ZETECH UNIVERSITY

UNIT NAME: BLOCK CHAIN TECHNOLOGY

UNIT CODE: BCE 413

LESSON 2 <u>Hash algorithm and cryptography</u>

- What is hash algorithm and its objectives?
- Hash working mechanism
- asymmetric encryption vs symmetric encryption

The first concept we need to discuss in our exploration of Cryptography is that of a Hashing Algorithm.

Cryptography - *is the practice and study of techniques for securing communication and data in the presence of adversaries.* maintaining *data security, confidentiality* and *integrity.*

The three primary cryptographic algorithms are:

1. Hash function
2. Public key or asymmetric encryption
3. Secret key or symmetric encryption

What is hash algorithm and its objectives?

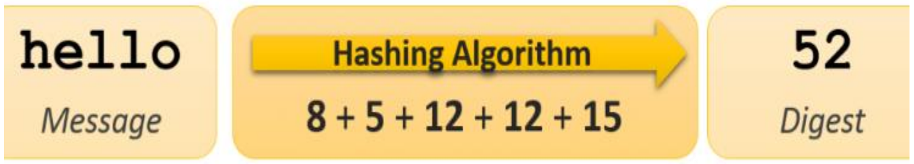
Hash definitions

Hashing Def 1

- *refers to the process of generating a fixed-size output from an input of variable size using the mathematical formulas known as hash functions. This technique determines <u>an index or location</u> for the storage of an item in a data structure.*
  - *the **Bitcoin** hash algorithm is SHA-256 or Secure Hashing Algorithm 256 bits.*

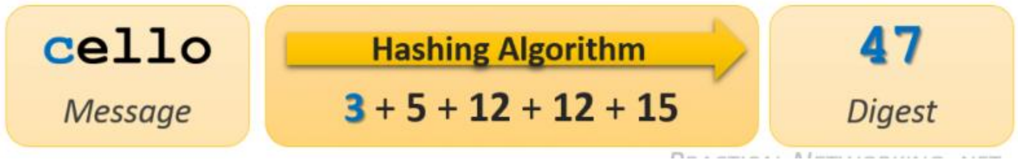
Hashing Def 2

- *A Hashing Algorithm is a **mathematical formula** that takes a Message of arbitrary length as input and produces as output a representational sample of the original data.*
  - ***For instance,** a rudimentary **<u>example</u>** of a hashing algorithm is simply <u>adding up all the letter values of a particular message</u>*



The result of a hashing algorithm is called a message Digest (or sometimes Checksum, or Fingerprint).

- If someone were to change our original message and process it through the same hashing algorithm, the result would be different:



A legitimate hashing algorithm must maintain four qualities before it is approved for industry usage: <u>Basic properties of hashing</u>

1. **It is mathematically impossible to extract the original message from the digest.**
    - *Hashing is sometimes referred to as one-way encryption: the message can be encrypted but is impossible to decrypt. This is accomplished using one-way functions within the hashing algorithm.*
        - *It is impossible to derive hello knowing only a resulting digest of 52. Mostly because there could be thousands of messages that result in the identical digest.*
2. **A slight change to the original message causes a drastic change in the resulting digest.**
    - *Hashing algorithm is not simply one calculation. It is a series of calculations, done iteratively over and over.*
3. **The result of the hashing algorithm is always the same length.**
    - *A digest should not grow in size as the length of the Message increases.*
4. **It is infeasible to construct a message which generates a given digest.**
    - *With our example hashing algorithm, if given the digest of 52, it would not be overly difficult to generate a list of words that might have been the original message.*

Digest Lengths

Below is a table with commonly seen, industry recognized hashing algorithms:

Algorithm	Digest Length
MD5	128 Bits
SHA or SHA1	160 Bits
SHA384	384 Bits
SHA256	256 Bits

- Each of these Hashing algorithms satisfy the four cryptography hashing algorithm properties, as described above. The primary difference between each of them is the size of the resulting digest.
- As with passwords, it is typically considered that a hashing algorithm which results in a *longer digest tends to be regarded as more secure.*

Data Integrity - How Hashing is used to ensure data isn't modified - Cryptography - Practical TLS video link

https://youtu.be/doN3lzzNEIM?list=PLIFyRwBY_4bTwRX__Zn4-letrtpSj1mzY&t=326

Need for Hash data structure

Hashing - *is an important method designed to solve the problem of efficiently finding and storing data in an array.*

- Hashing data structure known *for searching purpose*
- So now we are looking for a data structure that can **store** the data and search in it in constant time, i.e. in **O (1) time.**
  - This is how Hashing data structure came into play. With the introduction of the Hash data structure, it is now *possible to easily store data in constant time and retrieve them in constant time as well.*

Complexity of calculating hash value using the hash function

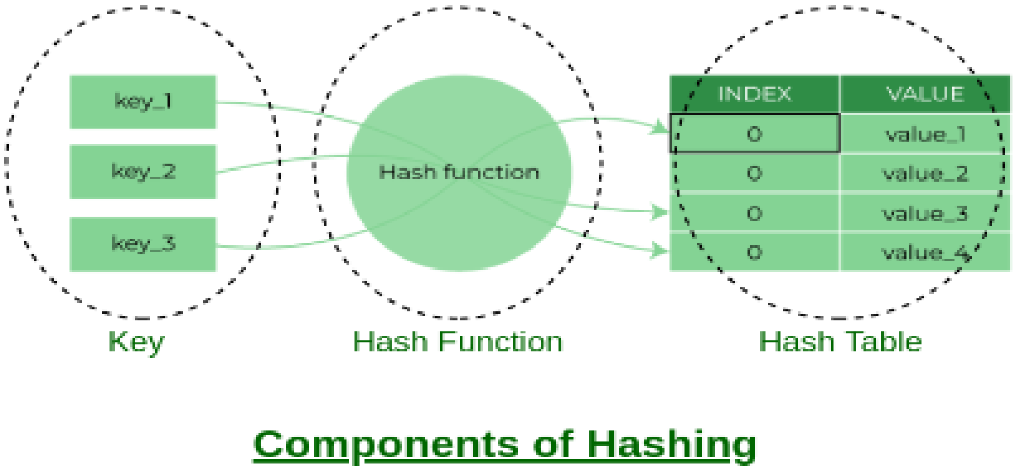
Time complexity: O(n)

Space complexity: O(1)

Components of Hashing

There are majorly three components of hashing:

1. **Key**: - *A Key can be anything string or integer which is fed as input in the hash function the technique that determines an index or location for storage of an item in a data structure.*
2. **Hash Function:** -*The hash function receives the input key and returns the index of an element in an array called a hash table. The index is known as the hash index.*
3. **Hash Table:** - *Hash table is a data structure that maps keys to values using a special function called a hash function. Hash stores the data in an associative manner in an array where each data value has its own unique index.*



Components of Hashing

<u>**There are three ways of calculating the hash function:**</u>

1. Division method
2. Folding method
3. Mid square method

In the division method, the hash function can be defined as:

h(ki) = ki % m;

where m is the size of the hash table.

For example, if the key value is 6 and the size of the hash table is 10. When we apply the hash function to key 6 then the index would be:

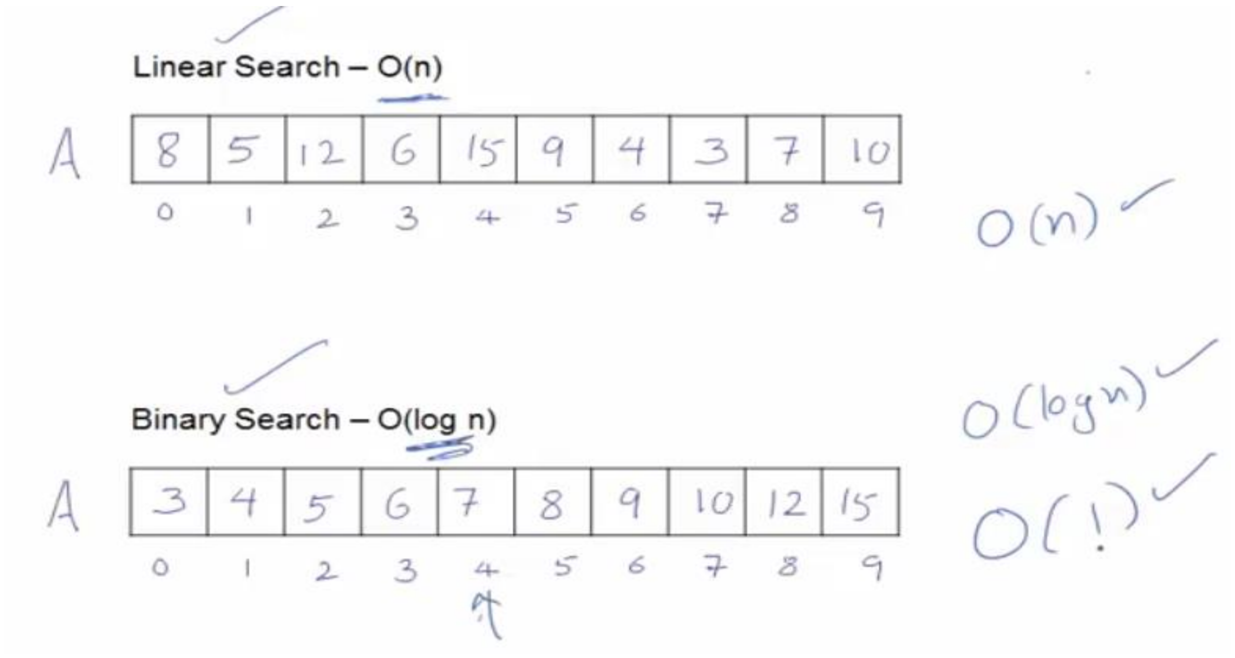
h(6) = 6%10 = 6

The index is 6 at which the value is stored.

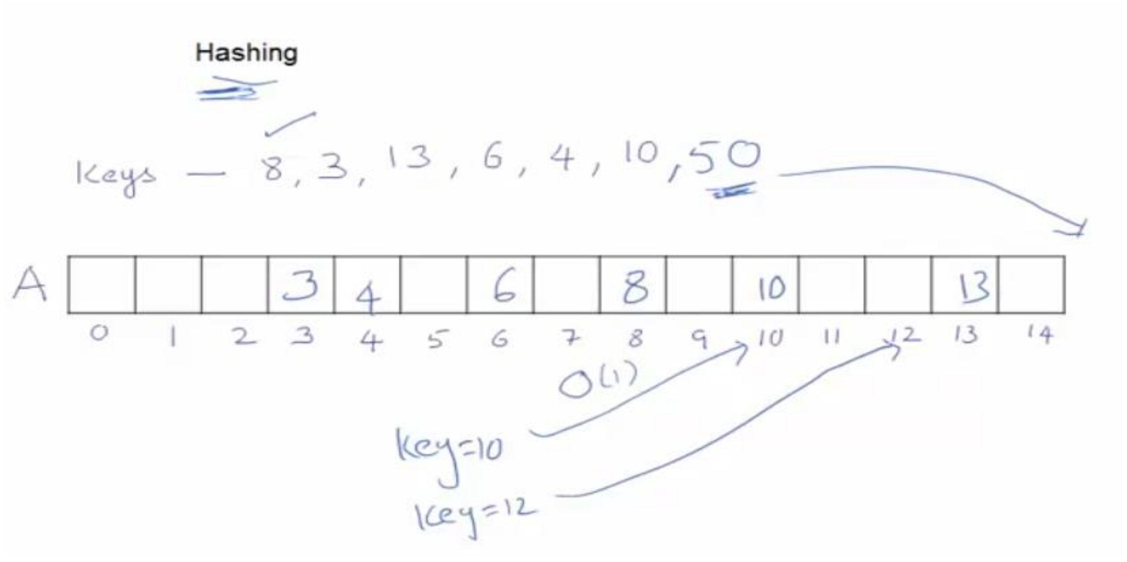
Working mechanism of Hash Algorithm

- Hashing data structure known *for searching purpose*

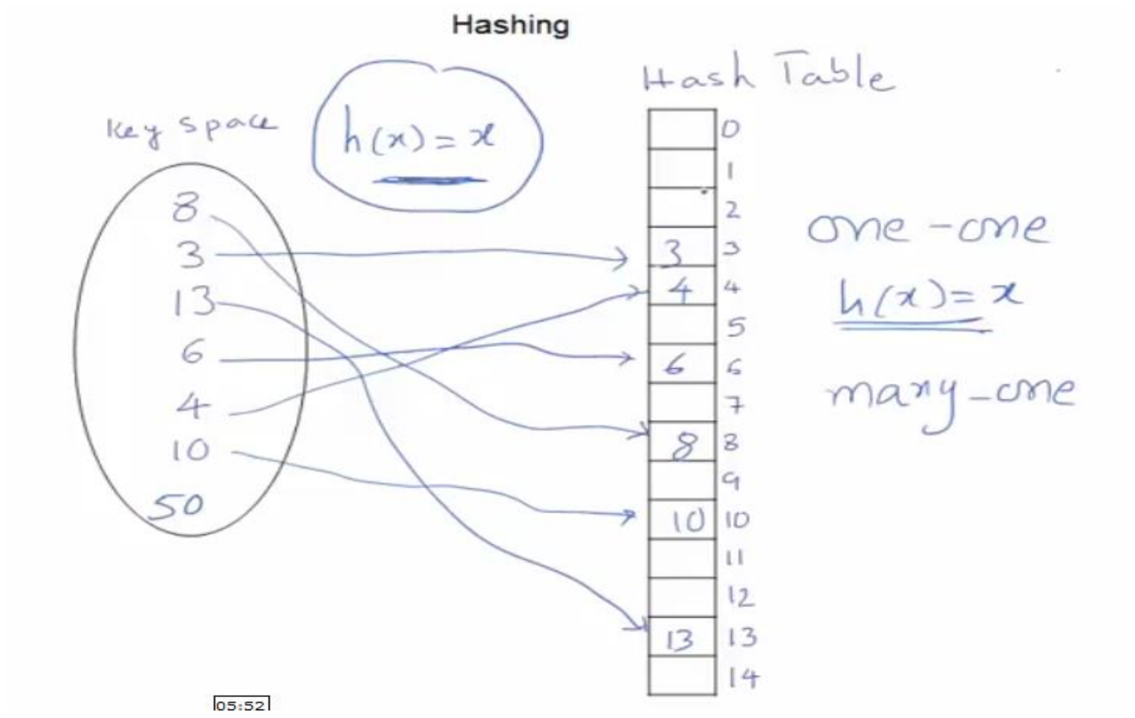
Two hashing search methods



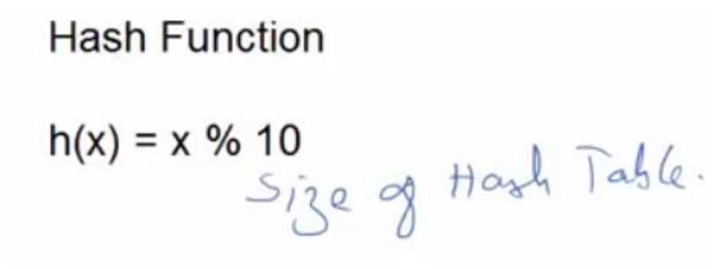
- So now we are looking for a data structure that can **store** the data and search in it in constant time, i.e. in **O (1) time. And that's harsh algorithm.**



The draw back in this method a lot free space in the memory so rectify that mathematical model called Hash function mapping is done.



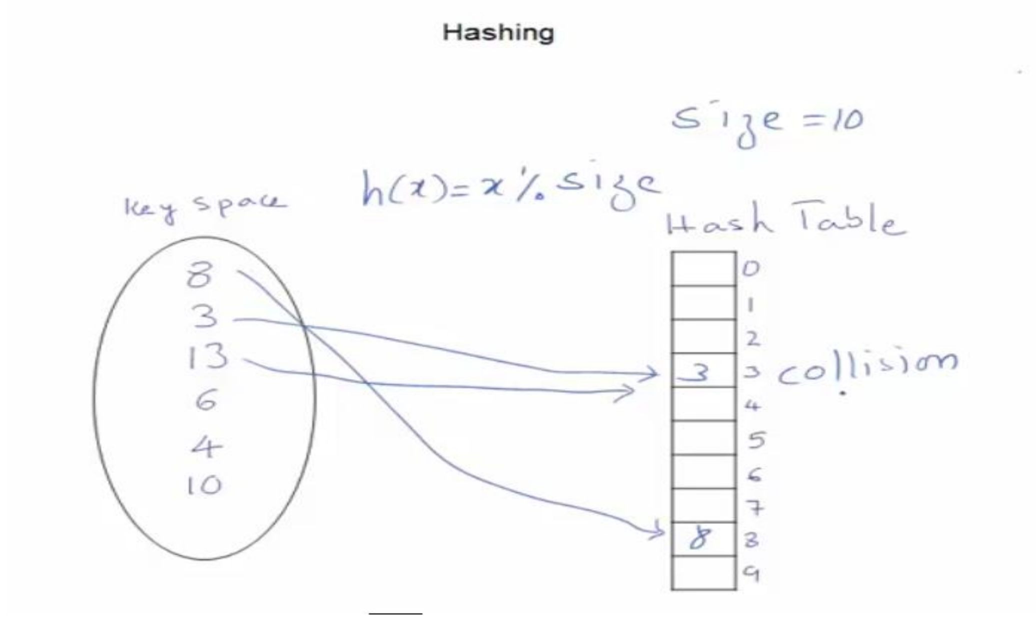
Next step: We again modify Hash function to efficiently utilize the space

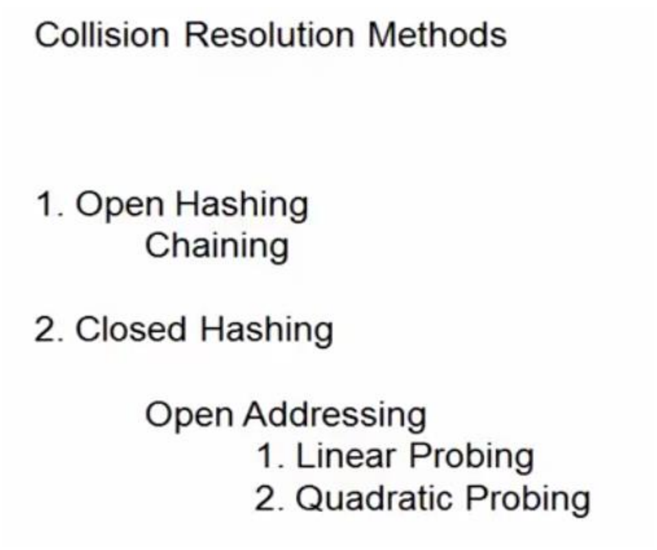


Hash Function

h(x) = x % 10

Size of Hash Table.





Collision Resolution Methods

1. Open Hashing
   Chaining

2. Closed Hashing

   Open Addressing
   1. Linear Probing
   2. Quadratic Probing

<u>The difference between the two hashing</u> (open and closed) has to do with whether collisions are stored outside the table (open hashing),

or whether collisions result in storing one of the records at another slot in the table (closed hashing).

The simplest form of open hashing defines each slot in the hash table to be the head of a linked list.

A collision occurs when two keys are hashed to the same index in a hash table.

- Collisions are a problem because every slot in a hash table is supposed to store a single element.

<u>The benefits of chaining</u>

1. Through chaining, insertion in a hash table always occurs in O(1) since linked lists allow insertion in constant time.
2. Theoretically, a chained hash table can grow infinitely as long as there is enough space.
3. A hash table which uses chaining will never need to be resized.

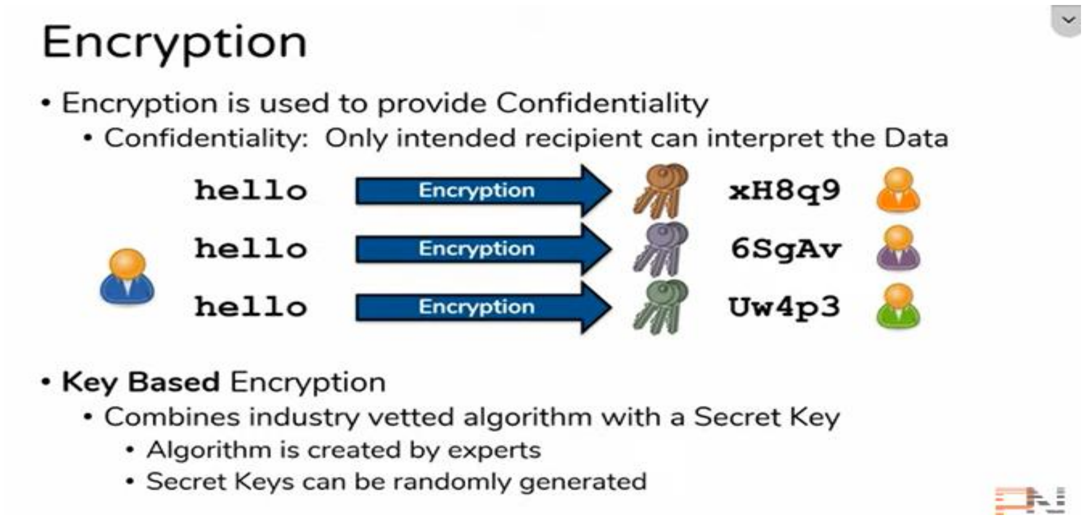
Hash chain video Link

https://www.youtube.com/watch?v=kF74yQLxulQ

Objectives of Hash Function

- Minimize collisions
- Uniform distribution of hash value
- Easy to calculate
- Resolve any collisions

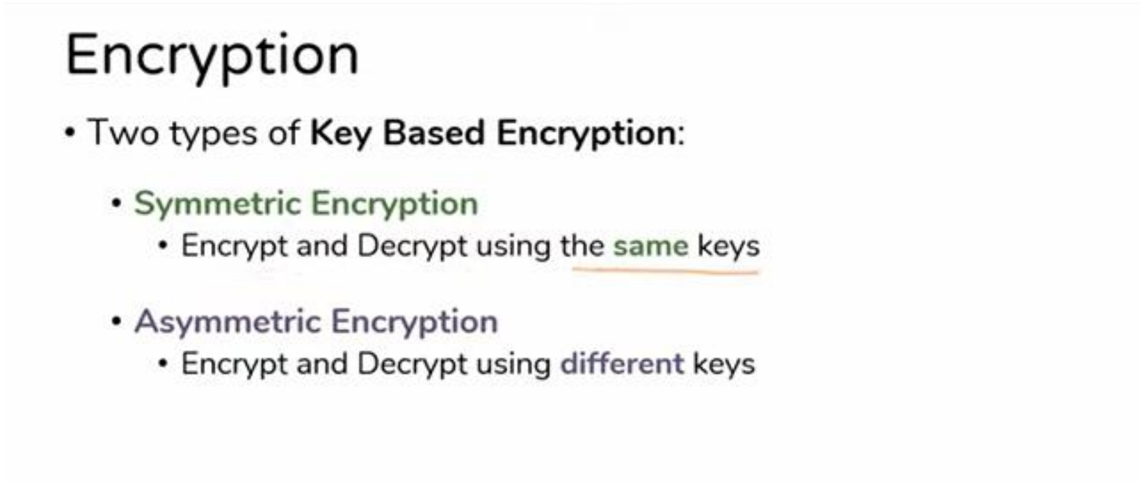
2. Encryption - Symmetric Encryption vs Asymmetric Encryption – Cryptography



Encryption

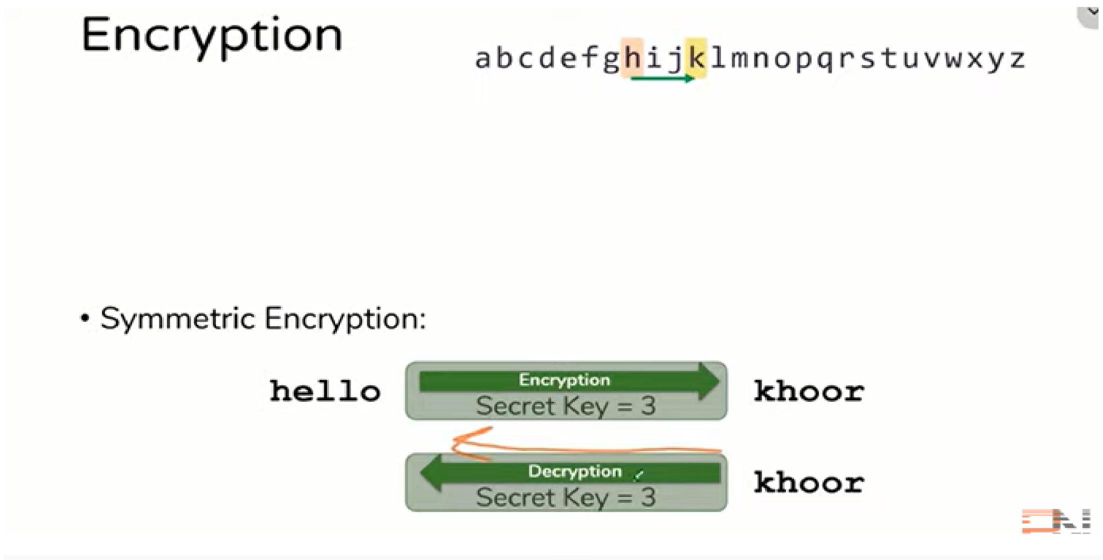
- Encryption is used to provide Confidentiality
  - Confidentiality: Only intended recipient can interpret the Data

- **Key Based** Encryption
  - Combines industry vetted algorithm with a Secret Key
    - Algorithm is created by experts
    - Secret Keys can be randomly generated



Encryption

- Two types of **Key Based Encryption**:
  - Symmetric Encryption
    - Encrypt and Decrypt using the same keys
  - Asymmetric Encryption
    - Encrypt and Decrypt using different keys



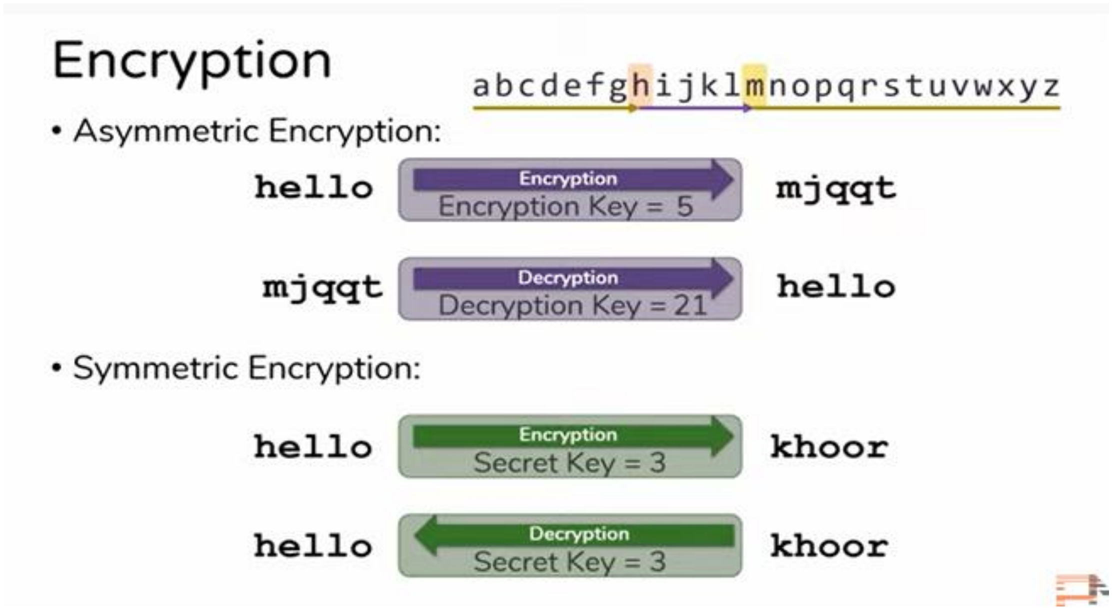
Encryption

abcdefghijklmnopqrstuvwxyz

- Symmetric Encryption:

hello → Encryption, Secret Key = 3 → khoor

khoor → Decryption, Secret Key = 3 →



Encryption

abcdefghijklmnopqrstuvwxyz

- Asymmetric Encryption:

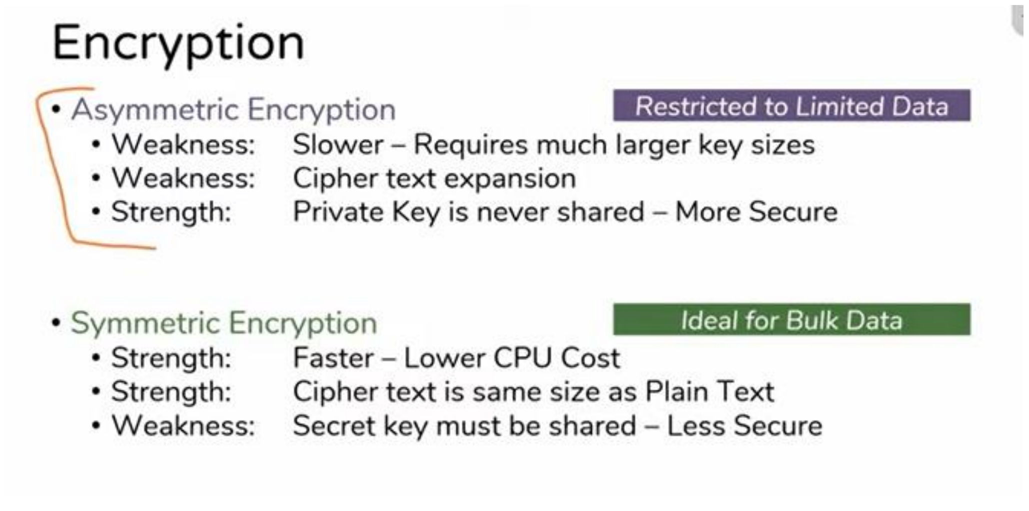
hello → Encryption, Encryption Key = 5 → mjqqt

mjqqt → Decryption, Decryption Key = 21 → hello

- Symmetric Encryption:

hello → Encryption, Secret Key = 3 → khoor

hello ← Decryption, Secret Key = 3 ← khoor



Encryption

- Asymmetric Encryption — Restricted to Limited Data
  - Weakness: Slower – Requires much larger key sizes
  - Weakness: Cipher text expansion
  - Strength: Private Key is never shared – More Secure

- Symmetric Encryption — Ideal for Bulk Data
  - Strength: Faster – Lower CPU Cost
  - Strength: Cipher text is same size as Plain Text
  - Weakness: Secret key must be shared – Less Secure

How to do cryptography in block chain

The goal of cryptography is to *ensure the security, authenticity, and integrity of data.*

two main types of cryptography in block chain

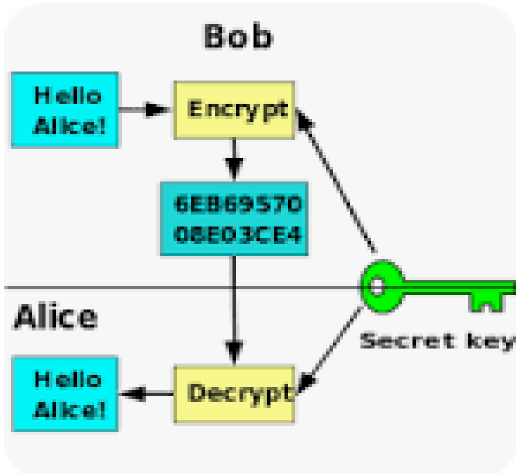
- **Public keys,** which are publicly known and essential for identification, and
- **Private keys**, which are kept secret and are used for authentication and encryption.

symmetric-key

- *cryptography is used to encrypt the data in each of the blocks in the block chain, thereby maintaining the integrity of the data in the block.*
- *Symmetric encryption uses the* same key *for both encryption and decryption,*
- *Symmetric encryption is fast, simple, and efficient, as it requires less computational power and memory than asymmetric encryption*
- *With symmetric cryptography: Both parties share the same key (which is kept secret). Before communications begin, both parties must exchange the shared secret key.*

5 elements of a symmetric encryption scheme?

There are five main components of a symmetric encryption system: plaintext, encryption algorithm, secret key, ciphertext, and the decryption algorithm.



Two types of symmetric encryption?

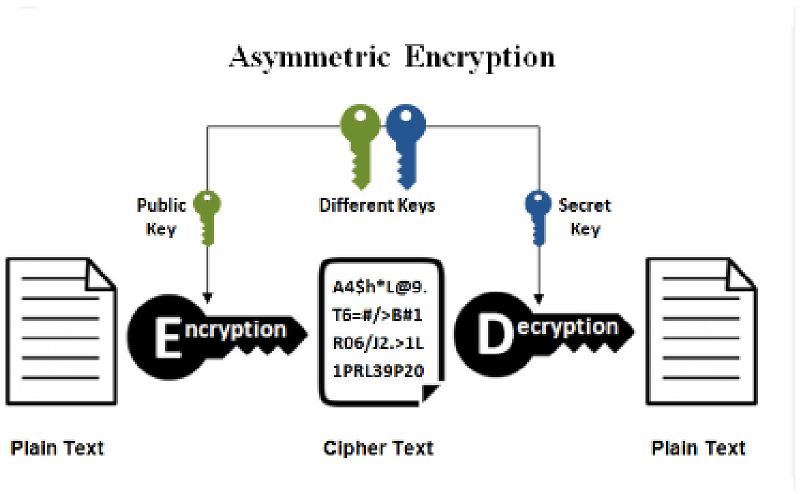
There are two types of symmetric algorithms (or ciphers): stream and block.

- **A block cipher** divides the data into blocks (often 64-bit blocks, but newer algorithms sometimes use 128-bit blocks) and encrypts the data one block at a time.
- **Stream ciphers** encrypt the data as a stream of bits, one bit at a time.

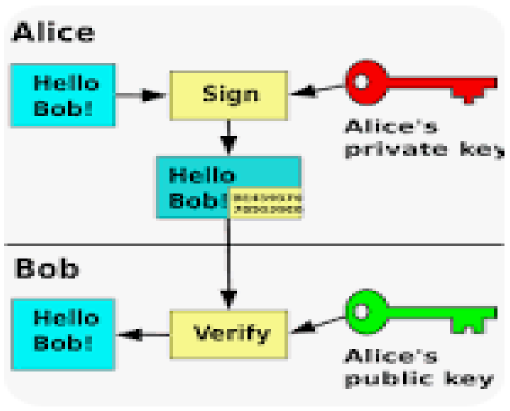
Use cases for symmetric encryption

- Payment applications, such as card transactions where PII (Personal Identifying Information) needs to be protected to prevent identity theft or fraudulent charges without huge costs of resources. ...
- Validations to confirm that the sender of a message is who he claims to be.

Asymmetric-key cryptography can be used in digital signatures and key management.



Digital signatures *employ asymmetric cryptography. In many instances, they provide a layer of validation and security to messages sent through a non-secure channel: Properly implemented, a digital signature gives the receiver reason to believe the message was sent by the claimed sender.*

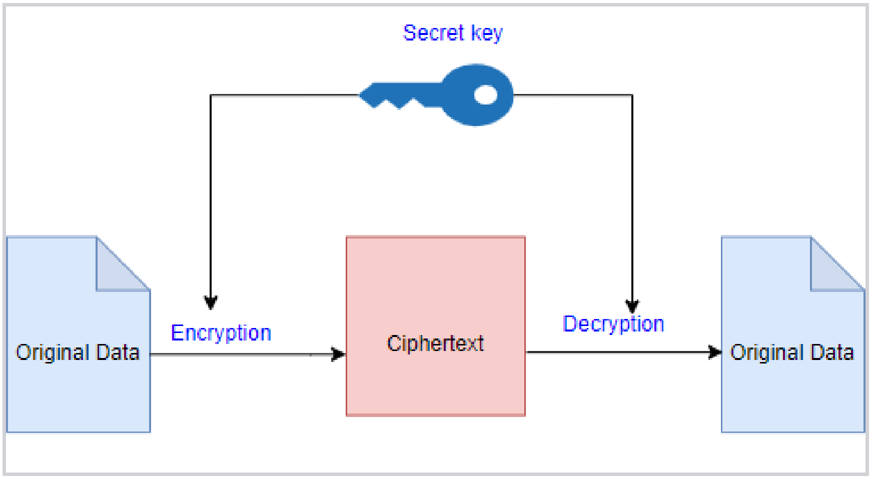


Symmetric-key cryptography

In symmetric-key cryptography, the same key used in encrypting the data is also used to decrypt it. Although it is not as secure as asymmetric-key cryptography, it is fast and efficient, which is why it is used to encrypt and decrypt a large amount of data.

Some of the symmetric-key cryptography algorithms are

- *Data Encryption Standard (DES),*
- *Advanced Encryption Standard (AES),*
- *and Triple Data Encryption Standard (3DES).*

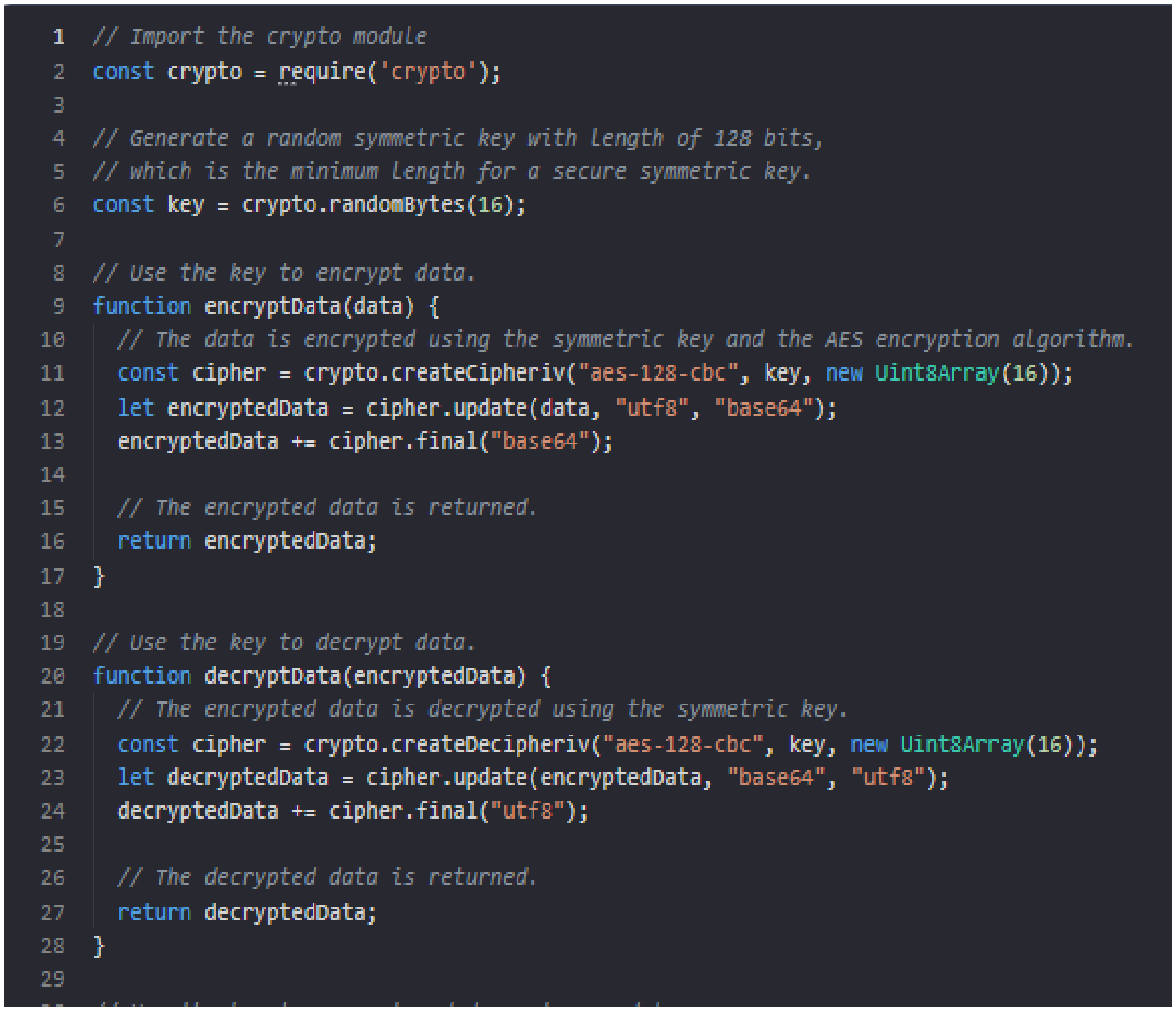


To perform symmetric-key cryptography in blockchain, follow the steps below:

1. **Symmetric-key generation:** When generating a symmetric key, choose a key length. The longer the length, the more secure the key will be. A key length of 128 bits or 256 bits is usually advised. *We can generate a symmetric key using tools that support symmetric key generation such as OpenSSL*. We can also use the cryptography library in *Python or the crypto module in JavaScript.*
2. **Encrypt data with generated key:** To encrypt the data, *we have to select a symmetric encryption algorithm such as AES or DES* that will be used along with the generated symmetric key for encrypting the data. This will change the data from its original form into a ciphertext, which is just a sequence of ambiguous characters. This ciphertext is then stored on the blockchain.
3. **Decrypt the stored encrypted data:** When a block is retrieved from the blockchain, the key is needed to decrypt the encrypted data. In symmetric-key cryptography, the security of data on the blockchain largely depends on the integrity of the symmetric key. If not securely shared, the data can be compromised. One way to securely share the symmetric key is by using key exchange protocols. The authorized recipient then applies the same encryption algorithm and the symmetric key on the retrieved data to decrypt it. The decrypted data can now be used in the blockchain.

Code example

An example of symmetric-key cryptography using JavaScript's crypto module.



```
1  // Import the crypto module
2  const crypto = require('crypto');
3
4  // Generate a random symmetric key with length of 128 bits,
5  // which is the minimum length for a secure symmetric key.
6  const key = crypto.randomBytes(16);
7
8  // Use the key to encrypt data.
9  function encryptData(data) {
10   // The data is encrypted using the symmetric key and the AES encryption algorithm.
11   const cipher = crypto.createCipheriv("aes-128-cbc", key, new Uint8Array(16));
12   let encryptedData = cipher.update(data, "utf8", "base64");
13   encryptedData += cipher.final("base64");
14
15   // The encrypted data is returned.
16   return encryptedData;
17 }
18
19 // Use the key to decrypt data.
20 function decryptData(encryptedData) {
21   // The encrypted data is decrypted using the symmetric key.
22   const cipher = crypto.createDecipheriv("aes-128-cbc", key, new Uint8Array(16));
23   let decryptedData = cipher.update(encryptedData, "base64", "utf8");
24   decryptedData += cipher.final("utf8");
25
26   // The decrypted data is returned.
27   return decryptedData;
28 }
29
```

```
Output
1.56s
```

```
Ciphertext:
Kuxb5Sc5D55cTXDn7z1A2yP+t36glIheHM8ejPKsP+PBrjerKCSxwOpT2omgMd0+4ZJZ0pLUr0
cnT2xvRu3xlA==

 Original data: How to do Cryptography in blockchain by oolawalebright
```

Code explanation

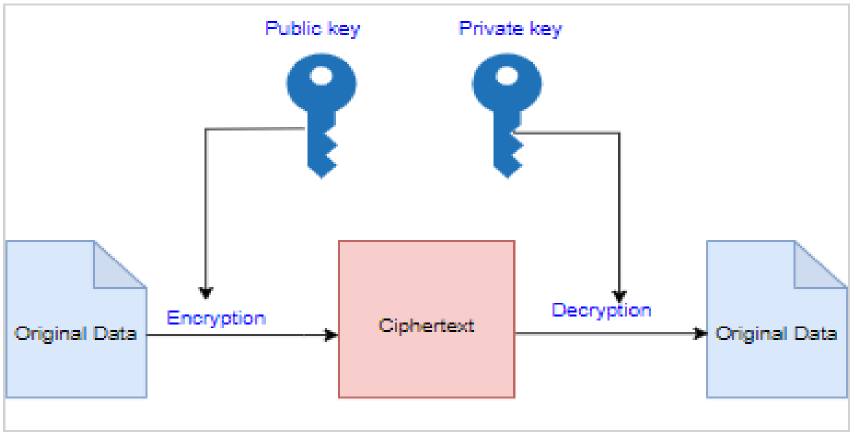
- **Line 2:** We import the crypto module.
- **Line 6:** We generate a random symmetric key with length of 128 bits.

- **Lines 9–17:** We create a function encryptData that accepts the data as an argument. With the help of the createCipheriv method. The function encrypts data using the AES encryption algorithm and the symmetric key. It returns the encrypted data as a base64 string.
- **Lines 20–28:** We create a function decryptData to decrypt the encrypted data. The function takes the encrypted data as an argument and uses the createDecipheriv method, which takes in the symmetric key as an argument for deciphering the parsed data. It returns the decrypted data, which is the original data as a utf-8 string.
- **Lines 31–37:** Here, we have an example to encrypt and decrypt data using the created function before logging their values to the console.

Asymmetric-key cryptography

In **asymmetric-key cryptography**, one public key is used to encrypt the data, and a different key (private key) is used to decrypt it. It is a more secure type of cryptography since the public key is used by the sender to encrypt the information while the receiver uses the private key to decrypt that data, this is why it is used in digital signatures. The public key is shared with others while the private key is kept secret. Some of the asymmetric-key cryptographic algorithms are Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), and Diffie-Hellman. Here is an example of how it works:

- James wants to send a message to Helen.
- James has Helen's public key but not the private key so James encrypts the message using Helen's public key.
- Helen receives the message in an encrypted form and uses the private key to decrypt it. So, only Helen reads the message since no one else has access to the private key for decrypting the message.

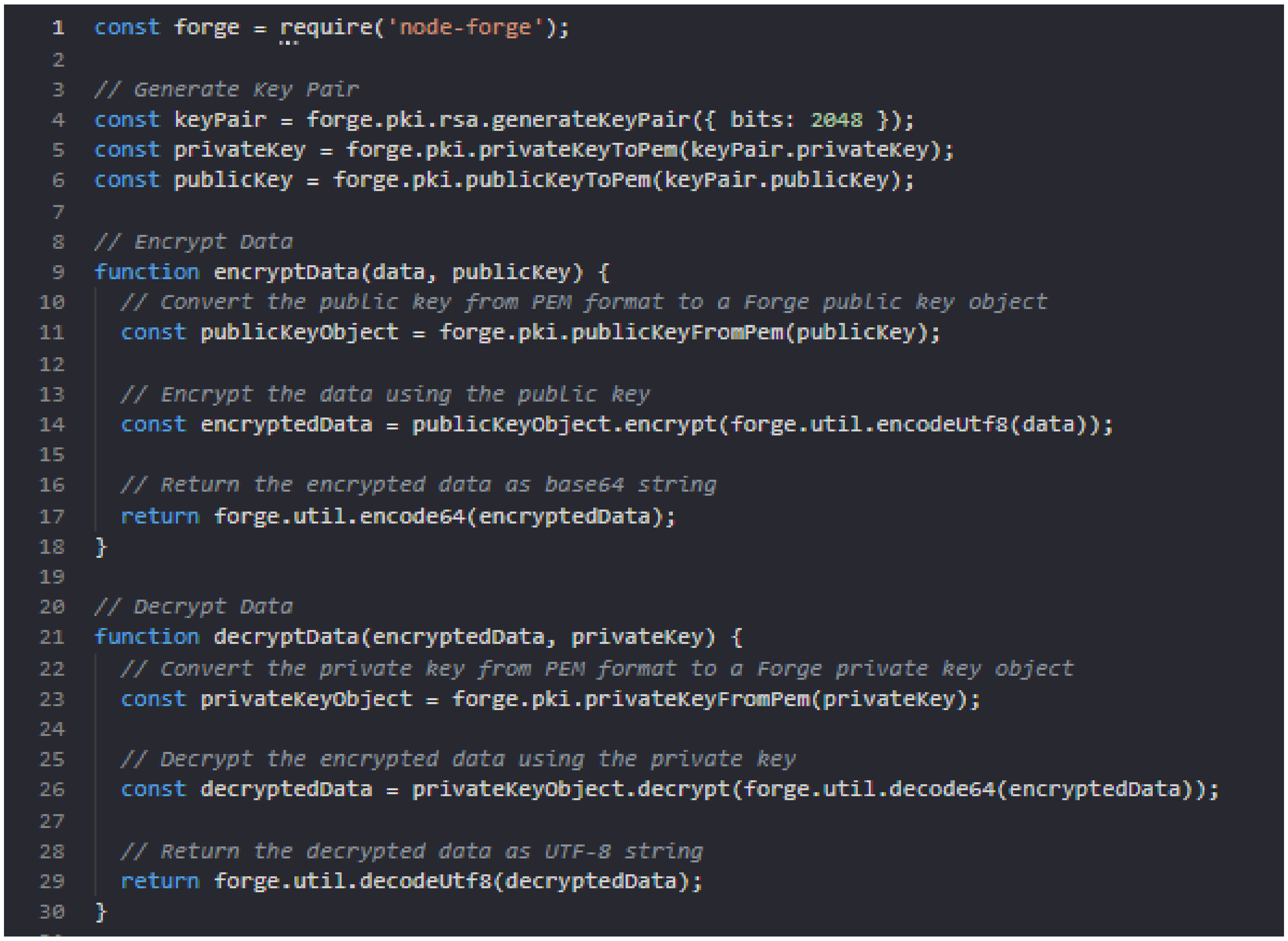


Asymmetric-key Cryptography

To perform symmetric-key cryptography in blockchain, follow the steps below:

1. **Asymmetric-key generation:** *To generate an asymmetric key pair, an asymmetric key generation algorithm such as RSA* has to be chosen based on its key size and efficiency. *A cryptographic tool like OpenSSL is then used to generate the pair.* We can also use cryptographic libraries and modules to achieve this. A private key is first created then a public key is created from the private key. *The private key can then be stored securely while the public key can be published or shared on the blockchain.*
2. **Encrypt the data:** To encrypt the data, the public key of the recipient is applied along with the appropriate encryption function or command of the tool being used. This would change the form of the original data into a ciphertext. *We can now store this ciphertext and send it to the recipient on the blockchain.*
3. **Decrypt the data:** When the recipient receives this encrypted data, *decryption is done by applying the corresponding private key to the public key used to encrypt the data*. After using the private key along with the decryption function or command of the tool being used, the data becomes readable and understandable for the recipient only.

An example of symmetric-key cryptography using JavaScript's `node-forge` library. To run this code, we first need to open your terminal and install the library using `npm install node-forge`.



```javascript
const forge = require('node-forge');

// Generate Key Pair
const keyPair = forge.pki.rsa.generateKeyPair({ bits: 2048 });
const privateKey = forge.pki.privateKeyToPem(keyPair.privateKey);
const publicKey = forge.pki.publicKeyToPem(keyPair.publicKey);

// Encrypt Data
function encryptData(data, publicKey) {
  // Convert the public key from PEM format to a Forge public key object
  const publicKeyObject = forge.pki.publicKeyFromPem(publicKey);

  // Encrypt the data using the public key
  const encryptedData = publicKeyObject.encrypt(forge.util.encodeUtf8(data));

  // Return the encrypted data as base64 string
  return forge.util.encode64(encryptedData);
}

// Decrypt Data
function decryptData(encryptedData, privateKey) {
  // Convert the private key from PEM format to a Forge private key object
  const privateKeyObject = forge.pki.privateKeyFromPem(privateKey);

  // Decrypt the encrypted data using the private key
  const decryptedData = privateKeyObject.decrypt(forge.util.decode64(encryptedData));

  // Return the decrypted data as UTF-8 string
  return forge.util.decodeUtf8(decryptedData);
}
```

```
Output
1.75s
```

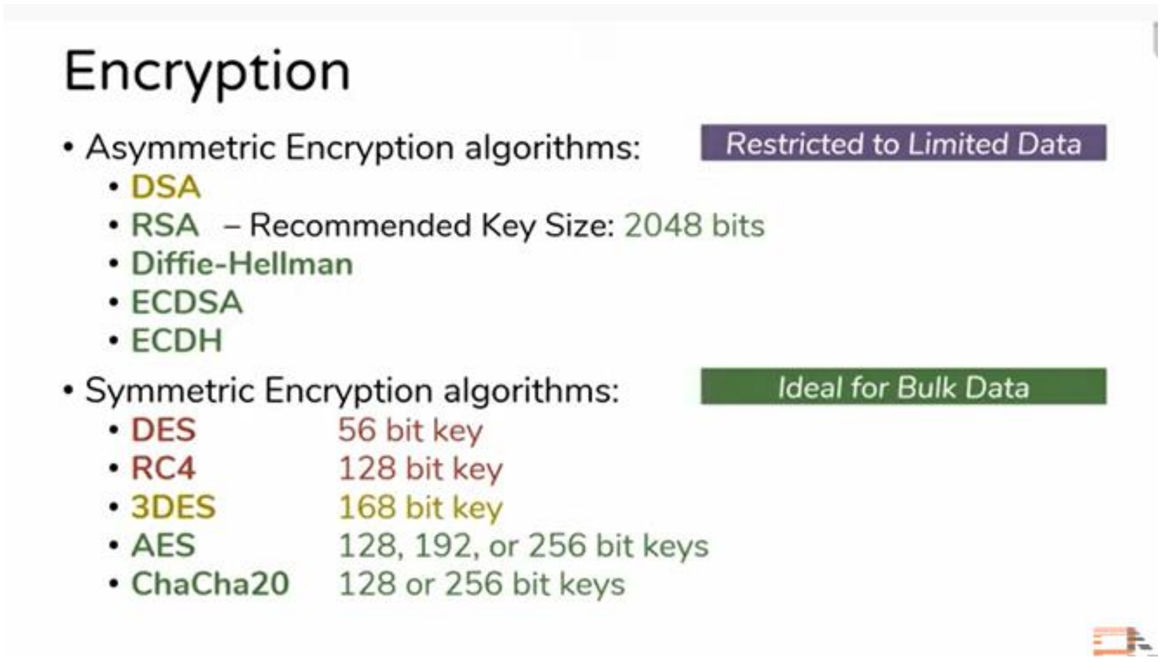
```
Encrypted Data:
n2llBv3zNDGavKAjRViql1E3B+R2fQHZfeUfICcGmR+JU8MQOEU6+Qh3v1h6ijmCpYLZrR/R44
bcMhznx8eLr4wI4aGC/JdJeY8sUTpqK207FuSBgsg/Jkn6QnjLlaDfxNtvh5eAbemOj7wgCLMi
Enf4MluJS/9YA74/OQtvV+wXLh7/H3orCTqogsVs8Ub18WHX4SYMeqcyvn4jRd+7g0ihlZpeat
YjW1u7mHzYzbqnLZgCFgbMTfTjhwR5/q3FHpRXbVqPGP4boAJlteCj+CVsMRWjyNi+R1ILEfcU
kLBEjG/IEb+ATe/LUbjYcN+TlzSvFc8xeWmUOWJXZgY3rg==

Decrypted Data: How to do asymmetric-key cryptography by Oolawalebright
```

Code explanation

- **Line 1:** We import the node-forge module after installing the module onto your local machine.
- **Lines 4–6:** We generate the key pair, both public and private key.
- **Line 3–18:** We create a function encryptData that takes in the data to encrypt and the public key. This function encrypts the data using some of the methods provided by node-forge. It returns the encrypted data as a base64 string.
- **Lines 21–30:** We create a function decryptData that takes the encrypted data and the private key as an argument. The private key is used by a node-forge method to decrypt the data. It returns the original data as a utf-8 string.
- **Lines 32–41:** Here, we have an example to encrypt and decrypt data using the created function before logging their values to the console.

Conclusion

It is important to note that both symmetric-key and asymmetric-key cryptography are important in the blockchain. Symmetric-key cryptography is popularly used in encrypting the content of the block while asymmetric-key cryptography finds its use in digital signature and key management. When properly implemented, cryptography ensures the security of transactions and the confidentiality of sensitive information in the blockchain network. As the blockchain ecosystem continues to grow, cryptography will continue to secure and protect the future of the blockchain.



Encryption

- Asymmetric Encryption algorithms: Restricted to Limited Data
  - DSA
  - RSA – Recommended Key Size: 2048 bits
  - Diffie-Hellman
  - ECDSA
  - ECDH
- Symmetric Encryption algorithms: Ideal for Bulk Data

• DES	56 bit key
• RC4	128 bit key
• 3DES	168 bit key
• AES	128, 192, or 256 bit keys
• ChaCha20	128 or 256 bit keys

<u>What is SHA?</u>

✓ **Secure hash algorithm**
✓ **NSA & NIST** joint development
✓ Has Many families such as SHA-0, SHA-1, SHA-2 and SHA-3s

<u>Summary Hash algorithm</u>

1) Used to index large amounts of data
2) Address of each key calculated using the key itself
3) Collisions resolved with open or closed addressing
4) Hashing is widely used in database indexing, compilers, cashing, password authentication,