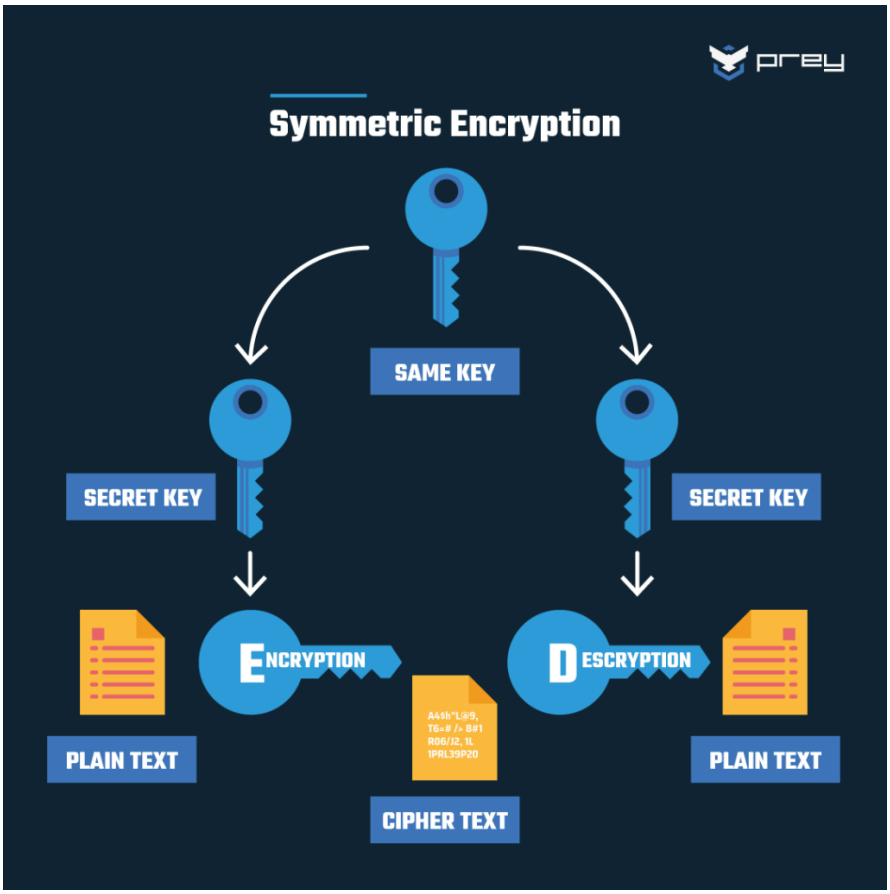
1. Cryptography:

- **Symmetric vs Asymmetric encryption**

1. ENCRYPTION :
   Process of converting data in an unreadable format (cipher text) using algorithm and a key, making it secure and only accessible to those with corresponding decryption key.

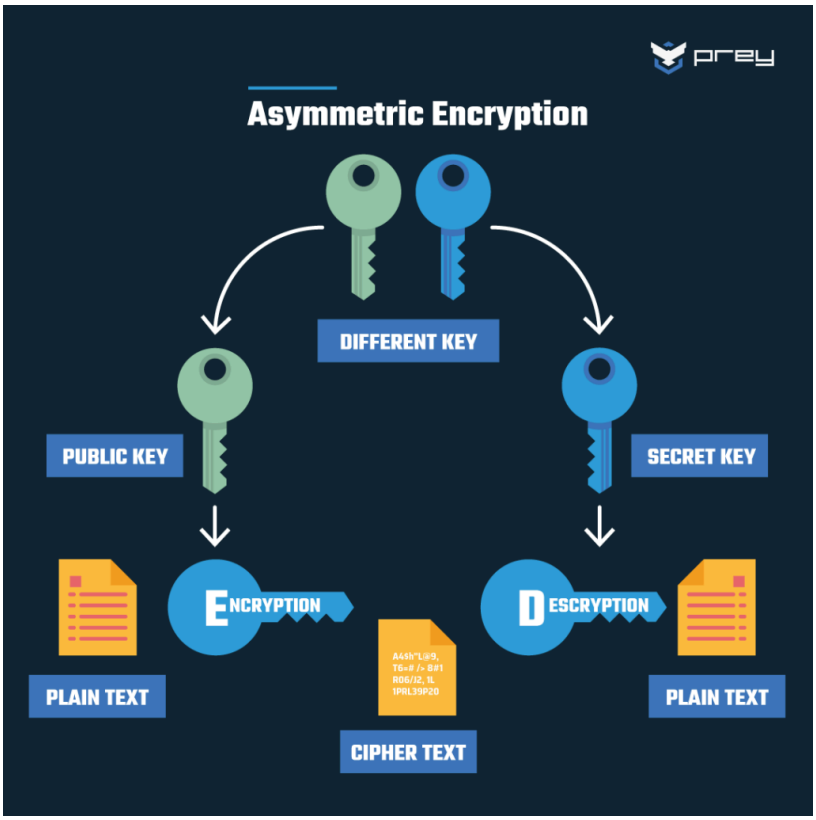
2. SYMMETRIC ENCRYPTION :

- Only one key used
- Parties us same secret key to encrypt and decrypt data
- It is essential to securely share and protect this key.
- Uses algorithms like - AES, DES, blowfish
- Using single key makes the process straight forward
- Faster performance and low resource consumption
- Inherently older and less secure as if you scale your encryption to wide scale then you're putting your trust into one single key.
- Real life ex - securing messaging apps(whatsapp), file encryption(veracrypt), bitlocker.



3. ASYMMETRIC ENCRYPTION :

- Uses a pair of key public (for encryption) and private (for decryption)
- Made to solve the issue of symmetric encryption
- Cornerstone of public key infrastructure - an encryption technique that requires 02 keys one to lock the plain text and other to unlock the ciphertext. Neither key performs both the functions.
- Public key : available to anyone who needs to encrypt a piece of information
- Private key : only held by an actor who decrypts the information without sacrificing security as you scale security.
- Real life ex - secure email communication ((PGP/MIME), digital signatures for document authenticity and integrity (adobesign/ docusign)
- Secure key exchange and establishment in SSL/TLS protocol for secure websites.



- **Hash functions (SHA-256, SHA-3, BLAKE3):**

  - A hash function transforms one numerical input value into another compressed numerical value.
  - It is an equation that is widely used to verify validity of data.
  - Translates data of various lengths or messages into a fixed size numerical string (hash).
  - It is a single-directional work, making it extremely difficult to reverse to recreate the info used to make it.

  - WORKING :

    - Accepts data of fixed length. Data blocks vary between algorithms.
    - If the block is too small, padding may be used.
    - However, the output or hash value always has the same set length.
    - Hash function is applied many times as the number of data blocks.

  1. SHA-256:
     - Secure Hash Algorithm
     - Has multiple families: SHA-0, SHA-1, SHA-2 & SHA-3
     - Part of SHA-2 family of Algo.
     - 256 stands for final hash digest value i.e., irrespective of the size of plaintext, the hash value will always be 256 bits.

     - Characteristics:
       1. Message length – should be less than 2^{64} bits. Size needs to be in the comparison area to keep digest as random as possible.
       2. Digest length – length of hash digest should be 256 bits. Digest outputs usually suggest significant mathematical calculations at the cost of speed & space.
       3. Irreversible:You should not get a plaintext when you have the digest beforehand nor should the digest provide its original value when you pass it through the hash function again.

     - STEPS :
       - Can divide in 05 steps:
         i) Padding bits -
         - Add '1' bit at end of message such that data is padded to the exact size of number of bits.

ii) Padding length -

- Padding length is added to the message to make the final plaintext as multiple of 512 bits by applying modulus 512.
- Original msg + padding bits + padding length = final data to be handled as multiple of 512.

iii) Initialising the buffers -

- When using buffer, digest result for 8 buffers to be used in hexadecimal are as follows:
- A = 0x6a09e667
- B = 0xbb67ae85
- C = 0x3c6ef372
- D = 0xa54ff53a
- E = 0x510e527f
- F = 0x9b05688c
- G = 0x1f83d9ab
- H = 0x5be0cd19
- These used as form of addition buffer input every round (K\[0] to K\[63])

iv) Compression function -

- During each iteration, the final output of the base is saved with the newer block, this block after padding the first block is hashed and the final output of the previous block works as input for the next block (chaining of output).

v) Output

2. <u>SHA-3:</u>

- Member of SHA-2 family
- Developed by NIST
- Part of bigger group of sponge constructions
- Works in a unique way called sponge construction
- Can absorb any amount of input data
- Squeezes out the output data like a sponge
- Goal: better output storage & better encoding

- WORKING :
  1. Uses sponge construction where:
     - Input string is divided into multiple blocks.
     - The padded string is XORed in successive blocks.
     - Each block is XORed with the extended state string to which it is padded.
  2. Uses a permutation function:
     - A fixed length state string is divided as a string of 1600 bits.
     - Applied to the result to get the required output.

3. BLAKE3
   - Cryptographic hash function
   - Takes input & creates a fixed-size output
   - Advantages:
     - Very quick to calculate
     - Based on strong model
     - Output by default is variable length
   - Internals of BLAKE3:
     - Tree-based
     - Uses Merkle tree (when we split input into chunks)
     - Each chunk is processed in parallel → ideal for parallel processing
     - Final hash taken as root hash of tree → unique fingerprint (hash) of original input
     -

- **Digital signatures and authentication Key derivation functions (PBKDF2, HKDF)**

- DIGITAL SIGNATURES :
- Mathematical technique used to validate the authenticity and integrity of a message, software or digital document.

- Key features :
  1. Key generation algorithms
  2. Signing algorithms : -
     - Like email programs create a one way hash of the electronic data which is to be signed.
     - Then this encrypts the hash value using private key which along with other information is the digital signature
  3. Signature verification algorithm

- Working :
  1. message digest is computed using hash function on the message and message digest is encrypted using private key of sender to form digital signature
  2. Message + digital signature is then transmitted
  3. Receiver decrypts using the public key of sender
  4. Receiver now has the message digest
  5. Message digest computed by the receiver and the message digest need to be the same for integrity

  - Note : message digest is computed using a one way hash function.

  - KEY DERIVATION FUNCTION :
  - transforms an input like a password or secret key into one or more cryptographically strong keys.
  - KDFs are essential for:
    1. Strengthening weak inputs like passwords
    2. Generating multiple keys from one secret
    3. Protecting against brute-force or rainbow table attacks

1. PBKDF2 (Password-Based Key Derivation Function 2)

   - PBKDF2 is used to derive secure keys from user passwords.
   - Commonly used in login systems, disk encryption, password managers, WPA2/WPA3.
   - Defined in RFC 8018 (PKCS #5).
   - Uses HMAC with a chosen hash function (e.g., HMAC-SHA256).
   - Adds randomness using a salt to prevent precomputed attacks (rainbow tables).
   - Applies a high number of iterations (e.g., 100,000 to 1,000,000) to slow down brute-force attacks.
   - Can produce output of any desired length (e.g., 128-bit, 256-bit, or more).

   - Parameters:
     1. Password (input string)
     2. Salt (random bytes, unique per password)
     3. Iterations (number of times hashing is repeated)

4. Key length (output size)
5. Hash function (e.g., SHA-256 or SHA-512)

- Security notes:
    1. Each user should have a unique salt
    2. High iteration counts are important for modern standards
    3. The result should be stored, not the password itself

2. <u>HKDF (HMAC-based Key Derivation Function)</u>

- HKDF is used when you already have a strong shared secret and want to derive one or more secure keys.
- Commonly used in secure communication protocols like TLS 1.3, Signal, WhatsApp, and Noise framework.
- Defined in RFC 5869.
- It is not meant for password hashing.
- Works in two phases: Extract and Expand.
    1. Extract: removes bias from input key material by applying HMAC with a salt
    2. Expand: uses the extracted key (PRK) to generate multiple output keys using HMAC

- Efficient and fast; suitable for real-time secure communication.

- Parameters:
    1. Input Keying Material (IKM): initial secret
    2. Salt: optional, improves security (recommended)
    3. Info: optional, context-specific label for output keys
    4. Output Length: number of bytes to derive

- Info parameter allows labeling of keys (e.g., encryption key, MAC key)
- Can derive multiple independent keys from the same input
- Suitable for systems requiring multiple keys from a single secret (key splitting)

- **Entropy and randomness**
  - Entropy in cryptography refers to the measure of randomness or unpredictability in a cryptographic system.
  - Determines the strength of cryptographic keys ensuring that they are resistant to attacks
  - High entropy indicates high level of unpredictability making it difficult for the attackers to predict or reproduce cryptographic keys
  - It can be sourced from various inputs which contribute to randomness including :
    1. Hardware random number
    2. Operating system sources
    3. User input
  - Measured in bits , higher the bits higher the security and randomness
  - Used in key generation , password hashing , digital signatures , secure communications etc.

2. PQC Fundamentals:

- **Quantum computing basics (qubits, superposition, entanglement)**

1. Qbits :

- Are like bits but can be zero and one both at the same time meaning quantum computers can do many things at once and work much faster than normal computers.
- Quantum computers use a unit called a qubit to process information. A qubit is similar to a bit, but it has unique quantum properties such as superposition and entanglement.
- qubits can be represented by various physical systems, such as electrons with spin, photons with polarization, trapped ions, and semiconducting circuits. With the ability to perform complex operations exponentially faster.

2. Superposition :

  - A quantum computer consisting n qubits can exist in a superposition of 2^n states
  - the famous Schrödinger's cat thought experiment, in which physicist Erwin Schrödinger imagined placing a cat in a sealed box along with a poisonous substance that has an equal chance of killing the cat or not within an hour. Schrödinger proposed that, at the end of the hour, the cat could be said to be both alive and dead, in a superposition of states until the box is opened, and that the act of observation randomly determines whether the cat is alive or dead.

- superposition can be thought of as an equation that has more than one solution.

3. Entanglement :
    - Entanglement is when multiple objects such as a pair of electrons or photons share a single quantum state. Like threads in a tangle of yarn, entangled objects cannot be described as independent entities.
    - refers to the interconnectedness of two or more qubits, where the quantum state of one qubit is directly linked to the state of the others, even if they are physically separated
    - Enables quantum computers to perform complex calculations faster and more efficiently than classical computers
    - Two qubits can become entangled, meaning the state of one affects the other, even if far apart.
    - Entangled qubits can represent complex relationships and help in powerful quantum computations.

- **Shor's algorithm (breaks RSA/ECC) :**

    - Prime factorization is a hard problem that underpins encryption like RSA.
    - Large numbers (e.g., 100-digit) take years to factor on classical computers.
    - Shor’s algorithm can factor large numbers efficiently using quantum computers.
    - It runs in polynomial time, making it much faster than classical methods.
    - Requires large quantum registers: 2048 and 1024 qubits working in entangled states.
    - Can factor a 1024-bit number in seconds on a 100 MIPS quantum computer.
    - Not effective on classical machines—other classical algorithms perform better.
    - So far, quantum computers have only factored small numbers (e.g., 21).
    - Can break RSA and similar encryption schemes if scaled successfully.
    - Drives the development of post-quantum cryptography.
    - Also useful for solving the period-finding problem.
    - WORKING :
        1. Choose a number to factor: Select a large number N that you want to factor. This number should be a product of two large prime numbers, p and q (N = p * q).
        2. Find the period: Shor's algorithm uses a quantum computer to find the period (r) of a function f(x) = a^x mod N, where a is a random number coprime to N (i.e., gcd(a, N) = 1). The period r is the smallest positive integer such that a^r ≡ 1 (mod N).

3. Quantum Fourier Transform (QFT): Apply a Quantum Fourier Transform to the quantum register containing the values of x. This transforms the register into a superposition of states representing the periodic function.
4. Measure the register: Measure the quantum register, which collapses the superposition into a single state. This measurement yields a value that's likely to be close to a multiple of the period r.
5. Classical post-processing: Use classical computing to find the period r from the measured value. This involves finding the greatest common divisor (GCD) of the measured value and N.
6. Find the factors: Once the period r is determined, use it to find the factors p and q of N. If r is even, compute gcd(a^(r/2) + 1, N) and gcd(a^(r/2) - 1, N) to obtain the factors p and q.

- **Grover's algorithm (weakens symmetric crypto) :**

- It's used for searching an unsorted database or solving problems where you are trying to find one specific solution among many possible ones.
- In simple terms:
- If you have a lock with 1000 combinations and no hints, a classical computer tries 500 on average to find the right one.
  Grover's algorithm can do it in about $\sqrt{1000} \approx 32$ tries using quantum computing.
- It gives a quadratic speedup for brute-force search problems.
- This includes breaking symmetric key encryption (like AES) and hash functions (like SHA-256) via brute force.
- It doesn't completely break symmetric crypto, but it makes them twice as weak in key length terms.
- WORKING : -

  - Imagine a black box (oracle) that marks the correct solution among N possible options.

  - Step 1: Superposition

  Start with n qubits in state $|0\rangle$.
  Apply the Hadamard gate to each qubit to create a superposition of all N states.
  So, instead of checking one option at a time, the quantum system is in all states at once.

  - Step 2: Oracle (Mark the Solution)

Use an oracle (a function) that flips the sign of the amplitude of the correct state.
Think of it as: "Hey, this is the correct answer!" — it inverts the amplitude of the correct solution from + to −.

-Step 3: Amplification (Grover Diffusion Operator)

Apply a special operation that amplifies the probability of the marked state (the correct answer) and reduces others.
This is done by reflecting all amplitudes about the average.
This step makes the "right" answer more likely to be measured.

- Step 4: Repeat

Repeat the oracle + amplification steps about $\sqrt{N}$ times (e.g., ~1000 items → 32 iterations).

- Step 5: Measure

Finally, measure the quantum state — the result will be the correct solution with high probability.

- **NIST PQC standardization process :**

  - Launched in 2016 by the U.S. National Institute of Standards and Technology (NIST).

  - Aimed at developing quantum-resistant cryptographic algorithms to replace RSA, ECC, and DSA.

  - Received 69 algorithm submissions from researchers globally.

  - After years of review and analysis, NIST selected finalists and announced standard candidates in July 2022.

  - Kyber (ML-KEM) was selected for public key encryption and key encapsulation.

- Dilithium, Falcon, and SPHINCS+ were selected for digital signatures.

- The process ensures that the selected algorithms are not only secure against quantum attacks but also efficient, scalable, and implementable in real-world systems.

- **Lattice-based cryptography (ML-KEM/Kyber) :**

  - ML-KEM is designed to withstand attacks from both classical and quantum computers, making it a promising solution for post-quantum cryptography. This means that even if a large-scale quantum computer is built, ML-KEM's encryption would remain secure.

  - used to securely establish a shared secret key between two parties over an insecure channel. Think of it like securely exchanging a secret code between two people, without anyone else being able to intercept it.

  - Based on Lattice Problems - its security relies on complex mathematical problems related to lattices, specifically the Module Learning with Errors (MLWE) problem. These problems are difficult for both classical and quantum computers to solve, ensuring the security of the encryption.

  - Has been standardized as FIPS 203 by NIST, which means it's been vetted and approved by a reputable standards organization. This standardization helps ensure its security and interoperability.

  - offers relatively small key sizes and high security, making it efficient for various applications. Its performance is also better than some other quantum-resistant algorithms in certain scenarios.

  - While ML-KEM provides quantum resistance, it may have trade-offs in terms of computational complexity and key sizes compared to other algorithms like RSA. This means that ML-KEM might require more computational power or have larger key sizes than some other encryption algorithms.

  - ML-KEM is being integrated into various libraries and systems for secure communication, including chat encryption. This means it's being used in real-world applications to provide secure communication, and its use cases will likely expand as post-quantum cryptography becomes more widespread.

3. For Brahmi:

Unicode block U+11000 to U+1107F

Character mapping and transliteration

@~Anoushka Shrivastav and @~Aaditya look at Statistical properties for cryptographic use

4. Mathematics:

- **Number Theory Transform (NTT)**
  - NTT is a fast algorithm for multiplying large polynomials and integers using properties of modular arithmetic. It's similar to the Fast Fourier Transform (FFT), but works with modular arithmetic instead of complex numbers.
  - The key idea is to break down complex calculations into smaller, more manageable pieces using modular arithmetic.
  - WORKING :
    - Step 1: Choose a prime number q and a primitive root ω (omega) modulo q. This sets up the modular arithmetic system.
    - Step 2: Map the polynomial coefficients to a new domain using ω. This is called the "evaluation" step.
    - Step 3: Perform calculations (like multiplication) in the new domain. This is where the magic happens, and NTT's efficiency shines.
    - Step 4: Map the results back to the original domain. This is called the "interpolation" step.
  - Much faster than traditional multiplication methods for large polynomials and integers. This makes it useful in cryptographic applications, such as lattice-based cryptography.
  - It is used in various cryptographic schemes, including digital signatures and key exchange protocols.

- **Polynomial arithmetic :**

  - Polynomial arithmetic is a fundamental concept in mathematics and computer science, and it plays a crucial role in various cryptographic applications.

  - the basic operations (addition, subtraction, multiplication, division) on polynomials. Polynomials are expressions consisting of variables and coefficients combined using mathematical operations.

  - Addition: Combine like terms by adding coefficients.
    - Example: $(2x + 3) + (4x + 5) = 6x + 8$
  - Subtraction: Combine like terms by subtracting coefficients.
    - Example: $(2x + 3) - (4x + 5) = -2x - 2$
  - Multiplication: Use the distributive property and combine like terms.
    - Example: $(2x + 3) \times (4x + 5) = 8x^2 + 10x + 12x + 15 = 8x^2 + 22x + 15$
  - Division: Use long division or synthetic division to divide polynomials.
    - Example: $(x^2 + 4x + 4) \div (x + 2) = x + 2$

  - Polynomial arithmetic is a fundamental building block for many cryptographic schemes, including lattice-based cryptography and homomorphic encryption.

- **Lattice problems (LWE, Ring-LWE)**
  - Lattice problems are mathematical problems based on lattices, which are regular arrangements of points in n-dimensional space. These problems are thought to be hard to solve, making them suitable for cryptography.

  - LWE (Learning With Errors):
    - Problem statement: Given a set of noisy linear equations, find the secret solution. Think of it like trying to solve a system of equations with some errors thrown in.
    - Imagine you have a system of linear equations, but some of the equations have errors in them. Your task is to find the secret solution despite these errors.
    - Given a set of samples (a_i, b_i) where a_i is a vector and b_i is a value, find a secret vector s such that $b_i \approx \langle a_i, s \rangle$ (dot product) for most samples. The $\approx$ symbol indicates that there might be some errors.

    - Hardness assumption: LWE is thought to be hard to solve, which makes it suitable for cryptography. This hardness assumption is based on the difficulty of finding short vectors in lattices.
    - LWE is used in various cryptographic schemes, including public-key encryption, key exchange, and digital signatures. Its hardness assumption provides a strong security guarantee.

- Ring-LWE:
    - Problem statement: Similar to LWE, but with additional structure using polynomial rings. This adds an extra layer of complexity and security.
    - Given a set of samples (a_i, b_i) where a_i and b_i are polynomials, find a secret polynomial s such that b_i ≈ a_i * s (polynomial multiplication) for most samples. The ≈ symbol indicates that there might be some errors.
    - Benefits: Ring-LWE is more efficient and compact than LWE, while maintaining security. This makes it attractive for cryptographic applications.
    - Ring-LWE is used in various cryptographic schemes, including public-key encryption, key exchange, and digital signatures. Its efficiency and compactness make it suitable for applications where performance is critical.

- Key differences between both are that :
    - LWE uses vectors, while Ring-LWE uses polynomials.
    - Ring-LWE is more efficient than LWE due to the use of polynomial rings.
    - Ring-LWE is more compact than LWE due to the use of polynomials.

- Lattice problems like LWE and Ring-LWE are used in cryptographic schemes, such as key exchange and encryption. These schemes are thought to be resistant to quantum computer attacks.

- **Error correction codes:**
  - Error correction codes are methods to detect and correct errors in digital data. These codes add redundancy to the data, allowing errors to be detected and corrected.
  - Key features :
    - Encoding: Add redundancy to data to enable error detection and correction. This is like adding extra information to help recover the original data.
    - Decoding: Detecting and correcting errors using redundant information. This is like using the extra information to fix the errors.
  - Types of error correction codes :
    - Block codes (e.g., Reed-Solomon codes): These codes work on blocks of data and can correct errors within those blocks.
    - Convolutional codes: These codes work on streams of data and can correct errors that occur during transmission.
  - Error correction codes are crucial in many applications, including data storage, communication systems, and cryptography.

- **Cryptographic reductions :**
  - Techniques to prove the security of a cryptographic scheme by reducing it to a well-established hard problem. This shows that breaking the scheme is at least as hard as solving the underlying problem.
  - A "hard problem" refers to a mathematical problem that is believed to be difficult or infeasible to solve in a reasonable amount of time, even with the most advanced computers and algorithms.
  - Some examples of hard problems used in cryptography include:
    1. Factoring large numbers: Given a large composite number, find its prime factors. This problem is the basis for RSA cryptography.
    2. Discrete logarithm problem: Given a number and its logarithm modulo a prime, find the logarithm. This problem is the basis for Diffie-Hellman key exchange and other cryptographic schemes.
  - The key idea is that if an attacker can break the cryptographic scheme, they can also solve the underlying hard problem. This provides confidence in the security of the scheme.
  - Types of reductions:
    - Direct reduction: Reduce the cryptographic scheme directly to the hard problem. This is like showing that breaking the scheme is equivalent to solving the hard problem.
    - Hybrid reduction: Use a combination of reductions to prove security. This is like using multiple steps to show that breaking the scheme is hard.

- Cryptographic reductions provide a rigorous way to establish the security of cryptographic schemes and ensure confidence in their use.

5. System Architecture:

- **Hybrid cryptographic systems:**
  - Hybrid cryptographic systems combine the strengths of symmetric and asymmetric encryption to provide both confidentiality and authenticity. These systems use symmetric encryption for bulk data encryption and asymmetric encryption for key exchange and digital signatures.
  - WORKING :
    - Key Exchange: Asymmetric encryption is used to exchange a shared secret key between two parties.
    - Symmetric Encryption: The shared secret key is used for symmetric encryption, which encrypts the bulk data.
    - Digital Signatures: Asymmetric encryption is used to create digital signatures, which provide authenticity and non-repudiation.
  - PGP (Pretty Good Privacy): PGP is a hybrid cryptographic system that uses symmetric encryption for bulk data encryption and asymmetric encryption for key exchange and digital signatures.
  - SSL/TLS (Secure Sockets Layer/Transport Layer Security): SSL/TLS is a hybrid cryptographic system that uses symmetric encryption for bulk data encryption and asymmetric encryption for key exchange and digital signatures.

- **Key management and rotation :**
  - process of generating, distributing, storing, and revoking cryptographic keys. Key rotation is an essential aspect of key management, which involves regularly changing or updating cryptographic keys to minimize the impact of a key compromise.
  - Generate keys using secure random number generators.
  - Store keys in a secure manner, such as using a hardware security module
  - Rotate keys regularly to minimize the impact of a key compromise.
  - Revoke keys that are no longer secure or have been compromised.
  - Proper key management and rotation are crucial for maintaining the security of cryptographic systems.
  -

- **Secure random number generation :**
    - Secure random number generators use physical phenomena or algorithms to generate truly random and unpredictable numbers.
    - Types of Random Number Generators:
        - Hardware Random Number Generators: Use physical phenomena, such as thermal noise or radioactive decay, to generate truly random numbers.
        - Software Random Number Generators: Use algorithms, such as Fortuna or Yarrow, to generate pseudorandom numbers.
    - Secure random number generation is essential for generating keys and other cryptographic parameters that are resistant to attacks.

- **Side-channel attack mitigation :**
    - Side-channel attacks exploit information about the implementation of a cryptographic system, such as timing or power consumption, to compromise the security of the system.
    - crucial for protecting cryptographic systems from attacks that exploit implementation weaknesses. By implementing mitigation techniques and following best practices, you can significantly reduce the risk of side-channel attacks.
    - Side-channel attacks exploit information about the implementation of a cryptographic system, rather than attacking the algorithm itself. This information can include:
        1. Timing: Measuring the time it takes to perform cryptographic operations.
        2. Power consumption: Analyzing the power consumption patterns of a device during cryptographic operations.
        3. Electromagnetic radiation: Measuring the electromagnetic radiation emitted by a device during cryptographic operations.
        4. Cache behavior: Analyzing the cache behavior of a system during cryptographic operations.
    - Types of Side-Channel Attacks
        1. Timing attacks: Exploiting differences in execution time to recover sensitive information.
        2. Power analysis attacks: Analyzing power consumption patterns to recover sensitive information.
        3. Electromagnetic analysis attacks: Analyzing electromagnetic radiation to recover sensitive information.
        4. Cache-based attacks: Exploiting cache behavior to recover sensitive information.

- Mitigation Techniques
    1. Constant-time implementation: Implementing cryptographic algorithms in a way that takes constant time, regardless of the input.
    2. Masking: Using masking techniques to hide sensitive data and prevent attackers from exploiting side-channel information.
    3. Secure coding practices: Using secure coding practices, such as secure coding guidelines and code reviews, to prevent side-channel vulnerabilities.
    4. Shielding: Using physical shielding to prevent electromagnetic radiation attacks.
    5. Noise generation: Generating noise to mask side-channel information.

- **Performance optimization :**
    - Performance optimization is essential for improving the efficiency and speed of cryptographic systems. Techniques include:
        1. Caching: Use caching to store frequently used cryptographic parameters or intermediate results.
        2. Parallel Processing: Use parallel processing to take advantage of multi-core processors and improve performance.
        3. Optimized Algorithms: Use optimized algorithms, such as optimized implementations of AES or elliptic curve cryptography.
        4. Importance: Performance optimization is important for improving the efficiency and usability of cryptographic systems, especially in high-performance applications.