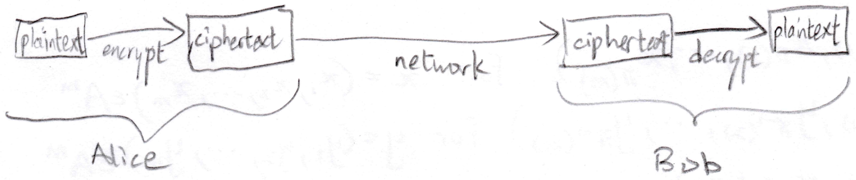
MA4261 Coding & Cryptography



Cryptosystem $= (P, C, K, E, D)$ where $\forall k \in K$, $d_k(e_k(x)) = x \ \forall x \in P$

- P: set of possible plaintexts
- C: set of possible ciphertexts
- K: set of possible keys (keyspace)
- E: set of encryption rules $\{e_k : P \to C\}$ ← key
- D: set of decryption rules $\{d_k : C \to P\}$ ← key

· key is shared via a separate secure channel

· **Good cryptosystem:** · easy to compute $e_k(x)$ and $d_k(y)$ $\forall k \in K, x \in P, y \in C$

· difficult to recover x from $e_k(x)$ if k is unknown

· **Multiplicative inverse thm:** $\forall a \in \mathbb{Z}_n$, $\gcd(a,n) = 1 \iff \exists$ unique $b \in \mathbb{Z}_n$ s.t. $ab \equiv 1 \pmod n$

To find b, use the extended Euclidean algorithm

Shift Cipher: A (alphabet) := n chars labelled $0, \dots, n-1$

$P = C = K = \mathbb{Z}_n$

$e_k(x) \equiv x + k \pmod n$

$d_k(y) \equiv y - k \pmod n$

Affine Cipher: $P = C = \mathbb{Z}_n$

$K = \{(a,b) \in \mathbb{Z}_n \times \mathbb{Z}_n \mid \gcd(a,n) = 1\}$

$e_k(x) \equiv ax + b \pmod n$

$d_k(y) \equiv a^{-1}(y - b) \pmod n$

Substitution cipher: $P = C$

$K = E = D = S_P$ (set of all permutations on P) (essentially, we can use any bijective mapping)

Monoalphabetic cipher: those where $P = A$.

Generalized affine cipher: $P = C = \mathbb{Z}_n^m$ ← i.e. we encode m-character chunks

$K = \{(A, b) \mid A$ is an $m \times m$ invertible matrix over $\mathbb{Z}_n$ and b is an m-vector in $\mathbb{Z}_n^m\}$ ← requires $\gcd(\det(A), n) = 1$

$e_k(x) = xA + b \pmod n$ $(x \in \mathbb{Z}_n^m)$

$d_k(y) = (y - b)A^{-1} \pmod n$ $(y \in \mathbb{Z}_n^m)$

Permutation cipher: $P = C = A^m$

$K = S_m$

$\forall \pi \in K$: $e_\pi(x) = (x_{\pi(1)}, x_{\pi(2)}, \cdots, x_{\pi(m)})$ for $x = (x_1, x_2, \dots, x_m) \in A^m$

$d_\pi(y) = (y_{\pi^{-1}(1)}, y_{\pi^{-1}(2)}, \cdots, y_{\pi^{-1}(m)})$ for $y = (y_1, y_2, \cdots, y_m) \in A^m$

(i.e. just permute m elements at a time)

- it is a subset of generalized affine cipher (using a permutation matrix)

Attack models:

- ciphertext-only attack : attacker only has the ciphertext string
- known-plaintext attack : attacker knows a plaintext string & its corresponding ciphertext string
- chosen-plaintext attack : attacker can choose a plaintext string to encrypt, and observe the corresponding ciphertext
- chosen-ciphertext attack: attacker can choose a ciphertext string to decrypt, and observe the corresponding plaintext

Frequency analysis: guess the key based on the frequency of letters in the ciphertext.

- monoalphabetic ciphers are easily broken by this

- known-plaintext attack for generalized affine ciphers : if we have (at least) $m+1$ distinct plaintext-ciphertext pairs,

i.e. $x^{(r)}A + b = y^{(r)}$, then let $X = \begin{pmatrix} x^{(1)} - x^{(0)} \\ \vdots \\ x^{(m)} - x^{(0)} \end{pmatrix}$, $Y = \begin{pmatrix} y^{(1)} - y^{(0)} \\ \vdots \\ y^{(m)} - y^{(0)} \end{pmatrix}$

then $XA = Y$.

so if X is invertible, then $A = X^{-1}Y$ and $b = y^{(0)} - x^{(0)}A$

Block ciphers: key does not change (i.e. there is some repeat)

Stream cipher: generate a keystream $z = z_1 \cdots z_s$, and to encrypt $x = x_1 \cdots x_s$ we get $y = y_1 \cdots y_s = e_{z_1}(x_1) \cdots e_{z_s}(x_s)$

$\shortparallel$

$$(P, C, K, \underset{\substack{\uparrow \\ \text{keystream} \\ \text{alphabet}}}{L}, E, D) \text{ and } \underset{\substack{\uparrow \\ g : K \to L^\infty}}{g}$$

is the keystream generator

- if keystreams are periodic then it is called a periodic stream cipher (then can use known-plaintext attack with the period)
- a possible keystream generator looks at the previous m keys to generate the new key (using some function)
- also vulnerable to known-plaintext attacks if used alone
- if the generator uses a linear recurrence relation that looks at the last m bits, then we only need a known-plaintext attack of size $2m$ (by solving m simultaneous linear equations). (obtained from a $2m$-length portion of the keystream)

S-box (substitution operation): $\pi_s : A^m \to A^{m'}$ (substitutes string x of length m by another string of length m')

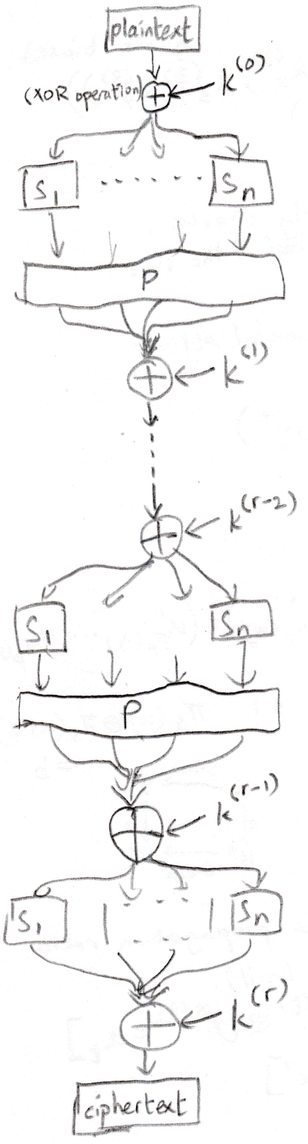
- usually, S-boxes are injective

P-box (permutation operation): $\pi_p : \{1,2,\dots,n\} \to \{1,2,\dots,n\}$

· is a permutation (i.e. bijective)

Substitution-permutation network: Plaintext & ciphertext are binary strings of mn bits.

Given a key k, use an algorithm to generate r+1 round keys $k^{(0)}, \dots, k^{(r)}$ each of mn bits



- For the different rounds, we might share the same S-boxes & P-box to make it cheaper... or have more separate different S-boxes & P-box.
- S-boxes have m-bit input & m-bit output
- P-boxes have mn-bit input
- it is desirable that an S-box have the property that changing one input bit will change about half the output bits
- small changes to input should lead to large changes in output so that two similar plaintexts look independent

no need a P-box for the last round, because P-box itself is easy to reverse

Linear cryptanalysis

Let X be a random variable taking values 0 or 1

Bias of X: $\varepsilon(X) = Pr[X=0] - 0.5$ (i.e. how unfair the coin is)

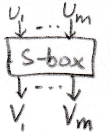
Hence: $Pr[X=0] = 0.5 + \varepsilon(X)$
$Pr[X=1] = 0.5 - \varepsilon(X)$

If $X_1, \dots, X_k$ are mutually independent random vars, then $\varepsilon(X_1 \oplus \cdots \oplus X_k) = 2^{k-1}\varepsilon(X_1)\cdots\varepsilon(X_k)$.

← Piling-up lemma

Bias of vector in S-box: $a = (a_1, \dots, a_m)$, $b = (b_1, \dots, b_m)$. then $\varepsilon(a,b)$ is the bias of: $(a_1U_1 \oplus \cdots \oplus a_mU_m) \oplus (b_1V_1 \oplus \cdots \oplus b_mV_m)$

where



$N_L(a,b)$ = number of 2m-tuples $(u_1, \dots, u_m, v_1, \dots, v_m) \in \mathbb{Z}_2^{2m}$
such that $(v_1, \dots, v_m) = \pi_s(u_1, \dots, u_m)$
and $(a_1u_1 \oplus \cdots \oplus a_mu_m) \oplus (b_1v_1 \oplus \cdots \oplus b_mv_m) = 0$.

So $\varepsilon(a,b) = \frac{N_L(a,b)}{2^m} - 0.5$. We want to pick a,b such that $|\varepsilon(a,b)|$ is large (i.e. it is biased)

④

E.g. if we find that $T_1 = U_2^{(1)} \oplus U_4^{(1)} \oplus V_3^{(1)} \oplus V_4^{(1)}$ is biased with $\varepsilon(T_1) = 0.375$ ↙ first row of S-boxes

(see slides Ch 2) and $T_2 = U_5^{(2)} \oplus U_7^{(2)} \oplus V_5^{(2)} \oplus V_6^{(2)}$ is biased with $\varepsilon(T_2) = 0.375$

then $T_1 \oplus T_2 = X_2 \oplus X_4 \oplus U_1^{(3)} \oplus U_4^{(3)} \oplus k_2^{(0)} \oplus k_4^{(0)} \oplus k_5^{(1)} \oplus k_7^{(1)} \oplus k_1^{(2)} \oplus k_4^{(2)}$

Let $c = k_2^{(0)} \oplus k_4^{(0)} \oplus k_5^{(1)} \oplus k_7^{(1)} \oplus k_1^{(2)} \oplus k_4^{(2)}$

so $$Pr[X_2 \oplus X_4 \oplus U_1^{(3)} \oplus U_4^{(3)} = 0] = \begin{cases} Pr[T_1 \oplus T_2 = 0] & \text{if } c = 0 \\ Pr[T_1 \oplus T_2 = 1] & \text{if } c = 1 \end{cases}$$

hence $\varepsilon(X_2 \oplus X_4 \oplus U_1^{(3)} \oplus U_4^{(3)}) = \pm\,\varepsilon(T_1 \oplus T_2) = \pm 2(0.375)^2 = \pm 0.28$

↑ quite biased

Since $(U_1^{(3)}, U_2^{(3)}, U_3^{(3)}, U_4^{(3)}) = \pi_S^{-1}((Y_1, Y_2, Y_3, Y_4) \oplus (k_1^{(3)}, k_2^{(3)}, k_3^{(3)}, k_4^{(3)}))$

Now suppose we have a plaintext-ciphertext pair $(x_1, \dots, x_8)$ and $(y_1, \dots, y_8)$,

we compute $(u_1', u_2', u_3', u_4') = \pi_S^{-1}((y_1, \dots, y_4) \oplus (k_1', \dots, k_4'))$

if $(k_1', \dots, k_4')$ is not the right key, then $x_2 \oplus x_4 \oplus u_1' \oplus u_4' = 0$ about half of the time. ↖ randomly chosen.

otherwise the bias is approx ±0.28.

<u>Lemma</u>: if the bias is ε, then we need $\Theta(\varepsilon^{-2})$ plaintext-ciphertext pairs to mount a successful attack

<u>Differential cryptanalysis</u>: picking pairs of plaintext-ciphertext pairs (x, x^*, y, y^*) such that $x \oplus x^* = x'$ is some fixed pattern we want.

- if u and u^* are xored with k, then $(u \oplus k) \oplus (u^* \oplus k) = u \oplus u^* = u'$
- if u and u^* pass through a P-box, then $(u_{\pi_P(1)}, \dots, u_{\pi_P(n)}) \oplus (u^*_{\pi_P(1)}, \dots, u^*_{\pi_P(n)}) = (u'_{\pi_P(1)}, \dots, u'_{\pi_P(n)})$
- <u>for S-boxes</u>: for $a, b \in \mathbb{Z}_2^m$, let $N_D(a,b)$ be the number of $u \in \mathbb{Z}_2^m$ s.t. $\pi_S(u) \oplus \pi_S(u+a) = b$
  - called the differential ↗ (a, b)

  (note: $N_D(0,0) = 2^m$)

  <u>propagation ratio</u>: $R_p(a,b) = \frac{N_D(a,b)}{2^m} = Pr[\pi_S(U) \oplus \pi_S(U^*) = b \mid U \oplus U^* = a]$ for uniformly distributed inputs U and U^*.

  i.e. $$\begin{tikzcd} u \arrow[r, "\text{S-box}"] \arrow[d, leftrightarrow, "a"'] & v \arrow[d, leftrightarrow, "b"] \\ u^* \arrow[r, "\text{S-box}"'] & v^* \end{tikzcd}$$

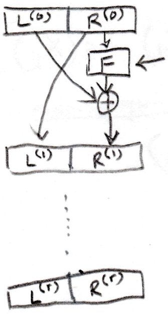
Find a <u>differential trail</u> with high propagation ratio (multiply together the propagation ratio of each item on the trail).

<u>propagation ratio</u> for the differential trail $= \prod_{t=1}^{r} Pr[V^{(t)} \oplus V^{*(t)} = b_t \mid U^{(t)} \oplus U^{*(t)} = a_t]$

$\leq Pr[Y \oplus Y^* = y' \mid X \oplus X^* = x']$

<u>Lemma</u>: if the bias is ε, then we need $\Theta(\varepsilon^{-1})$ tuples (x, x^*, y, y^*) to mount a successful attack.

<u>Feistel cipher</u>:



where $k^{(1)}, \dots, k^{(r)}$ are round keys generated from a key k,

$F : \underset{\substack{\uparrow \\ \text{half of the plaintext}}}{\mathbb{Z}_2^n} \times \underset{\substack{\uparrow \\ \text{round key}}}{\mathbb{Z}_2^m} \to \mathbb{Z}_2^n$ is the round function

$(L^{(0)}, R^{(0)}) \in \mathbb{Z}_2^{2n}$ is the plaintext

$(L^{(r)}, R^{(r)}) \in \mathbb{Z}_2^{2n}$ is the ciphertext

- <u>Decryption</u>: $(L^{(t-1)}, R^{(t-1)}) = (R^{(t)} \oplus F(L^{(t)}, k^{(t)}), L^{(t)})$
  - so F does not need to be injective.

DES/AES: see slides.

Finite fields : + and × : associative, commutative, identity, has inverse.
· distributivity of multiplication over addition

Characteristic of F : least n s.t. $\forall a \in F$, $na=0$

Order of F : number of elements in F ($|F|$)

· $|F| = p^k$ for some prime p, which is the characteristic of F.

Finding the finite field of order $p^s =: q$ (p: prime, $s \in \mathbb{N}$)

- if $s=1$: $F_p = \mathbb{Z}_p$
- if $s>1$: · find a monic reducible polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree s : $f(x) = x^s + a_1 x^{s-1} + \dots + a_s$, $a_1, \dots, a_s \in \mathbb{Z}_p$
  · let β be a new element s.t. $f(\beta)=0$. i.e. $\beta \notin \mathbb{Z}_p$ and $\beta^s = \underline{-(a_1\beta^{s-1} + \dots + a_s)}$
  then $F_q = \mathbb{Z}_p(\beta) = \{ b_1\beta^{s-1} + b_2\beta^{s-2} + \dots + b_s \mid b_1, \dots, b_s \in \mathbb{Z}_p \}$
  with usual addition & multiplication, except that we replace β^s with this
  note: every element in F_q is a zero of $x^q - x$. (hence $f(x)$ is a factor of $x^q - x$.

E.g. finite field of order 2^8 : use polynomial $a^8 = a^4 + a^3 + a + 1$

$$F = \{ b_1 a^7 + \dots + b_8 \mid b_1, \dots, b_8 \in \mathbb{Z}_2 \}$$ (with everything modulo 2)

AES : plaintexts and ciphertexts are 4×4 matrices where each element in F_{2^8}
(see slides)

Public-key cryptosystems: (P, C, K, E, D)

$K \Updownarrow (k_1, k_2)$

k_1: public key, k_2: private key

E, D: only depends on k_1

F: finite field

$F^* := F \setminus \{0\}$

For $\beta \in F^*$, $\langle \beta \rangle := \{ \beta^r \mid r = 1,2,3,\dots \}$

$o(\beta)$ = order of $\beta := |\langle\beta\rangle|$

Thm: $o(\beta) \mid (q-1)$

$\alpha \in F^*$ is a **primitive element** of F : $o(\alpha) = q-1$
(such $\alpha \in F^*$ always exists)

Double-and-add algorithm: $O(\log n)$ fast exponentiation to calculate a^n.

RSA cryptosystem:
· based on difficulty of factoring large numbers
(it is hard to find $\varphi(n)$ without knowing p and q)

(P, C, K, E, D)

P, C: $\mathbb{Z}_n$

K: $\{(n, E, D) \mid DE \equiv 1 \pmod{\varphi(n)}\}$

$e_k(x) \equiv x^E \pmod n$

$e_k(y) \equiv y^D \pmod n$

$n = pq$ — p, q: large primes

φ: Euler's totient function: number of positive integers in $[1, n]$ relatively prime to n.

so given the public key (n, E), it is difficult to obtain the private key D.

Proof: $DE \equiv 1 \pmod{\varphi(n)}$
so $DE \equiv 1 \pmod{p-1}$ and $\pmod{q-1}$ — where $DE = 1 + m(p-1)$
so if $x \equiv 0 \pmod p$ then $x^{DE} \equiv 0 \equiv x \pmod p$
otherwise, by Fermat's little theorem, $x^{p-1} \equiv 1 \pmod p$, so $x^{DE} = x(x^{p-1})^m \equiv x(1)^m \equiv x \pmod p$
... same for mod q, so $x^{DE} \equiv x \pmod n$

Factorization algorithms (i.e. how to find p and q from $n=pq$)

Pollard p-1 algorithm: if B is a positive integer s.t. $p-1 \mid B!$, then let $b \equiv a^{B!} \pmod{n}$ ← small pos. int, e.g. 2.

so $b \equiv a^{B!} \pmod{p}$.

By FLT, $a^{p-1} \equiv 1 \pmod{p}$ (if $\gcd(a,p)=1$)

so (since $p-1 \mid B!$), $b \equiv 1 \pmod{p}$

$\therefore p \mid b-1$

then $d=\gcd(b-1,n)$ is a nontrivial divisor of n.

Algorithm:

① pick B and compute $b_B \equiv a^{B!} \pmod{n}$

② if $b_B = 1$ then repeat step ① with a different a
else compute $d = \gcd(b_B - 1, n)$.

③ if $d=1$, then repeat step ① with $B \leftarrow B+1$
else d is a nontrivial divisor of n.

- To prevent this algorithm from working well, we need to pick p and q s.t. both $p-1$ and $q-1$ have some very large prime factors.

Fermat's factorization: relies on $(x+y)(x-y) = x^2 - y^2$ identity.

if $n = x^2 - y^2$ then $x = \sqrt{n+y^2}$.

we start by picking $x = \lceil \sqrt{n} \rceil$ and checking if $x^2 - n$ is square, increment x and retry until we get a square.

- To prevent this algorithm from working well, we need to pick p and q not too close.

Dixon's random squares algorithm: also relies on $(x+y)(x-y) = x^2 - y^2$.

if we have $x \not\equiv \pm y \pmod{n}$ and $x^2 \equiv y^2 \pmod{n}$ then $\gcd(x-y, n)$ and $\gcd(x+y, n)$ are non-trivial factors of n.

- let $B = \{\underset{\text{primes}}{p_1, \ldots, p_k}\}$ be a factor base
- generate some z_j randomly s.t. $\sqrt{n} < z_j < n$ and all prime factors of z_j^2 are in B
- let $v_j := (a_{j1}, \ldots, a_{jk})$ (i.e. $z_j^2 \equiv p_1^{a_{j1}} p_2^{a_{j2}} \cdots p_k^{a_{jk}} \pmod{n}$)

find some v_j's (a subset of them) s.t. $v_{j_1} + v_{j_2} + \cdots + v_{j_s} \equiv (0, \ldots, 0) \underline{\underline{\pmod{2}}}$

Then let $v_{j_1} + \cdots + v_{j_s} = (2c_1, \ldots, 2c_k)$

$$\text{so } \underbrace{(z_{j_1} \cdots z_{j_s})}_{x}^2 \equiv \underbrace{(p_1^{c_1} \cdots p_k^{c_k})}_{y}^2 \pmod{n}$$

if $x \not\equiv \pm y \pmod{n}$ then $\gcd(x-y, n)$ is a non-trivial factor of n.

$O\left(e^{(1+o(1))\sqrt{\ln(n)\ln(\ln(n))}}\right)$

sub-exponential running time w.r.t. m: slower than $O(m^c)$ for any c, but faster than $O(e^{dm})$ for any d.

Diffie-Hellman key exchange: relies on difficulty of finding x s.t. $c^x \equiv d \pmod{p}$ for known c, d, p.

← i.e. the discrete logarithm problem

- p and θ are published
- Alice picks a secret a and Bob picks a secret b.
- Alice sends $\theta^a \pmod{p}$ and Bob sends $\theta^b \pmod{p}$
- The shared secret is $\theta^{ab} \equiv \theta^{ba}$

El Gamal cryptosystem: (P, C, K, E, D) where $P = \mathbb{Z}_p^*$, $C = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$,

$$K = \{(\underbrace{p, \Theta, A}_{\substack{\text{public}\\ \text{key}}}, \underbrace{a}_{\substack{\text{private}\\ \text{key}}}) \mid A \equiv \Theta^a \pmod p\}$$

$$e_k(\underset{\substack{\uparrow\\ \text{plaintext}\\ (\text{in } \mathbb{Z}_p^*)}}{x}, \underset{\substack{\uparrow\\ \text{random}\\ \text{number}\\ (\text{in } \mathbb{Z}_p^*)}}{b}) = (\underset{\substack{\uparrow\\ \in \mathbb{Z}_p^*}}{\Theta^b \pmod p}, \underset{\substack{\uparrow\\ \in \mathbb{Z}_p^*}}{xA^b \pmod p})$$

$$d_k(B, y) = y(B^a)^{-1}$$ ← find inverse using Euclidean Algorithm or FLT.

works because
$$\begin{aligned} d_k(e_k(x, b)) &= d_k(\Theta^b \pmod p, xA^b \pmod p) \\ &\equiv xA^b(\Theta^{ba})^{-1} \\ &\equiv x\Theta^{ab}(\Theta^{ba})^{-1} \\ &\equiv x \pmod p \end{aligned}$$

Algoritms for solving the discrete logarithm problem:

(i.e. find x s.t. $c^x \equiv d \pmod p$)

Shanks' algorithm

(baby-step giant-step algorithm)

Let $n \geq \sqrt{p}$ (usually $n = \lceil \sqrt{p} \rceil$)

- create two lists: (mod p)
  $$1,\ c,\ c^2,\ \ldots,\ c^{n-1}$$
  $$d,\ dc^{-n},\ dc^{-2n},\ \ldots,\ dc^{-(n-1)n}$$
- Find a match between the two lists, e.g. $c^s \equiv dc^{-tn} \pmod p$.

  Then with $x = s + tn$, $c^x = c^s c^{tn} \equiv dc^{-tn}c^{tn} \equiv d \pmod p$

Running time: $O(n \ln n) = O(e^{\frac{1}{2}(\ln(p) + \ln(\ln(p)))})$

Pollard-rho algorithm:

Let $n = \underset{\substack{\uparrow\\ \text{in } \mathbb{Z}_p^*}}{o(c)}$ (i.e. smallest n s.t. $c^n \equiv 1 \pmod p$)

- create a sequence $z_1 = c^{a_1}d^{b_1}$, $z_2 = c^{a_2}d^{b_2}$, $z_3 = c^{a_3}d^{b_3}$, …

  s.t. $z_{i+1} = f(z_i)$ for some $f: \langle c\rangle \to \langle c\rangle$

e.g. $f(z) = \begin{cases} dz & \text{if } z \in S_0 \\ z^2 & \text{if } z \in S_1 \\ cz & \text{if } z \in S_2 \end{cases}$ →

where $S_0 = \{z \in \mathbb{Z}_p^* \mid 0 < z < \frac{p}{3}\}$
$S_1 = \{z \in \mathbb{Z}_p^* \mid \frac{p}{3} < z < \frac{2p}{3}\}$
$S_2 = \{z \in \mathbb{Z}_p^* \mid \frac{2p}{3} < z < p\}$
and initially, $z_1 = d$.

- for each z_{2i}, check whether $z_i = z_{2i}$.

  if $z_i = z_{2i}$ then $c^{a_i}d^{b_i} \equiv c^{a_{2i}}d^{b_{2i}} \pmod p$

  then if $b_{2i} - b_i$ is invertible modulo n, then
  with $x \equiv (a_i - a_{2i})(b_{2i} - b_i)^{-1} \pmod n$, $c^x \equiv d \pmod p$

Running time: $O(\sqrt{p}) = O(e^{\frac{1}{2}\ln(p)})$

Index calculus method.

- Take a factor base $B = \{p_1, \ldots, p_k\}$
- generate a list of integers z_j with all prime factors in B,
  say $c^{z_j} \equiv p_1^{a_{j1}} \cdots p_k^{a_{jk}} \pmod p$ for $j = 1, \ldots, m$.
- Then solve linear system (cannot do division)
  $$\begin{cases} z_1 \equiv a_{11}\log_c(p_1) + \cdots + a_{1k}\log_c(p_k) \pmod n \\ \vdots \\ z_m \equiv a_{m1}\log_c(p_1) + \cdots + a_{mk}\log_c(p_k) \pmod n \end{cases}$$
  to find $\log_c(p_1), \ldots, \log_c(p_k)$ ← discrete logarithms.
  $\log_c(d) = x$ if $c^x \equiv d \pmod p$
- Then, find y s.t. $dc^y \pmod p$ is in B.
  say $dc^y \equiv p_1^{f_1} \cdots p_k^{f_k} \pmod p$.
- Then $\log_c(d) + y \equiv f_1 \cdot \log_c(p_1) + \cdots + f_k \cdot \log_c(p_k)$.

Running time: $O(e^{(\frac{3}{\sqrt{2}} + o(1))\sqrt{\ln(n)\ln(\ln(n))}})$ →

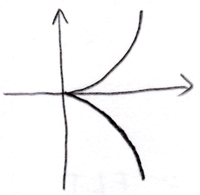
Elliptic curves over $\mathbb{R}$:

standard form : $y^2 = x^3 + Ax + B$
(after affine transformation)

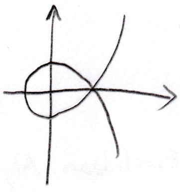
discriminant : $\Delta = -16(4A^3 + 27B^2)$

curve is "smooth" (i.e. no singular point) if $\Delta \neq 0$

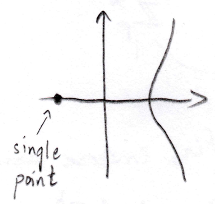
Not smooth



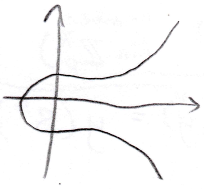
Not smooth



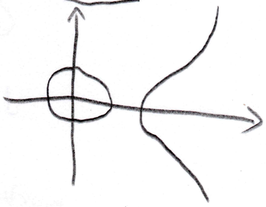
Not smooth



Smooth



Smooth



$$\mathcal{U} = \{ (x,y) \in \mathbb{R} \mid y^2 = x^3 + Ax + B \} \cup \{ \underset{\text{point at infinity}}{\mathcal{O}} \}$$

- The number of times any line in $\mathbb{R}^2$ intersects $\mathcal{U}$ can only be 1 or 3.
- $-P := \begin{cases} \mathcal{O} & \text{if } P = \mathcal{O} \\ (x,-y) & \text{if } P = (x,y) \end{cases}$
- Addition : $\mathcal{O} + \mathcal{O} := \mathcal{O}$

$P + Q = -R$ where P, Q, R are on a straight line : (a)(i) $P = \mathcal{O}$: $P + Q = Q$
(ii) $Q = \mathcal{O}$: $P + Q = P$
(iii) $P = -Q$: $P + Q = \mathcal{O}$
(b) $\underset{(x_1,y_1)}{\overset{\shortparallel}{P}} \neq \mathcal{O}$ and $\underset{(x_2,y_2)}{\overset{\shortparallel}{Q}} \neq \mathcal{O}$ and $P \neq \pm Q$:
$P + Q = (x_3, -(\lambda x_3 + c))$
where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$
and $c = y_1 - \lambda x_1$
and $x_3 = \lambda^2 - x_1 - x_2$
(c) $P \neq \mathcal{O}$ and $Q \neq \mathcal{O}$ and $P \neq -Q$ and $P = \underset{(x_0,y_0)}{\overset{\shortparallel}{Q}}$:
$P + Q = (x', -(\lambda x' + c))$
where $\lambda = \frac{3x_0^2 + A}{2y_0}$
and $c = y_0 - \lambda x_0$
and $x' = \lambda^2 - 2x_0$

Elliptic curves over $\mathbb{Z}_p$:

$\mathcal{U} = \{ (x,y) \in \mathbb{Z}_p^2 \mid y^2 = x^3 + Ax + B \} \cup \{ \mathcal{O} \}$

(p: odd prime)

Addition: (a)(i) $P = \mathcal{O}$: $P + Q = Q$
(ii) $Q = \mathcal{O}$: $P + Q = P$
(iii) $P = (x,y)$, $Q = (x, \underset{\text{mod } p}{p-y})$: $P + Q = \mathcal{O}$
(b) $P = (x_1, y_1)$, $Q = (x_2, y_2)$, $x_1 \not\equiv x_2 \pmod p$:
$P + Q = (x_3, -(\lambda x_3 + c) \pmod p)$
where $\lambda \equiv (y_2 - y_1)(x_2 - x_1)^{-1} \pmod p$
and $c \equiv y_1 - \lambda x_1 \pmod p$
and $x_3 \equiv \lambda^2 - x_1 - x_2 \pmod p$
(c) $P = Q = (x_0, y_0)$, $y_0 \not\equiv 0 \pmod p$:
$P + Q = (x', -(\lambda x' + c) \pmod p)$
where $\lambda \equiv (3x_0^2 + A)(2y_0)^{-1} \pmod p$
and $c \equiv y_0 - \lambda x_0 \pmod p$
and $x' \equiv \lambda^2 - 2x_0 \pmod p$

Scalar Multiplication : if $P \in \mathcal{U}$ and $m \in \mathbb{Z}$:

$$mP := \begin{cases} \underbrace{P + \dots + P}_{m \text{ times}} & \text{if } m > 0 \\ \mathcal{O} & \text{if } m = 0 \\ \underbrace{(-P) + \dots + (-P)}_{-m \text{ times}} & \text{if } m < 0 \end{cases}$$

Elliptic curves over fields with characteristic 2 :

← different standard form needed because 2 is even — the curve for $\mathbb{Z}_p$ would always be singular

$\mathcal{U} = \{ (x,y) \in \underset{\mathbb{F} = \mathbb{F}_{2^r}}{\mathbb{F}^2} \mid y^2 + xy = x^3 + Ax^2 + B \} \cup \{ \mathcal{O} \}$

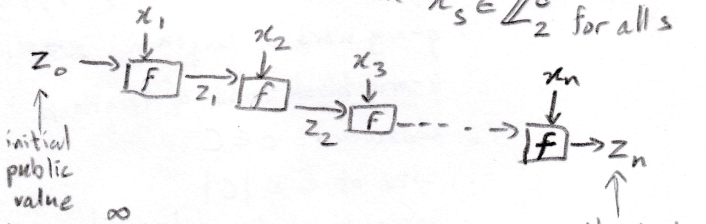
$-P := \begin{cases} \mathcal{O} & \text{if } P = \mathcal{O} \\ (x, x+y) & \text{if } P = (x,y) \end{cases}$

- Note : $-(-P) = P$ (since $x + x = 0$)

Addition: (a)(i) $P = \mathcal{O}$: $P + Q = Q$
(ii) $Q = \mathcal{O}$: $P + Q = P$
(iii) $P = (x,y)$, $Q = (x, x+y)$: $P + Q = \mathcal{O}$
(b) $P = (x_1, y_1)$, $Q = (x_2, y_2)$, $x_1 \neq x_2$:
$P + Q = (x_3, x_3 + \lambda x_3 + c)$ where $\lambda = (y_1 + y_2)(x_1 + x_2)^{-1}$
and $c = y_1 + \lambda x_1$
and $x_3 = \lambda^2 + \lambda + x_1 + x_2 + A$
(c) $P = Q = (x_0, y_0)$, $x_0 \neq 0$:
$P + Q = (x', x' + \lambda x' + x_0^2)$ where $\lambda = x_0 + y_0 x_0^{-1}$
and $x' = \lambda^2 + \lambda + A$

Hash functions : $h : \bigcup_{r=1}^{\infty} \mathbb{Z}_2^r \to \mathbb{Z}_2^m$

How to hash: $h(x)$ · add bits to the end of x to produce $x' = x_1 x_2 \cdots x_n$ with $x_s \in \mathbb{Z}_2^d$ for all s



Compression function : $f : \mathbb{Z}_2^m \times \mathbb{Z}_2^d \to \mathbb{Z}_2^m$

Security considerations ① Pre-image problem: Given $y \in \mathbb{Z}_2^m$, Find $x \in \bigcup_{r=1}^{\infty} \mathbb{Z}_2^r$ s.t. $h(x) = y$

② Second pre-image problem: Given $x \in \bigcup_{r=1}^{\infty} \mathbb{Z}_2^r$, Find x' s.t. $x \neq x'$ and $h(x) = h(x')$.

③ Collision problem: Find $x, x' \in \bigcup_{r=1}^{\infty} \mathbb{Z}_2^r$ s.t. $x \neq x'$ and $h(x) = h(x')$

Digital signatures to address these problems: ① Authentication : the receiver of the message needs to confirm sender's identity

② Integrity : the receiver has to be sure that the message was not altered during the transmission

③ Non-repudiation : the sender cannot later deny sending the message.

Signature scheme : (P, C, K, S, V)

set of possible messages

set of possible signatures

keyspace (set of possible keys)

signing algorithms

verification algorithms

$(k_1, k_2) = k \in K$

public key

private key (known to signee only)

$S \ni \mathrm{sig}_k : P \to C$

$V \ni \mathrm{ver}_k : P \times C \to \{\text{true}, \text{false}\}$ (depends only on public key)

s.t. $y = \mathrm{sig}_k(x) \Rightarrow \mathrm{ver}_k(x,y) = \text{true}$

(it is ideal that the converse is true, but usually not the case since $|C| < |P|$)

RSA signature scheme : (P, C, K, S, V)

P, C: $\mathbb{Z}_n$ $\quad K$: $\{(\underbrace{n, E}_{\text{public}}, \underbrace{D}_{\text{private}}) \mid DE \equiv 1 \pmod{\varphi(n)}\}$

$n = pq$ where p, q are large primes

$\mathrm{sig}_k(x) \equiv x^D \pmod{n}$

$\mathrm{ver}_k(x,y) = \begin{cases} \text{true} & \text{if } x \equiv y^E \pmod{n} \\ \text{False} & \text{otherwise} \end{cases}$

ElGamal signature scheme : (P, C, K, S, V)

P: $\mathbb{Z}_p^*$ $\quad C$: $\mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$ $\quad K$: $\{(\underbrace{p, \theta, A}_{\text{public}}, \underbrace{a}_{\text{private}}) \mid A \equiv \theta^a \pmod{p}\}$

b randomly chosen s.t. $b \in \mathbb{Z}_{p-1}^*$ (i.e. $\gcd(b, p-1) = 1$)

$\mathrm{sig}_k(x, b) = (\theta^b \pmod{p}, (x - aB)b^{-1} \pmod{(p-1)}))$

$\mathrm{ver}_k(x, (B, c)) = \begin{cases} \text{true} & \text{if } A^B B^c \equiv \theta^x \pmod{p} \\ \text{False} & \text{otherwise} \end{cases}$

Proof : $A^B B^c \equiv (\theta^a)^B (\theta^b)^{(x-aB)b^{-1}}$

$\equiv \theta^{aB + b(x-aB)b^{-1}}$

$\equiv \theta^x \pmod{p}$

(since $aB + b(x - aB)b^{-1} \equiv x \pmod{(p-1)}$ and $\theta^{p-1} \equiv 1 \pmod{p}$ (Fermat's little thm))

Hash-and-sign : to send message x, we send $(x, \mathrm{sig}_k(h(x)))$

to verify, check $\mathrm{ver}_k(h(x), y)$ where (x, y) is received

Digital signature algorithm (DSA)
Elliptic curve digital signature algorithm (ECDSA) } ⇒ see slides

Coding Theory

Block code :
- code alphabet : $A = \{a_1, \dots, a_q\}$
- q-ary word of length n : $x = x_1 \cdots x_n$ where $x_i \in A$ (i.e. $x \in A^n$)
- q-ary block code of length n : $C \subseteq A^n$
- codeword : $c \in C$
- size of C : $|C|$
- information rate : $\frac{\log_q |C|}{n}$ (amount of bits of information in one bit of the channel)
- (n, M)-code : code with length n and $|C| = m$
- binary code : $A = \mathbb{Z}_2$

Discrete communication channel : $A = \{a_1, \dots, a_q\}$, $A' = \{b_1, \dots, b_{q'}\}$,
- characterised by $p_{ij} = Pr[b_j \text{ received} \mid a_i \text{ sent}]$
- memoryless channel : $Pr[x_1 \cdots x_n \text{ received} \mid c_1 \cdots c_n \text{ sent}] = \prod_{i=1}^{n} Pr[x_i \text{ received} \mid c_i \text{ sent}]$
- q-ary symmetric channel with prob. of error p : (1) $p_{ii} = 1-p \quad \forall i$
  (2) $p_{ij} = \frac{p}{q-1} \quad \forall i, j,\ i \neq j$
- binary symmetric channel : (1) $p_{ii} = 1-p$ (same)
  (2) $p_{ij} = p$ (different)

Maximum likelihood decoding : decode x to $\underset{c \in C}{\operatorname{argmax}} Pr[x \text{ received} \mid c \text{ sent}]$
- complete (CMLD) : if tie, choose arbitrarily
- incomplete (IMLD) : if tie, request retransmission

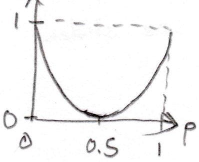
Shannon's channel coding theorem : (for memoryless channels)

capacity : $c = \max_{\Omega(X)} \sum_{\substack{x \in A \\ y \in A'}} Pr[X=x, Y=y] \log_q \left(\frac{Pr[X=x, Y=y]}{Pr[X=x]\, Pr[Y=y]} \right)$

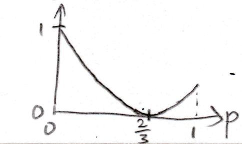
(max possible amount of info in one bit of channel)

all probability distributions of X

- thm : $\forall r < c$ and $\forall \varepsilon > 0$, $\exists n_0 \in \mathbb{N}$ s.t. $\forall n \geq n_0$, there exists a code $C \subseteq A^n$
  s.t. $r \leq \frac{\log_q |C|}{n} < c$ and probability of decoding wrongly is less than ε.
  (i.e. C is an (n, M)-code over A with $q^{rn} \leq M \leq q^{cn}$)
  (i.e. we can find codes that approach optimal information rate)
- thm converse : if $\{C_t\}_{t=1,2,\dots}$ is a sequence of codes
  where C_t is an (n_t, M_t)-code over A, where $\{n_t\}_{t=1,2,\dots}$ is strictly increasing,
  then if $\lim_{t \to \infty} \varepsilon_t = 0$ then $\limsup_{t \to \infty} \frac{\log_q M_t}{n_t} \leq c$
- For q-ary symmetric channel with err. prob. p, capacity $= c = 1 + \underbrace{(1-p)\log_q(1-p) + p\log_q\left(\frac{p}{q-1}\right)}_{\text{these are negative}}$
  - for binary symmetric channel : capacity $= c = 1 + (1-p)\log_2(1-p) + p\log_2 p$



- 3-ary symmetric channel :



Minimum distance decoding (= maximum likelihood decoding for q-ary symmetric channels where $p < \frac{q-1}{q}$)

- decode to $\underset{c\in C}{\operatorname{argmin}}\, d(x,c)$

$d(x,y)$: Hamming distance (is a metric)

- $d(x,y) \geq 0$
- $d(x,y)=0 \iff x=y$
- $d(x,y) = d(y,x)$
- $d(x,y) \leq d(x,z) + d(z,y)$ (Δ ineq.)
- if A is an additive group, then $w(x-y) = d(x,y)$ (and hence $w(x) = d(x,0)$)

$d(C) = \min\limits_{\substack{x,y\in C\\ x\neq y}} d(x,y)$

(n,M,d)-code : length n, size M, distance d

u-error-detecting : $d(C) \geq u+1$

v-error-correcting : $d(C) \geq 2v+1$

Thm : if $d(C)=d$, then C is exactly $(d-1)$-error-detecting and exactly $\lfloor \frac{d-1}{2} \rfloor$-error-correcting.

Linear code : A is a field $\mathbf{F}$ and C is a subspace of F^n.

C is nonempty subset of F^n and $\forall x,y\in C$, $\forall a,b\in F$, $ax+by\in C$

- $[n,k]$-linear code : length n, $\dim(C)=k$
  - is a (n,q^k)-code
- $[n,k,d]$-linear code : length n, $\dim(C)=k$, distance d
  - is a (n,q^k,d)-code
- $w(C) = \min\limits_{\substack{x\in C\\ x\neq 0}} w(x)$
- if $C \neq \{0\}$ then $d(C) = w(C)$

Dual code of a linear code :

- $C^\perp = \{x\in F^n \mid x\cdot c = 0 \;\; \forall c\in C\} \subseteq F^n$ is a linear code over F

  dot product (not a true inner product because not necessarily $\langle u.u\rangle > 0 \;\; \forall u\neq 0$)
- dimension theorem : $\dim(C) + \dim(C^\perp) = n$

Generator matrix of a linear code : a matrix G ($k\times n$ matrix) such that $C = \{aG \mid a\in F^k\}$

- G is not unique (a way to make G is $G = \begin{pmatrix} g_1 \\ \vdots \\ g_k \end{pmatrix}$ where $\{g_1,\dots,g_k\}$ is a basis for C)
- standard form generator matrix : $G = (\underbrace{I_k}_{\substack{k\times k\\ \text{identity}\\ \text{matrix}}} | X)$ — some $k\times(n-k)$ matrix over F
  - not all lin. codes have a generator matrix in std. form.
  - in standard form, $a\in F^k$ is encoded to the codeword $c = aG = a(I_k|X) = (aI_k, aX)$
    $= (a, aX)$

Parity-check matrix of a linear code:

- parity-check matrix for C = generator matrix for $C^\perp =: H$ ↙ $(n-k) \times n$ matrix
- if $G = (I_k \mid X)$ is a std. form generator matrix for C, then $H = (-X^T \mid I_{n-k})$ is a parity-check matrix for C
- $C = \{c \in F^n \mid cH^T = 0\}$ (i.e. C is the nullspace of H)
- $d(C) \geq d \iff$ any d-1 columns of H are lin. indep.
- $d(C) < d \iff \exists$ d-1 colums of H that are lin. dep.
- $d(C) = d \iff \begin{cases} \text{any } d-1 \text{ colums of } H \text{ are lin. indep.} \\ H \text{ has } d \text{ colums that are lin. dep.} \end{cases}$

Syndrome decoding:

- Given $x \in F^n$, the syndrome of x is $S_H(x)^T := Hx^T$ (so $s_H(x) = xH^T$)
- If $c \in C$ is sent and $x = c + \underset{\substack{\uparrow \\ \text{transmission error}}}{e}$ is received, then $S_H(x) = s_H(e)$
- Coset containing x:

  $$x + C := \{x + y \mid y \in C\}$$
- $x, y \in F^n$ are in the same coset of C $\iff S_H(x) = S_H(y)$
- coset leader: the vector in the coset with least weight (something like the delta to the closest codeword)
- syndrome decoding:
  - (init): choose a coset leader and compute its syndrome for every coset, and list down the syndromes and corresponding coset leaders in a table
  - (query): when receiving y, compute $S_H(y)$, and lookup the table to get the coset leader e (s.t. $S_H(e) = S_H(y)$), and decode y to y-e.
- syndrome decoding is a minimal distance decoding (pf. on slides)

Information rate: $R(C) = \frac{\log_q M}{n}$ (amount of information per digit (bit) sent)
(aka. efficiency)

Relative minimum distance: $\delta(C) = \frac{d-1}{n}$
(aka. error-correcting capacity)

$A_q(n,d)$: largest possible M such that a q-ary (n,M,d)-code exists

$B_q(n,d)$: largest possible q^k such that a $[n,k,d]$-linear code over F_q exists

Basic results

- $B_q(n,d) \leq A_q(n,d) \leq q^n$ for $1 \leq d \leq n$
- $B_q(n,1) = A_q(n,1) = q^n$
- $B_q(n,2) = A_q(n,2) = q^{n-1}$ ← parity check digit
- $B_q(n,n) = A_q(n,n) = q$

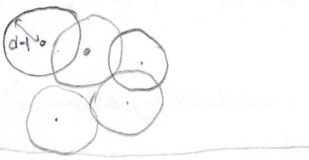
(B: if q is a prime power only)

Sphere of radius r and centre x: $S_A(x,r) = \{y \in A^n \mid d(x,y) \leq r\}$

- $V_q^n(r) = |S_A(x,r)| = \begin{cases} \sum_{s=0}^{r} \binom{n}{s} (q-1)^s & \text{if } r < n \\ q^n & \text{if } r \geq n \end{cases}$

Lower bounds:

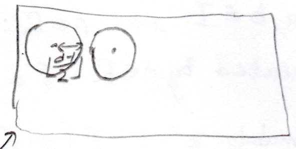
- Sphere-covering bound: $A_q(n,d) \geq \dfrac{q^n}{V_q^n(d-1)}$



- Gilbert-Varshamov theorem: If $2 \leq d \leq n$ and $1 \leq k \leq n$ and $V_q^{n-1}(d-2) < q^{n-k}$ then there exists an $[n,k]$-linear code over F_q with min. dist. at least d
- (equiv.) Gilbert-Varshamov bound: $A_q(n,d) \geq B_q(n,d) \geq q^{n - \lceil \log_q(V_q^{n-1}(d-2)+1) \rceil}$

Upper bounds:

- Hamming bound (Sphere-packing bound): $A_q(n,d) \leq \dfrac{q^n}{V_q^n(\lfloor \frac{d-1}{2} \rfloor)}$
  - usually only useful when d is small
  - Perfect code: where equality holds (i.e. no uncovered spaces)



- Hamming code: $Ham(r,q)$: code formed when parity check matrix H has r rows and $\dfrac{q^r-1}{q-1}$ colums
  - this is maximum possible, because $\forall x \in F_q^r \setminus \{0\}$, there are $q-1$ nonzero scalar multiples of x, and H cannot have two lin. indep columns
  - Hence $Ham(r,q)$ is a $\left[\dfrac{q^r-1}{q-1}, \dfrac{q^r-1}{q-1} - r, 3\right]$-linear code over F_q.
  - All Hamming codes are perfect.
- Thm: Let q be a prime power such that there exists an (n,M,d)-perfect code with $1 < d < n$. Then either:
  - it is a Hamming code (i.e. $n = \dfrac{q^r-1}{q-1}$, $M = q^{n-r}$, $d=3$), or
  - $\underbrace{(q,n,M,d) = (2,23,2^{12},7) \text{ or } (3,11,3^6,5)}_{\text{Golay codes}}$
- Singleton bound: Given $1 \leq d \leq n$,

$$B_q(n,d) \leq A_q(n,d) \leq q^{n-d+1}$$

  - For linear codes, if q is a prime power and $B_q(n,d) = q^{n-d+1}$ (i.e. the bound is attained), then they are maximum distance separable (MDS) codes
  - Equiv. MDS codes: (1) C is an MDS code
    - (2) any $n-k$ columns of H are lin. indep.
    - (3) any k columns of G are lin. indep.
    - (4) $C^\perp$ is an MDS code
  - Binary MDS codes: either one of the following must be true:
    - (1) $[n,k,d] = [n,n,1]$ (i.e. $C = \mathbb{Z}_2^n$)
    - (2) $[n,k,d] = [n,1,n]$ (i.e. $C = \{0 \dots 0, 1 \dots 1\}$)
    - (3) $[n,k,d] = [n,n-1,2]$ (i.e. C is the dual code of $\{0 \dots 0, 1 \dots 1\}$)
- Plotkin bound: If $rn < d$ where $r = \dfrac{q-1}{q}$, then $A_q(n,d) \leq \left\lfloor \dfrac{d}{d-rn} \right\rfloor$
  - Improved version for binary codes ($q=2$): When d is even: $A_2(n,d) \leq \begin{cases} 2\left\lfloor \frac{d}{2d-n} \right\rfloor & \text{if } n < 2d \\ 4d & \text{if } n = 2d \end{cases}$

    When d is odd: $A_2(n,d) \leq \begin{cases} 2\left\lfloor \frac{d+1}{2d+1-n} \right\rfloor & \text{if } n < 2d+1 \\ 4d+4 & \text{if } n = 2d+1 \end{cases}$
  - useful when d is large relative to n

Asymptotic bounds (when $n \to \infty$): $\alpha_q(\delta) := \limsup_{n \to \infty} \dfrac{\log_q A_q(n, \lfloor \delta n \rfloor)}{n}$ (i.e. amount of codewords relative to n)

- Asymptotic singleton bound: $\alpha_q(\delta) \leq 1 - \delta$
- $\lim_{n \to \infty} \frac{1}{n} \log_q(V_q^n(\lfloor \delta n \rfloor)) = H_q(\delta) := -(1-\delta)\log_q(1-\delta) - \delta \log_q\left(\frac{\delta}{q-1}\right)$ (i.e. some kind of entropy)
  - asymptotic Gilbert-Varshamov bound: $\alpha_q(\delta) \geq 1 - H_q(\delta)$
  - asymptotic Hamming bound: $\alpha_q(\delta) \leq 1 - H_q(\frac{\delta}{2})$ if $\delta \leq \frac{q-1}{q}$

Cyclic Codes : a linear code where $\forall c = c_0 \cdots c_{n-1} \in C$, $\underset{\substack{\uparrow \\ \text{cyclic shift}}}{\sigma(c)} := c_{n-1}c_0 \cdots c_{n-2} \in C$

Rings $(R, +, \cdot)$: "+" satisfies commutativity, associativity, identity, invertibility
"·" is associative
"·" is distributive over "+".

- If R has multiplicative identity, then R is a ring with unity
- If R has commutative multiplication, then R is a commutative ring

Ideal $I \subseteq R$: $\forall a, b \in I$ and $r \in R$: $a-b, ra, ar \in I$

- Principal ideal generated by $a \in R$ of a commutative ring with identity : $aR := \{ar \mid r \in R\} = \{ra \mid r \in R\}$

Ring of polynomials modulo $x^n - 1$: $\underset{\substack{\uparrow \\ \text{Field}}}{F}[x;n] := \{f(x) \in F[x] \mid \deg(f(x)) < n\}$ is a commutative ring with unity

- addition is usual polynomial addition
- multiplication is polynomial multiplication modulo $x^n - 1$
- $F[x;n]$ is a vector space over F.
- $\tilde{a} \in F^n \longleftrightarrow a(x) = a_0 + a_1 x + \dots + a_{n-1}x^{n-1} \in F[x;n]$
- $\tilde{b} = \sigma(\tilde{a}) \longleftrightarrow b(x) \equiv x a(x) \pmod{x^n - 1}$
- $C \subseteq F^n$ is a cyclic code $\Longleftrightarrow C' := \{c(x) \mid \tilde{c} \in C\}$ is an ideal of $F[x;n]$
- weight: $w(f(x)) :=$ number of nonzero coefficients in $f(x)$
- distance : $d(f(x), g(x)) := w(f(x) - g(x))$

Generator polynomials: $g(x) \in F[x]$ such that:

(i) $g(x)$ divides $x^n - 1$
(ii) $\deg(g(x)) = n-k$
(iii) $C' = \{f(x)g(x) \mid f(x) \in F[x] \text{ and } \deg(f(x)) \le k-1\}$ (note: C' is defined here)
(iv) $g(x)$ is monic

- C is an $[n,k]$-cyclic code $\Longleftrightarrow$ such $g(x)$ exists
- **Encoding**: a message $u \in F^k$ (polynomial $u(x)$) is encoded to :
  (1) Non-systematic encoding : $c(x) = u(x)g(x)$
  (2) Systematic encoding : Find the remainder when $x^{n-k}u(x)$ is divided by $g(x)$, let the remainder be $r(x) = r_0 + r_1 x + \dots + r_{n-k-1}x^{n-k-1}$, then $c(x) = x^{n-k}u(x) - r(x) = q(x)g(x)$
  (where $x^{n-k}u(x) = \underset{\substack{\uparrow \\ \text{quotient}}}{q(x)}g(x) + \underset{\substack{\uparrow \\ \text{remainder}}}{r(x)}$)
  (so the codeword is $(-r_0, \dots, -r_{n-k-1}, \underbrace{u_0, \dots, u_{k-1}}_{\text{message is part of the codeword!}})$)
- **Decoding** :
  - Syndrome polynomial: given $a \in F^n$, (i.e. $a(x) \in F[x;n]$) $s_a(x)$ is the remainder when $a(x)$ is divided by $g(x)$ (i.e. $s_a(x) \equiv a(x) \pmod{g(x)}$ and $\deg(s_a(x)) < \underset{\substack{\shortparallel \\ n-k}}{\deg(g(x))}$)
  - Thm: given $a, b \in F^n$, a and b are in the same coset of C $\Longleftrightarrow s_a(x) = s_b(x)$
  - $s_{\sigma(a)}(x) \equiv x s_a(x) \pmod{g(x)}$
  - Meggitt decoding algorithm: see slides : Ch9 p. 42
  - Runs of zeroes : $a = a_0 \cdots a_{n-1} \in F^n$ has a run of 0 of length s : $\exists i \ (0 \le i < n)$ s.t. $a_i = a_{i+1} = \dots = a_{i+s-1} = 0$
    mod n, so it wraps around
  - Error trapping : see slides: Ch9. p.48
  - Error trapping decoding : see slides: Ch9 p.51

BCH codes

Characteristic of a ring: $\mathrm{char}(R)$ = smallest $p \in \mathbb{N}$ s.t. $\forall a \in R$, $pa := \underbrace{a + \cdots + a}_{p \text{ times}} = 0$

Lemma 1: In any commutative ring R where $\mathrm{char}(R) =: p$ is prime,

$$\forall a, b \in R,\ \forall s \in \mathbb{N},\ (a+b)^{p^s} = a^{p^s} + b^{p^s}$$

Lemma 2: If K and F are finite fields s.t. F is a subfield of K and order of F is q, then:

(1) $|K| = q^t$ for some $t \in \mathbb{N}$

(2) $\forall \beta \in K$, $\beta^{|K|} = \beta$

(3) $\beta \in F \iff \beta^q = \beta$

Minimal polynomial of $\beta \in K$ over F: Monic polynomial $f(x) \in F[x]$ s.t.

(1) $f(x)$ is irreducible over F

(2) β is a root of $f(x)$ (computed in K), i.e. $f(\beta) = 0$

- see examples: Ch 10 p. 4
- if m is the smallest positive integer s.t. $\beta^{q^m} = \beta$, then $\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}$ are distinct elements in K
- Thm: $M_\beta(x) := \prod_{j=0}^{m-1} (x - \beta^{q^j})$ is the minimal polynomial of β over F

Primitive element $\alpha \in K$: order of α in K is $|K|-1$ (i.e. every $x \in K \setminus \{0\}$ can be written as $x = \alpha^i$ for some integer i)

- so $\alpha, \alpha^2, \ldots, \alpha^{q^t - 1}$ are distinct (where $|K| = q^t$)
- so $\deg(M_\alpha(x)) = t$
- $M_\alpha(x)$ is called a primitive polynomial
- Examples: Ch 10 p. 10.

BCH code: Given $\beta \in K^*$ and order of β is n (n is a divisor of $q^t - 1$),

For any θ, δ, the q-ary cyclic code with generator polynomial

$$g(x) = \mathrm{lcm}\{M_{\beta^\theta}(x), M_{\beta^{\theta+1}}(x), \ldots, M_{\beta^{\theta+\delta-2}}(x)\}$$

is a BCH code over F of length n with designed distance δ.

- If β is a primitive element of K (so $n = q^t - 1$), then it is a primitive BCH code.
- If $\theta = 1$, then it is a narrow-sense BCH code.
- See examples: Ch 10 p. 18.
- min distance thm: $d(C) \geq \delta$

Binary Hamming code: ($F = \mathbb{Z}_2$ and K is a finite extension of F and $|K| = 2^t$ for some $t \in \mathbb{N}$)

Given a primitive element α of K, with $\beta = \alpha$ and $\theta = 1$ and $\delta = 3$, we obtain a narrow-sense primitive BCH code C of length $n = 2^t - 1$ using the generator polynomial $g(x) = \mathrm{lcm}\{M_\alpha(x), M_{\alpha^2}(x)\} = M_\alpha(x)$.

Then $\deg(g(x)) = \deg(M_\alpha(x)) = t$, and C is a $[2^t - 1, 2^t - 1 - t]$-cyclic code, and C is a binary Hamming code $\mathrm{Ham}(t, 2)$.

Reed-Solomon code:

- If $F = K$ and $|F| = q$, then $\forall \beta \in K^* = F^*$, $M_\beta(x) = x - \beta$
- For any primitive element $\alpha \in K$ and any δ, we obtain a primitive BCH code C with length $n = q-1$ with generator polynomial $g(x) = \mathrm{lcm}\{M_{\alpha^\theta}(x), M_{\alpha^{\theta+1}}, \ldots, M_{\alpha^{\theta+\delta-2}}(x)\}$

$$= (x - \alpha^\theta)(x - \alpha^{\theta+1}) \cdots (x - \alpha^{\theta+\delta-2})$$

- This code C is a Reed-Solomon code, and is a $[q-1, q-\delta]$-cyclic code, and $d(C) = \delta$, and are MDS codes.