Absenat

in first words, **what is Absenat?** I think we have to say it's a decentralized network for communications. The purpose of this project is to reach an unrecognizable and completely anonymous network for its members. At the same time, we are trying to implement all the features of a standard messenger network in this project.

The project currently consists of two separate parts, the core and the user interface. core project has been developed by nodejs tools. It's open source and you can review the source on gitlab repository. core project has an command line interface too. the second project is web ui that is developed by ReactJS. It's open source and you can review the source on gitlab repository.

Theory

First we need to explain how it works. Absenat is essentially a mass of nodes in a network that are connected to each other and transmit their messages to others. Any message can only be read by the sender and receiver (or recipients) using cryptography. As a result, a message can be anywhere on the network and transferable between the two nodes. But spreading the message across the network in order to reach the recipient may increase the network load. For this reason, using greedy techniques can greatly reduce the network load.

The protocol that exists between the nodes involves different parts. Below we describe and critique its various sections.

Definitions

Node: The smallest unit in the network. In fact, the network consists of units called nodes. Network nodes have different types. A regular node is the simplest type of node in the network. This node can actually be a user who is using the service. Publishes its messages on the network and other nodes send messages to others. The more advanced type of nodes are bridges. Bridges, in addition to being a regular node, are also links to other nodes. The main difference between a bridge and a normal node is its network infrastructure. The bridge must have a static IP. The bridge must have more processing resources than regular nodes. The network connection time must be longer than the normal node type. It should also have more bandwidth.

All the nodes in the network have in common is their signature. Each node has a pair of public and private keys to authenticate. The private key named after it should not be published. But the public key of each node is considered to be its identity string, which is used to prove that the public key belongs to the node.

Packet: The smaller the data unit on the network that is portable. Each packet has a message. A data capsule is generated if we put all the packets of a message together and put them in sequence. The capsule itself is a smaller data unit on the network that has meaning and has a message in it. Each packet is encrypted by its destination public key. As a result, only the destination can decode it and place it alongside the rest of the packets to make the data capsule. As a result, it can be said that the data capsule can only have its origin and destination.

Start Connection

Before doing so, the two nodes must be connected and their identities identified. They also need to make sure that the communication channel that exists between them remains secure and stable.

There are two nodes in each ideal connection. We call one the destination node (DN) and the other the source node (SN). Each also has a pair of public/private keys which we call the source public key (SBK), destination public key (DBK) , source private key (SVK) and destination private key (DVK) . The connection request starts from the source node. The steps to start the connection are:

step	SN sends to DN	DN sends to SN
0	`SBK` and SN `nickname`	*nothing*
1	*nothing*	`DBK`, nickname and a random generated number as dst trust number or `DTN` encrypted by `SBK`
2	`DTN` decrypted from recieved message and a random generated number as src trust number or `STN` encrypted by `DBK`	*nothing*
3	*nothing*	`STN` decrypted from recieved message, encrypted by `SBK`
4	`finish` message encrypted by `DBK`	*nothing*

In the third step, the DN checks for the DTN received to be the same as the number it sent. Also in the fourth step the SN checks for the STN number. If one of the numbers is not the same, the process of connecting the two nodes will result in a lack of trust and the process will be canceled.

> Each process will be canceled if the process takes more than a certain amount of time. The timeout is currently 5 seconds but may later increase or decrease on the network.

The type of connection described is straightforward. This type of connection exists only between bridge nodes and normal nodes (in a few cases the two normal nodes can also connect directly). The second type of connection that is common between two nodes is the mediator connection. As the name implies, there is an intermediary node in this connection. The intermediate node should preferably be a bridge node. Because bridge nodes are more stable than other types of nodes and can perform their job of delivering messages to their destination.

Intermediate connection is not simply a direct connection. Because there is an intermediary in its path. One of the principles of security in decentralized networks is the lack of trust in everything. As a result, the intermediary node should not be trusted either. For the source node to send a message to the destination node, it just needs to have the public key of the destination. Creates your message, puts your signature next to the message and sends it. Now it can be guessed that if the origin and destination have a short conversation, then the node will be the key to both public keys (though the public key is in direct connection of the nodes to the bridge) so it can (eg) Replace the source node and send the message to the destination. At this point, the network becomes spoofed. In order to prove that the sender's identity is real or fake, the trust string technique must be used. The trust string is actually a 256-bit string. The process of trusting with the help of this field is quite similar to the process of trusting with the trust number of origin and destination in direct connection. The following table describes the intermediate connection steps:

1. The source node that initiates the connection must have the public key of the destination node. The source node generates a message containing its public key and trust string. It encrypts it with the public key of the destination node. It then sends it to the intermediate node.

2. The intermediary node waits for its connection to the destination node to be activated. If enabled, it will send the source node message to the destination node.
3. The destination node decodes the message. Generates a new message containing the trust string. It then generates a random number and puts it in the message. It encrypts the message and sends it to the intermediary node.
4. The intermediary node sends the destination message to the destination node after activating its connection to the source node.
5. The destination node decodes the message and, if the trust string is correct, means that the destination node is trusted by the source node.
6. The source node decodes the message. If the trust string is correct, the trust operation is terminated by the originator. It then encrypts the generated random number with its public key and sends it to the intermediary node.
7. The intermediary node waits for its connection to the destination node to be activated. If enabled, it will send the source node message to the destination node.
8. The destination node decodes the message. If the random number produced is correct. Then the trust operation from the destination was also successful. At this point, an indirect connection is established and the two nodes trust each other. Now they can continue to send messages to one another.

Send Message

Each message has three main characteristics. The first feature is the text and content of the message. A message can be in text, audio-visual and audio-visual types. If the message volume is high, we should divide it into smaller sections and send each segment separately. The second feature is the message recipient. For the message to reach its destination, the recipient must have a unique attribute. In the service introduced, the unique feature of the receiver or in other words all the nodes in the network is their public key. The third characteristic is the message sender. The sender of the message must identify itself in the message. At first it sounds simple, but if there is no oversight of the third characteristic of the message, each node can put itself somewhere else and send the message. That's why we use trust in connections. The trust string in each indirect connection is specific to the same connection and differs from other connections. The sender of the message (who is responsible for making the message) must have the trust string of that connection and place it in the message. If the message reaches the recipient and the trust string is wrong, the recipient considers it a fake message. The steps for sending a message are as follows:

1. First the content sender divides the message into small parts. He then puts each of the parts into an packet. It then places the trust string inside each packet. Finally encrypts the packet with the recipient's public key and sends it to the intermediary node.
2. The intermediary node sends the message packets as soon as it is connected to the receiver.
3. The receiver decodes the packets. If it discovers that one of the packets contains a false trust string while decrypting the packets, it deletes it from the packets list. After the decryption is finished check that all parts of the message are present. It then generates the original message from its parts. Finally, it receives a message and encrypts it with the public key of the source. Sends the message to the intermediary node.
4. The intermediary node sends the message packets as soon as it is connected to the sender.
5. The sender decodes the message and considers the received message.

ACK operation

The operation is designed to detect whether a node is connected to the network or not. Its base is heartbeat inspired. Each node within a specified time frame must send a heartbeat message to its associated intermediate nodes. The heartbeat message contains the following information:

1. The current time in the node in timestamps format
2. Recent user profile edits such as: image, name and other information. If it is first connected to a bridge node, it sends all its data.

One of the nodes' responsibilities is to maintain information and connectivity status of the nodes in the network. They have two important functions against the heartbeat message sent by the nodes:

1. Each node can forward the public key of another node to an intermediary node. If the intermediary node has that node's information, it must send it in response. The following will be described separately.
2. Each node has an indirect target audience through the nodes. The intermediary node must inform its audience of its latest heartbeat message. The following will be described separately.

Get Information

Receiving information is two separate steps in time. The first step is when the source node sends the heartbeat message to the intermediate node. The next step is when the destination node requests the node from the broker to receive the latest source node information.

Each node in the network should typically be connected to several intermediate nodes. This is because if one intermediary node is out of reach for hours, the other intermediaries can take responsibility for it. In what algorithm a node sends its messages to which nodes to send less traffic to the network, it will be explained below. The heartbeat message that a node sends to the intermediate nodes varies for each intermediate node. For example, suppose node a is connected to two intermediate nodes b and c. Node a changes its name to nine in the morning. Ironically, node c is not available at that moment. Node a sends its heartbeat message only to b. At two in the afternoon, she changes her profile picture. At that moment, node b becomes available. Node a sets its heartbeat message to b and c. However, the heartbeat message set to b has both the new name and the new profile image, while the heartbeat message set to b has only the new profile picture.

In fact, heartbeat messages are sent at shorter intervals (usually every forty-five seconds) so it is very unlikely that different heartbeat messages will be sent from one node through it. Unless an intermediary node is out of network for a long time. In order for each node to be able to easily send the needed changes to the heartbeat, it needs to keep a versioning system of its profile information and also know what the latest version of its profile provided to each intermediary.

Core

core project is developed by nodes and socket io.

Web UI

web ui project is developed by ReactJS and impletements core for comunications