# 1. Introduction

- Encryption process:
  - Alice has a message (**plaintext**) which is **encrypted** using an **encryption key** to produce the **ciphertext**, which is sent to Bob.
  - Bob uses a **decryption key** (which depends on the encryption key) to **decrypt** the ciphertext and recover the original plaintext.
  - It should be computationally infeasible to determine the plaintext without knowing the decryption key.
- **Caesar cipher**:
  - Add constant $k$ to each letter in plaintext to produce ciphertext:

    $$\text{ciphertext letter} = \text{plaintext letter} + k \mod 26$$
  - To decrypt,

    $$\text{plaintext letter} = \text{ciphertext letter} - k \mod 26$$
  - The key is $k \bmod 26$.
- Cryptosystem objectives:
  - **Secrecy**: an intercepted message is not able to be decrypted
  - **Integrity**: it is impossible to alter a message without the receiver knowing
  - **Authenticity**: receiver is certain of identity of sender
  - **Non-repudiation**: sender cannot claim they sent a message; the receiver can prove they did.
- **Kerckhoff's principle**: a cryptographic system should be secure even if the details of the system are known to an attacker.
- Types of attack:
  - **Ciphertext-only**: the plaintext is deduced from the ciphertext.
  - **Known-plaintext**: intercepted ciphertext and associated stolen plaintext are used to determine the key.
  - **Chosen-plaintext**: an attacker tricks a sender into encrypting various chosen plaintexts and observes the ciphertext, then uses this information to determine the key.
  - **Chosen-ciphertext**: an attacker tricks the receiver into decrypting various chosen ciphertexts and observes the resulting plaintext, then uses this information to determine the key.

# 2. Symmetric key ciphers

- **Converting letters to numbers**: treat letters as integers modulo 26, with $A = 1$, $Z = 0 \equiv 26 \pmod{26}$. Treat string of text as vector of integers modulo 26.
- **Symmetric key cipher**: one in which encryption and decryption keys are equal.
- **Key size**: $\log_2$(number of possible keys).
- Caesar cipher is a **substitution cipher**. A stronger substitution cipher is this: key is permutation of $\{a, ..., z\}$. But vulnerable to plaintext attacks and ciphertext-only attacks, since different letters (and letter pairs) occur with different frequencies in English.

- **One-time pad**: key is uniformly, independently random sequence of integers $\bmod 26$, $(k_1, k_2, ...)$, known to sender and receiver. If message is $(m_1, m_2, ..., m_r)$ then ciphertext is $(c_1, c_2, ..., c_r) = (k_1 + m_1, k_2 + m_2, ..., k_r + m_r)$. To decrypt the ciphertext, $m_i = c_i - k_i$. Once $(k_1, ..., k_r)$ have been used, they must never be used again.
  - One-time pad is information-theoretically secure against ciphertext-only attack: $\mathbb{P}(M = m \mid C = c) = \mathbb{P}(M = m)$.
  - Disadvantage is keys must never be reused, so must be as long as message.
  - Keys must be truly random.
- **Chinese remainder theorem**: let $m, n \in \mathbb{N}$ coprime, $a, b \in \mathbb{Z}$. Then exists unique solution $x \bmod mn$ to the congruences

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

- **Block cipher**: group characters in plaintext into blocks of $n$ (the **block length**) and encrypt each block with a key. So plaintext $p = (p_1, p_2, ...)$ is divided into blocks $P_1, P_2, ...$ where $P_1 = (p_1, ..., p_n)$, $P_2 = (p_{n+1}, ..., p_{2n})$, .... Then ciphertext blocks are given by $C_i = f(\text{key}, P_i)$ for some encryption function $f$.
- **Hill cipher**:
  - Plaintext divided into blocks $P_1, ..., P_r$ of length $n$.
  - Each block represented as vector $P_i \in (\mathbb{Z}/26\mathbb{Z})^n$
  - Key is invertible $n \times n$ matrix $M$ with elements in $\mathbb{Z}/26\mathbb{Z}$.
  - Ciphertext for block $P_i$ is

$$C_i = MP_i$$

    It can be decrypted with $P_i = M^{-1}C$.
  - Let $P = (P_1, ..., P_r)$, $C = (C_1, ..., C_r)$, then $C = MP$.
- **Confusion**: each character of ciphertext depends on many characters of key.
- **Diffusion**: each character of ciphertext depends on many characters of plaintext. Ideal diffusion is when changing single character of plaintext changes a proportion of $(S - 1)/S$ of the characters of the ciphertext, where $S$ is the number of possible symbols.
- For Hill cipher, $i$th character of ciphertext depends on $i$th row of key - this is medium confusion. If $j$th character of plaintext changes and $M_{ij} \neq 0$ then $i$th character of ciphertext changes. $M_{ij}$ is non-zero with probability roughly $25/26$ so good diffusion.
- Hill cipher is susceptible to known plaintext attack:
  - If $P = (P_1, ..., P_n)$ are $n$ blocks of plaintext with length $n$ such that $P$ is invertible and we know $P$ and the corresponding $C$, then we can recover $M$, since $C = MP \Longrightarrow M = CP^{-1}$.
  - If enough blocks of ciphertext are intercepted, it is very likely that $n$ of them will produce an invertible matrix $P$.

# 3. Public key encryption and RSA

- **Public key cryptosystem**:
  - Bob produces encryption key, $k_E$, and decryption key, $k_D$. He publishes $k_E$ and keeps $k_D$ secret.
  - To encrypt message $m$, Alice sends ciphertext $c = f(m, k_E)$ to Bob.
  - To decrypt ciphertext $c$, Bob computes $g(c, k_D)$, where $g$ satisfies
    $$g(f(m, k_E), k_D) = m$$
    for all messages $m$ and all possible keys.
  - Computing $m$ from $f(m, k_E)$ should be hard without knowing $k_D$.
- **Converting between messages and numbers**:
  - To convert message $m_1 m_2 ... m_r$, $m_i \in \{0, ..., 25\}$ to number, compute
    $$m = \sum_{i=1}^{r} m_i 26^{i-1}$$
  - To convert number $m$ to message, add character $m \bmod 26$ to message. If $m < 26$, stop. Otherwise, floor divide $m$ by 26 and repeat.
- **Fermat's little theorem**: let $p$ prime, $a \in \mathbb{Z}$ coprime to $p$, then $a^{p-1} \equiv 1 \pmod{p}$.
- **Euler $\varphi$ function**:
  $$\varphi : \mathbb{N} \to \mathbb{N}, \varphi(n) = |\{1 \leq a \leq n : \gcd(a, n) = 1\}| = |(\mathbb{Z}/n\mathbb{Z})^\times|$$
- $\varphi(p^r) = p^r - p^{r-1}$, $\varphi(mn) = \varphi(m)\varphi(n)$ for $\gcd(m, n) = 1$.
- **Euler's theorem**: if $\gcd(a, n) = 1$, $a^{\varphi(n)} \equiv 1 \pmod{n}$.
- **RSA algorithm**:
  - $k_E$ is pair $(n, e)$ where $n = pq$, the **RSA modulus**, is product of two distinct primes and $e \in \mathbb{Z}$, the **encryption exponent**, is coprime to $\varphi(n)$.
  - $k_D$, the **decryption exponent**, is integer $d$ such that $de \equiv 1 \pmod{\varphi(n)}$.
  - $m$ is an integer modulo $n$, $m$ and $n$ are coprime.
  - Encryption: $c = m^e \pmod{n}$.
  - Decryption: $m = c^d \pmod{n}$.
  - It is recommended that $n$ have at least 2048 bits. A typical choice of $e$ is $2^{16} + 1$.
- **RSA problem**: given $n = pq$ a product of two unknown primes, $e$ and $m^e \pmod{n}$, recover $m$. If $n$ can be factored, the RSA is solved.
- **Factorisation problem**: given $n = pq$ for large distinct primes $p$ and $q$, find $p$ and $q$.
- **RSA signatures**:
  - Public key is $(n, e)$ and private key is $d$.
  - When sending a message $m$, message is **signed** by also sending $s = m^d \bmod n$, the **signature**.
  - $(m, s)$ is received, **verified** by checking if $m = s^e \bmod n$.
  - Forging a signature on a message $m$ would require finding $s$ with $m = s^e \bmod n$. This is the RSA problem.

  - However, choosing signature $s$ first then taking $m = s^e \bmod n$ produces valid pairs.
  - To solve this, $(m, s)$ is sent where $s = h(m)^d$, $h$ is **hash function**. Then the message receiver verifies $h(m) = s^e \bmod n$.
  - Now, for a signature to be forged, an attacker would have to find $m$ with $h(m) = s^e \bmod n$.
- **Hash function** is function $h : \{\text{messages}\} \to \mathcal{H}$ that:
  - Can be computed efficiently
  - Is **preimage-resistant**: can't quickly find $m$ given $h(m)$.
  - Is **collision-resistant**: can't quickly find $m, m'$ such that $h(m) = h(m')$.
- **Attacks on RSA**:
  - If you can factor $n$, you can compute $d$, so can break RSA (as then you know $\varphi(n)$ so can compute $e^{-1} \bmod \varphi(n)$).
  - If $\varphi(n)$ is known, then we have $pq = n$ and $(p-1)(q-1) = \varphi(n)$ so $p + q = n - \varphi(n) + 1$. Hence $p$ and $q$ are roots of $x^2 - (n - \varphi(n) + 1)x + n$.
  - **Known $d$ attack**:
    - $de - 1$ is multiple of $\varphi(n)$ so $p, q \mid x^{de-1} - 1$.
    - Look for factor $K$ of $de - 1$ with $x^K - 1$ divisible by $p$ but not $q$ (or vice versa) (equivalently, $(p-1) \mid K$ but $(q-1) \nmid K$).
    - Let $de - 1 = 2^r s$, $\gcd(2, s) = 1$, choose random $x \bmod n$. Let $y = x^s$, then $y^{2^r} = x^{2^r s} = x^{de-1} \equiv 1 \bmod n$.
    - If $y \equiv 1 \bmod n$, restart with new random $x$. Find first occurence of 1 in $y, y^2, ..., y^{2^r}$: $y^{2^j} \not\equiv 1 \bmod n$, $y^{2^{j+1}} \equiv 1 \bmod n$ for some $j \geq 0$.
    - Let $a := y^{2^j}$, then $a^2 \equiv 1 \bmod n$, $a \not\equiv 1 \bmod n$. If $a \equiv -1 \bmod n$, restart with new random $x$.
    - Now $n = pq \mid a^2 - 1 = (a+1)(a-1)$ but $n \nmid (a+1), (a-1)$. So $p$ divides one of $a+1$, $a-1$ and $q$ divides the other. So $\gcd(a-1, n)$, $\gcd(a+1, n)$ are prime factors of $n$.
- **Theorem**: it is no easier to find $\varphi(n)$ than to factorise $n$.
- **Theorem**: it is no easier to find $d$ than to factor $n$.
- **Miller-Rabin algorithm** for probabilistic primality testing of $n$:
  1. Let $n - 1 = 2^r s$, $\gcd(2, s) = 1$.
  2. Choose random $x \bmod n$, compute $y = x^s \bmod n$.
  3. Compute $y, y^2, ..., y^{2^r} \bmod n$.
  4. If 1 isn't in this list, $n$ is **composite** (with witness $x$).
  5. If 1 is in list preceded by number other than ±1, $n$ is **composite** (with witness $x$).
  6. Other, $n$ is **possible prime** (to base $x$).
- **Theorem**:
  - If $n$ prime then it is possible prime to every base.
  - If $n$ composite then it is possible prime to $\leq 1/4$ of possible bases.

  In particular, if $k$ random bases are chosen, probability of composite $n$ being possible prime for all $k$ bases is $\leq 4^{-k}$.

## 3.1. Factorisation

- **Trial division algorithm**: for $p = 2, 3, 5, ...$ test whether $p \mid n$.
- If $x^2 \equiv y^2 \bmod n$ but $x \not\equiv \pm y \bmod n$, then $x - y$ is divisible by factor of $n$ but not by $n$ itself, so $\gcd(x - y, n)$ gives proper factor of $n$ (or 1).
- **Fermat's method**:
  - Let $a = \lceil \sqrt{n} \rceil$. Compute $a^2 \bmod n$, $(a + 1)^2 \bmod n$ until a square $x^2 \equiv (a + i)^2 \bmod n$ appears. Then compute $\gcd(a + i - x, n)$.
  - Works well under special conditions on the factors: if $|p - q| \leq 2\sqrt{2}\sqrt[4]{n}$ then Fermat's method takes one step: $x = \lceil \sqrt{n} \rceil$ works.
- **Definition**: an integer is ***B*-smooth** if all its prime factors are $\leq B$.
- **Quadratic sieve**:
  - Choose $B$ and let $m$ be number of primes $\leq B$.
  - Look at integers $x = \lceil \sqrt{n} \rceil + k$, $k = 1, 2, ...$ and check whether $y = x^2 - n$ is $B$-smooth.
  - Once $y_1 = x_1^2 - n, ..., y_t = x_t^2 - n$ are all $B$-smooth with $t > m$, find some product of them that is a square.
  - Deduce a congruence between the squares.
  - Time complexity is $\exp\left(\sqrt{\log n \log \log n}\right)$.

# 4. Diffie-Hellman key exchange

- **Primitive root theorem**: let $p$ prime, then there exists $g \in \mathbb{F}_p^\times$ such that $1, g, ..., g^{p-2}$ is complete set of residues $\bmod p$.
- Let $p$ prime, $g \in \mathbb{F}_p^\times$. **Order** of $g$ is smallest $a \in \mathbb{N}_0$ such that $g^a = 1$. $g$ is **primitive root** if its order is $p - 1$ (equivalently, $1, g, ..., g^{p-2}$ is complete set of residues $\bmod p$).
- Let $p$ prime, $g \in \mathbb{F}_p^\times$ primitive root. If $x \in \mathbb{F}_p^\times$ then $x = g^L$ for some $0 \leq L < p - 1$. Then $L$ is **discrete logarithm** of $x$ to base $g$. Write $L = L_g(x)$.
- **Proposition**:
  - $g^{L_g(x)} \equiv x \pmod{p}$ and $g^a \equiv x \pmod{p} \Longleftrightarrow a \equiv L_g(x) \pmod{p - 1}$.
  - $L_g(1) = 0$, $L_g(g) = 1$.
  - $L_g(xy) \equiv L_g(x) + L_g(y) \pmod{p - 1}$.
  - $L_g(x^{-1}) = -L_g(x)$ $(\bmod p - 1)$.
  - $L_g(g^a \bmod p) \equiv a$ $(\bmod p - 1)$.
  - $h$ is primitive root mod $p$ iff $L_g(h)$ coprime to $p - 1$. So number of primitive roots mod $p$ is $\varphi(p - 1)$.
- **Discrete logarithm problem**: given $p, g, x$, compute $L_g(x)$.
- **Diffie-Hellman key exchange**:
  - Alice and Bob publicly choose prime $p$ and primitive root $g \bmod p$.
  - Alice chooses secret $\alpha \bmod (p - 1)$ and sends $g^\alpha \bmod p$ to Bob publicly.
  - Bob chooses secret $\beta \bmod (p - 1)$ and sends $g^\beta \bmod p$ to Alice publicly.
  - Alice and Bob both compute shared secret $\kappa = g^{\alpha\beta} = (g^\alpha)^\beta) = \left(g^\beta\right)^\alpha \bmod p$.
- **Diffie-Hellman problem**: given $p, g, g^\alpha, g^\beta$, compute $g^{\alpha\beta}$.

- If discrete logarithm problem cna be solved, so can Diffie-Hellman problem (since could compute $\alpha = L_g(g^a)$ or $\beta = L_g\left(g^\beta\right)$).
- **Elgamal public key encryption**:
  - Alice chooses prime $p$, primitive root $g$, private key $\alpha \mod(p-1)$.
  - Her public key is $y = g^\alpha$.
  - Bob chooses random $k \operatorname{mod}(p-1)$
  - To send message $m$ (integer mod $p$), he sends the pair $(r, m') = \left(g^k, m y^k\right)$.
  - To decrypt message, Alice computes $r^\alpha = g^{\alpha k} = y^k$ and then $m' r^{-\alpha} = m' y^{-k} = m g^{\alpha k} g^{-\alpha k} m$.
  - If Diffie-Hellman problem is hard, then Elgamal encryption is secure against known plaintext attack.
  - Key $k$ must be random and different each time.
- **Decision Diffie-Hellman problem**: given $g^a, g^b, c$ in $\mathbb{F}_p^\times$, decide whether $c = g^{ab}$.
  - This problem is not always hard, as can tell if $g^{ab}$ is square or not. Can fix this by taking $g$ to have large prime order $q \mid (p-1)$. $p = 2q + 1$ is a good choice.
- **Elgamal signatures**:
  - Public key is $(p, g)$, $y = g^\alpha$ for private key $\alpha$.
  - **Valid Elgamal signature** on $m \in \{0, ..., p-2\}$ is pair $(r, s)$, $0 \leq r, s \leq p-1$ such that

    $$y^r r^s = g^m \pmod{p}$$

  - Alice computes $r = g^k$, $k \in (\mathbb{Z}/(p-1))^\times$ random. $k$ should be different each time.
  - Then $g^{\alpha r} g^{ks} \equiv g^m \mod p$ so $\alpha r + ks \equiv m \pmod{p-1}$ so $s = k^{-1}(m - \alpha r) \mod p - 1$.
- **Elgamal signature problem**: given $p, g, y, m$, find $r, s$ such that $y^r r^s = m$.
- **Discrete logarithm problem**: given prime $p$, primitive root $g \operatorname{mod} p$, $x \in \mathbb{F}_p^\times$, calculate $L_g(x)$.
- **Baby-step giant-step algorithm** for solving DLP:
  - Let $N = \left\lceil\sqrt{p-1}\right\rceil$.
  - Baby-steps: compute $g^j \operatorname{mod} p$ for $0 \leq j < N$.
  - Giant-steps: compute $x g^{-Nk} \operatorname{mod} p$ for $0 \leq k < N$.
  - Look for a match between baby-steps and giant-steps lists: $g^j = x g^{-Nk} \Longrightarrow x = g^{j+Nk}$.
  - Always works since if $x = g^L$ for $0 \leq L < p - 1 \leq N^2$, $L$ can be written as $j + Nk$ with $j, k \in \{0, ..., N-1\}$.
- **Index calculus** method for solving DLP $x = g^L$:
  - Fix smoothness bound $B$.
  - Find many multiplicative relations between $B$-smooth numbers and powers of $g \operatorname{mod} p$.
  - Solve these relations to find discrete logarithms of primes $\leq B$.
  - For $i = 1, 2, ...$ compute $x g^i \operatorname{mod} p$ until one is $B$-smooth, then use result from previous step.

- **Pohlig-Hellman algorithm** computes discrete logarithms $\bmod p$ with approximate complexity $\log(p)\sqrt{\ell}$ where $\ell$ is largest prime factor of $p-1$, so is fast if $p-1$ is $B$-smooth. Therefore $p$ is chosen so that $p-1$ has large prime factor, e.g. choose **Germain prime** $p = 2q+1$, with $q$ prime.

## 5. Elliptic curves

- **Definition**: **abelian group** $(G, \circ)$ satisfies:
  - **Associativity**: $\forall a, b, c, \in G, a \circ (b \circ c) = (a \circ b) \circ c$.
  - **Identity**: $\exists e \in G : \forall g \in G, e \times g = g$.
  - **Inverses**: $\forall g \in G, \exists h \in G : g \circ h = h \circ g = e$
  - **Commutativity**: $\forall a, b \in G, a \circ b = b \circ a$.
- **Definition**: $H \subseteq G$ is **subgroup** of $G$ if $(H, \circ)$ is group.
- To show $H$ is subgroup, sufficient to show $g, h \in H \Rightarrow g \circ h \in H$ and $h^{-1} \in H$.
- **Notation**: for $g \in G$, write $[n]g$ for $g \circ \cdots \circ g$ $n$ times if $n > 0$, $e$ if $n = 0$, $[-n]g^{-1}$ if $n < 0$.
- **Definition**: **subgroup generated by $g$** is

  $$\langle g \rangle = \{[n]g : n \in \mathbb{Z}\}$$

  If $\langle g \rangle$ finite, it has **order $n$**, and $g$ has **order $n$**. If $G = \langle g \rangle$ for some $g \in G$, $G$ is **cyclic** and $g$ is **generator**.
- **Lagrange's theorem**: let $G$ finite group, $H$ subgroup of $G$, then $|H| \mid |G|$.
- **Corollary**: if $G$ finite, $g \in G$ has order $n$, then $n \mid |G|$.
- **DLP for abelian groups**: given $G$ a cyclic abelian group, $g \in G$ a generator of $G$, $x \in G$, find $L$ such that $[L]g = x$. $L$ is well-defined modulo $|G|$.
- **Definition**: let $(G, \circ)$, $(H, \bullet)$ abelian groups, **homomorphism** between $G$ and $H$ is $f : G \to H$ with

  $$\forall g, g' \in G, \quad f(g \circ g') = f(g) \bullet f(g')$$

  **Isomorphism** is bijective homomorphism. $G$ and $H$ are **isomorphic**, $G \cong H$, if there is isomorphism between them.
- **Fundamental theorem of finite abelian groups**: let $G$ finite abelian group, then there exist unique integers $2 \leq n_1, ..., n_r$ with $n_i \mid n_{i+1}$ for all $i$, such that

  $$G \simeq (\mathbb{Z}/n_1) \times \cdots \times (\mathbb{Z}/n_r)$$

  In particular, $G$ is isomorphic to product of cyclic groups.
- **Definition**: let $K$ field, $\operatorname{char}(K) > 3$. An **elliptic curve** over $K$ is defined by the equation

  $$y^2 = x^3 + ax + b$$

  where $a, b \in K$, $\Delta_E := 4a^3 + 27b^2 \neq 0$.
- **Remark**: $\Delta_E \neq 0$ is equivalent to $x^3 + ax + b$ having no repeated roots (i.e. $E$ is smooth).

- **Definition**: for elliptic curve $E$ defined over $K$, a ***K*-point** (**point**) on $E$ is either:
  - A **normal point**: $(x, y) \in K^2$ satisfying the equation defining $E$.
  - The **point at infinity** $\overline{O}$ which can be thought of as infinitely far along the $y$-axis (in either direction).

  Denote set of all $K$-points on $E$ as $E(K)$.
- Any elliptic curve $E(K)$ is an abelian group, with group operation $\oplus$ is defined as:
  - We should have $P \oplus Q \oplus R = \overline{O}$ iff $P, Q, R$ lie on straight line.
  - In this case, $P \oplus Q = -R$.
  - To find line $\ell$ passing through $P = (x_0, y_0)$ and $Q = (x_1, y_1)$:
    - If $x_0 \neq x_1$, then equation of $\ell$ is $y = \lambda x + \mu$, where

      $$\lambda = \frac{y_1 - y_0}{x_1 - x_0}, \quad \mu = y_0 - \lambda x_0$$

      Now

      $$\begin{aligned} y^2 = x^3 + ax + b &= (\lambda x + \mu)^2 \\ \Longrightarrow 0 &= x^3 - \lambda^2 x^2 + (a - 2\lambda\mu)x + (b - \mu^2) \end{aligned}$$

      Since sum of roots of monic polynomial is equal to minus the coefficient of the second highest power, and two roots are $x_0$ and $x_1$, the third root is $x_2 = \lambda^2 - x_0 - x_1$. Then $y_2 = \lambda x_2 + \mu$ and $R = (x_2, y_2)$.
    - If $x_0 = x_1$, then using implicit differentiation:

      $$\begin{aligned} y^2 &= x^3 + ax + b \\ \Longrightarrow \frac{\mathrm{d}y}{\mathrm{d}x} &= \frac{3x^2 + a}{2y} \end{aligned}$$

      and the rest is as above, but instead with $\lambda = \frac{3x_0^2 + a}{2y_0}$.
- **Definition**: **group law** of elliptic curves: let $E : y^2 = x^3 + ax + b$. For all normal points $P = (x_0, y_0), Q = (x_1, y_1) \in E(K)$, define
  - $\overline{O}$ is group identity: $P \oplus \overline{O} = \overline{O} \oplus P = P$.
  - If $P = -Q =: (x_0, -y_0)$, $P \oplus Q = \overline{O}$.
  - Otherwise, $P \oplus Q = (x_2, -y_2)$, where

    $$\begin{aligned} x_2 &= \lambda^2 - (x_0 + x_1), \\ y_2 &= \lambda x_2 + \mu, \\ \lambda &= \begin{cases} \frac{y_1 - y_0}{x_1 - x_0} & \text{if } x_0 \neq x_1 \\ \frac{3x_0^2 + a}{2y_0} & \text{if } x_0 = x_1 \end{cases}, \\ \mu &= y_0 - \lambda x_0 \end{aligned}$$
- **Example**:
  - Let $E$ be given by $y^2 = x^3 + 17$ over $\mathbb{Q}$, $P = (-1, 4) \in E(\mathbb{Q})$, $Q = (2, 5) \in E(\mathbb{Q})$. To find $P \oplus Q$,

$$\lambda = \frac{5-4}{2-(-1)} = \frac{1}{3}, \quad \mu = 4 - \lambda(-1) = \frac{13}{3}$$

So $x_2 = \lambda^2 - (-1) - 2 = -\frac{8}{9}$ and $y_2 = -(\lambda x_2 + \mu) = -\frac{109}{27}$ hence

$$P \oplus Q = \left(-\frac{8}{9}, -\frac{109}{27}\right)$$

To find $[2]P$,

$$\lambda = \frac{3(-1)^2 + 0}{2 \cdot 4} = \frac{3}{8}, \quad \mu = 4 - \frac{3}{8} \cdot (-1) = \frac{35}{8}$$

so $x_3 = \lambda^2 - 2 \cdot (-1) \frac{137}{64}$, $y_3 = -(\lambda x_3 + \mu) = -\frac{2651}{512}$ hence

$$[2]P = (x_3, y_3) = \left(\frac{137}{64}, -\frac{2651}{512}\right)$$

- **Hasse's theorem**: let $|E(\mathbb{F}_p)| = N$, then
$$|N - (p+1)| \leq 2\sqrt{p}$$
- **Theorem**: $E(\mathbb{F}_p)$ is isomorphic to either $\mathbb{Z}/k$ or $\mathbb{Z}/m \times \mathbb{Z}/n$ with $m \mid n$.
- **Elliptic curve Diffie-Hellman**:
  - Alice and Bob publicly choose elliptic curve $E(\mathbb{F}_p)$ and $P \in \mathbb{F}_p$ with order a large prime $n$.
  - Alice chooses random $\alpha \in \{0, ..., n-1\}$ and publishes $Q_A = [\alpha]P$.
  - Bob chooses random $\beta \in \{0, ..., n-1\}$ and publishes $Q_B = [\beta]P$.
  - Alice computes $[\alpha]Q_B = [\alpha\beta]P$, Bob computes $[\beta]Q_A = [\beta\alpha]P$.
  - Shared key is $K = [\alpha\beta]P$.
- **Elliptic curve Elgamal signatures**:
  - Use agreed elliptic curve $E$ over $\mathbb{F}_p$, point $P \in E(\mathbb{F}_p)$ of prime order $n$.
  - Alice wants to sign message $m$, encoded as integer mod $n$.
  - Alice generates private key $\alpha \in \mathbb{Z}/n$ and public key $Q = [\alpha]P$.
  - Valid signature is $(R, s)$ where $R = (x_R, y_R) \in E(\mathbb{F}_p)$, $s \in \mathbb{Z}/n$, $[\widetilde{x_R}]Q \oplus [s]R = [m]P$.
  - To generate a valid signature, Alice chooses random $0 \neq k \in \mathbb{Z}/n$ and sets $R = [k]P$, $s = k^{-1}(m - \widetilde{x_R}\alpha)$.
  - $k$ must be randomly generated for each message.
- **Baby-step giant-step algorithm for elliptic curve DLP**: given $P$ and $Q = [\alpha]P$, find $\alpha$:
  - Let $N = \lceil \sqrt{n} \rceil$, $n$ is order of $P$.
  - Compute $P, [2]P, ..., [N-1]P$.
  - Compute $Q \oplus [-N]P, Q \oplus [-2N]P, ..., Q \oplus [-(N-1)N]P$ and find a match between these two lists: $[i]P = Q \oplus [-jN]P$, then $[i + jN]P = Q$ so $\alpha = i + jN$.
- For well-chosen elliptic curves, the best algorithm for solving DLP is the baby-step giant-step algorithm, with run time $O(\sqrt{n}) \approx O(\sqrt{p})$. This is much slower than the index-calculus method for the DLP in $\mathbb{F}_p^\times$.

- **Pollard's $p - 1$ algorithm** to factorise $n = pq$:
  - Choose smoothness bound $B$.
  - Choose random $2 \leq a \leq n - 2$. Set $a_1 = a$, $i = 1$.
  - Compute $a_i = a_{i-1}^i \bmod n$. Find $d = \gcd(a_i - 1, n)$. If $1 < d < n$, we have found a nontrivial factor of $n$. If $d = n$, pick new $a$ and retry. If $d = 1$, increment $i$ by 1 and repeat this step.
  - A variant is instead of computing $a_i = a_{i-1}^i$, compute $a_i = a_{i-1}^{m_{i-1}}$ where $m_1, ..., m_r$ are the prime powers $\leq B$ (each prime power is the maximal prime power $\leq B$ for that prime).
  - The algorithm works if $p - 1$ is ***B*-powersmooth** (all prime power factors are $\leq B$), since if $b$ is order of $a \bmod p$, then $b \mid (p - 1)$ so $b \mid B!$ (also $b \mid m_1 \cdots m_r$). If the first $i$ for which $i!$ (or $m_1 \cdots m_i$) is divisible by $d$ and order of $a \bmod q$, then $a_i - 1 = a^{i!} - 1 \bmod n$ is divisible by both $p$ and $q$, so must retry with different $a$.
- Let $n = pq$, $p, q$ prime, $a, b \in \mathbb{Z}$, $\gcd(4a^3 + 27b^2, n) = 1$. Then $E : y^2 = x^3 + ax + b$ defines elliptic curve over $\mathbb{F}_p$ and over $\mathbb{F}_q$. If $(x, y) \in \mathbb{Z}/n$ is solution to $E \bmod n$ then can reduce coordinates $\bmod p$ to obtain non-infinite point of $E(\mathbb{F}_p)$ and $\bmod q$ to obtain non-infinite point of $E(\mathbb{F}_q)$.
- **Proposition**: let $P_1, P_2 \in E \bmod n$, with

$$(P_1 \bmod p) \oplus (P_2 \bmod p) = \overline{O}$$
$$(P_1 \bmod q) \oplus (P_2 \bmod q) \neq \overline{O}$$

  Then $\gcd(x_1 - x_2, n)$ (or $\gcd(2x_1, n)$ if $P_1 = P_2$) is factor of $n$.
- **Lenstra's algorithm** to factorise $n$:
  - Choose smoothness bound $B$.
  - Choose random elliptic curve $E$ over $\mathbb{Z}/n$ with $\gcd(\Delta_E, n) = 1$ and $P = (x, y)$ a point on $E$.
  - Set $P_1 = P$, attempt to compute $P_i$, $2 \leq i \leq B$ by $P_i = [i]P_{i-1}$. If one of these fails, a divisor of $n$ has been found (by failing to compute an inverse $\bmod n$). If this divisor is trivial, restart with new curve and point.
  - If $i = B$ is reached, restart with new curve and point.
  - Again, a variant is calculating $P_i = [m_i]P_{i-1}$ instead of $[i]P_{i-1}$ where $m_1, ..., m_r$ are the prime powers $\leq B$
- Lenstra's algorithm works if $|E(\mathbb{Z}/p)|$ is $B$-powersmooth but $|E(\mathbb{Z}/q)|$ isn't. Since we can vary $E$, it is very likely to work eventually.
- Running time depends on $p$ (the smaller prime factor):

$$O\left(\exp\left(\sqrt{2\log(p)\log\log(p)}\right)\right)$$

  Compare this to the general number field sieve running time:

$$O\left(\exp\left(C(\log n)^{1/3}(\log\log n)^{2/3}\right)\right)$$

## 5.1. Torsion points

- **Definition**: let $G$ abelian group. $g \in G$ is a **torsion** if it has finite order. If order divides $n$, then $[n]g = e$ and $g$ is ***n*-torsion**.
- **Definition**: ***n*-torsion subgroup** is

$$G[n] := \{g \in G : [n]g = e\}$$

- **Definition**: **torsion subgroup** of $G$ is

$$G_{\text{tors}} = \{g \in G : g \text{ is torsion}\} = \bigcup_{n \in \mathbb{N}} G[n]$$

- **Example**:
  - In $\mathbb{Z}$, only 0 is torsion.
  - In $(\mathbb{Z}/10)^{\times}$, by Lagrange's theorem, every point is 4-torsion.
  - For finite groups $G$, $G_{\text{tors}} = G = G[|G|]$ by Lagrange's theorem.

## 5.2. Rational points

- **Note**: for elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{Q}$, can assume that $a, b \in \mathbb{Z}$.
- **Nagell-Lutz theorem**: let $E$ elliptic curve, let $P = (x, y) \in E(\mathbb{Q})_{\text{tors}}$. Then $x, y \in \mathbb{Z}$, and either $y = 0$ (in which case $P$ is 2-torsion) or $y^2 \mid \Delta_E$.
- **Corollary**: $E(\mathbb{Q})_{\text{tors}}$ is finite.
- **Example**: can use Nagell-Lutz to show a point is not torsion.
  - $P = (0, 1)$ lies on elliptic curve $y^2 = x^3 - x + 1$. $[2]P = \left(\frac{1}{4}, -\frac{7}{8}\right) \notin \mathbb{Z}^2$. Then $[2]P$ is not torsion, hence $P$ is not torsion. So $E(\mathbb{Q})$ contains distinct points $..., [-2]P, -P, \overline{O}, P, [2]P, ...$, hence $E$ has infinitely many solutions in $\mathbb{Q}$.
- **Mazur's theorem**: let $E$ be elliptic curve over $\mathbb{Q}$. Then $E(\mathbb{Q})_{\text{tors}}$ is either:
  - cyclic of order $1 \leq N \leq 10$ or order 12, or
  - of the form $\mathbb{Z}/2 \times \mathbb{Z}/2N$ for $1 \leq N \leq 4$.
- **Definition**: let $E : y^2 = x^3 + ax + b$ defined over $\mathbb{Q}$, $a, b \in \mathbb{Z}$. For odd prime $p$, taking reductions $\overline{a}$, $\overline{b} \bmod p$ gives curve over $\mathbb{F}_p$:

$$\overline{E} : y^2 = x^3 + \overline{a}x + \overline{b}$$

  This is elliptic curve if $\Delta_E \not\equiv 0 \bmod p$, in which case $p$ is **prime of good reduction** for $E$.
- **Theorem**: let $E : y^2 = x^3 + ax + b$ defined over $\mathbb{Q}$, $a, b \in \mathbb{Z}$, $p$ be odd prime of good reduction for $E$. Then $f : E(\mathbb{Q})_{\text{tors}} \to \overline{E}(\mathbb{F}_p)$ defined by

$$f(x, y) := (\overline{x}, \overline{y}), \quad f(\overline{O}) := \overline{O}$$

  is injective (note $x, y \in \mathbb{Z}$ by Nagell-Lutz).
- So $E(\mathbb{Q})_{\text{tors}}$ can be thought of as subgroup of $E(\mathbb{F}_p)$ for any prime $p$ of good reduction, so by Lagrange's theorem, $|E(\mathbb{Q})_{\text{tors}}|$ divides $|E(\mathbb{F}_p)|$.
- **Mordell's theorem**: if $E$ is elliptic curve over $\mathbb{Q}$, then

$$E(\mathbb{Q}) \cong E(\mathbb{Q})_{\text{tors}} \times \mathbb{Z}^r$$

  for some $r \geq 0$ the **rank** of $E$. So for some $P_1, ..., P_r \in E(\mathbb{Q})$,

$$E(\mathbb{Q}) = \{n_1 P_1 + \cdots + n_r P_r + T : n_i \in \mathbb{Z}, T \in E(\mathbb{Q})_{\text{tors}}\}$$

$P_1, ..., P_r, T$ are **generators** for $E(\mathbb{Q})$.

# 6. Basic coding theory

## 6.1. First definitions

- **Definition**:
  - **Alphabet** $A$ is finite set of symbols.
  - $A^n$ is set of all lists of $n$ symbols from $A$ - these are **words of length $n$**.
  - **Code of block length $n$ on $A$** is subset of $A^n$.
  - **Codeword** is element of a code.
- **Definition**: if $|A| = 2$, codes on $A$ are **binary** codes. If $|A| = 3$, codes on $A$ are **ternary codes**. If $|A| = q$, codes on $A$ are **$q$-ary** codes. Generally, use $A = \{0, 1, ..., q - 1\}$.
- **Definition**: let $x = x_1...x_n, y = y_1...y_n \in A^n$. **Hamming distance** between $x$ and $y$ is number of indices where $x$ and $y$ differ:

  $$d : A^n \times A^n \to \{0, ..., n\}, \quad d(x, y) := |\{i \in [n] : x_i \neq y_i\}|$$

  So $d(x, y)$ is minimum number of changes needed to change $x$ to $y$. If $x$ transmitted and $y$ received, then $d(x, y)$ **symbol-errors** have occurred.
- **Proposition**: let $x, y$ words of length $n$.
  - $0 \leq d(x, y) \leq n$.
  - $d(x, y) = 0 \iff x = y$.
  - $d(x, y) = d(y, x)$.
  - $\forall z \in A^n, d(x, y) \leq d(x, z) + d(z, y)$.
- **Definition**: **minimum distance** of code $C$ is

  $$d(C) := \min\{d(x, y) : x, y \in C, x \neq y\} \in \mathbb{N}$$
- **Notation**: code of block length $n$ with $M$ codewords and minimum distance $d$ is called $(n, M, d)$ (or $(n, M)$) code. A $q$-ary code is called an $(n, M, d)_q$ code.
- **Definition**: let $C \subseteq A^n$ code, $x$ word of length $n$. A **nearest neighbour** of $x$ is codeword $c \in C$ such that $d(x, c) = \min\{d(x, y) : y \in C\}$.

## 6.2. Nearest-neighbour decoding

- **Definition**: **nearest-neighbour decoding (NND)** means if word $x$ received, it is decoded to a nearest neighbour of $x$ in a code $C$.
- **Proposition**: let $C$ be code with minimum distance $d$, let word $x$ be received with $t$ symbol errors. Then
  - If $t \leq d - 1$, then we can detect that $x$ has some errors.
  - If $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$, then NND will correct the errors.

## 6.3. Probabilities

- **Definition**: **$q$-ary symmetric channel with symbol-error probability $p$** is channel for $q$-ary alphabet $A$ such that:
  - For every $a \in A$, probability that $a$ is changed in channel is $p$.
  - For every $a \neq b \in A$, probability that $a$ is changed to $b$ in channel is

$$\mathbb{P}(b \text{ received} \mid a \text{ sent}) = \frac{p}{q-1}$$

i.e. symbol-errors in different positions are independent events.

- **Proposition**: let $c$ codeword in $q$-ary code $C \subseteq A^n$ sent over $q$-ary symmetric channel with symbol-error probability $p$. Then

$$\mathbb{P}(x \text{ received} \mid c \text{ sent}) = \left(\frac{p}{q-1}\right)^t (1-p)^{n-t}, \quad \text{where } t = d(c, x)$$

- **Example**: let $C = \{000, 111\} \subset \{0,1\}^3$.

| $x$ | $t = d(000, x)$ | chance 000 received as $x$ | chance if $p = 0.01$ | NND decodes correctly? |
|---|---|---|---|---|
| 000 | 0 | $(1-p)^3$ | 0.970299 | yes |
| 100 | 1 | $p(1-p)^2$ | 0.009801 | yes |
| 010 | 1 | $p(1-p)^2$ | 0.009801 | yes |
| 001 | 1 | $p(1-p)^2$ | 0.009801 | yes |
| 110 | 2 | $p^2(1-p)$ | 0.000099 | no |
| 101 | 2 | $p^2(1-p)$ | 0.000099 | no |
| 011 | 2 | $p^2(1-p)$ | 0.000099 | no |
| 111 | 3 | $p^3$ | 0.000001 | no |

- **Corollary**: if $p < \frac{q-1}{q}$ then $P(x \text{ received} \mid c \text{ sent})$ increases as $d(x, c)$ decreases.
- **Remark**: by Bayes' theorem,

$$\mathbb{P}(c \text{ sent} \mid x \text{ received}) = \frac{\mathbb{P}(c \text{ sent and } x \text{ received})}{\mathbb{P}(x \text{ received})} = \frac{\mathbb{P}(c \text{ sent})\mathbb{P}(x \text{ received} \mid c \text{ sent})}{\mathbb{P}(x \text{ received})}$$

- **Proposition**: let $C$ be $q$-ary $(n, M, d)$ code used over $q$-ary symmetric channel with symbol-error probability $p < (q-1)/q$, and each codeword $c \in C$ is equally likely to be sent. Then for any word $x$, $\mathbb{P}(c \text{ sent} \mid x \text{ received})$ increases as $d(x, c)$ decreases.

## 6.4. Bounds on codes

- **Proposition (singleton bound)**: for $q$-ary code $(n, M, d)$ code, $M \leq q^{n-d+1}$.
- **Definition**: code which saturates singleton bound is called **maximum distance separable (MDS)**.
- **Example**: let $C_n$ be **binary repetition code** of block length $n$,

$$C_n := \{\underbrace{00...0}_{n}, \underbrace{11...1}_{n}\} \subset \{0,1\}^n$$

$C_n$ is $(n, 2, n)_2$ code, and $2 = 2^{n-n+1}$ so $C_n$ is MDS code.

- **Definition**: let $A$ be alphabet, $|A| = q$. Let $n \in \mathbb{N}$, $0 \leq t \leq n$, $t \in \mathbb{N}$, $x \in A^n$.
  - **Ball of radius $t$ around $x$** is

    $S(x, t) := \{y \in A^n : d(y, x) \leq t\}$

- Code $C \subseteq A^n$ is **perfect** if

  $$\exists t \in \mathbb{N} : A^n = \coprod_{c \in C} S(c, t)$$

  where $\coprod$ is disjoint union.
- **Example**: for $C = \{000, 111\} \subset \{0, 1\}^3$, $S(000, 1) = \{000, 100, 010, 001\}$ and $S(111, 1) = \{111, 011, 101, 110\}$. These are disjoint and $S(000, 1) \cup S(111, 1) = \{0, 1\}^3$, so $C$ is perfect.
- **Example**: let $C = \{111, 020, 202\} \subset \{0, 1, 2\}^3$. $\forall c \in C, d(c, 012) = 2$. So 012 is not in any $S(c, 1)$ but is in every $S(c, 2)$, so $C$ is not perfect.
- **Lemma**: let $|A| = q$, $x \in \mathbb{A}^n$, then

  $$|S(x, t)| = \sum_{k=0}^{t} \binom{n}{k} (q - 1)^k$$
- **Example**: let $C = \{111, 020, 202\} \subset \{0, 1, 2\}^3$, so $q = 3$, $n = 3$. So $|S(x, 1)| = \binom{3}{0} + \binom{3}{1}(3 - 1) = 7$, $|S(x, 2)| = \binom{3}{0} + \binom{3}{1}(3 - 1) + \binom{3}{2}(3 - 1)^2 = 19$. But $|\{0, 1, 2\}|^3 = 27$ and $7 \nmid 27$, $19 \nmid 27$, so $\{0, 1, 2\}^3$ can't be partioned by balls of either size. So $C$ can't be perfect. $|S(x, 3)| = 27$, but then $C$ must contain only one codeword to be perfect, and $|S(x, 0)| = 1$, but then $C = A^n$ to be perfect. These are trivial, useless codes.
- **Proposition (Hamming/sphere-packing bound)**: $q$-ary $(n, M, d)$ code satisfies

  $$M \sum_{k=0}^{t} \binom{n}{k} (q - 1)^k \leq q^n, \quad \text{where } t = \left\lfloor \frac{d - 1}{2} \right\rfloor$$
- **Corollary**: code saturates Hamming bound iff it is perfect.

# 7. Linear codes

## 7.1. Finite vector spaces

- **Definition**: **linear code** of block length $n$ is subspace of $\mathbb{F}_q^n$.
- **Example**: let $\boldsymbol{x} = (0, 1, 2, 0)$, $\boldsymbol{y} = (1, 1, 1, 1)$, $\boldsymbol{z} = (0, 2, 1, 0) \in \mathbb{F}_3^4$. $C_1 = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{0}\}$ is not linear code since e.g. $\boldsymbol{x} + \boldsymbol{y} = (1, 2, 0, 1) \notin C_1$. $C_2 = \{\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{0}\}$ is linear code.
- **Notation**: spanning set of $S$ is $\langle S \rangle$.
- **Proposition**: if linear code $C \subseteq \mathbb{F}_q^n$ has $\dim(C) = k$, then $|C| = q^k$.
- **Definition**: a $q$-ary $[n, k, d]$ code is linear code: a subspace of $\mathbb{F}_q^n$ of dimension $k$ with minimum distance $d$. Note: a $q$-ary $[n, k, d]$ code is a $q$-ary $\left(n, q^k, d\right)$ code.

## 7.2. Weight and minimum distance

- **Definition**: **weight** of $\boldsymbol{x} \in \mathbb{F}_q^n$, $w(\boldsymbol{x})$, is number of non-zero entries in $\boldsymbol{x}$:

  $$w(\boldsymbol{x}) = |\{i \in [n] : x_i \neq 0\}|$$
- **Lemma**: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^n$, $d(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{x} - \boldsymbol{y})$. In particular, $w(\boldsymbol{x}) = d(\boldsymbol{x}, \boldsymbol{0})$.
- **Proposition**: let $C \subseteq \mathbb{F}_q^n$ linear code, then

$$d(C) = \min\{w(\boldsymbol{c}) : \boldsymbol{c} \in C, \boldsymbol{c} \neq \boldsymbol{0}\}$$

- **Remark**: to find $d(C)$ for linear code with $q^k$ words, only need to consider $q^k$ weights instead of $\binom{q^k}{2}$ distances.

# 8. Codes as images

## 8.1. Generator-matrices

- **Definition**: let $C \subseteq \mathbb{F}_q^n$ be linear code. Let $G \in M_{k,n}(\mathbb{F}_q)$, $f_G : \mathbb{F}_q^k \to \mathbb{F}_q^n$ be linear map defined by $f_G(\boldsymbol{x}) = \boldsymbol{x}G$. Then $G$ is **generator-matrix** for $C$ if
  - $C = \text{im}(f) = \{\boldsymbol{x}G : \boldsymbol{x} \in \mathbb{F}_q^k\} \subseteq \mathbb{F}_q^n$.
  - The rows of $G$ are linearly independent.

  i.e. $G$ is generator-matrix for $C$ iff rows of $G$ form basis for $C$ (note $\boldsymbol{x}G = x_1\boldsymbol{g_1} + \cdots + x_k\boldsymbol{g_k}$ where $\boldsymbol{g_i}$ are rows of $G$).
- **Remark**: given linear code $C = \langle \boldsymbol{a}_1, ..., \boldsymbol{a}_m \rangle$, a generator-matrix can be found for $C$ by constructing the matrix $A$ with rows $\boldsymbol{a}_i$, then performing elementary row operations to bring $A$ into RREF. Once the $m - k$ bottom zero rows have been removed, the resulting matrix is a generator-matrix.
- **Example**: let $C = \langle\{(0,0,3,1,4), (2,4,1,4,0), (5,3,0,1,6)\}\rangle \subseteq \mathbb{F}_7^5$.

$$A = \begin{bmatrix} 2 & 4 & 1 & 4 & 0 \\ 5 & 3 & 0 & 1 & 6 \\ 0 & 0 & 3 & 1 & 4 \end{bmatrix} \underset{A_{12}(1)}{\to} \begin{bmatrix} 2 & 4 & 1 & 4 & 0 \\ 0 & 0 & 1 & 5 & 6 \\ 0 & 0 & 3 & 1 & 4 \end{bmatrix} \underset{M_1(4)}{\to} \begin{bmatrix} 1 & 2 & 4 & 2 & 0 \\ 0 & 0 & 1 & 5 & 6 \\ 0 & 0 & 3 & 1 & 4 \end{bmatrix} \underset{A_{21}(3), A_{23}(4)}{\to} \begin{bmatrix} 1 & 2 & 0 & 3 & 4 \\ 0 & 0 & 1 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

  So $G = \begin{bmatrix} 1 & 2 & 0 & 3 & 4 \\ 0 & 0 & 1 & 5 & 6 \end{bmatrix}$ is generator matrix for $C$ and $\dim(C) = 2$.

## 8.2. Encoding and channel decoding

## 8.3. Equivalence and standard form

- **Definition**: codes $C_1, C_2$ of block length $n$ over alphabet $A$ are **equivalent** if we can transform one to the other by applying sequence of the following two kinds of changes to all the codewords (simultaneously):
  - Permute the $n$ positions.
  - In a particular position, permuting the $|A| = q$ symbols.
- **Proposition**: equivalent codes have the same parameters $(n, M, d)$.
- **Definition**: linear codes $C_1, C_2 \subseteq \mathbb{F}_q^n$ are **monomially equivalent** if we can obtain one from the other by applying sequence of the following two kinds of changes to all codewords (simultaneously):
  - Permuting the $n$ positions.
  - In particular position, multiply by $\lambda \in \mathbb{F}_q^\times$.

  If only the first change is used, the codes are **permutation equivalent**.
- **Definition**: $P \in M_n(\mathbb{F}_q)$ is **permutation matrix** if it has a single 1 in each row and column, and zeros elsewhere. Any permutation of $n$ positions of row vector in $\mathbb{F}_q^n$ can be described as right multiplication by permutation matrix.
- **Proposition**: permutation matrices are orthogonal: $P^T = P^{-1}$.

- **Proposition**: let $C_1, C_2 \subseteq \mathbb{F}_q^n$ linear codes with generator matrices $G_1, G_2$. Then if $G_1 = G_2 P$ for permutation matrix $P$, then $C_1$ and $C_2$ are permutation equivalent.
- **Definition**: $M \in M_m(\mathbb{F}_q)$ is **monomial matrix** if it has exactly one non-zero element in each row and column.
- **Proposition**: monomial matrix $M$ can always be written as $M = DP$ or $M = PD'$ where $P$ is permutation matrix and $D, D'$ are diagonal matrices. $P$ is **permutation part**, $D$ and $D'$ are **diagonal parts** of $M$.
- **Example**:

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 3 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

- **Proposition**: let $C_1, C_2 \subseteq \mathbb{F}_q^n$ be linear codes with generator-matrices $G_1, G_2$. Then if $G_2 = G_1 M$ for some monomial matrix $M$, then $C_1$ and $C_2$ are monomially equivalent.
- **Definition**: let $C \subseteq \mathbb{F}_q^n$ linear code. If $G = (I_k \mid A)$, with $A \in M_{k,n-k}(\mathbb{F}_q)$, is generator-matrix for $C$, then $G$ is in **standard form**.
- **Note**: not every linear code has generator-matrix in standard form.
- **Proposition**: every linear code is permutation equivalent to a linear code with generator-matrix in standard form.
- **Example**: let $C_1 \subseteq \mathbb{F}_7^5$ have generator matrix $G_1 = \begin{bmatrix} 1 & 2 & 0 & 3 & 4 \\ 0 & 0 & 1 & 5 & 6 \end{bmatrix}$. Then applying permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \Longrightarrow G_1 P = \begin{bmatrix} 1 & 0 & 2 & 3 & 4 \\ 0 & 1 & 0 & 5 & 6 \end{bmatrix} = (I_2 \mid A)$$

# 9. Codes as kernels

## 9.1. Dual codes

- **Definition**: let $C \subseteq \mathbb{F}_q^n$ linear code. **Dual** of $C$ is

$$C^\perp := \left\{ \boldsymbol{v} \in \mathbb{F}_q^n : \forall \boldsymbol{u} \in C, \boldsymbol{v} \cdot \boldsymbol{u} = 0 \right\}$$

- **Proposition**: if $G$ is generator matrix for linear code $C$ then

$$C^\perp = \{ \boldsymbol{v} \in \mathbb{F}_q^n : \boldsymbol{v} G^T = \boldsymbol{0} \} = \ker(f_{G^T})$$

  where $f_{G^T} : \mathbb{F}_q^n \to \mathbb{F}_q^k$, $f(x) = x G^T$ is linear map.
- **Proposition**: let $C \subseteq \mathbb{F}_q^n$ linear code. Then $C^\perp$ is also linear code and $\dim(C) + \dim(C^\perp) = n$.
- **Proposition**: let $C \subseteq \mathbb{F}_q^n$ linear code, then $\left(C^\perp\right)^\perp = C$.
- **Proposition**: let $C \subseteq \mathbb{F}_q^n$ have generator-matrix in standard form, $G = (I_k \mid A)$, then $H = \left(-A^T \mid I_{n-k}\right)$ is generator-matrix for $C^\perp$.

- **Proposition**: let $G$ be generator matrix of $C \subseteq \mathbb{F}_q^n$, let $P \in M_n(\mathbb{F}_q)$ permutation matrix such that $GP = (I_k \mid A)$ for some $A \in M_{k,n-k}(\mathbb{F}_q)$. Then $H = (-A^T \mid I_{n-k})P^T$ is generator matrix for $C^{\perp}$.
- **Algorithm**: to find basis for dual code $C^{\perp}$, given generator matrix $G = (g_{ij}) \in M_{k,n}(\mathbb{F}_q)$ for $C$ in RREF:
  1. Let $L = \{1 \leq j \leq n : G \text{ has leading 1 in column } j\}$.
  2. For each $1 \leq j \leq n$, $j \notin L$, construct $\boldsymbol{v}_j$ as follows:
     1. For $m \notin L$, $m$th entry of $\boldsymbol{j}$ is 1 if $m = j$ and 0 otherwise.
     2. Fill in the other entries of $\boldsymbol{v}_j$ (left to right) as $-g_{1j}, ..., -g_{kj}$.
  3. The $n - k$ vectors $\boldsymbol{j}$ are basis for $C^{\perp}$.
- **Example**: let $C \subseteq \mathbb{F}_5^7$ be linear code with generator-matrix

  $$G = \begin{bmatrix} 1 & 2 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix}$$

  Then $L = \{1, 3, 6\}$.
  - $v_2 = (3, 1, 0, 0, 0, 0, 0)$
  - $v_4 = (2, 0, 4, 1, 0, 0, 0)$
  - $v_5 = (1, 0, 3, 0, 1, 0, 0)$
  - $v_7 = (0, 0, 2, 0, 0, 1, 1)$
  - So generator matrix for $C^{\perp}$ is

    $$H = \begin{bmatrix} 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 & 1 \end{bmatrix}$$

## 9.2. Check-matrices