1. Symmetric key ciphers

- **Symmetric key cipher**: one in which encryption and decryption keys are equal.
- **Key size**: $\log_2$(number of possible keys).
- **Caesar cipher**: shift all characters by a constant amount. Key size is $\log_2(26)$
- **Substitution cipher**: key is permutation of $\{a, ..., z\}$. Key size is $\log_2(26!)$.
- **Stirling's formula**:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

- If any statistical properties of plaintext are reflected in cipher text, then we can use this as basis for an attack. We compare the most common letters in the English language with the most common letters in the message. We can also compare letter pairs.
- **One-time pad**: key is random sequence of integers $\bmod 26$, $(k_1, k_2, ...)$. If message is $(m_1, m_2, ..., m_r)$ then ciphertext is $(c_1, c_2, ...) = (k_1 + m_1, k_2 + m_2, ...)$. 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: $\mathbb{P}(M = m \mid C = c) = \mathbb{P}(M = m)$.
  - Keys must never be reused, so must be as long as message.
  - Keys must be truly random.
- **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.
- For Hill cipher, ith character of ciphertext depends on ith row of key - this is medium confusion.
- 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 $C = MP \implies M = CP^{-1}$.

2. Public key cryptography and the RSA algorithm

- **Euler φ function**:

$$\varphi : \mathbb{N} \to \mathbb{N}, \varphi(n) = |\{1 \le a \le n : \gcd(a, n) = 1\}| = |(\mathbb{Z}/n\mathbb{Z})^\times|$$

- $\varphi(p^r) = p^r - p^{r-1}$, $\varphi(mn) = \varphi(m)\varphi(n)$ for $\gcd(m, n) = 1$.

- **Euler's theorem**: if $\gcd(a, n) = 1$, $a^{\varphi(n)} \equiv 1 \pmod{n}$.
- **Public key cryptography**:
  - Create two keys, k_D and k_E. k_E is public, k_D is private.
  - Plaintext m is encrypted as $c = f(m, k_E)$.
  - Ciphertext decrypted by $m = g(c, k_D)$.
- **RSA**:
  - k_E is pair (n, e) where $n = pq$ is product of two distinct primes and $e \in \mathbb{Z}$ is coprime to $\varphi(n)$.
  - k_D 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: $m = c^d \pmod{n}$.
- **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.
- It is recommended that n have at least 2048 bits. A typical choice of e is $2^{16} + 1$.
- **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 $e^{-1} \pmod{\varphi(n)}$).
  - 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 $x^2 - (n - \varphi(n) + 1) + n = 0$.
  - **Known d**: we have $de - 1$ is multiple of $\varphi(n)$. Look for a factor A of $de - 1$ such that $(p-1) \mid A$, $(q-1) \nmid A$. Then try $x^A - 1$ for random x, this satisfies $x^A - 1$ is divisible by p, hence $\gcd(x^A - 1, n) = p$.
- **RSA signatures**:
  - Public key is (n, e) and private key is d.
  - When sending a message m, message is **signed** by also sending $s = m^d \bmod n$.
  - (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 $m = s^e \bmod n$.
  - 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 with given $h(m)$.
  - Is collision-resistant: can't quickly find m, m' with $h(m) = h(m')$.

  Example is SHA-256.
- **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**:
  1. Choose random $x \bmod n$.

2. Let $n-1=2^r s$, $y=x^s$.
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 bynumber other than ± 1, n is **composite** (with witness a).
6. Other, n is **possible prime** (to base x).

2.1. Factorisation

- **Trial division algorithm**: for $p=2,3,5,...$ test whether $p \mid n$.
- **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.
- An integer is B-smooth if all its prime factors are $\leq B$.
- **Quadratic sieve**:
  - Choose B and let m be numberof 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.
- **Other factorisation algorithms**:
  - Pollard's ρ algorithm.
  - Pollard's $p-1$ algorithm.
  - Lenstra's algorithm using elliptic curves.
  - General number field sieve
  - Shor's algorithm: $\ln(N)^2 \ln(\ln(N))$.

2.2. Primitive roots

- 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 $p-1$.
- 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 $L=L_g(x)$. It satisfies:
  - $g^{L_g(x)} \equiv x \pmod p$ and $g^a \equiv x \pmod p \iff a \equiv L_g(x) \pmod{p-1}$.
  - $L_g(1)=0$, $L_g(g)=1$.
  - $L_g(xy) \equiv L_g(x)+L_g(y) \pmod{p-1}$.
  - 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**:
  - Two parties agree on prime p and primitive root $g \bmod p$.
  - Alice chooses secret $\alpha \bmod (p-1)$ and sends $g^\alpha \bmod p$ to Bob.
  - Bob chooses secret $\beta \bmod (p-1)$ and sends $g^\beta \bmod p$ to Alice.

  - Alice and Bob both compute $\kappa = g^{\alpha\beta} = (g^{\alpha})^{\beta}) = \left(g^{\beta}\right)^{\alpha} \bmod p$.
- **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 $\beta = L_g\left(g^{\beta}\right)$).