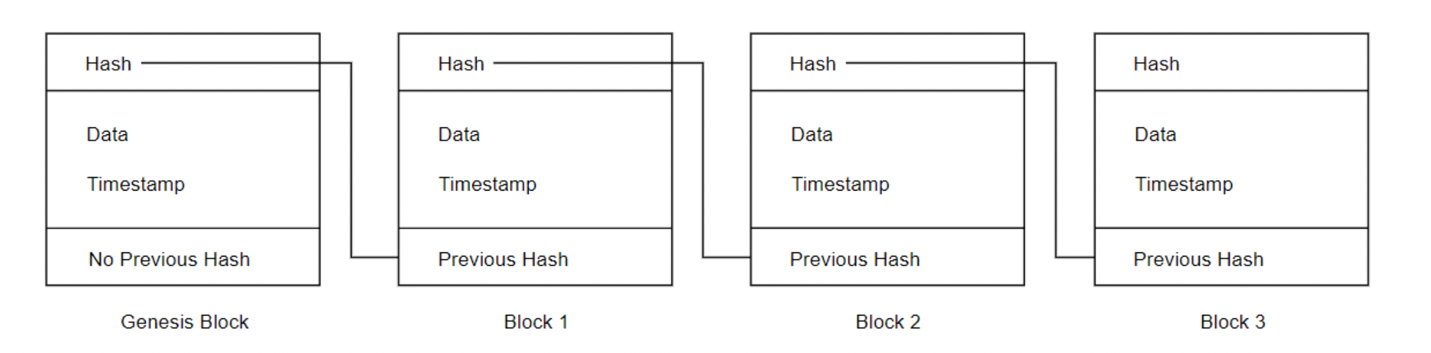
**Introduction:**

Sending messages over insecure channels has always been a security issue. Although there are many technologies to encrypt messages, but there are still ways to attack messages, such as: Eavesdropping, MITM, EFAIL. In many countries, such as Palestine, China and South Korea, citizens are persecuted by the government or the occupation. In addition, traditional applications maintain their data in a centralized database, which is also a problem. Our app ensures decentralization, immutability, censorship resistance and data security. data sent via Users are added directly to the blockchain and a global copy of the data is created in each node. Only legitimate users can do this. This data is accessed using a private key in the blockchain. It removes the need for a trusted intermediary. system complete Decentralizes and allows users to exchange messages securely.

**Disadvantage of central-based communication apps**

1. Central chatting apps collects meta information about the user
2. Confidentiality of a user can be compromised on the request of government
3. Single Point of Failure (SPF): If a single node fails then whole application can be compromised.

****

Blockchain is a shared, distributed ledger that facilitates the process of recording transactions and tracking assets in a network. It provides zero intermediary or 3rd party also achieved the security basics, confidentiality integrity and availability. The blockchain Network is collection of nodes and each node has a local copy of the blockchain

The benefits of blockchain-based communications and data storage technology are numerous: a. the stored data can be encrypted making it impervious to theft

b. the data is stored in a distributed manner, eliminating single-point failures

c. communications can be end-to-end encrypted making them spy-proof n

e. transactions and communications can be subject to incorruptible rules enforced by the blockchain itself

f. the system maintains a tamper and repudiation-proof history of past transactions and communications.

How to build the decentralize chat app

Ethereum is a platform allow the developers to develop dapp using smart contract,

Smart contract: an agreement, contract, or set of instruction that is deployed on a decentralized blockchain

* Cannot be altered “immutable”
* Automatically executed
* Every on sees the terms of agreement

Anyone can join the network can send transaction through the network and paying small amount of fees the fees may vary from one transaction to another. because it depends on the number of transactions sent in the same time, so paying fees for every message may not suitable for some users but for those who wants a high level of security our project is there right chouse.

URL to how the addresses are generated

https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch04.html

Elliptic Curve Cryptography Explained

Elliptic curve cryptography is a type of asymmetric or public-key cryptography based on the discrete logarithm problem as expressed by addition and multiplication on the points of an elliptic curve.

The elliptic curve cryptography (ECC) **does not directly provide encryption** method. Instead, we can design a **hybrid encryption scheme** by using the **ECDH** (Elliptic Curve Diffie–Hellman) key exchange scheme to derive a **shared secret key** for symmetric data encryption and decryption.

* **calculateEncryptionKey**(pubKey) --> (sharedECCKey, ciphertextPubKey)
  1. 1.

Generate **ciphertextPrivKey** = *new* ***random*** *private key*.

* 1. 2.

Calculate **ciphertextPubKey** = ciphertextPrivKey \* G.

* 1. 3.

Calculate the ECDH shared secret: **sharedECCKey** = pubKey \* ciphertextPrivKey.

* 1. 4.

Return both the **sharedECCKey** + **ciphertextPubKey**. Use the **sharedECCKey** for symmetric encryption. Use the randomly generated **ciphertextPubKey** to calculate the decryption key later.

* **calculateDecryptionKey**(privKey, ciphertextPubKey) --> sharedECCKey
  1. 1.

Calculate the the ECDH shared secret: **sharedECCKey** = ciphertextPubKey \* privKey.

* 1. 2.

Return the **sharedECCKey** and use it for the decryption.

The above calculations use the same math, like the **ECDH** algorithm (see the [previous section](file:////asymmetric-key-ciphers/ecdh-key-exchange)). Recall that EC points have the following property:

* (***a*** \* **G**) \* ***b*** = (***b*** \* **G**) \* ***a***

Now, assume that ***a*** = privKey, ***a*** \* **G** = pubKey, ***b*** = ciphertextPrivKey, ***b*** \* **G** = ciphertextPubKey.

The above equation takes the following form:

* pubKey \* ciphertextPrivKey = ciphertextPubKey \* privKey = **sharedECCKey**

This is what exactly the above two functions calculate, directly following the **ECDH key agreement** scheme. In the hybrid encryption schemes the encapsulated **ciphertextPubKey** is also known as "**ephemeral key**", because it is used temporary, to derive the symmetric encryption key, using the ECDH key agreement scheme.

the **encryption key** (derived from the public key) and the **decryption key** (derived from the corresponding private key) **are the same**. This is due to the above discussed property of the ECC: pubKey \* ciphertextPrivKey = ciphertextPubKey \* privKey. These keys will be used for data encryption and decryption in an integrated encryption scheme. The above output will be different if you run the code (due to the randomness used to generate ciphertextPrivKey, but the encryption and decryption keys will always be the same (the ECDH shared secret).

The above demonstrated mechanism for generating a shared ephemeral secret key, based on a ECC key pair, is an example of **KEM** (key encapsulation mechanism), based on the ECC and ECDH.

Src =https://cryptobook.nakov.com/asymmetric-key-ciphers/ecc-encryption-decryption