CUP2PY - Proof-of-Concept for recursive decentralised peer-to-peer (P2P) communication protocol

Lucy Betts, Ilia Bolgov, Jodie Furnell, Annabel May

12 May 2023

Abstract

This paper introduces and describes the functionality of CUP2PY: a python library that allows decentralised communication with peer-to-peer (P2P) communication protocol. This P2P protocol uses asymmetric cryptography (RSA) for encryption and authentication.

1 Introduction

In 2001, the Internet Relay Chat (IRC) communication protocol [1] was at its peak with 6 million simultaneous users at the time. IRC allowed simple peer-to-peer messaging using TCP as its transport protocol.

By default IRC doesn't have either user authentication or end-to-end encryption. To connect to the IRC server it is required to know its IP address.

In this project we describe a primitive communication protocol concept, which solves these problems, and demonstrate the Proof-of-Concept of implementing it in a python library.

In this project, the communication protocol is completely decentralised, provides authentication and doesn't require knowledge of the other peer's IP address - only their public keys.

2 Protocol Description

We chose to use RSA public keys [2] and their SHA-256 hashes [3] as ”usernames” instead of directly using IP addresses, because IP addresses can change and do not directly provide user authentication.

The key idea is for each peer of the network to have a local ”routing table” - here we've called them ”Address Books”, and ”User Records”, which contain a user's public key, their IP address, the date and time of the generation of this record and the digital signature for this record. The digital signature is used by other peers to verify that the ”User Record” was in fact generated by the user with the given public key [4].

Every user can generate or load an existing RSA key pair and use it to generate a new User Record. This User Record will be later used by other peers to find the user's current IP address and directly communicate with them. Every time a user changes IP address, they can generate a new User Record from the RSA key pair.

Each peer stores all records of other users locally, and uses them for communication, sending and routing their requests or other user's requests.

There are two types of request for this communication protocol: UPDATE requests and SEARCH requests. These requests are used to exchange User Records in the network and are routed recursively with a predetermined ”recursion depth”.

UPDATE requests are sent by a user to all users in their Address Book and contain their User Record, recursion depth (it is decreased by 1 after it is received by another user) and list of hashes of users which this request was already sent to (to minimise repetition). After each user receives the request, they verify the digital signature. If it is successfully verified, they update the User Record in their Address Book if this user was in the Address Book, or add a new one if it didn't. They then pass this request to all users in their Address Book. This continues until the ”recursion depth” is 0.

SEARCH requests are sent by a user to all users in their Address Book when the user wants to find another user who is not in their Address Book, or to look for updates of a User Record in case one is unreachable via their IP address. As well as UPDATE requests, SEARCH requests are also recursive. Recipients of the SEARCH requests will send the searched User Record to the sender of request, if they have it in their Address Book, or they will pass the searched request to all users in their Address Book (except for the ones which already got the request). The request is going to spread until the recursion depth is achieved.

3 Python library

As a proof-of-concept for this communication protocol, the python library CUP2PY was created. In this library, the Address Books are represented as SQL tables and a dedicated class was created to represent the User Records. The CUP2PY library allows user to manage their Address Books, create User Records from RSA public/private key pairs and generate the RSA key pairs. The library has the functionality to create and handle UPDATE and SEARCH requests, and convert them from string representations for communication to objects of dedicated classes and back. The library also has the functionality to encrypt and decrypt local files with RSA private keys using AES algorithm [5].

The library uses notion of sessions - this allows the user to choose a specific Address Book and User Record, and to create and handle requests using chosen Address Book and User Record without conflicts.

The library does not contain functions for the actual network communication: sending and receiving the request strings, as well as actual messaging functionality. A different library can be used for that, such that 'socket'.

4 Potential problems

While the protocol allows completely decentralised communication, it needs for a new user to have a non-empty Address Book to be able to connect to a ”global network”. This problem can be solved by introducing extra servers hosting the most up-to-date Address Books for users to download, although this would make the system partially centralised.

Another problem of this protocol arises due to the recursive nature of requests. When users send UPDATE or SEARCH requests, the same user can receive the same request multiple times. This causes ”flood” and might become a problem on large scale.

5 Simulation results

This simulation demonstrates how other users are able to communicate with others with UPDATE and SEARCH requests. Every user is represented by a node on a directed graph, each edge represents that the user is in the Address Book of a user that is connected to him. Color of a node represents amount of other nodes it can reach through other nodes (red = many nodes, blue = no nodes). The graphs are generated from probabilities of each event: user update, new user etc. Here are the details of this simulation:

5 users at the start

Probability of random connection between two nodes = 0.05 (This represents situation when two users exchange their User Records outside of the network)

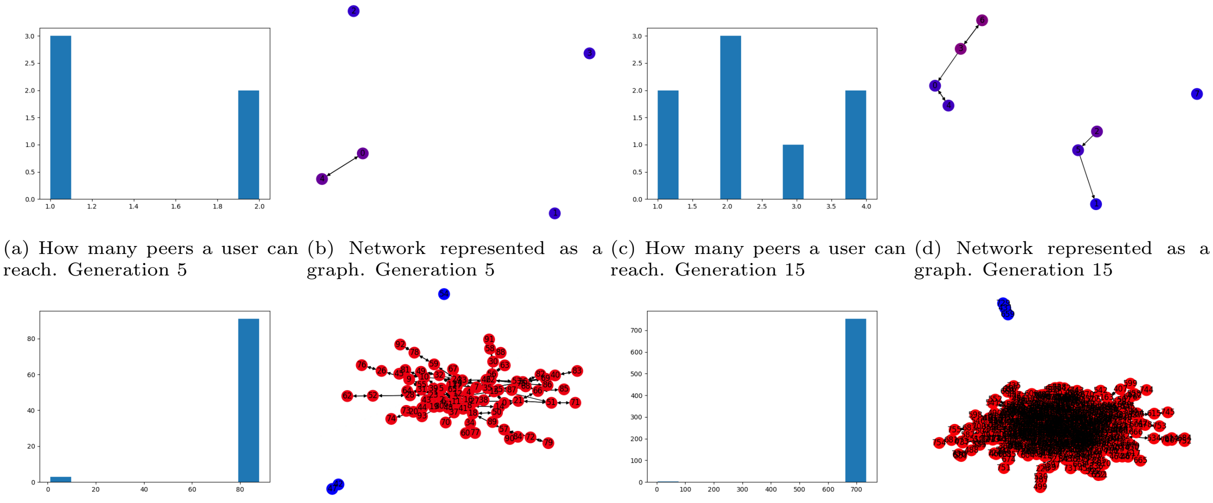
Probability of random "invite" = 0.1 (This represents situation when User "invites" another person and this person only has the User in their Address Book)

Probability of Update = 0.5

Probability of Successful connection = 0.7

Recursion depth for requests = 3

Probability that there will be a new node not connected to the graph = 0.1



(a) How many peers a user can reach. Generation 5

(b) Network represented as a graph. Generation 5

(c) How many peers a user can reach. Generation 15

(d) Network represented as a graph. Generation 15

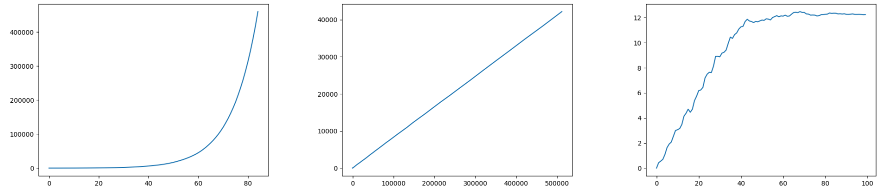
(e) How many peers a user can reach. Generation 25

(f) Network represented as a graph. Generation 25

(g) How many peers a user can reach. Generation 50

(h) Network represented as a graph. Generation 50

Generation 50: 756 nodes and 2198 edges, Unnecessary repeated UPDATE requests: 8325, Maximum reach: 734, Minimum reach: 3, Total nodes: 756, Average reach: 729.1375661375662.



(a) The number of "unnecessary repeated UPDATE requests" is growing exponentially (as well as the number of nodes in this simulation).

(b) The relation of number of peers and number of "unnecessary repeated UPDATE requests" is linear.

(c) "Unnecessary repeated UPDATE requests" over number of peers in the network tends to a constant. This shows that flood with UPDATE requests is not going to become a problem on a larger scale network.

These simulations shows that eventually pretty much all nodes are going to be reachable by others and the repetition of UPDATE requests on a larger scale is going to be manageable.

6 Conclusion

This communication protocol can be implemented in theory at least on a smaller scale. There might be problems with repetition of requests (especially SEARCH requests as these require larger recursion depth) and with resistance to all kinds of attacks, however the simulations we made shows, that the distribution of UPDATE requests will be manageable for a large number of peers.

The project requires a lot of further work to fix these problems. To make this protocol scalable we need to figure out correct distribution algorithm for SEARCH requests and have a probabilistic model for that distribution to prove that the protocol will work on any scale.

7 References

[1] Internet Relay Chat Protocol. $https://datatracker.ietf.org/doc/html/rfc1459$

[2] Pointcheval, D. (2011). RSA Public-Key Encryption. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. $https://doi.org/10.1007/978-1-4419-5906-5_153$

[3] Descriptions of SHA-256, SHA-384, and SHA-512 from NIST

$https://web.archive.org/web/20130526224224/https://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf$

[4] Kaliski, B. (2011). RSA Digital Signature Scheme. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. $https://doi.org/10.1007/978-1-4419-5906-5_432pp$1061–1064

[5] National Institute of Standards and Technology (NIST), Commerce Department. "Advanced Encryption Standard (AES)". Government. Commerce Department, November 1, 2001.

$https://www.govinfo.gov/app/details/GOVPUB-C13-5b70c4f21283f175408d9560480f73b3$