Basic Crypto Primitive

- A Hash Function is cryptographically secure if it
- satisfies the following 3 security properties:
- Property 1: Collision Resistance
- Property 2: Hiding
- Property 3: “Puzzle Friendliness

Puzzle-Friendly Hash Functions

- Definition: A hash function is puzzle-friendly if it is hard to compute in one direction (finding the original input) but easy to verify in the other direction.
- Importance: Puzzle friendliness is particularly relevant in scenarios where computational effort is required to solve puzzles or problems, such as in Proof-of-Work (PoW) systems. It ensures that participants need to invest significant computational resources to solve puzzles, adding a layer of security to the overall system.

- In applications such as blockchain and cryptocurrency, puzzle-friendly hash functions are commonly used in the process of mining. Miners need to find a solution to a computational puzzle by repeatedly hashing a block of data until they find a hash value that meets certain criteria (e.g., starts with a specific number of leading zeros). This requires significant computational power and effort.

- The puzzle-friendly nature ensures that participants cannot easily cheat by finding the solution without putting in the required computational work. However, once a solution is found, it is easy for others to verify its correctness by applying the hash function.

Collision-Resistant Hash Functions

- Definition: A hash function is collision-resistant if it is computationally infeasible to find two different inputs that produce the same hash value.
- Importance: Collision resistance ensures that it is difficult for an attacker to create two distinct inputs that map to the same hash, which is crucial for maintaining the integrity and security of various cryptographic applications.

Hiding

- Definition: The hiding property of a hash function ensures that the output (hash value) reveals no information about the input other than what is inherent in the hash itself. In other words, it should be computationally infeasible to determine the input from its hash.
- Importance: Hiding prevents attackers from gaining any knowledge about the input data by analyzing the hash output. It contributes to the confidentiality and security of cryptographic protocols.

- These three properties collectively contribute to the overall security of a hash function. A cryptographically secure hash function should resist collisions, hide input information effectively, and exhibit puzzle friendliness to prevent manipulation or cheating in computational tasks. Many widely used cryptographic hash functions, such as SHA-256 (Secure Hash Algorithm 256-bit), are designed with these properties in mind to meet the security requirements of various applications in the field of cryptography.

Digital Signatures:

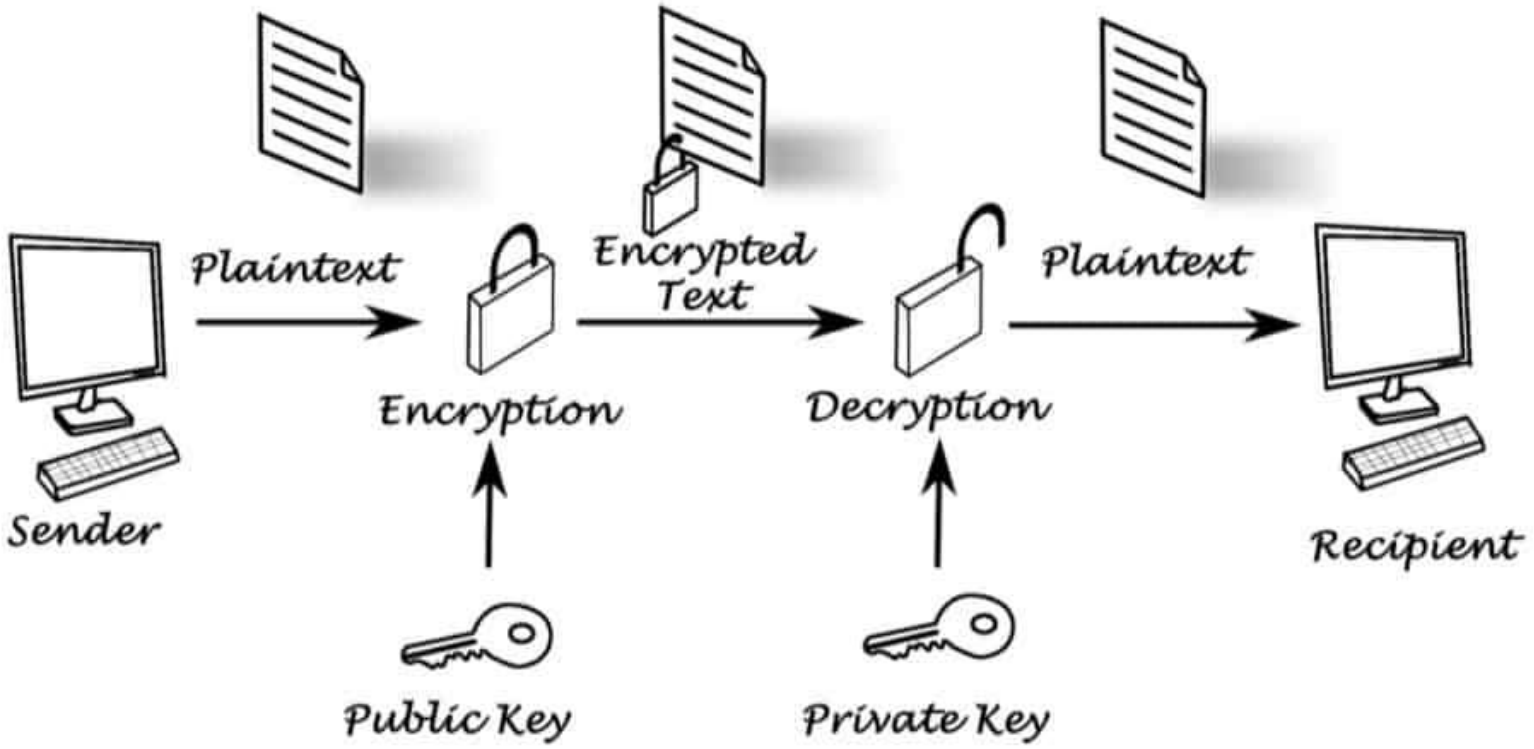
- Digital signatures provide a way for a person or entity to electronically sign a digital document to prove its authenticity and integrity.
- It involves using a private key to sign the document, and the recipient can verify the signature using the corresponding public key.

Source: https://techterms.com/definition/digitalsignature

Public Key Cryptography:

- Public key cryptography uses a pair of keys (public and private) for secure communication.
- The public key is shared openly, while the private key is kept secret. Messages encrypted with the public key can only be decrypted with the corresponding private key, and vice versa.

Public Key Cryptography



Verifiable Random Functions

- Verifiable Random Functions (VRFs) are cryptographic primitives that combine the properties of randomness and verifiability. Introduced by Mihir Bellare, Oded Goldreich, and Shafi Goldwasser in 1997, VRFs allow a party to generate a pseudorandom output based on a secret key and a public input. The important feature of VRFs is that the output is not only pseudorandom but also comes with a proof that can be publicly verified.
- They are useful in applications where a trusted third party generates random values, and participants can independently verify the randomness and correctness of the generated values.

- Randomness Generation:
- VRFs generate a pseudorandom output that appears indistinguishable from truly random values. The randomness is determined by both a secret key held by the owner of the VRF and a public input.

- Public Verifiability:
- The distinguishing feature of VRFs is their ability to produce a proof along with the pseudorandom output. This proof can be publicly verified by anyone, without access to the secret key. The verification process ensures that the output was indeed generated by a valid VRF with the corresponding secret key and public input.

- Deterministic Nature:
- While VRFs produce pseudorandom outputs, they are deterministic given the same secret key and public input. This means that the same input parameters will always produce the same pseudorandom output.

- Unforgeability:
- An essential property of VRFs is their resistance to forgery. Without the secret key, it should be computationally infeasible for an adversary to generate a valid VRF output along with a verifiable proof.

Applications:

- VRFs find applications in various cryptographic protocols, including but not limited to secure key generation, randomness beacon construction, and proof-of-stake consensus algorithms. They provide a way to generate unpredictable values that can be publicly verified, adding a layer of trust to certain cryptographic processes.
- Example VRFs:
- The concept of VRFs has been realized in different cryptographic constructions. One example is the Elliptic Curve VRF (ECVRF), which uses elliptic curve cryptography to achieve the desired properties.

- ECC is an alternative to the Rivest-Shamir-Adleman (RSA) cryptographic algorithm and is most often used for digital signatures in cryptocurrencies, such as Bitcoin and Ethereum, as well as one-way encryption of emails, data and software.

Zero-Knowledge Systems

- Zero-knowledge proofs are cryptographic protocols that allow one party (the prover) to prove to another party (the verifier) that they possess certain information without revealing the information itself.
- This concept is crucial for maintaining privacy and security in various cryptographic applications.

Examples of Zero-Knowledge Proofs

- ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge): These are efficient zero-knowledge proofs used in blockchain and cryptocurrency systems, such as Zcash, to prove the validity of transactions without revealing any details about the transactions.

- ZK-STARKs (Zero-Knowledge Scalable Transparent ARguments of Knowledge): Another form of zero-knowledge proof designed to be transparent and scalable, used in applications where a large number of verifications are needed.

- Proof of Knowledge of a Secret (e.g., Password or Private Key): A user can prove to a system that they know a secret (like a password or private key) without actually disclosing that secret.

Tabular Difference between various cryptographic hash function

Puzzle-Friendly Hash	Collision-Resistant Hash	Verifiable Random Functions	Zero-Knowledge Systems
Used in puzzles and protocols where participants invest computational resources.	Ensure uniqueness of hash values, crucial for integrity in various cryptographic applications.	Generate pseudorandom output with proof, publicly verifiable.	Prove knowledge of information without revealing the information itself.
Cryptographic puzzles, Proof-of-Work in blockchain.	Digital signatures, Data integrity, Cryptographic hash functions.	Randomness beacon, Secure key generation, Consensus algorithms.	Authentication, Identity verification, Privacy-enhanced transactions.
Puzzle difficulty ensures resource-intensive solving.	Prevents collisions, maintains integrity and authenticity.	Provides publicly verifiable pseudorandomness.	Ensures information can be proven without being revealed.
Cryptographic puzzles with specific difficulty criteria.	SHA-256, SHA-3, MD5 (though not recommended for cryptographic use).	Elliptic Curve VRF (ECVRF), zk-SNARKs, zk-STARKs.	zk-SNARKs, zk-STARKs, and various mathematical proofs in protocols like ZKPs.