq 0.$$

If moreover, $\frac{g(x)}{f(x)}$ is a reduced form, i.e. $\gcd(f(x), g(x)) = 1$, then $\mathrm{per}(\mathbf{a}) = \mathrm{per}(f)$.

Proof. On one hand, if $\mathbf{a}$ is a purely periodic sequence such that $\mathrm{per}(\mathbf{a}) = \ell$, then

$$\mathbf{a}(x) = (\sum_{i=0}^{\ell-1} a_i x^i)(\sum_{n=0}^{\infty} x^{n\ell}) = \frac{\sum_{i=0}^{\ell-1} a_i x^i}{1 - x^\ell}.$$

On the other hand, let $\ell = \mathrm{per}(f)$, then there exists $h(x)$ such that $f(x)h(x) = 1 - x^\ell$. Let $g(x)h(x) = \sum_{i=0}^{\ell-1} a_i x^i$. Then

$$\frac{g(x)}{f(x)} = \frac{g(x)h(x)}{1-x^\ell} = \sum_{n=0}^{\infty} a_n x^n$$

satisfying $a_{n+\ell} = a_n$ for $n \geq 0$. Hence $\mathbf{a} = (a_n)$ is purely periodic.

We now prove the second statement. If $\mathrm{per}(\mathbf{a}) = \ell$, then

$$\frac{g(x)}{f(x)} = \frac{\sum_{i=0}^{\ell-1} a_i x^i}{1-x^\ell} \implies f(x)\sum_{i=0}^{\ell-1} a_i x^i = g(x)(1-x^\ell).$$

Since $\gcd(f(x), g(x)) = 1$, $f(x) \mid 1 - x^\ell$. Hence $\mathrm{per}(f) \mid \ell$. On the other hand, let $\mathrm{per}(f) = k$. Then $\mathbf{a}(x) = \frac{g(x)}{f(x)} = \frac{\alpha(x)}{1-x^k}$. Write $\alpha(x) = \sum_{i=0}^{k-1} \alpha_i x^i$, then $\mathbf{a} = (\dot{\alpha}_0, \cdots, \dot{\alpha}_{k-1})$ and hence $\ell \leq k$. That is, $\mathrm{per}(f) = \mathrm{per}(\mathbf{a})$. $\square$

1.2. Decomposing rational fractions.

LEMMA 7.3. *Let $f(x) = f_1(x) \cdots f_s(x)$ where the f_i's are pairwise coprime. Then every proper fraction $\frac{g(x)}{f(x)}$ can be written uniquely as a sum of proper fractions*

$$\frac{g(x)}{f(x)} = \sum_{i=1}^{s} \frac{g_i(x)}{f_i(x)}, \ \deg(g_i) < \deg(f_i), \ i = 1, \cdots, s.$$

PROOF. For $i = 1, \cdots, s$, let $F_i(x) = \frac{f(x)}{f_i(x)}$. Then $\gcd(F_i) = 1$. By Bezout's Theorem, there exists $u_i(x) \in \mathbb{F}_q[x]$ such that $\sum_{i=1}^{s} u_i F_i = 1$. Then we have

$$\frac{g(x)}{f(x)} = \sum_{i=1}^{s} \frac{u_i(x)g(x)}{f_i(x)}.$$

Let $u_i(x)g(x) = g_i(x) + q_i(x)f_i(x)$ such that $\deg g_i < \deg f_i$, then

$$\frac{g(x)}{f(x)} = \sum_{i=1}^{s} \frac{g_i(x)}{f_i(x)} + \sum_{i=1}^{s} q_i(x).$$

Multiplying $f(x)$ to both sides and comparing their degrees, we get $\sum_{i=1}^{s} q_i(x) = 0$ and $\frac{g(x)}{f(x)} = \sum_{i=1}^{s} \frac{g_i(x)}{f_i(x)}$. This proves the existence.

For the uniqueness, if $\frac{g(x)}{f(x)} = \sum_{i=1}^{s} \frac{g_i(x)}{f_i(x)} = \sum_{i=1}^{s} \frac{g_i'(x)}{f_i(x)}$, then $\sum_{i=1}^{s} \frac{g_i(x) - g_i'(x)}{f_i(x)} = 0$. Multiplying $f(x)$ to both sides, we get:

$$\sum_{i=1}^{s} (g_i - g_i')F_i = 0.$$

Since $f_1 \mid F_i$ for $i \geq 2$ and $\gcd(f_1, F_1) = 1$, we have $f_1 \mid (g_1 - g_1')$. Then $g_1 - g_1' = 0$. Similarly we have $g_i - g_i' = 0$ for any i. $\square$

Lemma 7.4. *If $f(x) = p(x)^r$, where $p(x)$ is irreducible, then every proper fraction $\frac{g(x)}{f(x)}$ can be written uniquely as*

$$\frac{g(x)}{f(x)} = \sum_{i=0}^{r-1} \frac{g_i(x)}{p(x)^i}$$

with $\deg g_i < \deg p$ *for all* i.

Proof. $g(x)$ has a unique $p(x)$-adic expansion

$$g(x) = \sum_{i=0}^{r-1} g_i(x)p(x)^{r-i}, \ \deg g_i < \deg p.$$

Dividing $f(x)$ on both sides, we get

$$\frac{g(x)}{f(x)} = \sum_{i=0}^{r-1} \frac{g_i(x)}{p(x)^i}.$$

The uniqueness follows from the uniqueness of the $p(x)$-adic expansion. □

1.3. Sequences and trace.

Theorem 7.2. *Suppose $p_i(x)$ $(i = 1, \cdots, s)$ are distinct irreducible polynomials over $\mathbb{F}_q[x]$ and $p_i(0) \neq 0$, $\deg(p_i(x)) = d_i$ and α_i is a root of $p_i(x)$. Let T_{d_i} be the trace map $\mathrm{Tr}_{q^{d_i}/q} : \mathbb{F}_{q^{d_i}} \to \mathbb{F}_q$. Let $f(x) = p_1(x) \cdots p_s(x)$. For any proper fraction $\frac{g(x)}{f(x)}$, let $\mathbf{a} = (a_n)$ be the associated purely periodic sequence. then there exists a unique $\beta_i \in \mathbb{F}_{q^{d_i}}$ for $1 \leq i \leq s$ such that*

$$a_n = \sum_{i=1}^{s} T_{d_i}(\beta_i \alpha_i^{-n}).$$

Proof. By Lemma 7.3, we may assume $s = 1$ and $f(x) = p(x)$. Suppose $\deg(p(x)) = d$ and α is a root of $p(x)$. We may also assume $p(0) = 1$. Then α^{q^i} $(0 \leq i \leq d-1)$ are all the roots of $p(x)$. and we have a decomposition

$$p(x) = (1 - \alpha^{-1}x) \cdots (1 - \alpha^{-q^{k-1}}x)$$

in $\mathbb{F}_{q^d}$. By Lemma 7.3, a proper fraction $\frac{g(x)}{p(x)}$ has a unique decomposition

$$\frac{g(x)}{p(x)} = \sum_{i=0}^{d-1} \frac{\beta_i}{1 - \alpha^{-q^i}x}, \ \beta_i \in \mathbb{F}_{q^d}.$$

So

$$\frac{g(x)^q}{p(x)^q} = \sum_{i=0}^{d-1} \frac{\beta_i^q}{1 - \alpha^{-q^{i+1}}x^q} = \frac{g(x^q)}{p(x^q)} = \sum_{i=0}^{d-1} \frac{\beta_i}{1 - \alpha^{-q^i}x^q}.$$

Let $\beta = \beta_0$. By the uniqueness in Lemma 7.3 again, we have $\beta_i = \beta^{q^i}$ for $0 \leq i \leq d-1$ and

$$\frac{g(x)}{p(x)} = \sum_{i=0}^{d-1} \frac{\beta^{q^i}}{1 - \alpha^{-q^i} x}.$$

Comparing the coefficients we get $a_n = T_d(\beta\alpha^{-n})$ for $n \geq 0$. The uniqueness follows from the fact that the number of $\beta \in \mathbb{F}_{q^d}$ and the number of proper fractions of the form $\frac{g(x)}{p(x)}$ are both q^k. □

2. Linear Feedback Shift Registers(LFSR)

In practice, a linear feedback shift register (LFSR) is a register of bits that performs discrete step operations that shift all the bits one position to the left and replace the vacated bit by a linear function of its previous state. Over $\mathbb{F}_2$, the only linear function of single bits is the exclusive-or (xor), thus it is a shift register whose input bit is driven by xor of some bits of the overall shift register value.

This concept of LFSR can be generalized to an arbitrary finite field. Mathematically, we have the following definition.

Definition 7.3. Let $k \in \mathbb{Z}_+$ and $f(x) = 1 - a_1x - \cdots - a_kx^k \in \mathbb{F}_q[x]$ be a polynomial such that $f(0) = 1$ and $\deg(f) \leq k$. The linear feedback shift register $\mathrm{lsr}(f, k)$ is the map

$$\mathbb{F}_q^k \mapsto \mathbb{F}_q^k, \quad (x_0, x_1, \cdots, x_{k-1}) \mapsto (x_1, \cdots, x_{k-1}, x_k = \sum_{i=1}^{k} a_j x_{k-i}),$$

of which f is called the connecting polynomial or feedback polynomial.

Given $\mathbf{c}^k = (c_0, \cdots, c_{k-1})$, performing $\mathrm{lsr}(f, k)$ recursively, the infinite sequence $\mathbf{c} = (c_0, c_1, \cdots)$ obtained, i.e.,

$$c_n = \sum_{i=1}^{k} a_i c_{n-i} \text{ for } n \geq k,$$

is called the LFSR sequence generated by $\mathrm{LSR}(f, k)$ from $\mathbf{c}^k$. We denote

$$\mathbf{c} = \mathrm{LSR}(f, k)(\mathbf{c}^k) = \mathrm{lsr}(f, k)(c_0, \cdots, c_{k-1}),$$

and call $\mathbf{c}^k$ the seed or the initial state and $(c_i, c_{i+1}, \cdots, c_{i+k-1})$ an intermediate state of $\mathbf{c}$.

If $\deg(f) = k$, i.e. $a_k \neq 0$, we write $\mathrm{lsr}(f)$ and $\mathrm{LSR}(f)$ for $\mathrm{lsr}(f, k)$ and $\mathrm{LSR}(f, k)$.

We note that an LFSR sequence $\mathbf{c} = (c_i)$ is determined by its initial state $\mathbf{c}^k = (c_0, \cdots, c_{k-1})$, so there are q^k LFSR sequences generated by $\mathrm{LSR}(f, k)$ in total. Moreover, at least two intermediate states of $\mathbf{c}$ must be equal, say $(c_i, c_{i+1}, \cdots, c_{i+k-1}) = (c_j, c_{j+1}, \cdots, c_{j+k-1})$ and $j > i$, then $c_n = c_{n+j-i}$ for all $n \geq i$, hence $\mathbf{c}$ is periodic with period $j - i$.

LEMMA 7.5. *Suppose* $\deg(f) = j < k$. *Let* $(c_0, c_1, \cdots, c_{k-1})$ *be a seed of* $\mathrm{LSR}(f, k)$. *Then*

$$\mathrm{LSR}(f,k)(c_0,\cdots,c_{k-1}) = (c_0,\cdots,c_{k-j-1},\mathrm{LSR}(f)(c_{k-j},\cdots,c_{k-1})).$$

PROOF. Clear. □

By the above lemma, to study LFSR sequences generated by $\mathrm{LSR}(f, k)$, it suffices to study LFSR sequences generated by $\mathrm{LSR}(f)$.

DEFINITION 7.4. Let $V(f, k)$ be the set of LFSR sequences generated by $\mathrm{LSR}(f, k)$, and let $V(f) := V(f, \deg(f))$ be the set of LFSR sequences generated by $\mathrm{LSR}(f)$.

THEOREM 7.3. *All sequences in* $V(f, k)$ *are periodic and* $V(f)$ *is the* k*-dimensional vector space* $\{\frac{g(x)}{f(x)} \mid \deg g < k\}$. *In particular, if* $a_k \neq 0$, *then all sequences in* $V(f)$ *are purely periodic.*

PROOF. Suppose $\mathbf{c} = (c_n)$ is an LFSR sequence generated by $\mathrm{LSR}(f, k)$. Then

$$f(x)\mathbf{c}(x) = (1 - a_1x - \cdots - a_kx^k)(c_0 + c_1x + \cdots) = \sum_{\ell=0}^{\infty} b_\ell x^\ell$$

where

$$b_\ell = \begin{cases} c_0 & \text{if } \ell = 0, \\ c_\ell - \sum_{i=1}^{\ell} c_{\ell-i}a_i & \text{if } 1 \leq \ell \leq k-1, \\ 0 & \text{if } \ell \geq k. \end{cases}$$

Hence

$$\mathbf{c}(x) = \frac{\sum_{\ell=0}^{k-1} b_\ell x^\ell}{f(x)}.$$

Since there are only q^k fractions of the form $\frac{g(x)}{f(x)}$ such that $\deg(g) < k$, and $V(f)$ is also of order q^k, we must have $V(f, k) = \{\frac{g(x)}{f(x)} \mid \deg g < k\}$.

In the case $k = \deg(f)$, then $\frac{g(x)}{f(x)}$ is a proper fraction and $\mathbf{c}$ is purely periodic. □

THEOREM 7.4. *Suppose* $a_k \neq 0$. *Let* $\widetilde{f}(x) = -a_k^{-1}f(x)$.

(1) *Every sequence* $\mathbf{c}(x) \in V(f)$ *is of period at most* $q^k - 1$; *and if there exists a sequence of period* $q^k - 1$, *then* $\widetilde{f}(x)$ *is primitive.*
(2) *If* $f(x)$ *is irreducible, then every nonzero sequence in* $V(f)$ *is of period* $\mathrm{per}(f)$. *Hence if* $\widetilde{f}(x)$ *is primitive, then every non-zero sequence in* $V(f)$ *is of maximal length* $q^k - 1$.

PROOF. (1) Note that if $x_k = x_0a_k + \cdots + x_{k-1}a_1$, then $x_0 = -a_k^{-1}(x_k - x_{k-1}a_1 - \cdots - x_1a_{k-1})$. Thus $(x_0, \cdots, x_{k-1}) = 0 \Leftrightarrow (x_1, x_2, \cdots, x_k) = 0$.

Let $\mathbf{c} \in V(f)$. If $\mathbf{c} = 0$, then $\mathrm{per}(\mathbf{c}) = 1 \leq q^k - 1$. Otherwise, the seed $\mathbf{c}_k = (c_0, c_1, \cdots, c_{k-1}) \neq 0$, hence the state $(c_d, c_{d+1}, \cdots, c_{d+k-1}) \neq 0$ for all

d. There are only q^k-1 possible nonzero states, two of the first q^k states must be equal, hence $\mathrm{per}(\mathbf{c}) \le q^k-1$. Since $\mathbf{c}(x)=\frac{g(x)}{f(x)}$, $\mathrm{per}(c) \mid \mathrm{per}(f)$. We know $\mathrm{per}(f)$ is at most q^k-1 and the equality holds if and only if f is primitive. So if $\mathrm{per}(c)=q^k-1$, then f is primitive.

(2) If f is irreducible, then $\mathbf{c}(x)=\frac{g(x)}{f(x)}$ is of reduced form. Hence $\mathrm{per}(\mathbf{c})=\mathrm{per}(f)$. □

Definition 7.5. If $\widetilde{f}(x)$ is primitive, then every non-zero sequence in $V(f)$ is called a q-ary level-k m-sequence (maximal length sequence, pseudo random sequence or PN sequence).

Theorem 7.5. *Suppose $\widetilde{f}(x)=-a_k^{-1}f(x)$ is primitive and α^{-1} is a root of $f(x)$.*

(1) *For any $\mathbf{c}=(c_0,c_1,\cdots)\in V(f)$, there exists an unique element $\beta\in\mathbb{F}_{q^k}$ such that $c_n=T_k(\beta\alpha^n)$ for $n\ge 0$, where $T_k=\mathrm{Tr}_{q^k/q}$ is the trace map $\mathbb{F}_{q^k}\to\mathbb{F}_q$, $x\mapsto x^q+x^{q^2}+\cdots+x^{q^{k-1}}$.*

(2) *Suppose $1\le \ell\le k$. If $\mathbf{c}$ is an m-sequence, in the multiset $M_\ell := \{(c_i,c_{i+1},\cdots,c_{i+\ell-1}) \mid 0\le i\le q^k-2\}$, any $v=(v_1,\cdots,v_\ell)\in \mathbb{F}_q^\ell\setminus\{\vec{0}\}$ appears $q^{k-\ell}$ times, and $(0,\cdots,0)$ appears $q^{k-\ell}-1$ times. In particular, take $\ell=1$, then*

$$|\{i \mid c_i=b,\ 0\le i\le q^k-2\}| = \begin{cases} q^{k-1}-1, & \text{if } b=0;\\ q^{k-1}, & \text{if } b\in\mathbb{F}_q^\times.\end{cases}$$

(3) *For any two m-sequences $\mathbf{a}=(a_0,a_1,\cdots,a_n,\cdots)$ and $\mathbf{b}=(b_0,\cdots,b_n,\cdots)$ in $V(f)$, there exists t such that $b_n=a_{n+t}$ for all $n\ge 0$. As a consequence, for any m-sequence $\mathbf{c}$ generated by $\mathrm{lsr}(f)$,*

$$V(f)=\{0\}\cup\{(\dot{c}_i,c_{i+1},\cdots,\dot{c}_{i+q^k-2}) \mid 0\le i\le q^k-2\}.$$

Proof. (1): This is just Theorem 7.2. In particular, $\beta=0$ corresponds to the zero sequence.

(3): Suppose $\mathbf{a}$ corresponds to $\beta=\alpha^s$ and $\mathbf{b}$ corresponds to $\beta=\alpha^r$. Assume $r>s$, then $b_n=T(\alpha^{n+r})=T(\alpha^{n+s+t})=a_{n+t}$ for $t=r-s$.

(2): A non-zero vector in $\mathbb{F}_q^k$ is the seed of some m-sequence, by (3), it must be an intermediate state of the sequence $\mathbf{c}$, hence is contained in M_k at least once. But $|M_k|=q^k-1$, so $M_k=\mathbb{F}_q^k\setminus\{\vec{0}\}$ is actually a set, and (2) is true for the case $\ell=k$.

The case $\ell<k$ follows from the fact that there are $q^{k-\ell}$ vectors in $\mathbb{F}_q^k\setminus\{\vec{0}\}$ whose first ℓ terms are fixed and not all zero, and $q^{k-\ell}-1$ vectors whose first ℓ terms are all 0. □

Remark 7.1. The above fact is the reason why an m-sequence is also called a pseudo-random sequence (PN sequence).

3. Berlekamp-Massey Algorithm

Definition 7.6. For a sequence $\mathbf{c}^N = (c_0, c_1, \cdots, c_{N-1})$ of length $N \in \mathbb{Z}_+ \cup \{\infty\}$, we let $\mathbf{c}^n = (c_0, \cdots, c_{n-1})$ be the finite sequence consisting of the first n terms of $\mathbf{c}$.

Definition 7.7. A sequence $\mathbf{c}^n = (c_0, \cdots, c_{n-1})$ is said to be generated by the linear feedback shift register $\mathrm{LSR}(f, k)$ if it is the first n-terms of $\mathrm{LSR}(f, k)(c_0, \cdots, c_{k-1})$, i.e., $(\mathrm{LSR}(f, k)(c_0, \cdots, c_{k-1}))^n = \mathbf{c}^n$.

Lemma 7.6. *Let* $\mathbf{c}^n(x) = \sum_{i=0}^{n-1} c_i x^i$. *Then* $\mathrm{LSR}(f, k)$ *generates* $\mathbf{c}^n$ *if and only if there exists a polynomial* $g(x)$ *of degree* $< k$ *such that* $\mathbf{c}^n(x) \equiv \frac{g(x)}{f(x)} \bmod x^n$ *(as power series).*

Proof. Clear. □

Problem 7.1. Suppose an N-sequence $\mathbf{c}^N = (c_0, \cdots, c_{N-1})$ is received. Find an algorithm to determine the minimal integer ℓ_N and a connecting polynomial $f_N(x)$ such that $(\mathrm{LSR}(f_N, \ell_N)$ generated $\mathbf{c}^N$.

Remark 7.2. Certainly,

(1) $\ell_N \le N$ exists.
(2) f_N must satisfy the conditions $f_N(0) = 1$ and $\deg(f_N) \le \ell_N$.

The answer is the Berlekamp-Massey Algorithm (1969).

Algerithm 7.1 (Berlekamp-Massey Algorithm).

```
Input  $\mathbf{c}^N = (c_0, \cdots, c_{N-1})$.
Output For $1 \le n \le N$, find the minimal integer $\ell_n$ and a connecting
       polynomial $f_n$ such that $\mathrm{LSR}(f_n, \ell_n)$ generates $\mathbf{c}^n = (c_0, \cdots, c_{n-1})$.
Step 1 Suppose $c_{n_0} \ne 0$ and $c_i = 0$ for all $i < n_0$. Then we set
         (a) $f_i = 1$ and $\ell_i = 0$ for $i \le n_0$;
         (b) $d_i = 0$ for $i < n_0$ and $d_{n_0} = c_{n_0}$;
         (c) $f_{n_0+1}(x) = 1 - d_{n_0} x^{n_0+1}$ and $\ell_{n_0+1} = n_0 + 1$.
Step 2 Suppose we have already known $(f_i, \ell_i)$ and $d_i$ for $0 \le i \le n$ where
       $n \ge n_0+1$. Write $f_n(x) = \sum_{j=0}^{\ell_n} a_{n,j} x^j$ and set $d_n = \sum_{j=0}^{n} a_{n,j} c_{n-j}$.
         (a) If $d_n = 0$, let $f_{n+!} = f_n$ and $\ell_{n+1} = \ell_n$.
         (b) If $d_n \ne 0$, then there exists $m$ with $n_0 \le m \le n$ such that $\ell_m <$
             $\ell_{m+1} = \cdots = \ell_n$. Set $f_{n+1}(x) = f_n(x) - d_n d_m^{-1} x^{n-m} f_m(x)$ and
             $\ell_{n+1} = \max\{\ell_n, n + 1 - \ell_n\}$.
       Then $(f_N, \ell_N)$ is good for our problem.
```

Lemma 7.7. *Let* $(a_0, \cdots, a_n)$ *be a sequence over* $\mathbb{F}_q$.

(1) *Suppose* ℓ *and* f *satisfy that* $\mathrm{LSR}(f, \ell)$ *generates* $(a_0, \cdots, a_{n-1})$ *but not* $(a_0, \cdots, a_n)$. *Then if* $\mathrm{LSR}(f', \ell')$ *generates* $(a_0, \cdots, a_n)$, *one must have* $\ell' \ge n - 1 - \ell$.
(2) *Let* ℓ_i $(i \le n)$ *be the minimal integer such that there exists some* f_i *such that* $\mathrm{LSR}(f_i, \ell_i)$ *generates* $(a_0, \cdots, a_{i-1})$. *Then* $\ell_1 \le \ell_2 \le \cdots \le \ell_n$ *and* $\ell_i \le i$. *Moreover, if there exists* f_i *such that* $\mathrm{LSR}(f_i, \ell_i)$

generates $(a_0, \cdots, a_{i-1})$ *but not* $(a_0, \cdots, a_i)$, *then* $\ell_{i+1} \geq \max\{\ell_i, \ell + 1 - \ell_i\}$.

Proof. (1) Let $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$. By Lemma 7.6, $\mathrm{LSR}(f, \ell)$ generates $(a_0, \cdots, a_{n-1})$ but not $(a_0, \cdots, a_n)$ if and only if there exists a polynomial $g(x)$ such that $\deg g \leq \ell - 1$ and $\frac{g(x)}{f(x)} = a(x) + (b + a_n)x^n \mod x^{n+1}$ for some $b \in \mathbb{F}_q^\times$. Now if $\mathrm{LSR}(f', \ell')$ generates $(a_0, \cdots, a_n)$, then there exists a polynomial $g'(x)$ with $\deg g' \leq \ell' - 1$ such that $\frac{g'(x)}{f'(x)} = a(x) + a_n x^n \mod x^{n+1}$. So we have $\frac{g'(x)}{f'(x)} - \frac{g(x)}{f(x)} = bx^n \mod x^{n+1}$ with $b \neq 0$. So $g(x)f'(x) - g'(x)f(x) = bf(x)f'(x)x^n = bx^n \mod x^{n+1}$. If $\ell' \leq n - \ell$, then $\deg(gf') \leq \ell - 1 + \ell' < n$ and $\deg(g'f) \leq \ell' - 1 + \ell < n$, which is impossible.

(2) By definition we have $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_n$ and $\ell_i \leq i$. If there exists f_i such that $\mathrm{LSR}(f_i, \ell_i)$ generates $(a_0, \cdots, a_{i-1})$ but not $(a_0, \cdots, a_i)$, by (1) we have $\ell_{i+1} \geq \max\{\ell_i, \ell + 1 - \ell_i\}$. ☐

Theorem 7.6.

(1) *The Berlekamp-Massey Algorithm produces* $(f_n(x), \ell_n)$ *with* ℓ_n *minimal for* $1 \leq n \leq N$;
(2) *If* $\mathrm{LSR}(f_{n-1}, \ell_{n-1})$ *generates* $(c_0, \cdots, c_{n-1})$, *then* $\ell_n = \ell_{n-1}$, *otherwise* $\ell_n = \max\{\ell_{n-1}, n - \ell_{n-1}\}$ *for* $n \geq 2$.

Proof. We prove both (1) and (2) by induction on n.

Base case: By Step 1 in the Berlekamp-Massey Algorithm, (1) is true for $n \leq n_0$ and (2) is true for $n \leq n_0 + 1$, and $\mathrm{LSR}(f_{n_0+1}, n_0 + 1)$ generates $(0, \cdots, 0, c_{n_0})$. But $c_{n_0} \neq 0$, no $\mathrm{LSR}(f, \ell)$ for any f and $\ell \leq n_0$ can generate $(0, \cdots, 0, c_{n_0})$, thus $\ell_{n_0+1} = n_0 + 1$ and (1) is true for $n = n_0 + 1$. Hence (1) and (2) are both true for $n \leq n_0 + 1$.

Induction step: Now we suppose (1) and (2) are true for $n \geq n_0 + 1$.

(a) If $d_n = 0$, then $\mathrm{LSR}(f_n, \ell_n)$ generates $(a_0, \cdots, a_n)$, hence $\ell_{n+1} \leq \ell_n$. But we also have $\ell_n \leq \ell_{n+1}$, hence $\ell_{n+1} = \ell_n$.

(b) If $d_n \neq 0$, let $\ell_m < \ell_{m+1} = \cdots = \ell_n$, $m < n$. Then $d_m \neq 0$, which implies that $\ell_{m+1} = \max\{\ell_m, m + 1 - \ell_m\} > \ell_m$, so $\ell_n = \ell_{m+1} = m + 1 - \ell_m$. We have $\deg f_{n+1} \leq \max\{\ell_n, n - m + \ell_m\} = \max\{\ell_n, n + 1 - \ell_n\} := \ell'_{n+1}$. Then we have:

$$f_{n+1}(x) = \sum_{i=0}^{\ell'_{n+1}} a_{n+1,j}x^j = \sum_{i=0}^{\ell_n} a_{n,j}x^j + d_n d_m^{-1} \sum_{j=0}^{\ell_m} a_{m,j}x^{i+n-m}.$$

Hence for $\ell'_{n+1} \leq k \leq n$, we have:

$$\sum_{j=0}^{\ell_{m+1}} a_{n+1,j}c_{k-j} = \sum_{j=0}^{\ell_n} a_{n,j}c_{k-j} + d_n d_m^{-1} \sum_{j=0}^{\ell_m} a_{m,j}c_{k-j+m-n}.$$

If $\ell'_{n+1} \leq k \leq n-1$, this expression is of value 0; if $k = n$, then

$$\sum_{j=0}^{\ell_n} a_{n,j}c_{k-j} + d_n d_m^{-1} \sum_{j=0}^{\ell_m} a_{m,j}c_{k-j+m-n} = d_n - d_n d_m^{-1} d_m = 0.$$

So (f_{n+1}, ℓ'_{n+1}) generates $(c_0, \cdots, c_n)$. By lemma, we have $\ell_{n+1} \geq \max\{\ell_n, n+1-\ell_n\} = \ell'_{n+1}$. Hence the $\ell_{n+1} = \max\{\ell_n, n+1-\ell_n\}$ is minimal, and (1) and (2) are proved fo $n+1$. $\square$

THEOREM 7.7. *Suppose 2ℓ and $2\ell' \leq N$. Let $(c_0, \cdots, c_{2N-1})$ be given. Suppose the polynomials f, f, f', g' satisfy that $f(0) = f'(0) = 1$, $\gcd(f, g) = \gcd(f', g') = 1$, $\deg f \leq \ell$, $\deg f' \leq \ell'$, $\deg g \leq \ell - 1$, $\deg g' \leq \ell' - 1$ and*

$$\frac{g(x)}{f(x)} \equiv \frac{g'(x)}{f'(x)} \equiv c(x) = \sum_{i=0}^{N-1} c_i x^i \mod x^N.$$

Then $f = f'$ and $g = g'$.

In particular, if $2\ell_N \leq N$, then f_N is unique.

PROOF. The identity implies $f'(x)g(x) - f(x)g'(x) \equiv 0 \mod x^N$. Comparing the degrees, we must have $fg' - f'g = 0$. By the fact $\gcd(f, g) = \gcd(f', g') = 1$, then $f \mid f'$ and $f' \mid f$, and $f = cf'$ for some $c \neq 0$. But $f(0) = f'(0) = 1$, hence $f(x) = f'(x)$. $\square$

CHAPTER 8

Cyclic codes and BCH codes

1. Cyclic codes

1.1. The chain ring R_n and its ideals. For $n \geq 1$, let

$$R_n := \mathbb{F}_q[x]/(x^n - 1)$$

be the quotient ring of $\mathbb{F}_q[x]$ modulo $x^n - 1$, i.e., the polynomials $f(x)$ and $g(x)$ represent the same element in R_n if and only if $f(x) \equiv g(x) \bmod x^n - 1$. For $r(x) \in \mathbb{F}_q[x]$, by the division algorithm, $r(x) = q(x)(x^n - 1) + r'(x)$ for a unique $r'(x) \in \mathbb{F}_q[x]$ such that $\deg(r') < n$. Then the quotient element $\bar{r}(x) = r(x) + (x^n - 1)\mathbb{F}_q[x] \in R$ is represented by a unique polynomial $r'(x)$ such that $\deg(r') < n$. We identify $\bar{r}(x)$ and $r'(x)$ and hence

$$R_n = \{r(x) \mid r(x) \in \mathbb{F}_q[x],\ \deg(r) < n\}.$$

However, keep in mind that the sum $r_1 + r_2$ and product $r_1 \cdot r_2$ in R_n are the unique remainders of the sum and product in $\mathbb{F}_q[x]$ by division of $x^n - 1$. We also note that for $f(x) \in \mathbb{F}_q[x]$ and $r(x) \in R_n$, the multiplication $f(x)r(x) \in R_n$ is well defined. In fact, R_n is an $\mathbb{F}_q[x]$-module via the quotient map. In particular, one sees that

$$x \sum_{i=0}^{n-1} r_i x^i = r_{n-1} + \sum_{i=1}^{n-1} r_{i-1} x^i.$$

Note that R_n is an $\mathbb{F}_q$-vector space of dimension n. For $r(x) = \sum_{i=0}^{n-1} r_i x^i \in R_n$, set $\mathrm{wt}(r(x)) := \#\{0 \leq i < n \mid r_i \neq 0\}$ and define $d(r_1, r_2) := \mathrm{wt}(r_1 - r_2)$. Then R_n becomes a metric space. The map

$$\phi : \mathbb{F}_q^n \to R_n,\ \alpha = (a_1, \cdots, a_n) \mapsto \alpha(x) = \sum_{i=1}^{n} a_i x^{i-1}$$

is an isomorphism of $\mathbb{F}_q$-vector spaces and preserves the distance, we may identify R_n and $\mathbb{F}_q^n$.

Proposition 8.1. *Let I be a nonempty subset of $\mathbb{F}_q^n$ and hence of R_n under the identification map ϕ. Then the followings are equivalent:*

1. *For any $r = (r_1, r_2, \cdots, r_n) \in I$, $(r_2, \cdots, r_n, r_1) \in I$.*
2. *For every codeword $r = r(x) \in I$, $xr(x) \in I$.*
3. *For all function $f(x) \in \mathbb{F}_q[x]$, $f(x)r(x) \in C$.*
4. *I is an ideal of R_n.*

Lemma 8.1. *If I is an ideal of R_n, then there exists a unique monic factor $g(x)$ of $x^n - 1$, such that $I = g(x)R_n$.*

Proof. Since $\mathbb{F}_q[x]$ is a PID, for an ideal $I \subseteq R_n$, its inverse image $J \subseteq \mathbb{F}_q[x]$ is a principal ideal, i.e. $J = (g(x))$ for some $g(x) \in \mathbb{F}_q[x]$. Moreover, $(x^n - 1) \subseteq J$, $g(x)$ must be a factor of $x^n - 1$. We have $I = (g(x)) = g(x)R_n$. Note that two ideals $\beta_1(x)\mathbb{F}_q[x] = \beta_2(x)\mathbb{F}_q[x]$ if and only if $\beta_2(x) = \lambda\beta_1(x)$ for some $\lambda \in \mathbb{F}_q^\times$, so $g(x)$ is unique if it is assumed to be monic. □

1.2. Definition and basic properties. Let C be a linear code of length n over $\mathbb{F}_q$. Via the isomorphism ϕ we may and will identify C with

$$\phi(C) = \{c(x) := \sum_{i=1}^{n} c_i x^{i-1} \mid c = (c_1, \cdots, c_n) \in C\}.$$

Definition 8.1. A linear code C is called cyclic if for a codeword $(c_1, \cdots, c_n) \in C$, $(c_2, \cdots, c_n, c_1) \in C$.

Definition 8.2. The reflexive polynomial of $0 \neq f(x) \in \mathbb{F}_q[x]$ is the polynomial $f^*(x) := x^{\deg(f)} f(\frac{1}{x})$.

Lemma 8.2. *Let $f(x) \in \mathbb{F}_q[x]$ be of degree n. Then* $\deg(f^*) \leq \deg(f)$, *and the equality holds if and only if $f(0) \neq 0$. Moreover,*

(1) *if $f(0) \neq 0$, then the reflexive polynomial f^{**} of f^* is f;*
(2) *$(fg)^* = f^* g^*$;*
(3) *f is irreducible if only if f^* is.*

Theorem 8.1. *Let C be a cyclic $[n, k]_q$-code with $(q, n) = 1$.*

(1) *C is an ideal of $R_n = \mathbb{F}_q[x]/(x^n - 1)$ generated by a unique monic factor $g(x)$ of $x^n - 1$ whose degree is $n - k$. Moreover, let*
$$h(x) = \frac{x^n - 1}{g(x)}.$$
Then $C = \{c(x) \in R_n \mid c(x)h(x) = 0\}$ is annihilated by $h(x)$.
(2) *$C^\perp$ is a cyclic code of dimension $n - k$ generated by $h^*(x)$ and annihilated by $g^*(x)$:*
$$C^\perp = \{h^*(x)a(x) \mid a(x) \in R_n\} = \{c(x) \in R_n \mid c(x)g^*(x) = 0\}.$$

Proof. (1) We have already known $C = (g(x)) = \{g(x)a(x) \mid a(x) \in R_n\}$ with $g(x)$ a unique monic factor of $x^n - 1$. Moreover, for $a(x) \in \mathbb{F}_q[x]$, let $a(x) = q(x)h(x) + r(x)$, $\deg r < n - \deg(g)$. Then $g(x)a(x) = g(x)r(x) \in R_n$, hence C is generated by $\{x^i g(x) \mid 0 \leq i < n - \deg(g)\}$ as an $\mathbb{F}_q$-vector space, but $\{x^i g(x) \mid 0 \leq i < n - \deg(g)\}$ is linearly independent in R_n, hence C is an $\mathbb{F}_q$-vector space of dimension $n - \deg(g)$, which means $\deg(g) = n - k$.

For the second statement, the inclusion $\{g(x)a(x) \mid a(x) \in R_n\} \subseteq \{c(x) \in R_n \mid c(x)h(x) = 0\}$ is clear since $g(x)h(x) = 0 \in R_n$. On the other hand, if $c(x)h(x) = 0 \in R_n$, then $c(x)h(x) = \alpha(x)(x^n - 1)$ in $\mathbb{F}_q[x]$. So $c(x) = \alpha(x)\frac{x^n-1}{h(x)} = \alpha(x)g(x) \in R_n$. Hence

$$C = \{g(x)a(x) \mid a(x) \in R_n\} = \{c(x) \in R_n \mid c(x)h(x) = 0\}.$$

(2) Note that $\dim C^\perp = n - k$. Since $g(0) \neq 0$, g^* is also a factor of $x^n - 1$, $\deg(g^*) = \deg(g) = n - k$ and $g^*(x)h^*(x) = x^n - 1$. By (1), $\{c(x) \in R_n \mid c(x) = h^*(x)a(x)\} = \{c(x) \in R_n \mid c(x)g^*(x) = 0\}$ is of dimension $n - k$. We only need to show $c(x)g^*(x) = 0$ for $c \in C^\perp$.

Write $g(x) = \sum_{i=0}^{n-k} g_i x^i$ with $g_{n-k} = 1$, then $g^*(x) = \sum_{i=0}^{n-k} g_{n-k-i}x^i$. Let $g_i = 0$ for $i > n-k$, then $g(x)$ corresponds to $(g_0, \cdots, g_{n-1}) \in C$. For any $c = (c_1, \cdots, c_n) \in C^\perp$, then $c \cdot g = 0$, i.e. $c_1g_0 + \cdots + c_ng_{n-1} = 0$. Moreover, since C is a cyclic code, for $0 \leq i \leq n-1$, $(g_i, \cdots, g_{n-1}, g_0, \cdots, g_{i-1}) \in C$ and

$$c_1g_i + \cdots + c_{n-i}g_{n-1} + c_{n-i+1}g_0 + \cdots + c_ng_{i-1} = 0.$$

In other words, let $g_i^* = g_{n-i}$, then for $0 \leq i \leq n$,

$$\sum_{k+\ell=n+1-i} c_k g_\ell^* = 0$$

and hence $c(x)g^*(x) = 0$. □

Definition 8.3. Let C be a cyclic code and let $g(x)$ and $h(x)$ be as in Theorem 8.1. Then $g(x)$ is called the generator polynomial of C and $h(x)$ is called the parity check polynomial of C.

Proposition 8.2. *Suppose C is a cyclic $[n,k]_q$-code, $g(x)$ is its generator polynomial. Then*

(1) Write $g(x) = \sum_{i=0}^{n-k} x^i$ ($g_{n-k} = 1$). Corresponding to the basis $\{x^ig(x) \mid 0 \leq i < k\}$, C has a generator matrix

$$G = \begin{pmatrix} g_0 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k} & \ddots & 0 \\ 0 & \ddots & \ddots & & \ddots & 0 \\ 0 & \cdots & 0 & g_0 & \cdots & g_{n-k} \end{pmatrix}_{k \times n}.$$

(2) Write $h(x) = \frac{x^n-1}{g(x)} = \sum_{i=0}^{k} x^i$ ($h_k = 1$). Then C has a parity check matrix

$$H = \begin{pmatrix} h_k & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & h_k & \cdots & h_0 & \ddots & 0 \\ 0 & \ddots & \ddots & & \ddots & 0 \\ 0 & \cdots & 0 & h_k & \cdots & h_0 \end{pmatrix}_{(n-k) \times n}.$$

(3) Let $g^\perp(x) = h_0^{-1}h^*(x)$ and $h^\perp(x) = g_0^{-1}g^*(x)$. Then $h^\perp$ is the generator polynomial and $g^\perp$ is the parity check polynomial of $C^\perp$.

Proposition 8.3. *Assume* $\gcd(q,n)=1$. *Suppose* $\{p_1(x),p_2(x),\cdots,\ p_e(x)\}$ *is the set of monic irreducible factors of* x^n-1 *in* $\mathbb{F}_q[x]$. *Then the factorization of* x^n-1 *over* $\mathbb{F}_q[x]$ *is*

$$x^n-1=p_1(x)\cdots p_e(x),$$

and there are 2^e *different cyclic codes of length* n *over* $\mathbb{F}_q$.

Proof. Since $\gcd(q,n)=1$, x^n-1 is separable over $\mathbb{F}_q[x]$ and has no multiple roots, hence the factorization. Now a cyclic code of length n is uniquely determined by its generator polynomial which is a monic factor of x^n-1. There are 2^e monic factors of x^n-1 by the factorization of x^n-1. $\square$

Example 8.1. Let $q=3$ and $n=10$. Over $\mathbb{F}_3[x]$, the factorization of $x^{10}-1$ is

$$x^{10}-1=(x+1)(x-1)(x^4+x^3+x^2+x+1)(x^4-x^3+x^2-x+1).$$

Then the number of cyclic codes of length 10 over $\mathbb{F}_3$ is $2^4=16$.

Take $g(x)=x^4-x^3+x^2-x+1$, and let $C=(g(x))$ be the cyclic code generated by $g(x)$. For its parameters: $[n,k,d]$, then $n=10$ and $k=n-\deg g=6$. C has a generator matrix

$$G=\begin{pmatrix} 1&2&1&2&1&0&0&0&0&0\\ 0&1&2&1&2&1&0&0&0&0\\ 0&0&1&2&1&2&1&0&0&0\\ 0&0&0&1&2&1&2&1&0&0\\ 0&0&0&0&1&2&1&2&1&0\\ 0&0&0&0&0&1&2&1&2&1 \end{pmatrix}.$$

Since $h(x)=\frac{x^{10}-1}{g(x)}=x^6+x^5-x-1$, C has a parity check matrix

$$H=\begin{pmatrix} 1&1&0&0&0&2&2&0&0&0\\ 0&1&1&0&0&0&2&2&0&0\\ 0&0&1&1&0&0&0&2&2&0\\ 0&0&0&1&1&0&0&0&2&2 \end{pmatrix},$$

which implies $d=2$. So C is a linear $[10,6,2]_3$-code.

For $C^\perp$, it is a $[10,4,d^\perp]_3$-code. The generator polynomial of $C^\perp$ is $h(x)$ and the parity check polynomial is $g(x)$. So $C^\perp=\{(c_1,2c_1)\mid c_1\in C_1\}$ where C_1 is the code generated by:

$$\begin{pmatrix} 1&1&0&0&0\\ 0&1&1&0&0\\ 0&0&1&1&0\\ 0&0&0&1&1 \end{pmatrix}.$$

Since $d(C_1)=2$, we have $d^\perp=4$. Hence $C^\perp$ is $[10,4,4]_3$-code.

1.3. Binary Hamming Code is cyclic. Let $m \geq 2$, $g(x) \in \mathbb{F}_2[x]$ be a primitive polynomial of degree m. Note that this is equivalent to one of the following two conditions:

(1) any root α of $g(x)$ generates $\mathbb{F}_{2^m}^\times$, i.e., $\langle\alpha\rangle = \mathbb{F}_{2^m}^\times$;
(2) m is the minimal positive integer d such that $g(x) \mid (x^{2^d} - x)$.

Let $n = 2^m - 1$, then $g(x) \mid x^n - 1$ and $h(x) = \frac{x^n-1}{g(x)}$ is of degree $2^m - 1 - m$.

Let C be the cyclic code generated by $g(x)$, then C is a $[2^m - 1, 2^m - m - 1]_2$-code. In R_n, only x^ℓ with $0 \leq \ell \leq n$ and $x^\ell(1 + x^i)$ are of weight at most 2, and they are not in C. Hence $d(C) \geq 3$. By the Hamming bound we have $d(C) = 3$. So C is $[2^m - 1, 2^m - 1 - m, 3]$-code.

Let H be a parity check matrix of C, then $H \in \mathbb{F}_2^{m\times(2^m-1)}$. Note that:

- every column of H belongs to $\mathbb{F}_2^{m\times 1}$;
- any 2 columns of H are linearly independent (because $d(C) = 3$), hence different.

So the set of column vectors of H is the set of all nonzero vectors in $\mathbb{F}_2^{m\times 1}$, hence $C = H(2, m)$, the binary Hamming code. This means the Hamming code $H(2, m)$ and its dual $\mathrm{WH}_m = R(1, m)'$ are cyclic codes.

Example 8.2. Take $m = 3$. The polynomial $g(x) = x^3 + x + 1$ is a primitive polynomial over $\mathbb{F}_2[x]$. Then the cyclic code C generated by $g(x)$ is the Hamming code $H(2, 3)$, which is a linear $[7, 4, 3]$-code over $\mathbb{F}_2$. It has the parity check polynomial $h(x) = \frac{x^7-1}{g(x)} = x^4 + x^2 + x + 1$, a generator matrix

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

and a parity check matrix

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

2. Trace Expression of Cyclic codes

Suppose C is a cyclic code of length n over $\mathbb{F}_q$ with $(q, n) = 1$. Let $g(x)$ be the generator polynomial of C and $h(x)$ be the parity check polynomial of C. Since $x^n - 1$ has no multiple root, we can assume $h(x)$ has a decomposition

$$h(x) = p_1(x) \cdots p_s(x)$$

where p_i is a monic irreducible polynomial over $\mathbb{F}_q$ of degree d_i. Let α_i be a root of $p_i(x)$. Then $\mathbb{F}_q(\alpha_i) = \mathbb{F}_{q^{d_i}}$. Set $T_{d_i} = \mathrm{Tr}_{q^{d_i}/q} : \mathbb{F}_{q^{d_i}} \to \mathbb{F}_q$.

THEOREM 8.2. *For every codeword $c = (c_0, \cdots, c_{n-1}) \in C$, there exists a unique $\beta_i \in \mathbb{F}_q(\alpha_i) = \mathbb{F}_q^{d_i}$ for each $i \leq s$ such that for any $0 \leq \lambda \leq n-1$,*

$$c_\lambda = \sum_{i=1}^{s} T_{d_i}(\beta_i \alpha_i^{-\lambda}).$$

PROOF. For $c = (c_0, \cdots, c_{n-1})$, let $\hat{c} = (\dot{c_0}, \cdots, \dot{c_{n-1}})$. Let $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$ and $\hat{c}(x) = \sum_{\lambda \geq 0} \hat{c}_\lambda x^\lambda$. Then $\hat{c}(x) = \frac{c(x)}{1-x^n} = -\frac{a(x)}{h(x)}$ for some unique $a(x)$ of degree $< \deg(h)$. By Theorem 7.2, there exists a unique $\beta_i \in \mathbb{F}_q(\alpha_i) = \mathbb{F}_{q^{d_i}}$ for $i \leq s$ such that:

$$\hat{c}_\lambda = \sum_{i=1}^{s} T_i(\beta_i \alpha_i^{-\lambda}) \text{ for all } \lambda \in \mathbb{N}.$$

So $c_\lambda = \hat{c}_\lambda = \sum_{i=1}^{s} T_i(\beta_i \alpha_i^{-\lambda})$ for $0 \leq \lambda \leq n-1$. □

EXAMPLE 8.3. Suppose $m \geq 2$, $h(x)$ is a primitive polynomial of degree m over $\mathbb{F}_q$. Then $\mathrm{per}(h) = n = q^m - 1$. Let $x^n - 1 = g(x)h(x)$ and let C be the cyclic code with the parity check polynomial $h(x)$. Then C is generated by $g(x)$ and $[n,k] = [q^m - 1, m]$. Let α be a root of $h(x)$ and $\gamma = \alpha^{-1}$. Then γ is also a primitive root of $\mathbb{F}_{q^n}$. i.e.,

$$\mathbb{F}_{q^n}^\times = \langle \gamma \rangle = \{\gamma^i \mid 0 \leq i \leq q^m - 2\}.$$

By Theorem 8.2, for a nonzero codeword $c \in C$, there exists a unique $\beta \in \mathbb{F}_{q^m}^\times$ such that $c_i = T_m(\beta\gamma^i)$, $0 \leq i \leq q^m - 2$. Since $\{\beta\gamma^i \mid 0 \leq i \leq q^m - 2\} = F_{q^m}^\times$ and the trace map T_m is a q^{m-1}-to-1 map, we know $|\{i \mid c_i = 0\}| = q^{m-1} - 1$ and $\mathrm{wt}(c) = q^{m-1}(q-1)$. Hence C is $[q^m - 1, m, q^{m-1}(q-1)]$-code over $\mathbb{F}_q$ whose nonzero codewords are all of weight $q^{m-1}(q-1)$. So its weight enumerator is

$$W_C(x,y) = x^{q^m-1} + (q^m - 1)x^{q^{m-1}-1}y^{q^m - q^{m-1}}.$$

For $c = (c_0, \cdots, c_{n-1})$, write $c^{(1)} = (c_0, c_1, \cdots, c_{\frac{q^m-1}{q-1}-1})$. Note that $\gamma^{\frac{q^m-1}{q-1}} \in \mathbb{F}_q$, then for $0 \leq j \leq \frac{q^m-1}{q-1} - 1$,

$$c_{k\frac{q^m-1}{q-1}+j} = T_m(\beta\gamma^k(\gamma^{\frac{q^m-1}{q-1}})^k) = (\gamma^{\frac{q^m-1}{q-1}})^k c_j$$

Thus $c = (c^{(1)}, \gamma^{\frac{q^m-1}{q-1}}c^{(1)}, \cdots, (\gamma^{\frac{q^m-1}{q-1}})^{q-2}c^{(1)})$ is uniquely determined by $c^{(1)}$.

Let $C' = \{c^{(1)} \mid c \in C\}$. Then C' is $[\frac{q^m-1}{q-1}, m]$-linear code over $\mathbb{F}_q$. Moreover, if $c^{(1)} \neq 0$, then $\mathrm{wt}(c^{(1)}) = \frac{1}{q-1}\mathrm{wt}(c) = q^{m-1}$. Hence $d(C') = q^{m-1}$ and

$$W_{C'}(x,y) = x^{\frac{q^m-1}{q-1}} + (q^m - 1)x^{\frac{q^m-1}{q-1}}y^{q^{m-1}}.$$

By MacWilliams Identity, we get $d(C'^\perp) = 3$ and $C'^\perp$ is a $[\frac{q^m-1}{q-1}, \frac{q^m-1}{q-1} - m, 3]$-code over $\mathbb{F}_q$. Then a parity check matrix of $C'^\perp$ is an $m \times \frac{q^m-1}{q-1}$ matrix over $\mathbb{F}_q$ such that every 2 rows are linearly independent, hence it must be a parity check matrix of $H(q,m)$ too. So $C'^{\perp}$ is nothing but the Hamming code $H(q,m)$.

If $q=2$, then $C=C'$ and $C^{\perp}=C'^{\perp}$, both are cyclic codes.

If $q>2$, then C' and $C'^{\perp}$ may not be cyclic. For C', let $n'=\frac{q^m-1}{q-1}$, note that C is cyclic, so $(c_0,\cdots,c_{n-1})\in C \Rightarrow (c_{n-1},c_0,\cdots,c_{n-2})\in C$. Hence $(c'_0,\cdots,c_{n'-1})\in C' \Rightarrow (\gamma^{-n'},c'_0,\cdots,c'_{n'-2})\in C'$.

Example 8.4. Suppose $\gcd(m,q-1)=1$ and let $n=\frac{q^m-1}{q-1}$. Let α be a root of a primitive polynomial of degree m over $\mathbb{F}_q$ and $\beta=\alpha^{q-1}$. Then β if of order n and $\beta,\beta^q,\cdots,\beta^{q^{m-1}}$ are all distinct.

The polynomial

$$g(x)=\prod_{i=0}^{m-1}(x-\beta^{q^i})$$

is a factor of x^n-1, and invariant under the Frobenius map $\sigma_q: a\mapsto a^q$ and hence belongs to $\mathbb{F}_q[x]$. Since $\gcd(n,q-1)=\gcd(m,q-1)=1$, there exists t such that $(q-1)t\equiv 1 \bmod n$. Then $(\alpha/\beta^t)^{q-1}=1$ and $\alpha=\lambda\beta^t$ for some $\lambda\in\mathbb{F}_q^{\times}$. This implies that $\mathbb{F}_q(\alpha)=\mathbb{F}_q(\beta)=\mathbb{F}_{q^m}$ and $g(x)$ is irreducible.

Let C be the cyclic code of length n generated by $g(x)$, then $[n,k]=[\frac{q^m-1}{q-1},\frac{q^m-1}{q-1}-m]$. Moreover, note that the only elements in $R_n=\mathbb{F}_q[x]/(x^n-1)$ of weight at most 2 are of the form c_ix^i and $c_ix^i+c_jx^j$.

- $c_ix^i\in C \Leftrightarrow c_i\beta^i=0 \Leftrightarrow c_i=0$, i.e., $c_ix^i=0$.
- $c_ix^i+c_jx^j\in C \Leftrightarrow \beta^{i-j}\in\mathbb{F}_q^{\times}$. Assume $i>j$, then $1\le s:=i-j<\frac{q^m-1}{q-1}$. Hence $\beta^s\in\mathbb{F}_q^{\times} \Leftrightarrow \beta^{s(q-1)}=1 \Leftrightarrow n\mid(q-1)s$. Since $\gcd(n,q-1)=1$, the equivalence is that $n\mid s$, which is impossible.

In conclusion, all nonzero codewords in C are of weight at least 3, that is, $d(C)\ge 3$. By Hamming bound we have $d=d(C)=3$. Consider the parity check matrix of C like the previous example, we have $C=H(q,m)$. Thus if m is coprime to $q-1$, the Hamming code $H(q,m)$ is a cyclic code.

3. The BCH codes

We fix an algebraic closure $\overline{\mathbb{F}}_p=\bigcup_{n\ge1}\mathbb{F}_{p^n}$ of the prime field $\mathbb{F}_p$.

3.1. Roots of cyclic codes.

Definition 8.4. The roots of a cyclic code C are the roots of its generator polynomial in $\overline{\mathbb{F}}_p$.

Assume $(n,q)=1$, $g(x)h(x)=x^n-1$, $\deg h=k$, and C is the cyclic code of length n with $g(x)$ the generator polynomial. Then the set of roots of C is

$$R:=\{\gamma\in\overline{\mathbb{F}}_p \mid g(\gamma)=0\}.$$

Then
(8.1)
$C = \{a(x)g(x) \mid \deg a < k\} = \{c(x) \in \mathbb{F}_q[x] \mid \deg c < n, c(\gamma) = 0 \text{ for } \gamma \in R\}.$
Moreover, note that $g(x)$ has no multiple root, if $g(x) = p_1(x) \cdots p_g(x)$ with p_i distinct, monic and irreducible. Let $\gamma_i \in \overline{\mathbb{F}}_q$ be a root of $p_i(x) = 0$. Then

$$C = \{c(x) \in \mathbb{F}_q[x] \mid \deg c < n,\ c(\gamma_i) = 0 \text{ for all } 1 \leq i \leq g\}. \tag{8.2}$$

Let

$$\widetilde{H} := (\tilde{h}_{ij}),\ \tilde{h}_{ij} = \gamma_i^{j-1} \in \mathbb{F}_q(\gamma_1, \cdots, \gamma_g),\ 1 \leq i \leq g,\ 1 \leq j \leq n. \tag{8.3}$$

Then we have

$$c = (c_0, \cdots, c_{n-1}) \in C \ \Leftrightarrow\ c\widetilde{H}^T = 0. \tag{8.4}$$

In this sense, $\widetilde{H}$ is also a parity check matrix for C, which has a much smaller row size g than the row size $n-k$ of a usual parity check matrix, but the coefficients are in a bigger field.

One can also recover the usual parity check matrix from $\widetilde{H}$. Suppose $\deg(p_i) = d_i$, then

$$\mathbb{F}_q(\gamma_i) = \mathbb{F}_{q^{d_i}} \quad \text{and} \quad \sum_{i=1}^{g} d_i = n-k.$$

Let $\{v_1^{(i)}, \cdots, v_{d_i}^{(i)}\}$ be a basis of $\mathbb{F}_q(\gamma_i)$ over $\mathbb{F}_q$, then

$$\gamma_i^j = \sum_{k=1}^{d_i} a_{k,j}^{(i)} v_k^{(i)},\ \text{where } a_{k,j}^{(i)} \in \mathbb{F}_q,$$

and

$$c(\gamma_i) = \sum_{k=1}^{d_i} (\sum_{j=0}^{n-1} c_i a_{k,j}^{(i)}) v_k^{(i)}.$$

Let

$$H_i = (a_{k,j}^{(i)}) \in \mathbb{F}_q^{d_i \times n},\ \ 1 \leq k \leq d_i,\ 0 \leq j \leq n-1.$$

Then $c(\gamma_i) = 0 \Leftrightarrow c^T H_i = 0$. Let

$$H := \begin{pmatrix} H_1 \\ \vdots \\ H_g \end{pmatrix}.$$

Then $H \in \mathbb{F}_q^{(n-k)\times n}$ and $c(x) \in C$ if and only if $c^T H = 0$, hence H is a parity check matrix for C in the usual sense.

Example 8.5. We set $q = 3$ and $n = 13$, $g(x) = (x-1)(x^3+x^2+x+2)$. The code C is a $[13, 9]$-code over $\mathbb{F}_3$. Let γ be a root of x^3+x^2+x+2, then $\{1, \gamma, \gamma^2\}$ is an $\mathbb{F}_3$-basis of $\mathbb{F}_3(\gamma) = \mathbb{F}_{27}$ over $\mathbb{F}_3$, and:

$$\widetilde{H} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \gamma & \cdots & \gamma^{12} \end{pmatrix}.$$

Under the basis $\{1, \gamma, \gamma^2\}$, the coordinates of γ^j can be determined, hence we can write H as follows:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 1 \\ 0 & 1 & 0 & 2 & 2 & 2 & 1 & 2 & 2 & 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 1 & 1 \end{pmatrix}.$$

3.2. BCH Codes. The Bose-Chaudhuri-Hocquenghen code, or BCH code in short, is a special type of cyclic code constructed by R. C. Bose-D. K. Ray-Chaudhuri in 1960 and A. Hocquenghen in 1959 independently.

Definition 8.5. Suppose $(n, q) = 1$, β is an element in $\overline{\mathbb{F}}_p$ of order n, ℓ and δ are positive integers and $2 \le \delta \le n - 1$. Then the cyclic code

$$C = \{c(x) = \sum_{i=0}^{n-1} c_i x^i \in \mathbb{F}_q[x] \mid c(\beta^i) = 0, \ell \le i \le \ell + \delta - 2\}$$

is called a BCH code of design distance δ.

By definition, the generator polynomial of C is the least common multiple of the minimal polynomials of β^i for $\ell \le i \le \ell + \delta - 2$.

Theorem 8.3. *The minimum distance d of a BCH code is at least the design distance δ of it.*

Proof. Let

$$H = (h_{ij})_{(\delta-1)\times n},\ h_{ij} = \beta^{(\ell+i-1)(j-1)} \in \mathbb{F}_q(\beta),\ 1 \le i \le \delta - 1,\ 1 \le j \le n.$$

Then for $c(x) = \sum_{i=0}^{n-1} c_i x^i \in \mathbb{F}_q[x]$, $c(x) \in C$ if and only $cH^T = 0$. To show $d \ge \delta$ or equivalently $\mathrm{wt}(c) \ge \delta$ for any nonzero codeword c, it suffices to show any $\delta - 1$ columns of H are $\mathbb{F}_q$-linear independent.

Let H' be the square matrix of size $\delta - 1$ formed by some arbitrarily chosen $\delta - 1$ columns of H. Then $\det(H')$ is a non-zero multiple of a Vandermonde determinant, hence $\det(H') \ne 0$ and H' is invertible. Then the columns of H' are actually $\mathbb{F}_q(\beta)$-linear independent. $\square$

3.3. An example. Suppose $q = 2$, $m \ge 3$ and α is a primitive element of $\mathbb{F}_{2^m}$. Then $n = 2^m - 1$ is the order of α. Let C be a binary cyclic code of length n with roots α and α^3. Then

$$c(\alpha) = 0 \Rightarrow c(\alpha^2) = c(\alpha)^2 = 0 \Rightarrow c(\alpha^4) = 0.$$

Hence $\alpha, \alpha^2, \alpha^3, \alpha^4$ are roots of C, and C is a BCH code of design distance $\delta = 5$, which implies that $d(C) \ge 5$.

Let $g_1(x)$ and $g_3(x)$ be the minimal polynomial of α and α^3 respectively. Then $g_1(x)$ is primitive of degree m with root set $\{\alpha^{2^i} \mid 0 \le i \le m-1\}$, and $g_3(x)$ has roots $\alpha^{3\cdot 2^i}$ for $0 \le i \le m - 1$. If $m \ge 3$, these roots are all distinct: in fact if $m \ge 3$, $3 \times 2^\ell = 3 \mod 2^m - 1$ and ℓ minimal, then $\ell = m$. Hence $g_3(x)$ is irreducible of degree m, and the generator polynomial

$g(x) = g_1(x)g_3(x)$ of C is of degree $2m$. Thus C is $[2^m - 1, 2^m - 1 - 2m, d]_2$-code with $d \geq 5$.

We now try to decode this code to correct at most 2 errors.

Problem 8.1. Suppose $c(x) = \sum_{i=0}^{n-1} c_i x^i \in C$ was sent and $u(x) = c(x) + \varepsilon(x) = \sum_{i=0}^{n-1} u_i x^i$ is received. Suppose the error $\varepsilon(x) = \sum_{i=0}^{n-1} \varepsilon_i x^i$ has weight $\mathrm{wt}(\varepsilon) \leq 2$. Recover c from u.

We know $c \in C$ if and only if $cH^T = 0$, where the parity check matrix

$$H = \begin{pmatrix} 1 & \alpha & \cdots & \alpha^{n-1} \\ 1 & \alpha^3 & \cdots & \alpha^{3(n-1)} \end{pmatrix}.$$

The syndrome of $u = (u_0, \cdots, u_{n-1})$ is

$$S_H(u) = uH^T = (u(\alpha), u(\alpha^3)) = (\varepsilon(\alpha), \varepsilon(\alpha^3)).$$

Let $A_1 = u(\alpha)$ and $A_3 = u(\alpha^3)$.

(1) If $\varepsilon(x) = 0$, then $A_1 = A_3 = 0$;
(2) If $\varepsilon(x) = x^i$ for some $0 \leq i \leq n-1$, then $A_1^3 = A_3 \neq 0$;
(3) If $\varepsilon(x) = x^i + x^j$ for some $0 \leq i \neq j \leq n-1$, then $A_1 = \alpha^i + \alpha^j \neq 0$ and $A_3 = \alpha^{3i} + \alpha^{3j} \neq 0$. Also we have $A_1^3 - A_3 + \alpha^{i+j}(\alpha^i + \alpha^j) \neq A_3$. Let $x_1 = \alpha^i$, $x_2 = \alpha^j$. Then

$$x_1 + x_2 = A_1, \ x_1 x_2 = A_1^2 + \frac{A_2}{A_1},$$

and

$$\sigma(z) = (1 - x_1 z)(1 - x_2 z) = 1 + A_1 z + \frac{A_2 + A_1^3}{A_1} z^2 \in \mathbb{F}_{2^m}[z].$$

This leads to the following algorithm:

Algerithm 8.1. *We decode C as follows:*

(1) *Compute $A_1 = u(\alpha)$ and $A_3 = u(\alpha)^3$.*
(2) *If $A_1 = A_3 = 0$, then $c(x) = u(x)$.*
(3) *If $A_1^3 = A_3 \neq 0$, then write $A_1 = \alpha^i$, $0 \leq i \leq n-1$, we have $\varepsilon(x) = x^i$ and $c(x) = u(x) + x^i$.*
(4) *If $A_1 \neq 0$, $A_3 \neq 0$ and $A_1^3 \neq A_3$, then the equation*

$$\sigma(z) = 1 + A_1 z + \frac{A_2 + A_1^3}{A_1} z^2 \in \mathbb{F}_{2^m}[z]$$

has 2 roots α^{-i} and α^{-j} in $\mathbb{F}_{2^m}$. Then $\varepsilon(x) = x^i + x^j$ and $c(x) = u(x) + \varepsilon(x)$.

3.4. Decoding algorithm for BCH codes. Let C be a BCH code of length n over $\mathbb{F}_q$ whose design distance $\delta = 2t + 1$. Suppose $\mathbb{F}_q(\beta) = \mathbb{F}_{q^m}$ and $\beta^i \mid (1 \leq i \leq 2t)$ are roots of C. Let $c = \sum_{i=0}^{n-1} c_i x^i \in C$ be the codeword sent, $v(x) = c(x) + \varepsilon(x) = \sum_{i=0}^{n-1} v_i x^i$ be the polynomial received, and $\varepsilon(x) = \sum_{i=0}^{n-1} \varepsilon_i x^i$ be the error polynomial whose weight $\mathrm{wt}(\varepsilon) \leq t$.

Definition 8.6. Let ε be the error. Then the error set M, the error locator polynomial $\sigma(x)$ and the error evaluator polynomial $\omega(x)$ are defined as follows:

$$M := \{i \mid \varepsilon_i \neq 0\}, \text{ whose order } |M| := \ell \leq t; \tag{8.5}$$

$$\sigma(x) := \prod_{i\in M}(1-\beta^i x) \in \mathbb{F}_{q^m}[x]; \tag{8.6}$$

$$\omega(x) := \sigma(x)\cdot \sum_{i\in M}\frac{\varepsilon_i\beta^i x}{1-\beta^i x}. \tag{8.7}$$

The aim of decoding is to recover c or ε from v. Then a decoding algorithm is to determine M and the exact value ε_i for each $i \in M$. Note that $i \in M$ if and only if $\sigma(\beta^{-i}) = 0$. Note also that if we can find $\sigma(x)$ and $\omega(x)$, then $\varepsilon_i = \frac{\omega(\beta^{-i})\beta^i}{\sigma(\beta^{-i})}$. Hence to decode C, it suffices to determine $\sigma(x)$ (and its roots) and $\omega(x)$.

For $\lambda \geq 1$, set

$$s_\lambda := \varepsilon(\beta^\lambda).$$

If $1 \leq \lambda \leq 2t$, then $c(\beta^\lambda) = 0$, and $s_\lambda = \varepsilon(\beta^\lambda) = v(\beta^\lambda) \in \mathbb{F}_{q^m}$ is known. Let $a(x) := \sum_{\lambda=0}^{t-1} s_{\lambda+1}x^\lambda$. Then

$$\frac{\omega(x)}{\sigma(x)} = \sum_{i\in M}\frac{\varepsilon_i\beta^i x}{1-\beta^i x} = \sum_{i\in M}\varepsilon_i\sum_{\lambda=1}^{\infty}(\beta^i x)^\lambda = \sum_{\lambda=1}^{\infty}s_\lambda x^\lambda.$$

Note that $\sigma(0) = 1$, $\deg(\sigma(x)) = \ell \leq t$, $\omega(0) = 0$ and $\deg(\omega(x)) \leq \ell$. Note also that $\gcd(\sigma(x), \omega(x)) = 1$. Let $\omega(x) = zb(x)$. Then $\frac{b(x}{\sigma(x)}$ is a proper fraction in reduced form and

$$\frac{b(x)}{\sigma(x)} = \sum_{\lambda=0}^{\infty}s_{\lambda+1}x^\lambda \equiv a(x) \bmod x^{2t}.$$

Thus the linear feedback shift generator $\mathrm{LSR}(\sigma(x), \ell)$ generates the sequence $(s_1, \cdots, s_{2t})$.

On the other hand, by the Berlekamp-Massey Algorithm we can find the minimal integer ℓ_{2t} and a connecting polynomial f such that $\mathrm{LSR}(f(x), \ell_{2t})$ generates the sequence $(s_1, \cdots, s_{2t})$. Since $\ell_{2t} \leq \ell \leq \frac{2t}{2}$, $\sigma(x) = f(x)$ must be unique by Theorem 7.7. Once $\sigma(x)$ is found, $b(x)$ is nothing but the sum of degree $< \ell$ terms of the polynomial $\sigma(x)a(x)$ and $\omega(x) = xb(x)$.

The above analysis leads to the following algorithm.

Algerithm 8.2. *Given $v(x)$.*

1. *Compute $s_\lambda = v(\beta^\lambda)$ for $1 \leq \lambda \leq 2t$.*
2. *Apply the Berlekamp-Massey Algorithm and find the shortest* $\mathrm{LSR}(\sigma(x), \ell)$ *generating $a(x) = \sum_{\lambda=0}^{2t-1} s_{\lambda+1}x^\lambda$, let $b(x) = (\sigma(x)a(x))_{\deg<\ell}$ and $\omega(x) = xb(x)$.*
3. *Find all roots of $\sigma(x)$, hence obtain M.*

(4) *Compute* $\varepsilon_i = \frac{\omega(\beta^{-i})\beta^i}{\sigma(\beta^{-i})}$, $\varepsilon(x)$ *and* $c(x)$.

Remark 8.1. Two key points for the effectiveness of this algorithm: (1) B-M Algorithm and (2) finding roots of $\sigma(z)$ in $\mathbb{F}_{q^m}$.

3.5. More examples of BCH Codes. We first show that the Reed-Solomon Code $\mathrm{RS}(q-1, q-d)$ is a BCH Code.

Theorem 8.4. *Let* $n = q-1$ *and* α *be a primitive element in* $\mathbb{F}_q$*, then the BCH code* C *with roots* $\alpha, \cdots, \alpha^{d-1}$ *has parameters* $[q-1, q-d, d]$ *and is isomorphic to the Reed-Solomon code* $\mathrm{RS}(q-1, q-d) = \{c_f = (f(1), \cdots, f(\alpha^{q-2})) \mid f(x) \in \mathbb{F}_q[x], \deg f < q-d\}$.

Proof. The generating polynomial of C is $g(x) = \prod_{i=0}^{d-1}(x-\alpha^i)$. So $k = n - \deg g = q - d$ and $d(C) \geq \delta(C) = d$. By Singleton bound we have $d(C) = d$.

Now $\mathrm{RS}(q-1, q-d)$ has dimension $k = q-d$, length $n = q-1$, and a generator matrix $G \in \mathbb{F}_q^{k\times n}$ whose (i,j)-th entry is $\alpha^{(i-1)(j-1)}$. Let

$$H = \begin{pmatrix} 1 & \alpha & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha^{n-k} & \cdots & \alpha^{(n-k)(n-1)} \end{pmatrix}.$$

Then one can easily check that $GH^T = 0$ and $\mathrm{rank}(H) = n-k = d-1$. So H is a parity check matrix of $\mathrm{RS}(q-1, q-d)$. This means that $\mathrm{RS}(q-1, q-d)$ is a cyclic code with roots $\alpha, \cdots, \alpha^{d-1}$, i.e., $\mathrm{RS}(q-1, q-d) = C$. $\square$

Example 8.6. Let $\alpha \in \overline{\mathbb{F}}_2$ satisfying $\alpha^{23} = 1$. Then $\mathbb{F}_2(\alpha) = \mathbb{F}_{2^11}$ since $2^11 = 1 \mod 23$. Let $g_1(x)$ be the minimal polynomial of α, then $\deg(g_1) = 11$, and the roots of $g_1(x)$ are $\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^9, \alpha^{18}, \alpha^{13}, \alpha^3, \alpha^6, \alpha^{12}$. The minimal polynomial $g_2(x)$ of $\alpha^{-1} = \alpha^{22}$ is the reflexive polynomial $g_1^*(x)$ of $g(x)$, which is also of degree 11, and

$$x^{23} - 1 = g_1(x)g_2(x)(x-1), \quad g_1(x)g_2(x) = \sum_{i=0}^{22} x^i.$$

Let C be the cyclic code of length 23 whose generator polynomial is $g_1(x)$. Then $k = 23 - 11 = 12$. Since α^i $(1 \leq i \leq 4)$ are roots of C, the design distance $\delta = 5$ and hence $d \geq 5$.

Let $c(x) \in \mathbb{F}_2[x]$ be a codeword in C, then its reflexive polynomial $c^*(x) = \sum_{i=0}^{23} c_i x^{23-i}$ and

$$c(x)c^*(x) = \sum_{i,j=0}^{23} c_i c_j x^{i-j+23} \equiv \sum_{i=0}^{23} c_i + \sum_{0\leq i<j\leq 23} c_i c_j (x^{23+i-j}) + x^{j-i}) \mod x^{23} - 1.$$

Since $g_1(x) \mid c(x)$ and $g_2(x) \mid c^*(x)$, we have $g_1(x)g_2(x) \mid c(x)c^*(x)$. Thus

$$A(x) := \sum_{i=0}^{23} c_i + \sum_{0\le i<j\le 23} c_i c_j (x^{23+i-j} + x^{j-i}) = 0 \text{ or } g_1(x)g_2(x) \in \mathbb{F}_2[x].$$

Suppose $c \neq 0$ and $\mathrm{wt}(c) = \ell$. Then $\#\{i \mid c_i = 1\} = \ell$ and note that $c_i c_j = 1 \Leftrightarrow c_i = c_j = 1$, $\#\{(i,j) \mid i < j,\ c_i c_j = 1\} = \ell(\ell-1)/2$.

If ℓ is even, the constant term of $A(x)$ is $\ell = 0 \in \mathbb{F}_2$, so $A(x) = 0$. Note that $23 + i - j$ and $j - i$ have different parity, to get $A(x) = 0$, $\ell(\ell-1)/2$ must be even, hence $4 \mid \ell$ and in particular $\ell \neq 6$.

If $\ell = 5$, the constant term of $A(x)$ is 1 and hence $A(x) = \sum_{i=0}^{22} x^i$. However, there are at most $5^2 - 5 = 20$ nonzero terms in $A(x) - 1$, but there are 22 nonzero terms in $x + \cdots + x^{22}$, so they cannot be the same polynomial. So $d \geq 7$.

By Hamming bound we have $d = 7$, thus C is $[23, 12, 7]$-code over $\mathbb{F}_2$. Actually, C is the binary Golay code G_{23}.

Example 8.7. Let $\alpha \in \overline{\mathbb{F}}_3$ such that $\alpha^{11} = 1$. Then $\mathbb{F}_3(\alpha) = \mathbb{F}_{3^5}$. Its minimal polynomial $g_1(x)$ has roots $\alpha, \alpha^3, \alpha^9, \alpha^5, \alpha^4$, and $\deg(g_1) = 5$. The minimal polynomial $g_2(x)$ of α^{-1} is also of degree 5, and

$$x^{11} - 1 = (x-1)g_1(x)g_2(x).$$

Let C be the cyclic code of length 11 with the generator polynomial $g_1(x)$. Then $[n, k] = [11, 6]$. Moreover, C is a BCH code with roots $\alpha^3, \alpha^4, \alpha^5$, so $d(C) \geq \delta = 4$.

If $c(x) = a_1 x^{n_1} + \cdots + a_4 x^{n_4} \in C$ is of weight 4, then $c(x)c^*(x) = a(1 + \cdots + x^{10}) \mod x^{11} - 1$ for some $a \in \mathbb{F}_3$. Note the constant term of $c(x)c^*(x)$ is $1 + 1 + 1 + 1 = 1$, so $a = 1$. Hence $(a_1 + a_2 + a_3 + a_4)^2 = 11 = 2 \in \mathbb{F}_3$, which is not possible. So $d(C) \geq 5$.

By Hamming Bound, then $d(C) = 5$ and C is a $[11, 6, 5]$-code over $\mathbb{F}_3$. Actually C is the ternary Golay code G_{11}.

4. Goppa Code

Recall the G-V bound: $B_q(n,d) \geq \frac{q^{n-1}}{V_q^{n-1}(d-1)}$.

Definition 8.7. Let $r = 1 - q^{-1}$, define $H_q(x)$ as follows:

1. $H_q(0) = 0$;
2. for $0 < x \leq r$, $H_q(x) = x \log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x)$.

We know that

Lemma 8.3. $\lim\limits_{\delta \to r} H_q(\delta) = 1$.

Lemma 8.4. *For $0 \leq \delta \leq r$, we have*

$$\lim_{n\to+\infty} \log_q(V_q^n([\delta n])) = H_q(\delta)$$

Proof. If $n \to +\infty$, then $V_q^n([\delta n])$ is dominated by the last term, we have

$$\binom{n}{[\delta n]}(q-1)^{[\delta n]} \le V_q^n([\delta n]) \le (1+[\delta n])\binom{n}{[\delta n]}.$$

Then apply Sterling's formula and we get the result. □

We take another look at BCH code: Suppse $(n,q)=1$, β is of order n and $\mathbb{F}_q(\beta)=\mathbb{F}_{q^m}$. Suppose C is the BCH code of length n with roots $\{\beta^j \mid 1 \le j \le d-1\}$ (hence the design distance $\delta = d$). We know that

$$c(x) \in C \;\Leftrightarrow\; c(\beta^j)=0 \text{ for all } 1 \le j \le d-1.$$

For $c_i \in \mathbb{F}_q$ $(0 \le i \le n-1)$, we have

$$\begin{aligned}(z^n-1)\sum_{i=0}^{n-1}\frac{c_i}{z-\beta^{-i}} &= \sum_{i=0}^{n-1} c_i \frac{(z^n-1)}{z-\beta^{-i}} \\ &= \sum_{i=0}^{n-1} c_i \sum_{k=0}^{n-1} (\beta^{-i})^{n-1-k} z^k \\ &= \sum_{k=0}^{n-1} z^k \sum_{i=0}^{n-1} c_i \beta^{i(k+1)}.\end{aligned}$$

Hence

$$(c_0,\cdots,c_{n-1}) \in C \;\Leftrightarrow\; \sum_{i=0}^{n-1}\frac{c_i}{z-\beta^{-i}} = 0 \mod z^{d-1}.$$

Goppa code is obtained by replacing z^{d-1} by $g(z) \in \mathbb{F}_{q^m}[z]$ and β^j $(1 \le j \le n-1)$ by other elements.

Definition 8.8. Let $g(z) \in \mathbb{F}_{q^m}[z]$ be a monic polynomial of degree t. Let $L = \{\gamma_i \mid 0 \le i \le n-1\} \subset \mathbb{F}_{q^m}$ such that $g(\gamma) \ne 0$ for $\gamma \in L$. Then the Goppa code is the code

$$G(L,g) = \{c=(c_0,\cdots,c_{n-1}) \in \mathbb{F}_q^n \mid \sum_{i=0}^{n-1}\frac{c_i}{z-\gamma_i} = 0 \mod g(z)\}.$$

Theorem 8.5. *For the code $G(L,g)$, $k \ge n-mt$ and $d \ge t+1$.*

Proof. Write $g(z)=\sum_{j=0}^t g_j z^j$ and $g_t = 1$. Then

$$\tilde{g}(z,x) := \frac{g(z)-g(x)}{z-x} = \sum_{\substack{j,k\ge 0\\ j+k \le t-1}} g_{j+k+1} x^k z^j.$$

For $\gamma \in \mathbb{F}_{q^m}$ such that $g(\gamma) \ne 0$,

$$\frac{1}{z-\gamma} = \frac{1}{g(\gamma)}\tilde{g}(z,\gamma) \mod g(z).$$

Write $h_i = g(\gamma_i)^{-1}$. Then

$$c \in G(L,g) \Leftrightarrow \sum_{i=0}^{n-1} c_i h_i \tilde{g}(z,\gamma_i) \equiv 0 \mod g(z).$$

Since $\deg \tilde{g}(z,\gamma_i) = t-1$ for every i, we must have

$$c \in G(L,g) \Leftrightarrow \sum_{i=0}^{n-1} c_i h_i \tilde{g}(z,\gamma_i) = 0.$$

This means that

$$c \in G(L,g) \Leftrightarrow \ cH^T = 0,$$

where H is the $t \times n$ matrix over $\mathbb{F}_{q^m}$ given by

$$H = \begin{pmatrix} h_0 g_t & \cdots & h_{n-1} g_t \\ h_0(g_{t-1} + g_t\gamma_0) & \cdots & h_{n-1}(g_{t-1} + g_t\gamma_{n-1}) \\ \vdots & & \vdots \\ h_0(g_1 + g_2\gamma_0 + \cdots + g_t\gamma_0^{t-1}) & \cdots & h_{n-1}(g_1 + g_2\gamma_{n-1} + \cdots + g_t\gamma_{n-1}^{t-1}) \end{pmatrix}.$$

By elementary row transformations, we obtain a new matrix H' from H:

$$H' = \begin{pmatrix} h_0 & \cdots & h_{n-1} \\ h_0\gamma_0 & \cdots & h_{n-1}\gamma_{n-1} \\ \vdots & & \vdots \\ h_0\gamma_0^{t-1} & \cdots & h_{n-1}\gamma_{n-1}^{t-1} \end{pmatrix}.$$

Note that any t columns of H' form an invertible matrix, then $\operatorname{rank}(H') = t$.

Let $\{e_1, \cdots, e_m\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then we can write

$$H' = H_1 e_1 + \cdots + H_m e_m$$

where $H_1, \cdots, H_m$ are $t \times n$ matrices over $\mathbb{F}_q$. Let

$$\widetilde{H} = \begin{pmatrix} H_1 \\ \vdots \\ H_m \end{pmatrix},$$

Then $\widetilde{H}$ is an $mt \times n$ matrix over $\mathbb{F}_q$ and

$$c \in G(L,g) \Leftrightarrow \ c\widetilde{H}^T = 0.$$

Thus $k = \dim_{F_q} G(L,g) = n - \operatorname{rank}(\widetilde{H}) \geq n - mt$. By the fact $\operatorname{rank}(H') = t$, then any t columns of $\widetilde{H}$ are linearly independent, so $d \geq t+1$. $\square$

Take $m = 1$ in the above Theorem, then $k \geq n - t$ and $d \geq t+1$, so $k + d \geq n+1$. But by the Singleton bound, $k + d \leq n+1$. So $k = n - t$ and $d = t+1$.

THEOREM 8.6. *Suppose* $0 \leq t \leq n-1$. *Let* $g \in \mathbb{F}_q[z]$ *be monic of* $\deg g = t$, *and* $L = \{\gamma_i \mid 0 \leq i \leq n-1\} \subset \mathbb{F}_q$ *such that* $g(\gamma_i) \neq 0$. *Then* $G(L,g)$ *has parameters* $[n, n-t, t+1]$, *which is an MDS code.*

For $q = 2$, one has a stronger result about the distance:

THEOREM 8.7. *If $q = 2$ and $g(z)$ has no multiple root, then $d \geq 2t + 1$.*

PROOF. For $c = (c_0, \cdots, c_{n-1}) \neq 0$, write $f(z) = \prod_{i=0}^{n-1}(z - \gamma_i)^{c_i}$, where $c_i = 0$ or 1. Then $\frac{f'(z)}{f(z)} = \sum_{i=0}^{n-1} \frac{c_i}{z-\gamma_i}$, so $c \in G(L, g) \Leftrightarrow \frac{f'(z)}{f(z)} = 0 \mod g(z)$. Since f has no multiple root, we have $\gcd(f, f') = 1$ and $g(z) \mid f'(z)$.

For any $f(z) \in \mathbb{F}_{2^m}[z]$, $f'(z) = h(z^2) = h(z)^2$. So $g(z) \mid f'(z) \Leftrightarrow g(z) \mid h(z)^2$. Since $g(z)$ has no multiple roots, this is equivalent to $g(z) \mid h(z)$. Hence $c \in G(L, g)$ if and only if $g(z)^2 \mid f'(z)$. This means

$$\mathrm{wt}(c) = \deg f = \deg f' + 1 \geq 2 \deg g + 1 = 2t + 1.$$

Hence $d \geq 2t + 1$. □

CHAPTER 9

Generalized GRS codes

1. Generalized GRS codes

1.1. Basic properties.

Definition 9.1. Let $\alpha_1, \cdots, \alpha_n \in \mathbb{F}_q^\times$ be distinct and $v_1, \cdots, v_n \in \mathbb{F}_q^\times$ not necessarily distinct.

A Generalized Reed-Solomon code (GRS code) is a linear code defined by the following parity check matrix:

$$H_{\mathrm{GRS}} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \cdots & v_n\alpha_n \\ \vdots & \vdots & & \vdots \\ v_1\alpha_1^{n-k-1} & v_2\alpha_2^{n-k-1} & \cdots & v_n\alpha_n^{n-k-1} \end{pmatrix}$$

where $\alpha_1, \cdots, \alpha_n$ are called the code locators and $v_1, \cdots, v_n \in \mathbb{F}_q^*$ are called the column multipliers.

Lemma 9.1. *The Vandermonde determinant*

$$\Delta(\alpha_1, \cdots, \alpha_n) := \det \begin{pmatrix} 1 & 2 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{pmatrix} = \prod_{1 \le i < j \le n} (\alpha_j - \alpha_i)$$

which is not zero if α_i are pairwise distinct.

Proposition 9.1. *H_{GRS} is a legal parity check matrix, and a GRS code is an MDS code, i.e. $d = n - k + 1$.*

Proof. Let $H[n-k]$ be the first $n-k$ columns of H_{GRS}. Then

$$\det(H[n-k]) = v_1 v \cdots v_{n-k} \Delta(\alpha_1, \cdots, \alpha_{n-k}) \neq 0.$$

Hence $n-k \geq \mathrm{rank} H_{\mathrm{GRS}} \geq \mathrm{rank}(H[n-k]) = n-k$ and $\mathrm{rank}(H_{\mathrm{GRS}}) = n-k$. This means: (i) H_{GRS} is legal and (ii) every $n-k$ columns of H_{GRS} are linearly independent. (ii) implies that $d \geq n-k+1$ and by Singleton bound the equality must hold. $\square$

Example 9.1. Here are some examples of GRS code:

(1) primitive GRS code: $n = q-1$, $\{\alpha_1, \cdots, \alpha_n\} = \mathbb{F}_q^\times$.
(2) normalized GRS: $v_1 = \cdots = v_n = 1$.
(3) narrow sense GRS: for all i, $\alpha_i = v_i$.

THEOREM 9.1. *Let C be a GRS code, then $C^{\perp}$ is also a GRS code which can be defined by the same code locators.*

PROOF. Let

$$G_{\mathrm{GRS}} = \begin{pmatrix} v_1' & v_2' & \cdots & v_n' \\ v_1'\alpha_1 & v_2'\alpha_2 & \cdots & v_n'\alpha_n \\ \vdots & \vdots & & \vdots \\ v_1'\alpha_1^{k-1} & v_2'\alpha_2^{k-1} & \cdots & v_n'\alpha_n^{k-1} \end{pmatrix}_{k\times n}.$$

Then the (i,ℓ)-th entry of $G_{\mathrm{GRS}}H_{\mathrm{GRS}}^T$ is

$$(G_{\mathrm{GRS}}H_{\mathrm{GRS}}^T)_{i,\ell} = \sum_{j=1}^{n} v_j' v_j \alpha_j^{\ell+i}.$$

So $G_{GRS}H_{GRS}^T = 0$ if and only if $(v_1',\cdots,v_n')A = 0$, where $A = (a_{ij})$ is an $n\times(n-1)$ matrix whose (i,j)-th entry is $a_{ij} = v_i\alpha_i^{j-1}$. By the same argument of computing $\mathrm{rank}(H_{\mathrm{GRS}})$, we know $\mathrm{rank}A = n-1$. Thus there exists a nonzero vector $v' = (v_1',\cdots,v_n')$ such that $v'A = 0$.

Moreover, let A_j be the matrix obtained by deleting the j-th row of A, then $\mathrm{rank}A_j = n-1$ for any $1 \le j \le n$. If $v_j' = 0$ for some j, then $(v_1',\cdots,v_{j-1}',v_{j+1}',\cdots,v_n')A_j = 0$ and $(v_1',\cdots,v_{j-1}',v_{j+1}',\cdots,v_n') = 0$. Hence $v' = 0$ not possible. Hence there exists a $v' = (v_1',\cdots,v_n') \in (\mathbb{F}_q^{\times})^n$ such that $v'A = 0$. This corresponding matrix G_{GRS} is a generator matrix of C and a parity check matrix of $C^{\perp}$. □

1.2. Polynomial Representation of GRS codes. Suppose the generalized Reed-Solomon Code C_{GRS} has a generator matrix

$$G_{\mathrm{GRS}} = \begin{pmatrix} v_1' & v_2' & \cdots & v_n' \\ v_1'\alpha_1 & v_2'\alpha_2 & \cdots & v_n'\alpha_n \\ \vdots & \vdots & & \vdots \\ v_1'\alpha_1^{k-1} & v_2'\alpha_2^{k-1} & \cdots & v_n'\alpha_n^{k-1} \end{pmatrix}.$$

Then we have a bijection

$$\mathbb{F}_q[x]_{<k} \to G_{GRS},\ f(x) \mapsto (v_1'f(\alpha_1),\cdots,v_n'f(\alpha_n)).$$

The Classical Reed-Solomon code is the case $v_1' = \cdots = v_n' = 1$.

2. Decoding GRS codes

Let G_{GRS} be an $[n,k,n-k+1]_q$ GRS code with the code locator $\alpha_i (1 \le i \le n)$ and the column multipliers v_i $(1 \le i \le n)$. Let $H = H_{\mathrm{GRS}} = (v_i\alpha_i^{j-1})$. Suppose $c = (c_1,\cdots,c_n)$ is the codeword sent, $r = (r_1,\cdots,r_n) = r = c+\varepsilon$ is the vector received and $\varepsilon = (\varepsilon_1,\cdots,\varepsilon_n)$ is error whose weight $\mathrm{wt}(\varepsilon) := \ell \le \frac{d-1}{2} = \frac{n-k}{2}$. Let $e := [\frac{d-1}{2}]$.

For $0 \le \lambda \ge d-2$, we let

$$s_\lambda = \sum_{j=1}^{n} r_j v_j \alpha_j^\lambda = \sum_{j=1}^{n} \varepsilon_j v_j \alpha_j^\lambda.$$

Then the syndrome vector and polynomial of r is

$$S_H(r) = rH^T = (s_0, \cdots, s_{d-2}), \quad S(x) := \sum_{\lambda=0}^{d-2} s_\lambda x^\lambda.$$

Definition 9.2. The error set M, the error locator polynomial $\sigma(x)$ and the error evaluator polynomial $\omega(x)$ are defined as follows:

$$M := \{j \mid \varepsilon_j \neq 0\}; \tag{9.1}$$

$$\sigma(x) := \prod_{j\in M}(1-\alpha_j x); \tag{9.2}$$

$$\omega(x) := \sum_{j\in M} \frac{\varepsilon_j v_j \sigma(x)}{1-\alpha_j x} = \sum_{j\in M} \varepsilon_j \alpha_j \prod_{k\in M\setminus\{j\}} (1-\alpha_k x). \tag{9.3}$$

Clearly $|M| = \mathrm{wt}(\varepsilon) = \ell \le e$. As shown in the decoding of BCH codes, the decoding for GRS codes is equivalent to finding $\sigma(x)$ and $\omega(x)$. We note that

- $(\omega(x), \sigma(x)) = 1$ since for all $j \in M$, $\omega(\alpha_j^{-1}) \neq 0$;
- $\sigma(0) = 1$ and $\deg \omega < \deg \sigma = \ell \le e$.

For $\lambda \ge 0$, we let

$$s_\lambda = \sum_{j=1}^{n} \varepsilon_j v_j \alpha_j^\lambda.$$

By definition, we have

$$\begin{aligned}\frac{\omega(x)}{\sigma(x)} &= \sum_{j\in M} \frac{\varepsilon_j v_j}{1-\alpha_j x} = \sum_{j\in M}\sum_{\lambda\ge 0} \varepsilon_j v_j (\alpha_j x)^\lambda \\ &= \sum_{\lambda\ge 0}(\sum_{j\in M} \varepsilon_j v_j \alpha_j^\lambda) x^\lambda \\ &= \sum_{\lambda \ge 0} s_\lambda x^\lambda \equiv s(x) \mod x^{d-1}.\end{aligned}$$

This means $\mathrm{LSR}(\sigma(x), \ell)$ generates the sequence $S(r) = (s_0, \cdots, s_{d-2})$. Moreover, since $2\ell \le d-1$, $\sigma(x)$ is the unique polynomial produced by the Berlekamp-Massey Algorithm and $\omega(x) = (\sigma(x)S(x))_{\deg < \ell}$. The algorithm is similar to the BCH case and we leave it as an exercise.

We now give another algorithm by applying the key congruent equation

$$\sigma(x)S(x) \equiv \omega(x) \mod x^{d-1}. \tag{9.4}$$

Suppose

$$0 \neq \lambda(x) = \sum_{m=0}^{e} \lambda_m x^m \text{ and } \gamma(x) = \sum_{m=0}^{e-1} \gamma_m x^m.$$

Then $\lambda(x)S(x) \equiv \gamma(x) \mod x^{d-1}$ if and only if $\{\lambda_m\}_{m=0}^{e}$ and $\{\gamma_m\}_{m=0}^{e-1}$ satisfy the linear equations

$$A(\lambda_0, \cdots, \lambda_e)^T = (\gamma_0, \cdots, \gamma_{e-1}, 0, \cdots, 0)^T \tag{9.5}$$

where $A = (a_{ij})$ is an $(d-1) \times (e+1)$ matrix with the (i,j)-entry $a_{ij} = s_{i-j}$ which is 0 if $i < j$. Write $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$ where A_1 is an $(e+1) \times (e+1)$ matrix, $\vec{\lambda} = (\lambda_1, \cdots, \lambda_e)^T$ and $\vec{\gamma} = (\gamma_0, \cdots, \gamma_{e-1})^T$. Then (9.5) becomes

$$\begin{cases} A_1\vec{\lambda} = \vec{\gamma}, \\ A_2\vec{\lambda} = 0. \end{cases} \tag{9.6}$$

The homogeneous linear equations $A_2\vec{\lambda} = 0$ has $d-e-2$ equations and $e+1$ variables, and $e+1 > d-e-2$, so it has a nonzero solution $\vec{\lambda}$. The identity $A_1\vec{\lambda} = \vec{\gamma}$ then gives $\vec{\gamma}$. We get polynomials $\lambda(x) \neq 0$ and $\gamma(x)$ satisfying

$$\deg(\gamma) < e, \ \deg(\gamma(x)) \leq e \text{ and } \lambda(x)S(x) \equiv \gamma(x) \mod x^{d-1}.$$

Certainly $\sigma(x)$ and $\omega(x)$ also satisfy the above conditions.

Theorem 9.2. *Suppose $\lambda(x)$ and $\gamma(x) \in \mathbb{F}_q[x]$ such that $\deg \gamma < e$ and $\deg \lambda \leq e$.*

(1) *$\lambda(x)S(x) = \gamma(x) \mod x^{d-1}$ if and only if there exist $c(x) \in \mathbb{F}_q[x]$ such that $\gamma(x) = c(x)w(x)$ and $\lambda(x) = c(x)\sigma(x)$;*
(2) *If $\lambda(x)S(x) = \gamma(x) \mod x^{d-1}$, $\lambda(x) \neq 0$ and $\lambda(x)$ is of the smallest possible degree, then $\gamma(x) = c\omega(x)$ and $\lambda(x) = c\sigma(x)$ where $c = \lambda(0) \in \mathbb{F}_q^\times$.*
(3) *If $\lambda(x)S(x) = \gamma(x) \mod x^{d-1}$, $\gcd(\lambda(x), \gamma(x)) = 1$ and $\lambda(0) = 1$, then $\gamma(x) = \omega(x)$ and $\lambda(x) = \sigma(x)$.*

Proof. (2) and (3) can be easily derived from (1), so we only prove (1). Note that $\lambda(x)\omega(x) \equiv \gamma(x)\sigma(x) \mod x^{d-1}$. The two polynomials on both sides of the equation are of degree $< 2e \leq d-1$, so they must be equal. However, $\gcd(\omega(x), \sigma(x)) = 1$, so $\gamma(x) = c(x)w(x)$ and $\lambda(x) = c(x)\sigma(x)$. □

Algorithm 9.1 (Peterson-Gorenstein-Zievler GRS decoding algorithm). *A GRS code can be decoded as follows:*

(1) *Compute $s_0, \cdots, s_{d-1}$ and $S(x)$.*
(2) *Solve the equation $A_2\vec{\lambda} = 0$ and find $\lambda(x) \neq 0$, then compute $\vec{\gamma} = A_1\vec{\lambda}$ and get $\gamma(x)$.*
(3) *Compute $\sigma(x) = \frac{\lambda(x)}{\gcd(\lambda(x),\gamma(x))}$ and $\omega(x) = (\sigma(x)S(x))_{\deg<\deg(\sigma)}$.*
(4) *Compute M by finding the roots of $\sigma(x)$.*
(5) *Compute $\varepsilon_j = -\frac{\omega(\alpha_j^{-1})}{v_j\sigma'(\alpha_j^{-1})}$ where σ' is the formal derive of σ.*

Part 2

Cryptography

CHAPTER 10

History and Basic knowledge about Cryptography

Cryptography is a method of storing and transmitting data in a particular form.

- At sender side, using an encryption algorithm, the message (plaintext) is converted into an unreadable form. The message in unreadable form is called as ciphertext.
- The ciphertext is sent to the receiver over the communication channel. Since the message is encrypted, the attackers can not read the message.
- At receiver side, Using a decryption algorithm, the message is again converted into the readable form. Then, receiver can read the message.

Cryptography techniques can be classified into *Symmetric Key Cryptography* and *Asymmetric Key Cryptography.*

1. Cryptography from early age

1.1. Caesar Cipher. The Caesar cipher, named after Julius Caesar(100BC-44BC) who apparently used it to communicate with his generals, is one of the earliest known and simplest ciphers. It is an example of a simple substitution cipher, which moves the alphabet table forward or backward by n positions. For example, with a shift of 3, the letters

$$A, B, C, D, E, F, G, H, \cdots$$

are replaced by

$$D, E, F, G, H, I, J, K, \cdots$$

respectively.

To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. For the Caesar cipher, the key is the number of characters to shift the cipher alphabet. Here is a quick example of the encryption and decryption steps involved with the Caesar cipher. The text we will encrypt is 'go to the east side of the village', with a shift key of 5. Then we get

- plaintext: go to the east side of the village
- ciphertext: lt yt ymj jfxy xnij tk ymj anqqflj

Obviously, if the encryption is a shift of n, the decryption is a shift of $-n$; if a different key is used, the cipher alphabet will be shifted a different amount.

The Caesar cipher offers essentially no security and is easy to break. For one thing, if you know the ciphertext was encrypted by the Caesar cipher, you can just try by brute force to shift the text in all possible ways (which is just the size of the alphabet, not a big number). Even if you don't know the way how the text was encrypted, you can still decrypt it by frequency analysis. However, during Caesar's time, not many people was educated.

Let us give a mathematical description of the Caesar cipher. First we translate all of our characters to numbers, a=0, b=1, c=2, ... , z=25. Now suppose k is the key. Then the encryption function of the Caesar cipher is $E_k(x) = x + k \mod 26$ and the decryption function is $D_k(x) = x - k \mod 26$.

Example 10.1 (ROT13). The widely known ROT13 encryption is simply a Caesar cipher with an offset of 13, for which the encryption and decryption are the same, i.e., $E(x) = D(x) \equiv x + 13 \mod 26$. For example, if the plaintext 'go to the east side of the village', then the ciphertext is 'tb gb gur rnfg fvqr bs gur ivyyntr'; if the plaintext 'tb gb gur rnfg fvqr bs gur ivyyntr', then the ciphertext is 'go to the east side of the village'.

ROT13 was originally devised to be used with newsgroup postings that contained offensive material so the more sensitive among us wouldn't be inadvertently exposed to them. The idea was that you had to take an action to decode the posting as a way of indicating that you understood that you might find the contents offensive. It was also used by the first generation web browser Netscape to store users' passwords by some strange unexplained reason.

1.2. Vigenère Cipher. The Vigenère Cipher is a polyalphabetic substitution cipher. The method was originally described by the Italian cryptologist Giovan Battista Bellaso (5105-?) in his 1553 book La cifra del. Sig. However, the scheme was later misattributed to the French diplomat and cryptographer Blaise de Vigenère (1523-1596) in the 19th century, and is now widely known as the Vigenère cipher. Blaise de Vigenère actually the inventor of the stronger Autokey cipher in 1586.

Vigenère Cipher is a variant of Caesar cipher. The key K is a word or phrase that is repeated as many times as required to generate the key stream K_i, then

- Encryption: $C_i \equiv P_i + K_i \mod 26$;
- Decryption: $P_i \equiv C_i - K_i \mod 26$.

Example 10.2. For example, suppose the key is "lemon". Then

- Plaintext: gototheeastsideofthevillage
- Keystream: lemonlemonlemonlemonlemonle
- Ciphertext: rsfcgsiqofewurrzjfvrgmxznri

Example 10.3. During the USA Civil war (1861-1865), the Vigenère Cipher used by the South army has three keys: Manchester Bluff; Complete victory; Come retribution. The North broke all three keys, partly the reason for its victory in the battlefield.

The Vigenère Cipher was known as le chiffre indéchiffrable (literally "the unbreakable cipher" in French) for 300 years, until in 1863 a former German army officer and cryptanalyst Friedrich Kasiski published a successful attack on the Vigenère cipher. Charles Babbage had, however, already developed the same test in 1854. Gilbert Vernam worked on the Vigenère cipher in the early 1900s, and his work eventually led to the one-time pad, which is a provably unbreakable cipher.

So how to break the Vigenère cipher? The key point is that the key is repeated, once we known the length of the key, then it is easy to break. Kasiski developed a way (Kasiski Examination) to find the length of the key. Let κ be the length of the key. If one sequence in the plaintext was repeated at the same position (after modulo κ), then it would appear in the ciphertext repeatedly. Therefore the positions apart of repeated sequences in the ciphertext are highly likely a multiple of κ. For example, suppose we have the ciphertext: dyduxrmntvdvnqdqnwdyduxrmhartjgwndq. The sequence "dyduxr" was repeated 20 letters apart in the ciphertext, so probably the length of the key is a factor of 20; the sequence "ndq" was repeated 18 letters apart, so probably the length of the key is a factor of 18. One then can try to exam if $\gcd(18, 20) = 2$ is really the length of the key.

William Friedman (1891-1969) then developed the index coincidence method (Friedman Test) to find the length of the key. Let k_p be the probability of which any two letters are equal in a language, k_r be the probability of which any two letters are equal in an alphabet table, let $k_0 = \frac{\sum_{i=1} n_i(n_i-1)}{N(N-1)}$ where N is the length of ciphertext and n_i is the number of letter i in the ciphertext, then

$$\kappa \sim \frac{k_p - k_r}{k_0 - k_r}. \tag{10.1}$$

The Autokey Cipher, the code which was actually invented by Vigenère in 1586, is identical to the Vigenère cipher with the exception that instead of creating a keyword by repeating one word over and over, the keyword is constructed by appending the keyword to the beginning of the actual plaintext message. For example, if the plaintext is

This is a secret message;

and the keyword was "zebra", then the actual key stream would be

zebrathisisasecretmessage

Enciphering and deciphering the message is performed using the exact same method as the Vigenère cipher. In general, Autokey cipher is more secure than the Vigenère cipher.

1.3. Cryptography Techniques.

1.4. Symmetric Key Cryptography. In this technique, both sender and receiver uses a common key to encrypt and decrypt the message. This secret key (the symmetric key) is known only to the sender and to the receiver. The key must not be known to anyone else other than sender and receiver. If the secret key is known to any intruder, he could decrypt the message. It is also called as *secret key cryptography*, since the key has to be kept secret between the sender and receiver.

Before starting the communication, sender and receiver shares the secret key. This secret key is shared through some external means. At sender side, sender encrypts the message using his copy of the key. The ciphertext is then sent to the receiver over the communication channel. At receiver side, receiver decrypts the ciphertext using his copy of the key. After decryption, the message converts back into readable format.

Comparing to the asymmetric key algorithm, the symmetric key algorithms are more efficient and they take less time to encrypt and decrypt the message.

However, it has two disadvantages. First, in symmetric key cryptography, each pair of users require a unique secret key. Thus if n people in the world wants to use this technique, then $n(n-1)/2$ secret keys are needed. It is a huge number even n is just 1 million. Secondly, sharing the secret key between the sender and receiver is an important issue. While sharing the key, attackers might intrude. This led to a very difficult key management problem.

The most common used symmetric key encryption algorithms are

- Advanced Encryption Standard (AES);
- Data Encryption Standard (DES).

1.5. Asymmetric Key Cryptography. The asymmetric key cryptography is more commonly known as *Public Key Cryptography* (PKC). To use public key cryptography, each individual requires two keys, one public key which is publicly available and known to everyone and one private key which is known only to himself. Using the public key, it is not possible for anyone to determine the receiver's private key.

Sender encrypts the message using receiver's public key. Encryption converts the message into a ciphertext, which can be decrypted only using the receiver's The ciphertext is sent to the receiver over the communication channel. Receiver then decrypts the ciphertext using his private key to convert it back into a readable format.

Public Key Cryptography is more robust and less susceptible to third-party security breach attempts than Symmetric Key Cryptography. It is widely used in the age of internet, for example, E-commence. But it involves high computational requirements and is slower than symmetric key cryptography.

The famous asymmetric encryption algorithms are based on hard computational problems:

- RSA Algorithm, based on RSA problem;
- Diffie-Hellman Key Exchange, based on discrete logarithm problem.

CHAPTER 11

Hard Computational Problems

1. Trapdoor function and One-way function

In Public Key Cryptography, it is required that it is not possible for anyone to determine the receiver's private key using the public key only. This makes trapdoor functions important in cryptography.

A trapdoor function is a function that is easy to compute in one direction, and difficult to compute without extra information and easy with some special information (called the trapdoor) in the opposite direction. Mathematically, a function $f : X \to Y$ is a trapdoor function if $x \mapsto f(x)$ is easy to compute and finding x given $f(x) \in Y$ is hard without extra information but easy with a trapdoor.

Trapdoor functions rose to fame in cryptography in the mid-1970s with the publication of the groundbreaking paper *New Directions in Cryptography* of Diffie and Hellman in 1976. However, no example of trapdoor functions was presented in their paper. The RSA problem proposed by Rivest, Shamir and Adleman in 1977 gave the first practical example of trapdoor functions. Since then, several function classes have been proposed, and it soon became obvious that trapdoor functions are harder to find than was initially thought. As of now, the best known trapdoor function (family) candidates are still the RSA and Rabin families of functions.

Functions related to the hardness of the discrete logarithm problem are not known to be trapdoor functions, because there is no known "trapdoor" information about the group that enables the efficient computation of discrete logs. However, the discrete logarithm problem can be used as the basis for a trapdoor when the related computational Diffie–Hellman problem (CDHP) and/or its decisional variant(DDP) are used.

A trapdoor and a backdoor are not the same concepts in cryptography, though nowadays many people incorrectly use these two interchangeably. A backdoor is a mechanism added deliberately to a cryptographic algorithm or operating system, for example, that permits unauthorized parties to bypass or subvert the security of the system in some fashion.

Another related notion is one-way function. A function f is called a one-way function if f is easy to compute but f^{-1} is hard even with extra information.

2. Factoring and RSA

Suppose p and q are prime, $N = pq$. We have the following four computational problems:

(1) Factoring: Given N, find p and q.
(2) RSA: Given e and c, find m such that $m^e \equiv c \mod N$.
(3) Quadres: Given a, determine if a is a square module N or not.
(4) Squaroot: Given a, find x such that $a \equiv x^2 \mod N$.

Lemma 11.1. Factoring $\Leftrightarrow$ Squaroot $\Rightarrow$ Quadres.

Proof. Squaroot $\Rightarrow$ Quadres is trivial by definition. Now we show Factoring $\Leftrightarrow$ Squaroot.

$\Rightarrow$: If we know the factorization $N = pq$, then $a \equiv x^2 \mod N \Leftrightarrow a \equiv x^2 \mod p$ and $a \equiv x^2 \mod q$ by the Chinese Remainder Theorem. But it easy to find the value of the Legendre symbol $(\frac{a}{p}) = \pm 1$. If $(\frac{a}{p}) = 1$, By Shanks algorithm or Pocklington algorithm, we can find x such that $x^2 \equiv a \mod p$ as follows:

- If $p = 4m + 3$, then $x = \pm a^{m+1}$.
- If $p = 8m + 5$, if $a^{2m+1} = 1$, then $x = \pm a^{m+1}$; if $a^{2m+1} = -1$, then $(4a)^{2m+1} = 1$, let $y = (4a)^{m+1}$, then $x = \pm\frac{y}{2}$ or $\pm\frac{p+y}{2}$.
- If $p = 8m + 1$, the situation is a little bit more complicated.

$\Leftarrow$: Suppose we know how to find the square root. Take any x, compute $z = x^2 \mod N$, use Squaroot, we can find y such that $y^2 = z \mod N$. There are 4 solutions satisfying $y^2 = z \mod N$, two of them are $\pm x$. Thus there are $\frac{1}{2}$ chance that $y \neq \pm x \mod N$. Repeat this procedure until we find x and y satisfying $y \neq x \mod N$ and $y^2 = x^2 \mod N$. Then $x^2 - y^2 = (x - y)(x + y) = 0 \mod N$, and $\gcd(x + y, N) = p$ or q. □

Lemma 11.2. Factoring $\Rightarrow$ RSA.

Proof. If we know the factorization $N = pq$, then $\varphi(N) = (p-1)(q-1)$ is known. By the Euclidean Algorithm, we compute $d = e^{-1} \mod \varphi(N)$. Then $(c^d)^e = c^{de} = c \mod N$ by Euler's Theorem. Then $m = c^d$ is a solution of the RSA problem. □

The RSA problem is a trapdoor function which laid the foundation in the RSA cryptosystem, the first and maybe most successful cryptosystem in the era of Public Key Cryptography:

Algerithm 11.1 (RSA cryptosystem). *Let $N = pq$, choose e such that $\gcd(e, \varphi(N)) = 1$, find $d \equiv e^{-1} \mod \varphi(N)$. Then (N, e) is the public key which is available to everyone in the network, (d, p, q) is the private key just for the person himself.*

If m is the plaintext, Bob want to send m to Alice, he found Alice's public key $N(, e)$, computed $m^e \equiv c \mod N$ and then sent the ciphertext c to Alice. Alice uses her private key d to recover the original message $m = c^d \mod N$.

Example 11.1. Let $p = 7$ and $q = 11$, then $N = 77$ and $\varphi(N) = 60$. Let $e = 37$, by Euclidean algorithm, $d = 37^{-1} = 13 \mod 77$. Suppose Alice got $c = 51$ from Bod, then $m = 51^31 \equiv 2 \mod 77$, i.e. $m = 2$ is the original message.

Remark 11.1. By Factoring $\Rightarrow$ RSA, thus once Factoring is solved, the RSA cryptosystem is broken. However, at present nobody knows if RSA $\Rightarrow$ Factoring is true or not.

RSA is also used for Signature and Authentication.

Algerithm 11.2 (RSA Authentication). *Bob sends m to Alice, Alice computes $m^d = s$, and sends s back to Bob, Bob checks whether $s^e \equiv m \mod N$ or not to verify the authenticity of Alice.*

Lemma 11.3. *If N, e and d are all known, then* Factoring *is solved.*

Proof. Pick any $x \neq 0$, we may assume $\gcd(x, N) = 1$. Note that $ed = 1 \mod \varphi(N)$, by Euler's theorem, $x^{ed-1} \equiv 1 \mod N$. Suppose $2^t \| (ed-1)$. Since $4 \mid (p-1)(q-1) = \varphi(N)$, we know $t \geq 2$.

(1) Let $y_1 = x^{\frac{ed-1}{2}} \mod N$, then $y_1^2 = 1 \mod N$. If $y_1 \neq \pm 1 \mod N$, then compute $\gcd(y_1 \pm 1, N)$. Otherwise, either (i) $y_1 = -1 \mod N$, choose another x and start again; or (ii) $y_1 = 1 \mod N$, then go to next step.

(2) If $y_1 = 1 \mod N$, let $y_2 = x^{\frac{ed-1}{4}} \mod N$. If $y_2 \neq \pm 1 \mod N$, then compute $\gcd(y_2 \pm 1, N)$. Otherwise, either (i) $y_2 = -1 \mod N$ or $t = 2$ and $y_2 = 1 \mod N$, choose another x and start again; or (ii) $t > 2$ and $y_2 = 1 \mod N$, then go to next step.

(3) In general, if $y_k = 1 \mod N$ and $t < k$, then compute $y_{k+1} = x^{\frac{ed-1}{2^{k+1}}} \mod N$ and repeat the above procedure. $\square$

Example 11.2. Supose we know $N = 1441499$, $e = 17$ and $d = 507905$. Let $m_1 = \frac{ed-1}{2} = 4317192$. Let $x = 2$. Then $y_1 = 2^{m_1} \equiv 1 \mod N$; $m_2 = \frac{t_1}{2} = 2158596$, $y_2 = 2^{m_2} \equiv 1 \mod N$; $m_3 = \frac{t_2}{2} = 1079298$, $y_3 = 2^{m_3} \equiv 119533 \mod N$. Then we compute $\gcd(y_3 - 1, N) = 1423$ and get $N = 1441499 = 1423 \times 1013$.

Lemma 11.4. *If we know N and $\varphi(N)$, then* Factoring *is solvable.*

Proof. Since $\varphi(N) = pq - p - q + 1$, the sum $p + q = N + 1 - \varphi(N) = s$ and the product $pq = N$ are all known. This means that p, q are the roots of $x^2 - sx + N$, i.e., $p, q = \frac{s \pm \sqrt{s^2 - 4N}}{2}$. $\square$

Example 11.3. If $N = 18923$, $\varphi(N) = 18648$, then $s = N + 1 - \varphi(N) = 276$. Hence p, q are roots of $x^2 - 276x + 18923$, which are 149 and 127.

CHAPTER 12

Primality Testing and Factoring

1. Primality Test

Generating prime numbers is needed for almost all public key algorithms, for example:

(1) In the RSA system, we need to find primes p and q to compute the public key $N = pq$.
(2) In the ElGamal encryption we need to find primes p and q with q dividing $p - 1$.

The Prime Number Theorem tells us the distribution of primes among all positive integers:

THEOREM 12.1 (Prime Number Theorem). *Let $\pi(X)$ be the function counting the number of primes less than X. Then*

$$\pi(X) \sim \frac{X}{\log X}.$$

COROLLARY 12.1. *Choose randomly a number n, then the probability that n is a prime is $\frac{1}{\log n}$,*

EXAMPLE 12.1. Let $X = 2^{512}$, then $\pi(X) \sim 2^{503}$.

Thus a random number of 512 bits in length is a prime with probability $\sim \frac{1}{\log 2^{512}} \sim \frac{1}{355}$.

Pick randomly an integer $n > 0$, the primality test is to decide if n is a prime or not.

1.1. Trivial division.

1.1. Trivial division. Take all numbers between 2 and $\sqrt{n}$ and see if they divide n, if not then n is prime. This algorithm is simple and easy to implement, but it is usually a bad algorithm since when n is a prime, the algorithm requires $\sqrt{n}$ steps to run, which is an exponential function in terms of the size of input $\log n$.

Despite its drawbacks trivial division is however the method of choice for numbers which are very small. It is useful for eliminating composite numbers with small prime factors. Partial trivial division, up to a bound Y, is able to eliminate all but a proportion of $\prod_{p<Y}(1 - \frac{1}{p})$. Take $Y = 100$, then $\prod_{p<Y}(1 - \frac{1}{p}) \approx 0.12$.

1.2. Fermat's primality test. By Fermat's Little Theorem, if p is a prime, then $a^{p-1} \equiv 1 \mod p$ for all $0 < a < p$. Thus if there exists a such that $a^{p-1} \not\equiv 1 \mod p$, then n is not a prime.

Fermat's primality test is then the following algorithm: pick randomly k (about 50) numbers a between 0 and pn, check if $a^{n-1} \equiv 1 \mod n$ or not. If not, then n is a composite number; if yes for all a, then n is highly likely a prime, called a pseudom-prime.

Passing Fermat's primality test is a necessary but not sufficient condition for a number to be prime. This is because the existence of Carmichael numbers. By Euler's Theorem, $a^{\varphi(N)} \equiv 1 \mod N$ if $\gcd(a, N) = 1$. If n is a composite number and $\varphi(n) \mid (n-1)$, then for any a prime to n, $a^{n-1} \equiv 1 \mod n$. Numbers of this type are called Carmichael numbers. Unfortunately there are infinitely many Carmichael numbers.

Example 12.2. The first three Carmichael numbers are 561, 1105 and 1729.

Take $X = 10^{16}$, the number of primes $\leq X$ is about 2.7×10^{14} and the number of Carmichaes numbers $\leq X$ is about 2.4×10^5.

1.3. Miller-Rabin Test. This is a modification of Fermat's test. Write $n-1 = 2^s m$ with m odd. For $a \in \{2, \cdots, n-2\}$, if n is a prime, then $a^{n-1} = 1 \mod n$, thus either $a^m = a^{2m} = \cdots = a^{2^s m} = 1$ or one of $a^{2^i s} = -1 \mod n$ for $0 \leq i < s$. This leads to the Miller-Rabin Test:

Algerithm 12.1 (Miller-Rabin Test). *Aim: to determine n is either a composite number or highly likely a prime number.*

(1) *Pick $a \in \{2, \cdots, n-2\}$.*
- *If $a^m = 1 \mod n$, then n passes.*
- *Otherwise, for $i = 0, \cdots, s-1$, see if $a^{2^i m} = -1 \mod n$. If yes, then n passes.*
- *Otherwise n is composite, stop and exit the algorithm.*

(2) *If n passes Round 1, choose another a and repeat.*

(3) *If n passes k rounds for some fixed $k \geq 20$, stop and output that n passes the Miller-Rabin test.*

Again, this test is not deterministic. If n passes the Miller-Rabin Test, n is a pseudo-prime, more likely a prime than those who pass Fermat's test.

1.4. Primality Proofs. The most successful primality proving algorithm in practical use is the Elliptic Curve Primality Prover (ECPP) based on elliptic curves, due to Goldwasser and Kilian, using ideas from an older primality proving algorithm based on finite fields due to Pocklington and Lehmer.

In August 2002, Agrawal, Kayal and Saxena from Indian Institute of Technology Kanpur discovered a polynomial time deterministic algorithm to test if an input number is prime or not (see Agrawal, M.; Kayal, N.; and Saxena, N. "Primes is in P." Ann. Math. 160, 781-793, 2004.). This test is now known as the Agrawal-Kayal-Saxena primality test or AKS primality test. This is the first explicit polynomial time deterministic algorithm for primality, though this had long believed possible.

2. Factoring Algorithms

2.1. Overview of factoring algorithms. The factoring algorithms are the central piece of computation number theory. Modern factoring algorithms lie somewhere between polynomial and exponential time, in an area called sub-exponential time. These algorithms have complexity measured by the function $L_N(\alpha, \beta)$.

Definition 12.1. For $0 \leq \alpha \leq 1$ and $\beta > 0$, let

$$L_N(\alpha, \beta) = \exp((\beta + o(1))(\log N)^{\alpha}(\log\log N)^{1-\alpha}). \tag{12.1}$$

If the running time of an algorithm is $L_N(\alpha, \beta)$ where N is the input, then

(1) If $\alpha = 0$, i.e., the running time $L_N(0, \beta) = (\log N)^{\beta+o(1)}$, the algorithm is called a polynomial time algorithm.

(2) If $\alpha = 0$, i.e., the running time $L_N(1, \beta) = N^{\beta+o(1)}$, the algorithm is called an exponential time algorithm.

(3) If $0 < \alpha < 1$, the algorithm is called a sub-exponential time algorithm.

We list the factoring methods here, details can be found in Chapters 8-10 of H. Cohen's book *A course in computational algebraic number theory* (GTM 138, Springer, 1993):

(1) Dark Age methods:

(A) Trivial division: Try every prime number up to $\sqrt{N}$ and see if it is a factor of N. This has complexity $L_N(1, \frac{1}{2})$, and is therefore an exponential algorithm.

(B) $p-1$ method: This was invented by Pollard in 1974, good for finding a prime factor p of N when $p-1$ has a decomposition in small prime factors. We shall study it in next subsection.

(C) $p+1$ method. This is a variant of the $p-1$ method invented by Williams in 1982, which uses Lucas sequences to achieve rapid factorization if some factor p of N has a decomposition of $p+1$ in small prime factors.

(D) Pollard ρ method. This is also an algorithm for discrete logarithm problem. We shall talk this more there.

(2) Modern methods:

(E) Continued Fraction Method (CFRAC): This is the first sub-exponential time algorithm with complexity $L_N(\frac{1}{2}, \sqrt{3})$. It was first described in 1931 by Lehmer and Powers and later in 1975 were developed into a computer algorithm by Morrison and Brillhart.

(F) Elliptic Curve Method (ECM): This is a very good method if $p < 2^{50}$, its complexity is $L_p(\frac{1}{2}, c)$, which is sub-exponential. We shall talk about it more when we study elliptic curves.

(G) Quadratic Sieve (QS): This is probably the fastest method for factoring integers of between 80 and 100 decimal digits. It has complexity $L_N(\frac{1}{2}, 1)$.

(H) Number Field Sieve (NFS): This is currently the most successful method for numbers with more than 100 decimal digits. It can factor numbers of the size of $10^{155} \approx 2^{512}$ and has complexity $L_N(\frac{1}{3}, 1.923)$.

Smoothness characterizes numbers with only small prime factors:

Definition 12.2 (Smooth Numbers). Let B be an integer, an integer N is called B-smooth if every prime factor of N is less than B.

Example 12.3. The number $N = 1999996 = 2^{31} \cdot 31 \cdot 127^2$, so N is 128-smooth.

We shall need the Dickman-de Bruijn function ρ, which is the solution of the following differential delay equation

$$\mu\rho'(\mu) + \rho(\mu - 1) = 0$$

for $\mu > 1$. In practice we approximate $\rho(\mu)$ via the expression $\rho(\mu) \approx \mu^{-\mu}$. Then the function

$$\psi(x, y) = \#\{n \leq x, n \text{ is } y\text{-smooth}\} \tag{12.2}$$

is approximated by $\psi(x, y) \approx x\rho(\mu) \approx x\mu^{-\mu}$ where $\mu = \frac{\log x}{\log y}$. This means

Theorem 12.2. *The proportion of integers less than x, which are $x^{\frac{1}{\mu}}$-smooth, is asymptotically equal to $\mu^{-\mu}$.*

Now if we set $x = N$ and $y = L_N(\alpha, \beta)$, then $\mu = \frac{\log N}{\log y} = \frac{1}{\beta}(\frac{\log N}{\log \log N})^{1-\alpha}$. Hence we have

Corollary 12.2. *The proportion of $L_N(\alpha, \beta)$-smooth integers $\leq N$ is approximately $\frac{1}{L_N(1-\alpha, \gamma)}$.*

Definition 12.3. A number N is called B-power smooth if every prime power dividing N is less than B.

Example 12.4. $N = 2^5 \cdot 3^3$ is 33-power smooth.

2.2. Pollard $p - 1$ method. Suppose the number we wish to factor is given by $N = pq$, in addition suppose we know (by some pure guess) an integer B such that $p - 1$ is B-power smooth, but $q - 1$ is not B-power smooth. We can then hope that $p-1$ divides $B!$, but $q-1$ is unlikely dividing $B!$. This means the number $b = 2^{B!} = 1 \mod p$ but unlikely $b = 1 \mod q$. Hence we can recover p by computing $p = \gcd(b - 1, N)$. This observation leads to Pollard's $p - 1$ method.

Algerithm 12.2 (Pollard $p - 1$ method). *Choose a test cap B.*

(1) *Use a simple method (such as the sieve of Eratosthenes) to find primes $p < B$, or even probable primes.*

(2) *Choose an integer a coprime to N, for example $a = 2$ if N is odd.*
(3) *Find the exponent e such that $p^e \leq B$. Then for each $p < B$, compute $b = a^{p^e} \mod N$ and see if $1 < \gcd(b, N-1) < N$. If that's the case, return those results of the greatest common divisor function, exit.*
(4) *If the GCD function consistently returned 1s, one could try a higher test cap and try again from step 1.*
(5) *If the GCD function consistently returned n itself, this could indicate that a is in fact not coprime to N, in which case N is already factored.*
(6) *Throw a failure exception.*

As of version 5.2, Pollard's $p-1$ algorithm is one of the methods used by Mathematica's **FactorInteger** function after eliminating small factors by trial division.

EXAMPLE 12.5. For $N = 15770708441$, take $B = 180$ and $a = 2$. Then $b = 2^{180!} = 1162022425 \mod N$. Then $p = \gcd(b-1, N) = 135979$ and $N = 135979 \times 115979$.

Note in this case $p - 1 = 135978 = 2 \cdot 3 \cdot 131 \cdot 173$ and $q - 1 = 115978 = 2 \cdot 103 \cdot 563$, hence $p-1$ is B-power smooth and $q-1$ is not.

Due to the $p-1$ method, it is often recommended that RSA primes are chosen to satisfy $p - 1 = 2p_1$ and $q - 1 = 2q_1$, where p_1 and q_1 are both primes. However, this is not really needed, because the probability that for a random 512-bit prime p, the number $p-1$ is B-power smooth for a small value of B is vary small.

Pollard's $p-1$ method also indicates the key point of factorization: produce two numbers x and y, of around the same size as N, such that $x^2 = y^2 \mod N$ and $x \neq \pm y$, then compute $\gcd(x \pm y, N)$ to factorize N.

2.3. Strategy of Modern Factoring Methods. Most modern factoring methods have the following strategy based on the difference of two squares method described at the end of the last section.

(1) Take a smoothness bound B.
(2) Compute a factor base F of all prime numbers p less than B.
(3) Find a large number of values of x and y, such that x and y are B-smooth and $x = y \mod N$. These are called relations on the factor base.
(4) Use linear algebra modulo 2 (i.e., linear algebra over $\mathbb{F}_2$), find a combination of the relations to give an X and Y with $X^2 = Y^2 \mod N$.
(5) Attempt to factor N by computing $\gcd(X - Y, N)$.

Let J be the set of all relations. For $j \in J$, the relation $x_j = y_j \mod N$ is equivalent to $x_j y_j^{-1} = 1 \mod N$. As x_j and y_j are both B-smooth, $x_j y_j^{-1} = \prod_{p<B} p^{n_{pj}}$. Let $c_j = (n_{pj})^T_{p<B}$. To find $X^2 = Y^2 \mod N$ is

equivalent to find $z_j \in \mathbb{Z}$ for $j \in J$ such that $\sum_{j\in J} z_j c_j = 0 \mod 2$. In other words, regard $A = (n_{pj})_{p<B, j\in J}$ as a matrix over $\mathbb{F}_2$ of $\#F$ rows and $\#J$ columns. Then to find a combination that $X^2 = Y^2 \mod N$ is equivalent to find a non-zero binary vector $z = (z_j)_{j\in J} \mod 2$ such that $zA = 0$ over $\mathbb{F}_2$. By linear algebra, if the number of relations $\#J > \#F$, then the homogeneous equation $zA = 0$ is guaranteed to have a non-zero solution. The key for any modern factoring method is to gather $\geq |F| + 1$ relations and then solve the linear equation.

Example 12.6. Suppose $F = \{p, q, r\}$ and the relations are

$$p^2q^5r^2 = p^3q^4r^3 \mod N, /$$
$$pq^3r^5 = pqr^2 \mod N,$$
$$p^3q^5r^3 = pq^3r^2 \mod N.$$

Dividing one side by the other in each of our relations we obtain

$$p^{-1}qr^{-1} = 1 \mod N,$$
$$q^2r^3 = 1 \mod N,$$
$$p^2q^2r = 1 \mod N.$$

Multiplying the last two equations together we obtain

$$p^2q^4r^4 = 1 \mod N$$

Hence if $X = pq^2r^2$ and $Y = 1$, then we obtain $X^2 = Y^2 \mod N$.

In this example, the matrix

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 0 & 2 & 3 \\ 2 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \mod 2.$$

Take $z = (0, 1, 1)$ then gives $X = pq^2r^2$ and $Y = 1$.

2.4. Linear Sieve. We let F denote a set of *small* prime numbers which form the factor base: $F = \{p : p \leq B\}$. A number with all its factors in F is therefore B-smooth. The idea of the linear sieve is to find two integers a and λ such that a and $b = a + N\lambda$ are both B-smooth.

We could write $a = \prod_{p\in F} p^{a_p}$ and $b = a + N\lambda = \prod_{p\in F} p^{b_p}$. We could have a relation:

$$\prod_{p\in F} p^{a_p} \equiv \prod_{p\in F} p^{b_p} \mod N.$$

So the main question is how to find the values of a and λ? This is done as follows:

(1) Fix a value of λ to consider.
(2) Initialize an array of length $A + 1$ indexed by 0 to A with zeros.
(3) For each prime $p \in F$, add $\log_2 p$ to every array location whose position is congruent to $-\lambda N \mod p$.
(4) Choose the a to be the position of those elements which exceed some threshold bound.

EXAMPLE 12.7. Suppose we take $N = 1159$, $F = \{2, 3, 5, 7, 11\}$, $B = 12$, $\lambda = -2$ and $A = 9$, we wish to find a smooth value of $b = a - 2N$.
We initialize the sieving array as follows:

0	1	2	3	4	5	6	7	8	9
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

We now take the first prime in F, namely $p = 2$, we compute $-\lambda N \mod p = 0$. We add $\log_2 2 = 1$ to every array location with index equal to 0 modulo 2. Our sieve array becomes:

0	1	2	3	4	5	6	7	8	9
1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0

We now take the next prime in F, namely $p = 3$, we compute $-\lambda N \mod p = 2$. We add $\log_2 3 = 1.6$ to every array location with index equal to 2 modulo 3. This results in our sieve array becoming:

0	1	2	3	4	5	6	7	8	9
1.0	0.0	2.6	0.0	1.0	1.6	1.0	0.0	2.6	0.0

Continuing in this way with $p = 5, 7$ and 11, eventually the sieve array becomes:

0	1	2	3	4	5	6	7	8	9
1.0	2.8	2.6	2.3	1.0	1.6	1.0	0.0	11.2	0.0

Hence, the value $a = 8$ looks like it should correspond to a smooth value, and indeed it does, since we find:

$$a - \lambda N = 8 - 2 \cdot 1159 = -2310 = -2 \cdot 3 \cdot 5 \cdot 7 \cdot 11.$$

To have the linear sieve successful, we need to have at least $|F| + 1$ such relations. Unfortunately, the basic linear sieve produces a very small number of relations.

2.5. Number Field Sieve. First we construct two monic, irreducible polynomials with integer coefficients f_1 and f_2, of degree d_1 and d_2 respectively, such that $f_1(m) \equiv f_2(m) \equiv 0 \mod N$ for some $m \in \mathbb{Z}$. The number field sieve will make use of arithmetic in the number fields K_1 and K_2 given by $K_1 = \mathbb{Q}(\theta_1)$ and $K_2 = \mathbb{Q}(\theta_2)$, where θ_1 and θ_2 are defined by $f_1(\theta_1) = f_2(\theta_2) = 0$. For $i = 1, 2$, we have $[K_i : \mathbb{Q}] = d_i$, and have two homomorphisms $\phi(i)$ given by:

$$\phi(i) : \begin{cases} \mathbb{Z}[\theta_i] \longrightarrow \mathbb{Z}/N\mathbb{Z}, \\ \theta_i \longrightarrow m. \end{cases}$$

We aim to use a sieve, just as in the linear sieve, to find a set

$$S \subseteq \{(a, b) \in \mathbb{Z}^2 \mid \gcd(a, b) = 1\}$$

such that

$$\prod_S (a - b\theta_1) = \beta^2, \qquad \prod_S (a - b\theta_2) = \gamma^2$$

where $\beta \in K_1$ and $\gamma \in K_2$. If we found two such values of β and γ, then we would have

$$\phi_1(\beta)^2 \equiv \phi_2(\gamma)^2 \mod N$$

then $\gcd(\phi_1(\beta) - \phi_2(\gamma), N)$ would be a factor of N.

This leads to three obvious problems:

(1) How do we find the set S?
(2) Given $\beta^2 \in \mathbb{Q}[\theta_1]$, how do we compute β?
(3) How do we find the polynomials f_1 and f_2 in the first place?

Nowadays the modern method of factorization of N is as follows:

(1) For any $p < 100$, check whether $(p, N) = 1$ or not.
(2) Find some f_1 and f_2, try if one can find many relations, otherwise change f_1 and f_2.

We give an example.

Example 12.8. Suppose $N = 290^2 + 1 = 84101$, we take $f_1(x) = x^2 + 1$ and $f_2(x) = x - 290$ with $m = 290$. We have $f_1(m) \equiv f_2(m) \equiv \mod N$. Then $K_1 = \mathbb{Q}(i)$, $K_2 = \mathbb{Q}$, $d_1 = 2$, $d_2 = 1$, $\theta_1 = i$ and $\theta_2 = m$.

We need to find x, y, such that $x - iy$ is smooth(algebraic smooth) and $x - my$ is also smooth. This leads to the following table:

x	y	$N(x-iy)$	Factors	$x-my$	Factors
-38	-1	1445	$(5)(17^2)$	252	$(2^2)(3^2)(7)$
-22	-19	845	$(5)(13^2)$	5488	$(2^4)(7^3)$

Then we obtain:

$$-38 + i = -(2+i)(4-i)^2, \qquad -22 + 19i = -(2+i)(3-2i)^2.$$
$$(-38+i)(-22+19i) = (2+i)^2(3-2i)^2(4-i)^2 = (31-12i)^2.$$
$$(-38+m)(-22+19m) = 2^6 3^2 7^4 = 1176^2.$$

we apply the map ϕ_1 to $31 - 12i$ to obtain

$$\phi_1(31 - 12i) = -3449.$$

Then we have

$$\begin{aligned}(-3449)^2 =& \phi_1(31-12i)^2 = \phi_1((-38+i)(-22+19i)) \\ \equiv& (-38+m)(-22+19m) = 1176^2 \mod N.\end{aligned}$$

We compute $\gcd(N, -3449 \pm 1176)$, hence 37 and 227 are factors of $N = 84101$.

CHAPTER 13

Public Key Encryption Algorithms

1. Public Key Cryptography

Recall that in symmetric key cryptography each communicating party needed to have a copy of the same secret key. This led to a very difficult key management problem. This was overcomed by Diffie-Hellman Key Exchange Algorithm, a center piece of Public Key Cryptography invented by Diffie and Hellman in their masterpiece *New Directions in Cryptography* in 1976. Although Diffie and Hellman invented the concept of public key cryptography it was not until a year or so later that the first system, namely RSA, was invented.

We recall public key cryptography here: the use of identical keys was replaced with two keys: one public and one private. As is custom we call the two parties exchanging messages Alice and Bob, and the intruder Eve.

- Bob: encrypts the message with Alice's public key to get ciphertext and sends it to Alice.
- Alice: uses her private key to decrypts the ciphertext to get the original message.
- Eve: intercepts the ciphertext but can not recover the message without Alice's private key.

In public key cryptography, the use of trapdoor functions/one-way functions is vital. The most important one-way function used in public key cryptography is that of factoring integers. Another important class of problems are those based on the discrete logarithm problem or its variants.

2. Factoring and RSA-based algorithms

2.1. RSA Algorithm. This algorithm is based on the trapdoor function RSA.

Suppose Alice wishes to enable anyone to send her secret messages, which only she can decrypt. This is achieved by the following algorithm:

Algerithm 13.1 (RSA Algorithm).

(1) *Pick two primes p and q, compute $N = pq$.*
(2) *Choose e, such that* $\gcd(e, (p-1)(q-1)) = 1$.
(3) *Compute $d = e^{-1} \mod \varphi(N)$.*

Then the private key of Alice is (d, p, q), the public key is (N, e).

Suppose Bob wishes to encrypt a message "m" to Alice. He computes $c = m^e \mod N$, *c is the ciphertext, then he sends c to Alice. Alice computes* $m = c^d \mod N$ *to recover m.*

LEMMA 13.1. *If the* RSA *problem is hard, then the* RSA *system is secure under a chosen plaintext attack, i.e. an attacker is unable to recover the whole plaintext from the ciphertext.*

PROOF. We wish to give an algorithm which solves the RSA problem using an algorithm to break the RSA system as an oracle. If we can show this, then we can conclude that breaking the RSA system is no easier than solving the RSA problem. Recall that the RSA problem is given $N = pq$, e and $y \in (\mathbb{Z}/N\mathbb{Z})^\times$, find x such that $x^e = y \mod N$. We use our oracle to break the RSA encryption algorithm to decrypt the message corresponding to the ciphertext $c = y$, this oracle will return the plaintext message m. Then our RSA problem is solved by setting $x = m$, since $m^e = y \mod N$. So if we can break the RSA algorithm, then we can solve the RSA problem. □

To build a secure RSA system, we shall avoid the following:

(I) Use of a shared modulus N.

(1) Internal person: the factorization $N = pq$ is know, then it is easy to get any other people's private keys.
(2) External person: Suppose Alice send message m to two users B_i ($i = 1, 2$), who has public key (N, e_i). Eve can see $c_i = m^{e_i} \mod N$ for $i = 1, 2$. Let $t_1 = e_1^{-1} \mod e_2$ and $t_2 = \frac{t_1 e_1 - 1}{e_2}$. Then Eve can get $m = c_1^{t_1} c_2^{-t_2}$.

(II) Use of small public e. Suppose m is sent to three different users with modulus N_1, N_2, N_3 and same exponent $e = 3$. Let the ciphertext $c_i \equiv m^3 \mod N_i$, for $i = 1, 2, 3$. By Chinese Remainder Theorem, one can find X such that $X \equiv m^3 \mod N_1 N_2 N_3$ and $X < N_1 N_2 N_3$. Note that $m < N_i$ and hence $m^3 < N_1 N_2 N_3$, then $X = m^3$ and $m = \sqrt[3]{X}$ (as real number) is the original message.

EXAMPLE 13.1. Let the modulus be $N_1 = 33$, $N_2 = 299$ and $N_3 = 341$, and $e = 3$. Suppose the ciphertext $c_1 = 50$, $c_2 = 268$ and $c_3 = 1$. Then $X = 300763 \mod N_1 N_2 N_3$ and the original message $m = X^{\frac{1}{3}} = 300763^{\frac{1}{3}} = 67$.

From the above analysis, to get a secure encryption algorithms, we should:

(1) Avoid small e, now usually $e = 65537 = 2^{12} + 1$.
(2) "Same message" should never be encrypted to two different people.
(3) Plaintext should be randomly packed before transmission.

2.2. Rabin Encryption. We first choose prime numbers of the form $p \equiv q \equiv 3 \mod 4$ (so that it is easy to find square roots $\mod p$ and $\mod q$). The private key is the pair (p, q), the public key is (N, B), where

$N = pq$ and B is chosen randomly from $\{0, \cdots, N-1\}$. Rabin Algorithm is as follows:

(1) Encryption: $c = m(m+B) \mod N$ (this is very fast).
(2) Decryption: $m = \sqrt{\frac{B^2}{4} + c} - \frac{B}{2} \mod N$.

At first sight this uses no private information, but a moment's thought reveals that you need the factorization of N to be able to find the square root. There are however four possible square roots modulo N, since N is the product of two primes. Hence on decryption you obtain four possible plaintexts. This means that we need to add redundancy to the plaintext before encryption in order to decide which of the four possible plaintexts corresponds to the intended one.

Example 13.2. Choose $p = 127$ and $q = 131$. Then $N = 16637$. Choose $B = 12345$. The message $m = 4410$ is Encrypted to $c = 4633 = m(m+B) \mod N$. For decryption, let $t = \frac{B^2}{4} + c = 1500 \mod N$. Then $\sqrt{t} \equiv \pm 22 \mod p$ and $\sqrt{t} \equiv \pm 7 \mod q$, hence $\sqrt{t} \equiv \pm 3705$ or $\pm 14373 \mod N$. Then $m = 4410$, 5851, 15078 or 16519.

2.3. Paillier Encryption. We first pick an RSA modulo $N = pq$, but instead of working with the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$ we work with $(\mathbb{Z}/N^2\mathbb{Z})^\times$. The private key is defined to be an integer d such that $d \equiv 1 \mod N$ and $d \equiv 0 \mod (p-1)(q-1)$. Such a value of d can be found by the Chinese Remainder Theorem. The public key is just the integer N.

(1) Encryption: $c = (1+N)^m r^N \mod N^2$, $r \in (\mathbb{Z}/N^2\mathbb{Z})^\times$, m is the plaintext.
(2) Decryption: Let $t \equiv c^d \mod N^2$ and $t < N^2$. Note that

$$\begin{aligned} t &\equiv (1+N)^{md} r^{dN} \\ &\equiv (1+N)^{md} \text{ (since } d \equiv 0 \mod (p-1)(q-1)) \\ &\equiv 1 + mdN \equiv 1 + mN \mod N^2 \text{ (since } d \equiv 1 \mod N), \end{aligned}$$

to recover the message, just compute $m = \frac{t-1}{N}$.

3. Discrete Logarithm Problem based algorithms

3.1. Discrete Logarithm Problem. Let G be a finite abelian group, the discrete logarithm problem, or DLP in short, in G is: given $g, h \in G$, find an integer x (if it exists) such that $g^x = h$, i.e. $x = \log_g h$.

For the multiplicative group of a finite field the best known algorithm for this task is the Number Field Sieve. The complexity of determining discrete logarithms in this case is given by $L_N(\frac{1}{3}, c)$, for some constant c, depending on the type of the finite field, e.g. whether it is a large prime field or an extension field of characteristic two. For other groups, such as the group of rational points of an elliptic curve over a finite field, the best known algorithm for finding discrete logarithm on a general elliptic curve defined over a finite field $\mathbb{F}_q$ is Pollard's Rho method which has complexity $\sqrt{q} = L_q(1, \frac{1}{2})$.

There are a number of related problems associated to discrete logarithms, suppose we are given a finite abelian group $(G, \cdot)$ and $g \in G$.

(1) DLP (Discrete Logarithm Problem): Given $g, h \in G$ such that $h = g^x$, find x.
(2) DHP (Diffie-Hellman Problem): Given $g \in G$, $a = g^x$ and $b = g^y$, find c, such that $c = g^{xy}$.
(3) DDH (Decision Diffie-Hellman): Given $g \in G$, $a = g^x$, $b = g^y$ and $c = g^z$, determine if $z = x \cdot y$.

Lemma 13.2. DLP $\Rightarrow$ DHP $\Rightarrow$ DDH.

Proof. Easy. $\square$

Remark 13.1. However, it is not known if DHP is easier than DLP or not, even for $G = \mathbb{F}_q^\times$ or $E(F_q)$. Probably these two are as easy as each other.

3.2. ElGamal Encryption (Finite field version). Unlike the RSA algorithm, in ElGamal encryption there are some public parameters which can be shared by a number of users. These are called the domain parameters and are given by:

(1) p is a large prime, by which we mean one with around 1024 bits, such that $p-1$ is divisible by another medium prime q of around 160 bits.
(2) g is an element of $\mathbb{F}_p^\times$ of prime order q, i.e. $g = \gamma^{\frac{p-1}{q}} \neq 1 \mod p$ for some $\gamma \in \mathbb{F}_p^\times$.

Once these domain parameters have been fixed, the public and private keys can then be determined. The private key is chosen to be an integer x, the public key is given by $h = g^x \mod p$. To encrypt a message $m \in \mathbb{F}_p^*$, we

(1) Generate a random ephemeral key k.
(2) Set $c_1 = g^k$, $c_2 = mh^k$.
(3) Output the ciphertext as $c = (c_1, c_2)$.

To decrypt a ciphertext $c = (c_1, c_2)$ we compute $m = c_1^{-x} c_2 = g^{-kx}(mh^k)$.

Lemma 13.3. *Assume* DHP *is hard, then ElGamal is secure under a chosen plaintext attack.*

Proof. Suppose there is an oracle to break ElGamal, i.e., given $c = (c_1, c_2)$, one can find the original message m without knowing the private key x. Now in DHP, suppose g^x and g^y are given. Let $h = g^x$ (even though don't know x), $c = (c_1, c_2) = (g^y, \text{random element in } \mathbb{F}_p^*)$. The oracle for ElGamal gives $m = \frac{c_2}{c_1^x}$. Then $\frac{c_2}{m} = c_1^x = g^{xy}$ and DHP is solved. $\square$

CHAPTER 14

Discrete Logarithms

In this chapter we survey the methods known for solving the discrete logarithm problem in various abelian groups G. Keep in mind that two cases are used most common in practice: $G = \mathbb{F}_q^\times$ or $E(\mathbb{F}_q)$ where E is an elliptic curve over $\mathbb{F}_q$.

1. Pohlig-Hellman Algorithm

1.1. Reduction to cyclic groups. Suppose G is a group of order N and $g \in G$. Let $G' = \langle g \rangle$ be the cyclic subgroup generated by g, which is of order N'. Then $N' \mid N$. Moreover, if $p^e \parallel N$, then $p^{e'} \parallel N'$ where e' is the first integer between 0 and e such that $g^{N/p^{e-e'}} = 1$. In this way, once the prime decomposition of N is known, it is easy to find the order N'. Now to find x such that $h = g^x$, it suffices to solve the related DLP in G'.

1.2. Reduction to groups of prime order. Suppose we have a finite cyclic abelian group $G = \langle g \rangle$ whose order is given by $N = \#G = \prod_{i=1}^{s} p_i^{e_i}$. For prime $p \mid N$, we let $p^e \parallel N$. Now suppose $h \in G$, find x such that $g^x = h$, we just need to find $x \mod N$ since $g^N = 1$. By the Chinese Remainder Theorem, to find $x \mod N$, it is equivalent to find $x \mod p_i^{e_i}$ for $1 \le i \le t$.

Let $C_{p^e} = \langle g^{\frac{N}{p^e}} \rangle$, then C_{p^e} is cyclic of order p^e and $h^{\frac{N}{p^e}} = (g^{\frac{N}{p^e}})^x$. Hence

$$\text{DLP for } G \iff \text{DLP for } C_{p^e} \text{ for all } p^e \parallel N.$$

Now suppose $C_{p^e} = \langle g \rangle$ and $h = g^x \in C_{p^e}$. We may assume $0 \le x < p^e$. Write $x = x_0 + x_1 p + \cdots + x_{e-1}p^{e-1}$ such that $0 \le x_i < p$. Then

(1) Let $g_1 = g^{p^{e-1}}$, then $\langle g_1 \rangle = C_p$, which is a cyclic group of order p.
(2) Let $h_1 = h^{p^{e-1}}$, then $h_1 \in C_p$ and $h_1 = g_1^x = g_1^{x_0}$. Solving the DLP for C_p, one can get the value of x_0.
(3) Suppose by induction we knew the values $x_0, \cdots, x_{t-1}, t \le e-1$. Let $x'_{t-1} = x_0 + x_1 p + \cdots + x_{t-1}p^{t-1}$. Then $x - x'_{t-1} = p^t(x_t + px_{t+1} + \cdots)$ and

$$h_t = (hg^{-x'_{t-1}})^{p^{e-t-1}} = (g^{x-x'_{t-1}})^{p^{e-t-1}} = g_1^{x_t}.$$

Once again solving the DLP for C_p to get te value of x_t, and the value x is recovered by induction.

In conclusion, we have

LEMMA 14.1. *Suppose $G = \langle g \rangle$ is cyclic of order N. For $p^e \parallel N$, let $C_{p^e} = \langle g^{\frac{N}{p^e}} \rangle$ and $C_p = \langle g^{\frac{N}{p}} \rangle$, which are cyclic of order p^e and p respectively. Then*

$$\text{DLP } \textit{for } G \iff \text{DLP } \textit{for } C_{p^e} \textit{ for } p^e \parallel N \iff \text{DLP } \textit{for } C_p \textit{ for } p \mid N,$$

i.e. DLP *for G is as hard as* DLP *for the largest subgroup of G of prime order.*

ALGERITHM 14.1 (Pahlig-Hellman Algorithm).

(1) *For $p \mid N$, find e such that $p^e \parallel N$.*
(2) *Let $g' = g^{\frac{N}{p^e}}, h' = h^{\frac{N}{p^e}}, g_1 = g^{\frac{N}{p}} = (g')^{p^{e-1}}$. Solve* DLP *in C_p for $h_1 = h^{\frac{N}{p}} = g_1^{x_0}, 0 \le x_0 \le p-1$.*
(3) *For $1 \le t \le e-1$, suppose $x_0, \cdots, x_{t-1}$ is known, let*
$$x'_{t-1} = x_0 + x_1 p + \cdots + x_{t-1} p^{t-1},$$
solve DLP *in C_p for*
$$h_t = (h' g'^{-x'_{t-1}})^{p^{e-t-1}} = g_1^{x_t},\ 0 \le x_t < p-1,$$
then
$$x \equiv x_0 + x_1 p + \cdots + x_{e-1} p^{e-1} \bmod p^e.$$
(4) *Use the Chinese Remainder Theorem to find $x \mod N$.*

EXAMPLE 14.1. Let $G = \mathbb{F}_{397}^{\times} = \langle 5 \rangle$ and $g = 5$. Then $N = 396 = 2^2 \cdot 3^2 \cdot 11$. Suppose $h = 298$. To find x such that $h = 298 = 5^x \mod 397$ is equivalent to find x_4, x_9 and x_{11} such that

$$\begin{cases} h^{\frac{396}{4}} = 334 \equiv 334^{x_4} \equiv (g^{\frac{396}{4}})^{x_4} \mod 397, \\ h^{\frac{396}{9}} = 286 \equiv 79^{x_9} \equiv (g^{\frac{396}{9}})^{x_9} \mod 397, \\ h^{\frac{396}{11}} = 273 \equiv 290^{x_{11}} \equiv (g^{\frac{396}{11}})^{x_{11}} \mod 397. \end{cases}$$

Need to determine

$$\begin{cases} x_4 \equiv x \mod 4 \\ x_9 \equiv x \mod 9 \\ x_{11} \equiv x \mod 11 \end{cases}$$

let $x_9 = x_{9,0} + x_{9,1} \cdot 3$, $0 \le x_{9,i} \le 2$. Then $g_1 = 334^3 = 362$ and $h_1 = 286^3 = 34$, then

$$\begin{cases} h_1 = g_1^{x_{9,0}} \mod 337 \Rightarrow x_{9,0} = 2 \\ 1 \equiv \frac{286}{334^2} \equiv 362^{x_{9,1}} \mod 397 \Rightarrow x_{9,1} = 0 \end{cases} \qquad \Longrightarrow x_9 = 2.$$

Similarly we get

$$\begin{cases} x_4 \equiv x \equiv 1 \mod 4, \\ x_9 \equiv x \equiv 2 \mod 9, \\ x_{11} \equiv x \equiv 6 \mod 11. \end{cases}$$

By the Chinese Remainder Theorem we get $x = 281$.

2. Baby-Step/Giant-Step Method

ALGERITHM 14.2 (Baby-Step/Giant-Step). *Suppose $G = \langle g \rangle$ is cyclic group of order p, $h \in G$, find x such that $h = g^x$. Suppose it is easy to store, sort and search a list of elements in G. We first write*

$$x = x_0 + x_1\lceil\sqrt{p}\rceil, 0 \le x_0, x_1 < \lceil\sqrt{p}\rceil.$$

(1) *Baby step: Compute $g_i = g^i$ for $0 \le i < \lceil\sqrt{p}\rceil$ and store (g_i, i). To compute and store the Baby-Steps clearly requires $O(\sqrt{p})$ time, and a similar amount of storage.*
(2) *Giant step: Compute $h_j = hg^{-j\lceil\sqrt{p}\rceil}$ for $0 \le j < \lceil\sqrt{p}\rceil$. If $h_i = g_i$, then $x = i+j\lceil\sqrt{p}\rceil$. Notice that the time to compute the Giant-Steps is at most $O(\lceil\sqrt{p}\rceil)$.*

Hence, the overall time and space complexity of the Baby-Step/Giant-Step method is $O(\sqrt{p})$.

EXAMPLE 14.2. Take the subgroup of order 101 in the multiplicative group of the finite field $\mathbb{F}_{607}$, generated by $g = 64$. Suppose we are given the discrete problem

$$h = 182 = 64^x \mod 607.$$

We first compute the Baby-Steps:

$$g_i = 64^i \mod 607, \text{ for } 0 \le i < \lceil\sqrt{101}\rceil = 11.$$

We get

i	0	1	2	3	4	5	6	7	8	9	10
g_i	1	64	454	527	343	100	330	482	498	308	288

Now we compute the Giant-Steps:

$$h_j = 182 \cdot 64^{-11j} \mod 607 \text{ for } 0 \le j < 11,$$

and check when we obtain a Giant-Step which occurs in our table of Baby-Steps:

j	0	1	2	3	4	5	6	7	8	9	10
g_j	182	143	69	271	343	573	60	394	483	76	580

So we obtain a match when $i = 4$ and $j = 4$, which means that $x = i+j\cdot 11 = 4 + 4 \cdot 11 = 48$, which we can verify to be the correct answer to the discrete logarithm problem by computing $64^{48} = 182 \mod 607$.

3. Pollard Type Methods

3.1. Birthday Paradox. Suppose there are n birthdays (for example, $n = 365$ in a solar year) in a calendar. We compute the probability that there exists at least 2 people among k people with the same birthday. Suppose this probability is p, then the probability of every people have distinct birthday is $q = 1 - p$. Let b_i be the birthday of the i-th people,

$$E = \{(b_1, \cdots, b_k) \mid b_1, \cdots, b_k \text{ are all different}\} \subseteq \{1, \cdots, n\}^k.$$

then

$$\#E = n(n-1)\cdots(n-k-1) = \frac{n!}{(n-k)!},$$

$$q = \frac{\#E}{n^k} = \prod_{i=1}^{k-1}(1-\frac{i}{n}).$$

By $1+x<e^x$, then

$$q \le e^{\frac{-k(k-1)}{2n}} \text{ and } p \ge 1-e^{\frac{-k(k-1)}{2n}}. \tag{14.1}$$

Proposition 14.1. *In a calendar which has n days in a year. if*

$$k \ge \frac{1+\sqrt{8n\log 2}}{2}, \tag{14.2}$$

then the probability that 2 of k people shares the same birthday is $\ge \frac{1}{2}$. In particular, for $n=365$, if $k \ge 23$, then $p > \frac{1}{2}$.

3.2. Deterministic Random Walk. Suppose $f : S \to S$ is a random map between a set S of size n and itself. Pick a random value $x_0 \in S$ and compute recursively

$$x_{i+1} = f(x_i) \text{ for } i \ge 0.$$

The sequence s $x_0, x_1, x_2, \cdots$ is called a deterministic random walk determined by the initial value x_0. Since S is finite, we must eventually obtain $x_i = x_j$ for some $i \ne j$. This is called a collision. Then

$$x_{i+1} = f(x_i) = f(x_j) = x_{j+1}$$

and $x_{i+n} = x_{j+n}$ for all $n \ge 0$. Hence the sequence $x_0, x_1, x_2, \cdots,$ will eventually become cyclic. If we draw this sequence in the board, then it looks like the Greek letter ρ. In other words it has a cyclic part and an initial tail.

By the Birthday Paradox, one has

Proposition 14.2. *Let $\alpha > 0$, $l = 1 + [\sqrt{2\alpha n}]$, then*

$$\frac{\#\{(f,x_0) : x_0, \cdots, x_l \textit{ are distinct }\}}{\#\{(f,x_0)\}} < e^{-\alpha}.$$

Proof. The denominator is equal to $\#f \cdot \#x_0 = n^n \cdot n = n^{n+1}$. Now we compute the numerator. Note that x_0 has n choices, $f(x_0) = x_1$ has $n-1$ choices, $\cdots$, $f(x_{l-1}) = x_l$ has $n-l$ choices, and $f(x_i)$ can take arbitrary elements for $i \ge l$. Then the numerator is $n(n-1)\cdots(n-l)n^{n-l}$. Then by $1+x<e^x$,

$$\frac{\#\{(f,x_0) : x_0, \cdots, x_l \text{ are distinct }\}}{\#\{(f,x_0)\}} = \prod_{j=1}^{l}(1-\frac{j}{n}) < e^{-\alpha}$$

if $l = 1 + [\sqrt{2\alpha n}]$. □

From this proposition, one can show that for a random mapping the tail has expected length (i.e. the number of elements in the tail) $\sqrt{\frac{\pi n}{8}}$, and the cycle has expected length (i.e. the number of elements in the cycle) $\sqrt{\frac{\pi n}{8}}$ as well. The goal of many of Pollard's algorithms is to find a collision in a random map like the one above.

The collision is actually obtained by the following *Floyd's cycle finding algorithm.* Given (x_1, x_2), we compute (x_2, x_4) and then (x_3, x_6) and so on, i.e. given the pair (x_i, x_{2i}), we compute

$$(x_{i+1}, x_{2i+2}) = (f(x_i), f(f(x_{2i}))).$$

We stop when we find $x_m = x_{2m}$. If the tail of the sequence $x_0, x_1, x_2, \cdots$ has length λ and the cycle has length μ, then one can show that $x_m = x_{2m}$ for $m = \mu(1 + \lfloor\frac{\lambda}{\mu}\rfloor)$. Since $\lambda < m \leq \lambda + \mu$ we see that $m = O(\sqrt{n})$.

3.3. Pollard ρ-Method. We now work on Pollard ρ-method in the discrete logarithm problem case. Let G denote a cyclic group of order n and let the discrete logarithm problem be given by $h = g^x$. We partition the group into three sets S_1, S_2 and S_3, and assume $1 \notin S_2$. Then we define the following random walk on the group G:

$$x_{i+1} = f(x_i) = \begin{cases} hx_i, & x_i \in S_1; \\ x_i^2, & x_i \in S_2; \\ gx_i, & x_i \in S_3. \end{cases}$$

In practice we actually keep track of three pieces of information (x_i, a_i, b_i), where

$$a_{i+1} = \begin{cases} a_i, & x_i \in S_1; \\ 2a_i \mod n, & x_i \in S_2, \\ a_i + 1 \mod n. & x_i \in S_3. \end{cases} \qquad b_{i+1} = \begin{cases} b_i + 1 \mod n, & x_i \in S_1; \\ 2b_i \mod n, & x_i \in S_2; \\ b_i, & x_i \in S_3. \end{cases}$$

If we start with the triple $(x_0, a_0, b_0) = (1, 0, 0)$, then for all i,

$$\log_g(x_i) = a_i + b_i \log_g(h) = a_i + b_i x.$$

Applying Floyd's cycle finding algorithm we obtain a collision, and so find a value of m such that $x_m = x_{2m}$. This leads us to deduce the following equality of discrete logarithms

$$\begin{aligned} a_m + b_m x &= a_m + b_m \log_g h \\ &= \log_g(x_m) = \log_g(x_{2m}) \\ &= a_{2m} + b_{2m} \log_g h = a_{2m} + b_{2m} x. \end{aligned}$$

Rearranging, we see that

$$(b_m - b_{2m})x = a_{2m} - a_m,$$

if $b_m \neq b_{2m}$, then

$$x = \frac{a_{2m} - a_m}{b_m - b_{2m}} \mod n.$$

Pollard's ρ method for factorization works in the same way. Suppose $n = pq$ is the number to be factorized. Let $f(x)$ be any polynomial of degree ≥ 2 (except $x^2 - 2$) and the random walk be $x_{i+1} = f(x_i) \mod n$. Now just check $\gcd(|x_{2m} - x_m|, n)$ for each $m \geq 1$. Stop when the gcd is inside $(1, n)$. The expected time complexity is $O(\sqrt{p})$ where p is the smaller prime factor of n.

3.4. Pollard's λ Method. The λ method is particularly tuned to the situation where one knows that the discrete logarithm x lies in a certain interval $[a, \cdots, b]$. Let $\omega = b - a$ denote the length of the interval in which the discrete logarithm x is known to lie. Take $S = \{s_0, \cdots, s_{k-1}\}$ of integers in non-decreasing order, such that the mean m of S should be around $N = \sqrt{\omega}$. It is common to choose

$$s_i = 2^i \text{ for } 0 \leq i < k,$$

which implies that the mean of the set is $\frac{2^k}{k}$, and so we choose $k \approx \frac{1}{2}\log_2(\omega)$.

In Pollard's Λ method, two deterministic random walks are computed. Once a collision can be found between the two walks, then the discrete logarithm can be solved. The algorithm is as follows:

(1) Partition the group into k sets S_i, for $i = 0, \cdots, k-1$ and define the following deterministic random walk:

$$x_{i+1} = f(x_i) = x_i g^{s_j} \text{ if } x_i \in S_j.$$

(2) Compute the first deterministic random walk, starting from $g_0 = g^b$, by setting $g_i = f(g_{i-1})$ for $i = 1, \cdots, N$. In the mean time set $c_0 = b$ and compute $c_{i+1} = c_i + s_j \mod n$, so that $c_i \log_g(g_i)$ for each i. Store (g_N, c_N).

(3) Compute the second deterministic random walk starting from $h_0 = h = g^x$ by setting $h_{i+1} = f(h_i) = h_i g^{s'_j}$. Set $d_0 = 0$ and compute $d_{i+1} = d_i + s'_j \mod n$. Note that

$$\log_g(h_i) = x + d_i \mod n.$$

If the path of the h_i's meets that of the path of the g_i's, then the h_i's will carry on the path of the g_i and we will be able to find a value M where h_M equals our stored point g_N. At this point we have

$$c_N = \log_g(g_N) = \log_g(h_M) = x + d_M,$$

and so the solution to our discrete logarithm problem is given by

$$x = c_N - d_M \mod q.$$

If a collision is not obtained, then increase N and continue both walks in a similar manner until a collision does occur.

3.5. Parallel Pollard's ρ Method. Suppose we are given the discrete logarithm problem $h = g^x$ in a group G of prime order n. We first decide on an easily computable function

$$H : G \to \{1, \cdots, k\},$$

where k is usually around 20 (i.e. partition G into k subsets). Then we define a set of multipliers m_i for $i \leq k$, who are produced by generating random integers $a_i, b_i \in \{0, \cdots, n-1\}$ and then setting

$$m_i = g^{a_i} h^{b_i}.$$

To start a deterministic random walk we pick randomly $s_0, t_0 \in \{0, \cdots, n-1\}$ and compute

$$g_0 = g^{s_0} h^{t_0},$$

the deterministic random walk is then defined on the triples (g_i, s_i, t_i) where

$$\begin{cases} g_{i+1} &= g_i \cdot m_{H(g_i)}; \\ s_{i+1} &= s_i + a_{H(g_i)} \quad \bmod n; \\ t_{i+1} &= t_i + b_{H(g_i)} \quad \bmod n. \end{cases}$$

Hence for every triple (g_i, s_i, t_i), $g_i = g^{s_i} h^{t_i}$.

Suppose there are m processors, each processor starts a different deterministic random walk from a different starting position using the same algorithm to determine the next element in the walk. When two processors, or even the same processor, meet an element of the group that has been seen before, then $g^{s_i} h^{t_i} = g^{s'_j} h^{t'_j}$ and one can solve for the discrete logarithm x. Hence we expect that after $O(\sqrt{\pi n/2}/m)$ iterations of these parallel walks we will find a collision.

However as described above, each processor needs to return every element in its computed deterministic random walk to a central server which then stores all the computed element, so the storage requirement is $O(\sqrt{\pi n/2})$, very large and highly inefficient. The storage can be reduced as follows.

Define a function $d : G \to \{0, 1\}$ such that $d(g) = 1$ around $\frac{1}{2^t}$ of the time, for example setting $d(g) = 1$ if a certain subset of t of the bits representing g are zero and $d(g) = 0$ if otherwise. Now only those triples such that $d(g_i) = 1$ will be stored and hence the storage becomes $O(\sqrt{\pi n/2}/2^t)$.

On the other hand, one expects to continue another 2^t steps before a collision is detected between two deterministic random walks. Hence, the computing time becomes $O(\sqrt{\pi n/2}/m + 2^t)$. This allows the storage to be reduced to any manageable amount, at the expense of a little extra computation.

4. Modern Method for DLP over Finite Fields

We need a definition

Definition 14.1. Let $\mathbb{F}_q$ be a finite field of characteristic p. Write

$$p = e^{(c+o(1))(\log q)^{l_p}(\log\log q)^{1-l_p}} = L_q(l_p, c).$$

Then

$$\mathbb{F}_q \text{ is of } \begin{cases} \text{high characteristic if } l_p > \frac{2}{3}, \\ \text{medium characteristic if } \frac{1}{3} \leq l_p \leq \frac{2}{3}, \\ \text{small characteristic if } l_p < \frac{1}{3}. \end{cases}$$

The case $l_p = \frac{1}{3}$ or $\frac{2}{3}$ is called the boundary case.

Definition 14.2. A polynomial f over a field $\mathbb{F}$ is called B-smooth if every irreducible factor of f is of degree $< B$.

Proposition 14.3. *One has*

(1) *The probability* $\Pr(x \leq n, x \text{ is } B\text{-smooth}) = \left(\dfrac{\log n}{\log B}\right)^{-\frac{\log n}{\log B}(1+o(1))}$.

(2) *The probability* $\Pr(f : \deg f \leq n, f \text{ is } B\text{-smooth}) = \left(\dfrac{n}{B}\right)^{-\frac{n}{B}(1+o(1))}$.

(3) *The probability* $\Pr(x \leq L_N(\alpha_1, c_1) \text{ is } L_N(\alpha_2, c_2)\text{-smooth})$ *is equal to* $L_N(\alpha_1 - \alpha_2, -\frac{c_1}{c_2}(\alpha_1 - \alpha_2))$.

4.1. Adleman Index Calculus Method. This method is to solve DLP for $\mathbb{F}_p^\times$ where p is a prime.

Algerithm 14.3 (Adleman's Algorithm for DLP over $\mathbb{F}_p^\times$). *Set* $B = L_p(\frac{1}{2}, c)$ *with* c *to be determined. The factor base* $\mathcal{F}$ *is the set of primes* $\leq B$*, i.e.,* $\mathcal{F} = \{\pi \in \mathbb{Z} \mid \pi \text{ is prime}, \pi \leq B\}$.

(1) *Relation Collection Phase: Randomly pick* $i \in \{0, \cdots, p-2\}$*, compute* $g^i \bmod p \in \{0, \cdots, p-1\}$. *Determine if* $g^i \bmod p$ *is* B*-smooth, if yes, then*

$$g^i = \prod_{\pi \in \mathcal{F}} \pi^{e_\pi} \mod p.$$

Then we get a relation

$$i \equiv \sum_{\pi} e_\pi \log_g \pi \mod (p-1).$$

Repeat until the linear system of the relations has rank $|\mathcal{F}|$.

(2) *Linear Algebra Phase: Solve the linear system, then compute* $\log_g \pi$ *for all* $\pi \in \mathcal{F}$.

(3) *Individual Algorithm: Randomly pick* $k \in \{0, \cdots, p-2\}$*, compute* $hg^k \bmod p$ *until it is* B*-smooth, then*

$$hg^k \equiv \prod_{\pi \in \mathcal{F}} \pi^{c_\pi} \mod p,$$

and hence

$$\log_g h = \sum_{\pi \in \mathcal{F}} c_\pi \log_g \pi - k \mod (p-1).$$

We now give Complexity Analysis of Adelman's Algorithm. Let T_1 be the time to produce the linear system, T_2 be the time to solve the linear system, and T_3 be the time to compute $\log_g h$.

We know $T_3 \leq T_1$ and $T_2 = O(|\mathcal{F}|^3) = L_p(\frac{1}{2}, 3c)$. By Prime Number Theorem, $|\mathcal{F}| \approx \frac{B}{\log B}$, then Step 1 needs to collect $|\mathcal{F}| \approx \frac{B}{\log B}$ relations, so the time cost T_1 is

$$|\mathcal{F}| \cdot \frac{1}{\Pr(g^i \text{ is } B\text{-smooth})} = |\mathcal{F}| \cdot L_p(\frac{1}{2}, -\frac{1}{2c}) = L_p(\frac{1}{2}, c + \frac{1}{2c}).$$

If $T_1 \approx T_2$, then $3c = c + \frac{1}{2c}$ and $c = \frac{1}{2}$. The overall running time is then $L_p(\frac{1}{2}, \frac{3}{2})$.

4.2. Number Field Sieve. This is the best method up to now to solve DLP of $\mathbb{F}_q^\times$. For medium and high characteristic, the complexity applying Number Field Sieve is $L_{p^n}(\frac{1}{3})$. We give a sketch of this method here.

Suppose $q = p^n$. Pick $f_1, f_2 \in \mathbb{Z}[x]$, two monic and irreducible polynomials of degree n. Then

$$\mathbb{F}_q = \mathbb{F}_p^n = \mathbb{Z}[x]/(f_1) = \mathbb{Z}[x]/(f_2).$$

Let θ_i be a root of f_i, $K_i = \mathbb{Q}(\theta_i)$, then $[K_i : \mathbb{Q}] = n$. Let the conductor $f_{\theta_i} = [O_{K_i} : \mathbb{Z}[\mathbb{Q}_i]]$, where O_{K_i} is the ring of integers of K_i. Let $B = L_{p^n}(\frac{1}{3}, c')$ with c' to be determined. The factor base $\mathcal{F}$ is

$$\mathcal{F} = \{\mathfrak{p} : \text{ is prime in } O_{K_i}, N_{K_i/\mathbb{Q}}(\mathfrak{p}) \leq B \text{ or } \mathfrak{p} \mid (f_{\theta_i})\}.$$

The pair $(a, b) \in \mathbb{Z}^2$ is called B-smooth if $N_{K_i/\mathbb{Q}}(a + b\theta_i) \in \mathbb{Z}$ is B-smooth. The idea of Number Field Sieve is to find a lot of relations just like Adleman's Algorithm so that the linear system is solvable and in turn DLP is solved. The best result we know so far is:

(1) High characteristic: $L_{p^n}\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right)$;
(2) Medium characteristic: $L_{p^n}\left(\frac{1}{3}, \sqrt[3]{\frac{96}{9}}\right)$;
(3) Boundary $l_p = \frac{2}{3}$ case: $L_{p^n}\left(\frac{1}{3}, \sqrt[3]{\frac{48}{9}}\right)$.

CHAPTER 15

Hash Functions

Hash functions represent a third cryptography type alongside symmetric and asymmetric cryptography. It provides a number quantity that represents the input data, just like a finger print to a person. A cryptographic hash function is easy to generate a hash value from the input, but very difficult to reproduce the input by performing calculations on the generated hash. This property makes hash functions very useful in cryptography, in particular, the integrity and authenticity of messages.

1. Hash Functions

1.1. Definition and basic properties.

Definition 15.1. A cryptographic hash function h is a function which takes arbitrary length bit strings as input (often called a message) and produces a fixed length bit string as output (often called a hash code or hash value or the hash of the message), i.e.

$$h : \bigcup_{i\geq 1} \mathbb{F}_2^i \to \mathbb{F}_2^n, \quad m \mapsto h(m)$$

satisfying the following three properties:

(1) Preimage Resistance: Given a hash value y, it is hard to find a message m such tat $h(m) = y$.
(2) Collision Resistance: It is hard to find two messages with the same hash value.
(3) Second Preimage Resistance: Given one message, it is hard to find another message with the same hash value.

Lemma 15.1. *Second Preimage Resistance implies Preimage Resistance, Collision Resistance implies Second Preimage Resistance and hence Preimage Resistance.*

Proof. We prove Second Preimage Resistance implies Preimage Resistance. Given a message m, compute $y = h(m)$. If finding the preimage is easy, we can find a message m' such that $y = h(m')$. Then m and m' have the same hash value. □

From definition, a Hash function is not reversible: there exist two messages sharing the same hash value, but the collision resistance property makes it very difficult to find a collision. Thus a hash function does create a largely unique and fixed-length hash value based on the original message. Any slight change to the message will change the hash. Hashes cannot be used to discover the contents of the original message, or any of its other characteristics, but can be used to determine whether the message has changed. In this way, hashes provide confidentiality, but not integrity. Sending a message along with its hash value, the receiver can verify its integrity by simply hashing the message again using the same algorithm and comparing the two hashes: if they agree, the message is not changed; if they do not match, the message has been altered.

1.2. Designing Hash Functions. The most common way of constructing a hash function is to iterate a compression function on the input message. The compression function is usually designed from scratch or made out of a block cipher. Once the compression function is given, Merkle-Damgärd find the following algorithm to construct Hash functions.

ALGERITHM 15.1 (Merkle-Damgärd Construction Algorithm). *Suppose $f : \mathbb{F}_2^s \to \mathbb{F}_2^n$ is a compression function with $s > n$, which is believed to be collision resistant. Let $l = s - n$.*

(1) *For a message m, add zeros to m so that the number of bits of m is a multiple of l bits in length.*
(2) *Add a final block of l bits which encodes the original length of m, (thus the original message m has $< 2^l$ bits).*
(3) *Divide the message m into t blocks of l bits long, $m_1, \cdots, m_t$, i.e. $m = (m_1|m_2|\cdots|m_t)$, set*

$$H_0 \in \mathbb{F}_2^n \text{ and } H_i = f(H_{i-1}|m_i),\ i = 1, \cdots, t.$$

Output $H_t = H(m)$, the hash value of m.

1.3. Families of Hash functions. The following families of hash functions have been used:

- MD4: This has 3 rounds of 16 steps and an output bitlength of 128 bits.
- MD5: This has 4 rounds of 16 steps and an output bitlength of 128 bits.
- SHA-1: This has 4 rounds of 20 steps and an output bitlength of 160 bits.
- RIPEMD-160: This has 5 rounds of 16 steps and an output bitlength of 160 bits.
- SHA-256: This has 64 rounds of single steps and an output bitlength of 256 bits.
- SHA-384: This is identical to SHA-512 except the output is truncated to 384 bits.
- SHA-512: This has 80 rounds of single steps and an output bitlength of 512 bits.

In recent years a number of weaknesses have been found in almost all of the early hash functions in the MD4 family, for example MD4, MD5 and SHA-1. Hence, it is wise to move all applications to use the SHA-2 algorithms.

2. Message Authentication Codes (MAC)

Given a message and its hash code, as output by a cryptographic hash function, the integrity of data is persevered by recomputing the hash and comparing the two hash values. However, using a hash function in this way requires the hash code itself to be protected in some way, by for example a digital signature, as otherwise the hash code itself could be tampered with. To avoid this problem one can use a form of keyed hash function called a message authentication code, or MAC.

Suppose two parties, who share a secret key, wish to ensure that data transmitted between them has not been tampered with. The sender uses the shared secret key and a keyed algorithm to produce a check-value, or MAC, which is sent with the data, i.e. he transmits

$$m \parallel \mathrm{MAC}_k(m)$$

where MAC is the check function, k is the secret key and m is the message.

If moreover the users wants the message to remain confidential, the sender should encrypt it before applying the MAC, i.e. the sender should transmit

$$e_{k_1}(m) \parallel \mathrm{MAC}_{k_2}(e_{k_1}(m)).$$

To produce a secure MAC from a hash function one needs to be a little more clever. A MAC, called HMAC, occurring in a number of standards documents works as follows:

$$\mathrm{HMAC} = h(k \parallel p_1 \parallel h(k \parallel p_2 \parallel m)),$$

where p_1 and p_2 are strings used to pad out the input to be hash function to a full block.

CHAPTER 16

Key Exchange and Signature Schemes

1. Diffie-Hellman Key Exchange

The Diffie-Hellman Key Exchange (DHKE) is a method that allows two parties that have no previous knowledge of each other to securely establish a shared secret key over a public channel. This key can then be used to encrypt communications using a symmetric key cipher. This was the first widely-used method of safely developing and exchanging keys over an insecure channel.

Diffie Hellman Key Exchange Algorithms was made public by Diffie and Hellman (again in their 1976 paper) to overcome the problem of key agreement and exchange of symmetric key cryptography. It was said that the scheme was created by Ralph Merkle in 1976. This development was extremely valuable among public key cryptography and is still implemented in today's security protocols.

ALGERITHM 16.1 (Diffie-Hellman Key Exchange, DHKE). *Suppose G is a abelian group generated by g. The basic message flows for the Diffie-Hellman protocol are given in the following diagram:*

$$\begin{array}{ccc} \textit{Alice} & & \textit{Bob} \\ a & g^a \longrightarrow & g^a \\ g^b & \longleftarrow g^b & b \end{array}$$

The two parties each have their own ephemeral secrets a and b. From these secrets both parties can agree on the same secret session key:

(1) *Alice can compute $K = (g^b)^a$, since she knows a and received g^b from Bob.*
(2) *Bob can also compute $K = (g^a)^b$, since he knows b and received g^a from Alice.*

The safeness of Diffie-Hellman key exchange is based on the hardness of the the Diffie-Hellman problem (DHP) for G. Eve, the attacker, can see the messages g^a and g^b. For her to recover the secret key $K = g^{ab}$, what she needs to do is exactly solving DHP.

EXAMPLE 16.1. Take $p = 2147483659$, $G = \mathbb{F}_p^\times = \langle 2 \rangle$ and $g = 2$.

$$\begin{array}{lcl} \text{Alice} & & \text{Bob} \\ a = 12345 & & b = 654323 \\ A = g^a = 428647416 & \longleftrightarrow & B = g^b = 450904856 \end{array}$$

The shared secret key is then computed via

$$K = B^a = A^b = 2^{ab} = 1333327162.$$

It seems that the key distribution problem is solved by DHKE, but there is an important issue: you need to be careful who you are agreeing a key with. Alice has no assurance that she is agreeing a key with Bob, which can lead to the following person in the middle attack:

$$\begin{array}{ccccc} \text{Alice} & & \text{Eve} & & \text{Bob} \\ a & \longrightarrow & g^a, g^b & \longleftarrow & b \\ g^m & \longleftarrow & m, n & \longrightarrow & g^n \\ & k_1 = g^{ma} & & k_2 = g^{nb} & \end{array}$$

Hence, Eve can be the middle person between Alice and Bob (both not knowing this). Then, we need to know the person you are agreeing with is really the one you are intending to communicate with!

2. Digital Signature Schemes

2.1. Requirement for public key siganture. The basic idea behind public key signatures is as follows:

(1)

$$\text{Message+Alice's private key=Signature,}$$

$$\text{Message+Signature+Alice's public key=YES/NO.}$$

(2)

$$\text{Message+Alice's private key=Signature,}$$

$$\text{Signature+Alice's public key=YES/NO+Message.}$$

The main idea is that only Alice can sign a message, which could only from her since only Alice has access to the private key(Signing Process). On the other hand anyone can verify Alice's signature, since everyone can have access to her public key (Verification).

The main problem is how are the public keys to be trusted? How do you know a certain public key is associated to a given entity? You may think a public key belongs to Alice, but it may belong to Eve. Eve can therefore sign checks etc., and you would think they come from Alice. Thus if the signature is valid, the recipient gets a guarantee of three important security properties:

(1) Message Integrity: The message has not been altered in transit.
(2) Message Origin: The message was really sent by Alice.
(3) Non-Repudiation: Alice can not claim she did not send the message.

2.2. RSA signature algorithm. The RSA encryption algorithm is particularly interesting since it can be used directly as a signature algorithm with message recovery. Suppose $N = pq \approx 2^n$, (N, e) is the public key, (p, q, d) is the private key, m is the message and $h : \mathbb{F}_2^\infty \to \mathbb{F}_2^n$ is the cryptographic hash function. Alice applies private key d to generate a signature s:

$$s \equiv h(m)^d \mod N$$

Bob applies Alice's public key e to check if

$$s^e \equiv h(m) \mod N$$

.

Note that the three security guarantee suggest that a hash function needs to have the three properties in its definition:

(1) Preimage Resistance: If not, for some random r, Eve can compute $h' = r^e \mod N$ and $m = h^{-1}(h')$, then she now has Alice's signature (m, r) on the message m.
(2) Collision Resistance: If not, then the signature can be attacked by a legitimate signer. Choose two message m and m' with $h(m) = h(m')$. He signs m and outputs the signature (m, s). Later he can repudiate this signature, saying it was really a signature on the message m'.
(3) Second Preimage Resistance: If not, an attacker obtains your signature (m, s) on a message m. The attacker finds another message m' with $h(m') = h(m)$. Now the attacker has the signature (m', s).

3. Digital Signature Algorithm

We have already presented the RSA digital signature scheme. You may ask why we need another one?

(1) What if someone breaks the RSA algorithm or finds that factoring is actually easy?
(2) RSA is not suited to some applications since its signature generation is a very costly operation.
(3) RSA signatures are very large (1024 bits or longer), some applications require smaller signature footprints.

One algorithm which addresses all of these concerns is the Digital Signature Algorithm, or DSA. One sometimes sees this referred to as the DSS, or Digital Signature Standard. Although originally designed to work in the group $\mathbb{F}_p^\times$, where p is a large prime, it is now common to see it used using elliptic curves, in which case it is called EC-DSA. The elliptic curve variants of DSA run very fast and have smaller footprints and key sizes than almost all other signature algorithms.

The DSA domain parameters are all public information and are much like those found in the ElGamal encryption algorithm:

(1) p is a large prime number between 512 and 2048 bits;

(2) q is a 160 bit prime number such that $q \mid p-1$;

(3) Random choose an integer x less than p and compute $g = x^{\frac{p-1}{q}}$, if $g = 1$ then we pick a new value of x until we obtain $g \neq 1$, this ensures that g is an element of order q in the group $\mathbb{F}_p^\times$, i.e. $g^q = 1 \mod p$.

Then the domain parameters are (p, q, g).

Alice generates her own private signing key x such that $0 < x < q$. The associated public key is $y = g^x \mod p$. To sign a message m the Alice performs the following steps:

(S1) Compute the hash value $h = H(m)$.
(S2) Choose a random ephemeral key, $0 < k < q$.
(S3) Compute $r = (g^k \mod p) \mod q$.
(S4) Compute $s = (h + xr)/k \mod q$.

The signature on m is then the pair (r, s). Notice this signature is therefore around 320 bits long.

To verify the signature (r, s) on the message m, the verifier performs the following steps:

(V1) Compute the hash value $h = H(m)$.
(V2) Compute $a = h/s \mod q$.
(V3) Compute $b = r/s \mod q$.
(V4) Compute $v = (g^a y^b \mod p) \mod q$.
(V5) Accept the signature if and only if $v = r$.

Example 16.2. Let $q = 13$, $p = 4q + 1 = 53$ and $g = 16$. The private key $x = 3$, the public key $y = g^3 \mod p = 15$. To sign a message which has hash value $h = 5$, we first generate the ephemeral secret key $k = 2$ and then compute

$$r = (g^k \mod p) \mod q = 5, \quad s = (h + xr)/k \mod q = 10.$$

To verify this signature the recipient computes

$$a = h/s \mod q = 7, \quad b = r/s \mod q = 7.$$

$$v = (g^a y^b \mod p) \mod q = 5.$$

Remark 16.1. For security reason, the primes p and q should satisfy:

(1) $p > 2^{512}$, although $p > 2^{1024}$ may be more prudent, to avoid attacks via the Number Field Sieve.
(2) $q > 2^{160}$ to avoid attacks via the Baby-Step/Giant-Step method.

3.1. DSA for arbitrary groups. DSA can be generalized to an arbitrary finite abelian group in which the discrete logarithm problem is hard. We write $G = \langle g \rangle$ for a group generated by g. Assume that

(1) g has prime order $q > 2^{160}$.
(2) The discrete logarithm problem with respect to g is hard.
(3) There is a public function f such that $f : G \longrightarrow \mathbb{Z}/q\mathbb{Z}$.

(4) A secret key x is chosen, and the public key y is again given by $y = g^x$.

Signatures are computed via the steps:

(S1) Compute the hash value $h = H(m)$.
(S2) Choose a random ephemeral key, $0 < k < q$.
(S3) Compute $r = f(g^k)$.
(S4) Compute $s = (h + xr)/k \mod q$.

The signature on m is then the pair (r, s). To verify the signature (r, s) on the message m the verifier performs the following steps:

(V1) Compute the hash value $h = H(m)$.
(V2) Compute $a = h/s \mod q$.
(V3) Compute $b = r/s \mod q$.
(V4) Compute $v = f(g^a y^b)$.
(V5) Accept the signature if and only if $v = r$.

4. Schnorr Signatures

Suppose G is a public finite abelian group generated by an element g of prime order q. The public/private key pairs are just the same as in DSA, namely the private key is an integer x in the range $0 < x < q$ and the public key is the element $y = g^x$. To sign a message m using the Schnorr signature algorithm:

(1) Choose an ephemeral key k in the range $0 < k < q$.
(2) Choose the associated ephemeral public key $r = g^k$.
(3) Compute $e = h(m \parallel r)$, notice how the hash function depends both on the message and the ephemeral public key.
(4) Compute $s = k + xe \mod q$.

The signature is given by the pair (e, s). The verification step is simple, we first compute $r = g^s y^{-e}$, the signature is accepted if and only if $e = h(m \parallel r)$.

The Schnorr signature is often used in a challenge response situation. To see this, we give the following scenario. A smart card wishes to authenticate you to a building or an ATM machine. The card reader has a copy of your public key $y = g^x$, while the card has a copy of your private key x:

$$\begin{array}{ccc}
\text{Card} & & \text{Card reader} \\
x \text{ (private key)} & & y = g^x \text{ (public key)} \\
k, r = g^k & \longrightarrow & r \\
e & \longleftarrow & e \\
s = (k + xe) \mod q & \longrightarrow & s
\end{array}$$

Then the reader verifies the signature by checking if $g^s = r y^e$.

5. Authenticated Key Agreement

Recall that the man in the middle attack worked because each end did not know who he/she was talking to. We can now authenticate each end by requiring the parties to digitally sign their messages. We will still obtain forward secrecy, since the long-term signing key is only used to provide authentication and is not used to perform a key transport operation.

The protocol invented by Menezes, Qu and Vanstone, now called the MQV protocol, offers another way to share a key:

$$\begin{array}{ccc} \text{Alice} & & \text{Bob} \\ (A = g^a, b) \text{ (long term)} & & (B = g^b, b) \text{ (long term)} \\ (c, C = g^c) & \longrightarrow & C \\ D & \longleftarrow & (d, D = g^d) \end{array}$$

In this way Alice knows A, B, C, D, a and c, and Bob knows A, B, C, D, b and d.

Let l denote half the bit size of the order of the group G. To determine the session key, Alice now computes

(A1) Convert C to an integer i.
(A2) Put $s_A = i \mod (2^l) + 2^l$.
(A3) Convert D to an integer j.
(A4) Put $t_A = j \mod (2^l) + 2^l$.
(A5) Put $h_A = c + s_A a$.
(A6) Put $P_A = (DB^{t_A})^{h_A}$.

Bob runs the same protocol but with the public and private keys swapped around in the obvious manner, namely

(B1) Convert D to an integer i.
(B2) Put $s_B = i \mod (2^l) + 2^l$.
(B3) Convert C to an integer j.
(B4) Put $t_B = j \mod (2^l) + 2^l$.
(B5) Put $h_B = d + s_B b$.
(B6) Put $P_B = (CA^{t_B})^{h_B}$.

Then $P_A = P_B$ is the shared secret.

To see why the P_A computed by Alice and the P_B computed by Bob are the same, we notice that the s_A and t_A seen by Alice, are swapped when seen by Bob, i.e. $s_A = t_B$ and $s_B = t_A$. Setting $\log(P)$ to be the discrete logarithm of P to the base g, we see that

$$\begin{aligned} \log(P_A) &= \log((DB^{t_A})^{h_A}) \\ &= (d + bt_A)h_A \\ &= d(c + s_A a) + bt_A(c + s_A a) \\ &= d(c + t_B a) + bs_B(c + t_B a) \\ &= c(d + s_B b) + at_B(d + s_B b) \\ &= (c + at_B)h_B \\ &= \log((CA^{t_B})^{h_B}) \\ &= \log(P_B). \end{aligned}$$

CHAPTER 17

Elliptic Curves

1. Elliptic Curves over general fields

1.1. Definition. Let K be a field. Let $\overline{K}$ be a fixed algebraic closure of K.

Definition 17.1. An elliptic curve E is a non-singular cure of genus 1 with a point O. If moreover the curve is defined over K and the point O is a K-rational point, then E is called an elliptic cure over K.

In the set $K^3 - \{(0,0,0)\}$, we say $(X,Y,Z) \sim (\lambda X, \lambda Y, \lambda Z)$ for $\lambda \in K^\times$. This defines an equivalence relation on this set. The projective plane $\mathbf{P}^2(K)$ is the set of equivalent classes, if letting $(X : Y : Z)$ be the equivalent class of (X,Y,Z), then

$$\mathbf{P}^2(K) = \{(X : Y : Z) \mid (0,0,0) \neq (X,Y,Z) \in K^3\}.$$

The affine plane $\mathbf{A}^2(K) = K^2$ is a subset of $\mathbf{P}^2(K)$ via the embedding $(x,y) \mapsto (x : y : 1)$.

A point $(X : Y : Z) \in \mathbf{P}^2(\overline{K})$ is called K-rational if there exists $\lambda \in \overline{K}^\times$ such that $(\lambda X, \lambda Y, \lambda Z) \in K^3$. Then $\mathbf{P}^2(K)$ is just the set of K-rational points of $\mathbf{P}^2(\overline{K})$.

We now have a more explicit definition of elliptic curves:

Definition 17.2. An elliptic curve E over K is the locus

$$E = \{(X : Y : Z) \mid F(X,Y,Z) = 0\} \subset \mathbf{P}^2(\overline{K})$$

of a non-singular homogeneous polynomial $F(X,Y,Z)$ of degree 3 given by (17.1)

$$F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^2 - (X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3),$$

with $a_1, a_2, a_3, a_4, a_6 \in K$, where the non-singular condition means that the system of equations

$$F(X,Y,Z) = \frac{\partial F}{\partial X} = \frac{\partial F}{\partial Y} = \frac{\partial F}{\partial Y} = 0 \tag{17.2}$$

has no solutions in $\overline{K}$.

The set $E(K)$ is the set of K-rational points of E. The point $(0 : 1 : 0) \in E(K)$ is called the point at infinity and denoted by O. The equation $F(X,Y,Z) = 0$ is called the projective Weierstrass equation of E.

Suppose $(X : Y : Z)$ is a point in E. If $Z = 0$, then $X = 0$ by the defining equation and $(X : Y : Z) = (0 : 1 : 0) = O$. If $Z \neq 0$, then $(X : Y : Z) = (x : y : 1)$ where $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$, and (x, y) is a solution of $f(x, y) = 0$ where

(17.3) $$f(x,y) = F(x,y,1) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6.$$

By the identification of (x, y) and $(x : y : 1)$, then

$$E = \{(x, y) \mid f(x, y) = 0\} \cup \{O\}.$$

The equation $f(x, y) = 0$ is called the affine Weierstrass equation of E. Often we just say E is the elliptic curve $f(x, y) = 0$. For a point $P = (x, y) = (X : Y : Z)$ in E, we call (x, y) the affine coordinate and $(X : Y : Z)$ the projective or homogeneous coordinate of P.

Remark 17.1. Note that the projective equation can also be obtained from the affine equation:

$$F(X, Y, Z) = Z^3 f(\frac{X}{Z}, \frac{Y}{Z}).$$

The defining polynomial $F(X, Y, Z)$ is determined by $a_i \in K$ for $i = 1, 2, 3, 4, 6$. Set

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

$$c_4 = b_2^2 - 24b_4,\ c_6 = -b_2^3 + 3b_2b_4 - 216b_6.$$

Then the number

$$\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6 = (c_4^3 - c_6^2)/1728$$

where the latter equality requires $\mathrm{char}(K) \neq 2, 3$ characterizes the singularity condition:

$$F \text{ is non-singular if and only if } \Delta \neq 0.$$

Definition 17.3. The discriminant $\Delta(E)$ of E is the number Δ, and the j-invariant $j(E)$ of E is the number $j(E) = \frac{c_4^3}{\Delta}$.

Proposition 17.1. *Two elliptic curves E and E' are isomorphic over K if they are related by a linear change of variables of the form*

$$x' = \mu^2x + r, \quad y' = \mu^3y + s\mu^2x + t$$

with $\mu \in K^\times$ and $r, s, t \in K$.

Through change of variables, one can simplify the Weierstrass equation of E (up to isomorphism) as follows:

(1) If $\mathrm{char}K \neq 2$, changing $(x, y$ to $(x', y') = (x, \frac{1}{2}(y - a_1x - a_3))$, then the equation becomes $y'^2 = 4x'^3 + b_2x'^2 + b_4x' + b_6$.

(2) If moreover char$K \neq 3$, changing (x, y) to $(x', y') = (\frac{x-3b_2}{36}, \frac{y}{108})$, then the equation becomes $y'^2 = x'^3 - 27c_4x' - 54c_6$. This means that if char$K \neq 2, 3$, one can always assume E is defined by
$$E : y^2 = x^3 + ax + b, a, b \in K.$$
In this case
$$\Delta = -16(4a^3 + 27b^2), \quad j = -1728(4a)^3/\Delta.$$
(3) If char$K = 2$, then E is defined by (i) the equation $y^2 + xy = x^3 + a_2x^2 + +a_6$ with $a_6 \neq 0$ or (ii) $y^2 + a_3y = x^3 + a_2x + a_6$, where the latter if and only if $j(E) = 0$ or equivalently $a_1 = 0$.

Theorem 17.1. *Two elliptic curves E and E' are isomorphic over $\overline{K}$ if and only if their j-invariants $j(E) = j(E')$.*

Example 17.1. Let $E : y^2 = x^3+x+5$, $E' : y'^2+4x'y'+3y' = x'^3+x'+1$, and $E'' : y''^2 = x''^3 + 4x'' + 4$ are elliptic curves defined over $K = \mathbb{F}_7$. Then $j(E) = j(E') = j(E'') = 5$.
$$E \cong E' : \begin{cases} x = 4x' + 3, \\ y = y' + 2x' + 5. \end{cases} \qquad \text{and} \quad E \not\cong E'' \text{ over } \mathbb{F}_7.$$

1.2. Group Law. The following theorem tells us for an elliptic curve E over a field K, $E(K)$ is an abelian group.

Theorem 17.2. *There exist an operation "+", such that E is an abelian group with O the identity under this operation. This means*
1. *(Associativity): for points P, Q, R of E, $(P+Q)+R = P+(Q+R)$.*
2. *(Commutativity): for points P and Q of E, $P + Q = Q + P$.*
3. *(Identity element): for any point P of E, $P + O = O + P = P$.*
4. *(Inverse) for any point P of E, there exists a unique point $-P$ such that $(-P) + P = O$.*

The group law can be described as follows. Let $P : (x_1, y_1)$ and $Q : (x_2, y_2)$ be two different points on the elliptic curve E, we start by drawing the line $\overline{PQ}$ through P and Q. The line intersects E at three points, namely P,Q and one other point $R = (x_3, -y_3)$. We take that point R and reflect it across the x-axis to get a new point $R' = (x_3, y_3)$, the point R' is then the "sum of P and Q", we write $P + Q = R'$. If $P = Q$, then use the tangent line at P to replace the line $\overline{PQ}$.

For simplicity, suppose $E : y^2 = x^3 + ax + b$ and the coordinates of P and Q are (x_1, y_1) and (x_2, y_2). Then the line $\overline{PQ}$ has equation $y = \lambda x + \mu$ with λ and μ explicitly determined by x_i and y_i. Now x_1 and x_2 are roots of the cubic equation
$$(\lambda x + \mu)^2 = x^3 + ax + b.$$
Viète's Theorem then tells us the third root $x_3 = \lambda^2 - x_1 - x_2$. By this way we can write down the group law explicitly.

PROPOSITION 17.2. *Let E be defined by $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$. Suppose $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = (x_3, y_3) = P_1 + P_2$. Then*

(1) *The inverse of P is $-P_1 = (x_1, -y_1 - a_1x - a_3)$*
(2) *If $Q \neq \pm P$ (hence $x_1 \neq x_2$), let*

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad \mu = \frac{y_1x_2 - y_2x_1}{x_2 - x_1};$$

if $Q = P$, let

$$\lambda = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, \quad \mu = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}.$$

Then

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

COROLLARY 17.1. *Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and $P_3 = P_1 + P_2 = (x_3, y_3)$.*

(1) *If $\mathrm{char}K \neq 2, 3$, assume $E/K : y^2 = x^3 + ax + b$. Then $-P_1 = (x_1, -y_1)$ and*

$$x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1 \tag{17.4}$$

where

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq \pm Q, \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{if } P = Q,\ y_1 \neq 0. \end{cases}$$

(2) *If $\mathrm{char}K = 2$, and assume $j(E) \neq 0$ and $E : y^2 + xy = x^3 + a_2x^2 + a_6$ with $a_6 \neq 0$, then $-P = (x_1, x_1 + y_1)$ and*

$$x_3 = \lambda^2 + \lambda + a_2 + x_1 + x_2, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \tag{17.5}$$

where

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq \pm Q, \\ \dfrac{x_1^2 + y_1}{x_1}, & \text{if } P = Q,\ x_1 \neq 0. \end{cases}$$

1.3. Multiplication by n. For $n \in \mathbb{Z}$, if $n > 0$, we let $[n]P = P + \cdots + P$ (n times), and $[-n]P = -[n]P$. Then $[n]$ is a group homomorphism of E.

2. Elliptic Curves over Finite Fields

2.1. Basic properties. Let $K = \mathbb{F}_q$ be a finite field, $q = p^f$ and char $\mathbb{F}_q = p > 0$. Then $E(\mathbb{F}_q)$ is a finite abelian group determined by

$$E(\mathbb{F}_q) = \{(x, y) \mid\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}.$$

Note that if (x, y) is a point of E, then $f(x^q, y^q) = f(x, y)^q = 0$ and hence (x^q, y^q) is also a point of E.

Definition 17.4. For $E/\mathbb{F}_q$, the q-Frobenius of E is the homomorphism $\varphi = \varphi_q : (x, y) \mapsto (x^q, y^q)$ and $O \mapsto O$.

One easily checks that $\varphi(P+Q) = \varphi(P)+\varphi(Q)$ and φ is indeed a group homomorphism.

Theorem 17.3. *Let $t = q+1-\#E(\mathbb{F}_q)$. Then $\varphi^2-[t]\varphi+[q] = O$, which means for every point (x, y) of E, $(x^{q^2}, y^{q^2}) - [t](x^q, y^q) + [q](x, y) = O$.*

The number t is actually the trace of φ_q acting on the Tate module of E. Let α_q, β_q be two roots of $x^2 - tx + q = 0$, then

$$\alpha_q + \beta_q = t, \quad \alpha_q \cdot \beta_q = q.$$

Theorem 17.4 (Hasse). *One has $|\alpha_q| = |\beta_q| = \sqrt{q}$, and for all $n \geq 1$, $\#E(\mathbb{F}_{q^n}) = q^n + 1 - \alpha_q^n - \beta_q^n$. In particular,*

$$|t| = |q + 1 - \#E(\mathbb{F}_q)| \leq 2\sqrt{q}.$$

Definition 17.5. E is called anomalous if $t = 1$, i.e. $\#E(\mathbb{F}_q) = q$. E is called supersingular if $p \mid t$.

We note that the elliptic curve discrete logarithm problem (ECDLP) is then: given points P and Q on the elliptic curve E over a finite field $\mathbb{F}_q$, find n, such that $[n]P = Q$.

In application, elliptic curves over $\mathbb{F}_q$ for either $q = p$ a large prime or $q = 2^p$ with p prime are more important than other cases, as the associated ECDLP is harder to crack. Supersingular elliptic curves are now used in Post-quantum cryptography.

2.2. Addition using projective coordinates. As we know, division operation is needed for addition of points of an elliptic curve. This is expensive even for finite fields. To avoid this problem, in practice people often use the homogeneous coordinates instead of the affine coordinates.

2.3. Point Compression. To store a point $(x, y) \in E$ in a computer, one can use the point compression via a bijection: $(x, y) \in E \longleftrightarrow (x, b_1)$ where b_1 is just one bit.

(1) For large prime characteristic $q = p$. Suppose $E : x^3 + ax + b$. For $P = (x, y)$, then

$$y = \pm\sqrt{x^3 + ax + b} = \text{either } \sqrt{x^3 + ax + b} \text{ or } p - \sqrt{x^3 + ax + b} \in \mathbb{F}_p.$$

Identifying $\mathbb{F}_p$ and $\{0, \cdots, p-1\}$, then their parities are different. Let b_1 be the parity of y, i.e., $b_1 = 1$ if y is odd and 0 if y is even. Conversely, to recover (x, y) from (x, b_1), let $\beta = \sqrt{x^3 + ax + b}$, then $y = \beta$ if the parity of β is b_1 and $y = p - \beta$ if the parity of β is not b_1. This establishes the bijection.

(2) For even characteristic, let $q = 2^n$. Suppose $E : y^2 + xy = x^3 + a_2x + a_6$. For $P = (x, y)$, let $b_1 = 0$ if $y = 0$ and be the least significant bit of $z = \frac{y}{x}$ if $y \neq 0$.

To recover (x, y) given (x, b_1): If $x \neq 0$, let $\alpha = x + a_2 + \frac{a_6}{x^2}$ and β be a solution of $z^2 + z = \alpha$. If the least significant bit of β is b_1, set $y = x\beta$, otherwise set $y = x(\beta + 1)$. This works since y/x and $1 + y/x$ are the two roots of $z^2 + z = \alpha$ for (x, y) a point of $E : y^2 + xy = x^3 + a_2x + a_6$.

CHAPTER 18

Applications of Elliptic Curves

1. ECDLP-based algorithms

Analogue to DLP of finite fields, based on ECDLP, we have the elliptic curve variants of the cryptographic algorithms mentioned before.

1.1. EC-ElGamal Encryption. The public parameter: E is a given elliptic curve over a finite field $\mathbb{F}$, $P \in E(\mathbb{F})$ is of prime order q. The private key: x, the public key: $Y = [x]P$.

(1) Encryption: to encrypt $G \in E(\mathbb{F})$, generate some random key k, compute $c_1 = [k]P$ and $c_2 = G + [k]Y$. Then (c_1, c_2) is the ciphertext of G.

(2) Decryption: $G = c_2 - [x]c_1$.

1.2. EC-DH (EC Diffie-Hellman). Given (E, G) with E an elliptic curve over a finite field $\mathbb{F}$ and $G \in E(\mathbb{F})$.

$$\begin{array}{cc} \text{Alice} & \text{Bob} \\ a & b \\ A = [a]G & B = [b]G \end{array}$$

Then $[b]A = [a]B = [ab]G$. The shared key $K = x([ab]G)$ is the x-coordinate of $[ab]G$.

1.3. EC-DSA. The public parameter is $(E/\mathbb{F}, P)$ with $P \in E(\mathbb{F})$ a point of prime order q. The private key is x, the public key is $Y = [x]P$, the message is m, the hash value of m is $H(m) = h$.

(1) Signature: Choose a random k, $r = x([k]P)$, $s = (h + xr)/k \mod q$. The signature is (r, s).

(2) Verification: $a = \frac{h}{s} \mod q$, $b = \frac{r}{s} \mod q$, $Z = [a]P + [b]Y$, check if $r = x(z)$.

EXAMPLE 18.1. Suppose $E : Y^2 = X^3 + X + 3$ is an elliptic curve over $\mathbb{F}_{199}$. Then $\#E(\mathbb{F}_{199}) = 197 = q$ is prime and $P = (1, 76)$ is of order q. Let $x = 29$ be the private key, then the public key $Y = [x]P = (113, 91)$. Suppose $H = H(m) = 68$.

- Choose a random $k = 153$, then $[k]P = [153](1, 76) = (185, 35)$. Hence $r = 185$ and $s = \frac{68+29\cdot 185}{153} = 78 \mod 197$. The signature $(r, s) = (185, 78)$.
- $a = \frac{68}{78} = 112$, $b = \frac{185}{78} = 15$, $Z = [112](1, 76) = [15](113, 191) = (185, 35)$, hence $r = x(Z)$.

1.4. Development of attacks on ECDLP.

The attack on EC-DLP:

- Menezes-Okamato-Vanstone attack (MOV attack,1993): Using Weil-pairing to reduce it to DLP for finite field.
- Frey-Miiller-Riick (1999): Using Tate pairing.
- Diem (2011): Using Index calculus and summation polynomial based on Algebraic Geometry, this is the first sub-exponential algorithm

2. Elliptic curve primality test

The following is the Pocklington primality test which can prove a given number n is a prime or not.

Proposition 18.1 (Pocklington Primality Test). *Suppose there exist a prime q such that $q \mid n-1$. If there exists a such that*

(1) $a^{n-1} \equiv 1 \mod n$,
(2) $\gcd(a^{n-1/q}, n) = 1$.

Then n is a prime.

Proof. If n is not a prime, then there exists a prime p that $p \mid n$ and $p \le \sqrt{n}$. Since $q > p-1$, $\gcd(q, p-1) = 1$, there exists μ such that $\mu = q^{-1}$ mod $(p-1)$. Then

$$a^{(n-1)/q} \equiv a^{\mu q(n-1)/q} \equiv a^{\mu(n-1)} \equiv 1 \mod p.$$

This contradicts to (2). □

The elliptic curve primality test, due to Goldwasser-Lilian and in another variant by Atkin, is an analog of Pocklington's test.

Suppose n is a pseudo-prime which had already passed the Miller-Rabin primality test. We shall work on elliptic curves over the ring $\mathbb{Z}/n\mathbb{Z}$. Suppose $E: y^2 = x^3 + ax + b \mod n$ and $P = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. We want to add P_1 and P_2. This is done by the formula in Corollary 17.1(1). There are three cases: (i) if $x_1 = x_2$ and $y_2 = -y_1$, then $P_1 + P_2 = O$; (ii) if the denominator $x_1 - x_2$ or y_1 is invertible in $\mathbb{Z}/n\mathbb{Z}$, just apply (17.4); (iii) if the denominator is not invertible, then n must be composite and the gcd of the denominator and n is a factor of n. Hence we may assume (iii) will never happen.

Proposition 18.2. *Let $E: y^2 = x^3 + ax + b \mod n$. Let $m \in \mathbb{Z}$ such that it has a prime factor $q \ge (n^{\frac{1}{4}} + 1)^2$. If there exists a point $P \in E$ such that*

$$mP = O \text{ and } (\tfrac{m}{q})P \ne O,$$

then n is a prime.

Proof. If n is not a prime, there exists a prime factor p of n such that $p \le \sqrt{n}$. Let $E_p: y^2 = x^3 + ax + b \mod p$, $m' = \#E(\mathbb{F}_p)$. By Hasse's Theorem, $m' \le p + 1 + 2\sqrt{p} \le (n^{\frac{1}{4}} + 1)^2 < q$, hence $\gcd(q, m') = 1$.

Let $\mu = q^{-1} \mod m'$ and P' be the image of P in E_p, then

$$(\frac{m}{q})P' = \mu q(\frac{m}{q})P' = O.$$

However, kP' in E_p is calculated using the same addition formula like in E, then $(\frac{m}{q})P \neq O$ implies that $(\frac{m}{q})P' \neq O$ in E_p. □

Algerithm 18.1 (Elliptic curve Primality Test). *Suppose n is a pseudo-prime.*

(1) *Randomly choose $a, x, y \mod n$, let $b = y^2 - x^3 - ax \mod n$, $P = (x, y)$ and $E : y^2 = x^3 + ax + b \mod n$.*

(2) *Use Schoof's algorithm, assuming n is a prime, compute $m = \#E(\mathbb{F}_n)$ which is roughly n:*

- (A) *If m can not be written as $m = kq$ with k small and q is a pseudo-prime, choose new a, x, y, and start again.*
- (B) *Now $m = kq$, k is small, and q is a pseudo-prime, then $q > (n^{\frac{1}{4}} + 1)^2$. Compute mP and kP.*
  - *If $mP \neq O$, then n is composite.*
  - *If mP is undefined, then n is composite and can be factorized.*
  - *If $kP = O$, start again.*
  - *If $mp = O$ and $kP \neq O$, then by proposition, n is a prime if q is a prime.*

(3) *Prove the primality of $q < \frac{n}{2}$.*

3. Factorization Using Elliptic Curve

This method is due to H. W. Lenstra.

Suppose n is the number to be factorized and all its prime factors > 100.

Proposition 18.3. *Let $E : y^2 = x^3 + ax + b$, $a, b \in \mathbb{Z}$ such that* $\gcd(4a^3 + 27b^2, n) = 1$. *Let P_1 and $P_2 \neq -P_1$ be two point in E whose coordinates have denominators prime to n, then $P_1 + P_2 \in E$ has coordinates with denominators prime to n if and only if there exists no prime $p \mid n$ such that the sum pf their images $\bar{P}_1$ and $\bar{P}_2$ in the elliptic curve $\bar{E} = E \mod p$ over $\mathbb{F}_p$ is not the infinity point O.*

Proof. The condition $\gcd(4a^3 + 27b^2, n) = 1$ is nothing but the polynomial $x^3 + ax + b \mod p$ has no multiple roots for $p \mid n$, or equivalently $\gcd(x^3 + ax + b \mod p, 3x^2 + a \mod p) = 1$.

Let $p \mid n$. Let $\bar{x}_i$ and $\bar{y}_i$ be the images of x_i and y_i in $\mathbb{F}_p$, which are well-defined since the coordinates of P_1 and P_2 have denominators prime to n.

Suppose $P_1 + P_2 \in E$ has coordinates with denominators prime to n. By (17.4), if $\bar{x}_1 \neq \bar{x}_2$, then $\bar{P}_1 + \bar{P}_2 \neq O$. If $\bar{x}_1 = \bar{x}_2$, then $\bar{P}_1 = \pm\bar{P}_2$. If $\bar{P}_1 = \bar{P}_2$ and if $\bar{y}_1 \neq 0$, then $\bar{P}_1 + \bar{P}_2 \neq O$. If $\bar{P}_1 = \bar{P}_2$ and if $\bar{y}_1 = 0$, then $3\bar{x}_1^2 + a = 0$ since $P_1 + P_2 \in E$ has coordinates with denominators prime to n, this means $x^3 + ax + b \mod p$ has multiple roots, not possible. If $\bar{P}_1 = -\bar{P}_2$

and $P_2 \neq \pm P_1$, then $\bar{x}_1 = \bar{x}_2$ and $\bar{y}_1 = -\bar{y}_2$. But in this case $x_2 \neq x_1$. Let $x_2 = x_1 + p^r x$ with $p \nmid x$, The condition and (17.4) imply that $p^r \mid y_2 - y_1$. Then $\bar{y}_1 = \bar{y}_2 = 0$ and $p^{r+1} \mid (y_2^2 - y_1^2)$. But $y_2^2 - y_1^2 = p^r x(3x_1^2 + a) \mod p^{r+1}$, then $p \mid 3x_1^2 + a$ and again not possible. Thus for all $p \mid n$, $\bar{P}_1 + \bar{P}_2 \neq O$.

Conversely suppose $\bar{P}_1 + \bar{P}_2 \neq O$ for all $p \mid n$. If $\bar{x}_1 \neq \bar{x}_2$, (17.4) implies that the coordinates of $P_1 + P_2$ have no denominators divisible by p. If $\bar{x}_1 = \bar{x}_2$, then $\bar{y}_2 = \pm\bar{y}_1$, but since $\bar{P}_1 + \bar{P}_2 \neq O$, $\bar{y}_2 = \bar{y}_1 \neq 0$. If $P_1 = P_2$, (17.4) implies that the coordinates of $P_1 + P_2$ have no denominators divisible by p. If $P_1 \neq \pm P_2$, write $x_2 = x_1 + p^r x$ with $p \nmid x$, then

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y_2^2 - y_1^2}{(y_2 + y_1)(x_2 - x_1)} \equiv \frac{3x_1^2 + a}{y_2 + y_1} \mod p$$

and $P_1 + P_2$ has no coordinates with denominators divisible by p. □

Algerithm 18.2 (Elliptic curve Factorization). *Suppose n is a number to be factorized and all its prime factors* > 100.

(1) *Choose $a, x, y \in \mathbb{Z}$, $b = y^2 - x^3 - ax$ such that* $\gcd(4a^3 + 27b^2, n) = 1$. *Let $E : y^2 = x^3 + ax + b$ and $P = (x, y) \in E(\mathbb{Q})$.*

(2) *Let B=smooth bound, C=a given bound,*

$$k = \prod_{\substack{l \text{ prime} \\ l \leq B}} l^{\alpha_l} \leq C.$$

Compute successively $kP = (x_k, y_k)$ and $(x_k \mod n, y_k \mod n)$. *If at some point, x_k and y_k have no inverse* $\mod n$, *then the denominators and n are not prime to each other. Note that*

$$p \leq \sqrt{n}, \quad C \sim \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}.$$

CHAPTER 19

Stream Cipher

Suppose the encryption system of a symmetric cipher is $c = e_k(m)$, and the decryption system is $m = d_k(c)$, where m is the plaintext, e is the encryption function, d is the decryption function, k is secret key, and c is ciphertext.

1. Principles of Symmetric Ciphers

Kerokoff's principle is the guideline to design a symmetric cipher: e and d are public knowledge and the secrecy of the message given the ciphertext depends totally on the secrecy of the secret key k. Thus the number of possible keys must be large! Nowadays the size of the key space should be around 2^{80}.

The best scheme of symmetric ciphers should satisfy:

(1) It must have been studied extensively.
(2) Worked for a long time.
(3) This is no known way to break it.

Note that this is not the case for a commercial secret.

The stream cipher is secure if the key is different for every message and the key is as long as message. For the practical reason, we use short key for long message and reuse keys.

Passive attack: The Attacker is only allowed to listen to encrypted messages.

Active attack: The adversary is allowed to insert, delete or reply messages between the two communication parties.

2. Stream Cipher basic

A Stream Cipher is an encryption method and is part of symmetric cryptography. It encrypts an arbitrary length of plain text, one bit at a time, with an algorithm that uses a key. This method is not much used now in modern cryptography.

For a stream cipher, the key stream generator generates the key stream (k_i) by the secret key, then the plaintext m_i is encrypted eith the key k_i to the ciphertext $c_i = m_i \oplus k_i$. For example, if $m_i = (110010101)$, $k_i = (101001110)$, then $c_i = (011011011)$. Certainly the decryption is nothing but $m_i = c_i \oplus k_i$

For a stream cipher to be secure, the key should never be used more than once. Otherwise,

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2.$$

This means that one needs to change keys frequently, either on a per message or a per session basis. There are two ways to generate keys:

- Public key cryptography, which can generate session key and message key.
- Stream cipher/block cipher, which is based on actual data.

Moreover, the key stream should be unpredictable, i.e. it should satisfy the following properties:

(1) Long period: if there exists N such that $k_i = k_i + N$, N must be large.
(2) Pseudo-random properties: The generator should produce a sequence with appears to be random. In other words, it should pass a number of statistical random tests.
(3) Large linear complexity: Determining more of the sequence from a part should be computationally infeasible.

Stream ciphers have two primary advantages. The keystream can be precalculated and buffered, which increases the speed. The second advantage is that if there is a bit error in the ciphertext, only one bit in the plaintext is effected. Stream ciphers also have two disadvantages. First the effort for initialization is quite high, thus high speed can only be achieved with longer plain texts. The second disadvantage is that the entire ciphertext must always decrypt together. One can not decrypt the ciphertext by parts.

3. Linear feedback shift register

A stream cipher needs to generate a key stream which is a (pesudo)-random sequence generated through a number of random seed values that use digital shift registers. The Linear feedback shift register is often used to fulfill this purpose.

Recall for an initial internal state: $[s_{L-1}, \cdots, s_0]$, let $s_j = c_1 s_{j-1} + \cdots + c_L s_{j-L}$ for $j \geq L$, then we get an infinite periodic sequence: $(s_0, s_1, s_2, \cdots)$ which is called a linear feedback shift register sequence generated by LSR(f) where the connecting polynomial $f(x) = 1 - c_1 x - \cdots - c_l x^L$. Let

$$M = \begin{pmatrix} 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \\ c_L & c_{L-1} & \ldots & c_1 \end{pmatrix}.$$

then $(s_{j-L+1}, \ldots, s_j) = (s_{j-L}, \ldots, s_{j-1}) \cdot M$ and $f(x) = \det(xM - I_L)$. We also note that:

- If $c_L \neq 0$, the sequence is purely periodic.

- If $c_L = 0$: the sequence is not purely periodic, but periodic after some terms.

The following theorem about linear feedback shift register is pivotal for pseudo-randomness:

THEOREM 19.1. *Let* $f(x) = 1 - c_1x - \cdots - c_lx^L \in \mathbb{F}_q[x]$ *and* $\hat{f}(x) = -c_L^{-1}f(x), (c_L \neq 0)$. *If* $\hat{f}(x)$ *is a primitive polynomial over* $\mathbb{F}_q[x]$, *then any sequence generated by* LSR(f) *is an* m*-sequence, which means that:*

(1) *Its smallest period is* $q^L - 1$.

(2) *It is a pseudo-random sequence:* $\#\{i \mid s_i = 0\} = q^{L-1} - 1$ *and* $\#\{i \mid s_i = b\} = q^{L-1}$ *for all* $b \neq 0$.

(3) *Every non-zero sequence generated by* LSR(f) *is obtained by translation from any given non-zero sequence.*

DEFINITION 19.1 (Linear complexity). For an infinite sequence $\mathbf{s} = (s_0, s_1, \dots)$, the linear complexity of $\mathbf{s}$, denoted by $L(\mathbf{s}) = \mathrm{LC}(\mathbf{s})$, is defined as:

(1) $L(\mathbf{s}) = 0$ if $s = 0$,

(2) $L(\mathbf{s}) = \infty$ if no LFSR generates $\mathbf{s}$,

(3) otherwise $L(\mathbf{s})$ is the length of the shortest LFSR generates $\mathbf{s}$.

Note that $L(\mathbf{s}) \leq N$ if $\mathbf{s}$ is periodic of period N.

EXAMPLE 19.1. Let $s^n = (s_0, \dots, s_{n-1})$. Then

(1) $0 \leq L(s^n) \leq n$,

(2) $L(s^n) = l_n$ where l_n is given in Berlekamp-Massey's Algorithm.

THEOREM 19.2. *If* $2l_n \leq n$ *and* LSR(f_n) *generates* $(s_0, \dots, s_{n-1})$ *from* $(s_0, \dots, s_{l_n-1})$, *then* f_n *is unique.*

To generate key streams for a stream cipher, we need the case that $q = 2$ and $\hat{f}(x)$ is primitive over $\mathbb{F}_2[x]$. We now assume this is the case.

If we use LFSR of size L to generate a key stream for a stream cipher, and the aggressor obtains at $2L$ bits of the key stream, then they can determine LFSR by Berlekamp-Massey's Algorithm. Then we need to use LFSR in a non-linear way (high non-linearity, producing sequences with high linear complexity).

4. Combining LFSRs

In practice, one needs to non-linearly combine LFSRs to achieve high linear complexity and low correlation.

EXAMPLE 19.2 (Gefle Generator). Three LFSRs of periods L_1, L_2, L_3 respectively. Let

$$z = f(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_3.$$

The linear complexity is $L_1L_2 + L_2L_3 + L_3$, the period is $(2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$, but $\Pr(z = x_1) = \Pr(z = x_3) = \frac{3}{4}$.

Example 19.3 (Filler Generator). One single primitive LFSR, $z = F(s_1, \ldots, s_L)$ with F of non-linearity order m. Then $L(z) = \sum_{i=1}^{m} \binom{L}{i}$.

Example 19.4 (Alternating step generator). Three LFSR of size L_1, L_2, L_3, which are pairwise coprime of roughly the same size.

$$\text{If } x_1 = 1 \begin{cases} \text{LFSR-2}(x_2) = x_2 \\ x_3 = x_3 \end{cases} \qquad \text{If } x_1 = 0 \begin{cases} x_2 = x_2 \\ x_3 = \text{LFSR-3}(x_3) \end{cases}$$

Then the period is $2^{L_1}(2^{L_2} - 1)(2^{L_3} - 1)$ and the linear complexity is $(L_2 + L_3)2^{L_1}$.

Example 19.5 (Shrinking generator).

$$\begin{cases} \text{LFSR-1} \longrightarrow \\ \text{LFSR-2} \longrightarrow \end{cases} \boxed{\text{If } x_1 = 1\text{,output } x_2\text{, otherwise output nothing}} \longrightarrow .$$

Period: $(2^{L_2} - 1)2^{L_1 - 1}$, linear complexity: $L_2 \cdot 2^{L_1}$.

Example 19.6 (As over 1 generator for GSM mobile phone).

$$\text{LFSR:} \begin{cases} x^{18} + x^5 + x^2 + x + 1, L_1 = 19 \\ x^{21} + x + 1, L_2 = 22 \\ x^{22} + x^{15} + x^2 + x + 1, L_3 = 23 \end{cases}$$

For the three LFSRs

$$\begin{cases} c_1 = \text{ the 10-th position in LFSR-1,} \\ c_2 = \text{ the 11-th position in LFSR-2,} \\ c_3 = \text{ the 12-th position in LFSR-3,} \\ m = c_1c_2 + c_2c_3 + c_3c_1. \end{cases}$$

If $m = c_i$, compute LFSR-i, output exclusive-or of the three sequences.

CHAPTER 20

Block Cipher

The block cipher is also a symmetric cpher, just like the stream cipher, in which a key and algorithm are applied to blocks of data rather than individual bits in a stream. Block cipher prefers in the DES and AES crypto standards and can also be resource-saving and fast, which makes it more widely used than stream cipher.

In a block cipher: block, the plaintext is a block m, the secret key is k, The cipher function is e, the ciphertext is c, i.e. $e(m, k) = c$.

The most famous block ciphers are

- DES: Data Encryption standard (mid-1970s). (NSA: national security agency. NIST: national institute of standards)
- AES: Advanced Encryption Standard (2000, NIST, invented by Rijndael)

1. Feistel Cipher and DES

1.1. Feistel Cipher. This was invented by H. Feistel at IBM in 1970's.

Suppose the alphabet is $\mathbb{F}_2 = 0, 1$, t is the block length. For a key k, the encryption function f is the function $f(k, R) = f_k(R)$ with both R and the values of f in $\mathbb{F}_2^t$. Feistel cipher is a $2t$ length block cipher as follows. Let $\mathbf{K}$ be the key space and r be the rounds.

(1) Take $K \in \mathbf{K}$, generate round keys: $K_1, K_2, \ldots, K_r$.
(2) (Encryption) For $P = (L_0, R_0)$ the plaintext of length $2t$ with L_0 and R_0 both t bits, for $1 \le i \le r$, compute $(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f_{K_i}(R_{i-1}))$. Then $E_k(P) = E_k(L_0, R_0) = (R_r, L_r)$.
(3) (Decryption) For $1 \le i \le r$, $(R_{i-1}, L_{i-1}) = (L_i, R_i \oplus f_{K_i}(L_i))$, repeat this and get (L_0, R_0).

We see that Feistel Cipher has very nice properties:

- f can be chosen arbitrarily.
- Encryption and decryption are the same, just using the round keys in the reverse order for decryption.

This leaves the following problems:

(1) How to generate $K_1, \cdots, K_r$ from $K \in \mathbf{K}$?
(2) How many rounds to take?
(3) How to choose f?

1.2. DES. This is a variant of Feistel cipher.

In DES, the rounds $r = 16$, the block length $n = 64$, the key length is 56 bits. Then $\mathbf{K} = \mathbb{F}_2^{56}$ and $|\mathbf{K}| = 2^{56}$. The round key length is 48 bits. Since the key space is small ($|\mathbf{K}| = 2^{80}$ is required), for safety reason, now 3-DES (use DES for three times and use different keys K) is used.

To describe the encryption of DES, we first define:

(1) Initial permutation $\mathrm{IP} : \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2^{64}$ is given by the table

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

whose inverse IP^{-1} is

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(2) Expansion permutation $E : \mathbb{F}_2^{32} \longrightarrow \mathbb{F}_2^{48}$ is given by

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(3) S-boxes: there are 8 DES S-boxes: for $1 \le i \le 8$, $S_i = (s^i_{s,t})_{0\le s\le 3, 0\le t\le 15}$, with $s^i_{s,t} \in \mathbb{F}_2^4 = \{0, 1, \cdots, 15\}$, see Figure 1.

For $(a_1, a_2, a_3, a_4, a_5, a_6) \in \mathbb{F}_2^6$, let $s = a_1a_6 = a_1 + a_6 \cdot 2$ and $t = a_2a_3a_4a_5 = a_2 + a_3 \cdot 2 + a_4 \cdot 4 + a_5 \cdot 8$, then S_i gives the map

$$S_i : \mathbb{F}_2^6 \to \mathbb{F}_2^4, \quad (a_1, a_2, a_3, a_4, a_5, a_6) \mapsto s^i_{s,t}.$$

TABLE 1. DES S-Boxes

S-Box 1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-Box 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-Box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box 5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-Box 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-Box 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

(4) P-box: $(\mathbb{F}_2^4)^8 \longrightarrow \mathbb{F}_2^{32}$ is given by

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Now the Encryption is as follows. Suppose P is the plaintext block.

(1) Compute $\text{IP}(P) = (L_0, R_0)$.
(2) Compute 16 DES rounds to get (R_{16}, L_{16}).
(3) The ciphertext block is $C = \text{IP}^{-1}(R_{16}, L_{16})$.

For each round, the function $f_{K_i}(R)$ is defined as follows:

- For $R \in \mathbb{F}_2^{32}$, by expansion get $E(R) \in \mathbb{F}_2^{48}$;
- Compute and write $E(R) \oplus K_i = B_1 B_2 \cdots B_8 \in \mathbb{F}_2^{48}, B_i \in \mathbb{F}_2^6$;
- $f_{K_i}(R) = (C_1 C_2 \cdots C_8)$ with $C_i = S_i(B_i)$.

The DES key schedule is as follows. First note that the 56-bit key is actually input as a bitstring of 64 bits comprising of the key and eight parity bits in bit positions 8, 16, $\cdots$, 64 for error detection, which ensure that each byte of the key contains an odd number of bits.

(1) $\text{PC} - 1$ is the permutation of the key K given by

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

hence no parity bits are left and a 56 bits is produced.

(2) Now suppose $K = (C_0, D_0)$ with C_0 and D_0 both 28 bits. For $1 \le i \le 16$, let

$$C_i = C_{i-1} \lll P_i,\ D_i = D_{i-1} \lll P_i$$

where $\lll$ means shift to the left by P_i positions, where

$$P_i = \begin{cases} 1, & \text{if } i = 1, 2, 9, 16, \\ 2, & \text{if otherwise.} \end{cases}$$

Now the map $\mathrm{PC}-2 : \mathbb{F}_2^{56} \to \mathbb{F}_2^{48}$ given by

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

produces the i-th round key $K_i = \mathrm{PC}-2(C_i, D_i)$.

2. AES (Rijndael)

AES, originally named Rijndal, is invented by Belgian cryptographers Daemen and Rijnmen.

Before we start, first note that the polynomial $m(x) = x^8+x^4+x^3+x+1$ is an irreducible polynomial in $\mathbb{F}_2[x]$, then we identify $\mathbb{F}_{2^8} = \mathbb{F}_2[x]/m(x)$ with $\mathbb{F}_2^8$ via

$$\begin{aligned} \mathbb{F}_{2^8} &\longleftrightarrow \mathbb{F}_2^8 \\ a(x) = a_0 + a_1x + \cdots + a_7x^7 &\longleftrightarrow (a_7, a_6, \cdots, a_0). \end{aligned}$$

i.e., a byte is identified with an element in $\mathbb{F}_{256}$. Note that a byte $(a_7, a_6, \cdots, a_0)$ is also identified with the hexadecimal string $\{a, b\}$ if $\sum a_i 2^i = a \times 16 + b$. A string of 32 bits is called a word, consisting 4 bytes via the following identification:

$$\begin{aligned} 32 \text{ bits} \in \mathbb{F}_2^{32} &\longleftrightarrow \mathbb{F}_{2^8}^4 = \mathbb{F}_{2^8}[X]/(X^4+1) \\ (a_0 \parallel a_1 \parallel a_2 \parallel a_3) &\longleftrightarrow a_3X^3 + a_2X^2 + a_1X + a_0. \end{aligned}$$

For Rijndael, blocks are of size 128, 192 or 256 bits (then number of words $Nb = 4, 6, 8$), and keys are of size 128, 192 or 256 bits (then number of words $Nk = 4, 6, 8$). The number of rounds $Nr = 10$, 12 or 14 if $Nk = 4, 6, 8$ respectively.

For simplicity, we consider the block size and the key size are both 128 and the number of rounds is 10. In this case, blocks and keys are regarded as 4×4 matrices of bytes, and if each word represents a column of 4 bytes, then and they can also be viewed as row vectors of four words.

The following four operations are used in Rijndael.

(1) SubBytes: There are two types of S-Boxes used in Rijndael: One for the encryption rounds and one for the decryption rounds, each one being the inverse of the other. We shall describe the encryption S-Box, the decryption one will follow immediately. This S-Box is performed in 2 steps:

- The inverse of $s = [s_7, \cdots, s_0] \in \mathbb{F}_{2^8}$ is computed to produce a new byte $x = [x_7, \cdots, x_0]$ (assume $0 \mapsto 0$).

- A new byte y is obtained via the linear transform:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

(2) ShiftRows: This is the transform

$$\begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix} \longrightarrow \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{11} & s_{12} & s_{13} & s_{10} \\ s_{22} & s_{23} & s_{20} & s_{21} \\ s_{33} & s_{30} & s_{31} & s_{32} \end{pmatrix}.$$

In general, the i-th row shift to the left for c_i positions with c_i given by the table:

Nb	c_0	c_1	c_2	c_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

(3) MixColumns: A word a, i.e. a column of 4 bytes is regarded as a polynomial $a_0 + a_1X + a_2X^2 + a_3X^3 \in \mathbb{F}_{2^8}[X]$, then the new word $b = b_0 + b_1X + b_2X^2 + b_3X^3$ is just given by

$$(a_0+a_1X+a_2X^2+a_3X^3)(\{0,2\}+\{0,1\}X+\{0,1\}X^2+\{0,3\}X^3) \mod (X^4+1).$$

This correspondence can be written as matrix multiplication.

(4) AddRoundKey: this is the map sending S to $S \oplus K_i$ (byte by byte).

Algerithm 20.1 (Rijndael Encryption). *Suppose S is the plaintext block. Then S is encrypted as follows:*

(1) AddRoundKey(S, K_0),
(2) *For $i = 1$ to 9,* SubBytes(S), ShiftRows(S), MixColumns(S), AddRoundKey(S, K_i).
(3) SubBytes(S), ShiftRows(S), AddRoundKey(S, K_{10}).

Algerithm 20.2 (Rijndael Decryption). *Suppose S is the ciphertext block. Then S is decrypted as follows:*

(1) AddRoundKey(S, K_{10}), InverseShiftRows(S), InverseSubBytes(S)
(2) *For $i = 9$ to 1,* AddRoundKey(S, K_i), InverseMixColumns(S), InverseShiftRows(S), InverseSubBytes(S).
(3) AddRoundKey(S, K_0).

We are left to describe the Key schedule.

ALGERITHM 20.3 (Rijndael Key Schedule). *Suppose the main key* $K = 128$ *bits is written as* (k_0, k_1, k_2, k_3) *of* 4 *words. The round keys* $K_i = (W_{4i}, W_{4i+1}, W_{4i+2}, W_{4i+3})$ *are produced by the following way:*

(1) $W_0 = k_0, W_1 = k_1, W_2 = k_2, W_3 = k_3$.

(2) *For* $i = 1$ *to* 10, $T = \mathrm{RotbBytes}(W_{4i-1})$ *where* RotbBytes *is the map rotating a word to the left by one byte,* $T = \mathrm{SubBytes}(T)$, $T = T \oplus \mathrm{RC}_i$ *where* $\mathrm{RC}_i = x^i$ *in* $\mathbb{F}_{2^8}$. *Then*

$$\begin{cases} W_{4i} = W_{4i-1} \oplus T, \\ W_{4i+1} = W_{4i-3} \oplus W_{4i}, \\ W_{4i+2} = W_{4i-2} \oplus W_{4i+1}, \\ W_{4i+3} = W_{4i-1} \oplus W_{4i+2}. \end{cases}$$