Greedy Delivery – Anonymity through Ambiguity

The Request-Response (RR) Axiom

A quintessential aspect of network communications is that in order for a connection to be established and data to be transferred there must be at least two parties involved. Consequently, a certain model emerges which dictates how information moves between hosts. This model can in essence be described the following way:

First, Alex wants some information which is stored on George's machine (Henceforth the names of the people shall be synonymous with their machines). Since Alex desires to have this information, she sends out a *request* for this data. Let's examine this request more closely. It must contain at least 3 things – a destination address (otherwise how can the request even be delivered to the right system?), a payload which says what information is sought on the destination host, and finally – a sender address. The sender address will tell the destination host where to send the requested information. Without a sender address the whole request becomes pointless!

Unfortunately, this sender address is also the biggest thorn in the back of anonymity, since it is what ties one to their data.

The Request-Response Axiom lies in the fact that it is impossible to not have your sender address *somewhere,* for the data you requested must eventually be delivered to you. Or stated more formally,

The RR Axiom: *Any request-response cycle may never be completely separated from its instigator.*

No matter how complex, obscure, and convoluted you make the transmission of data, the sender address will have to make an appearance at least in one place. And this is a fact of life which we are forced to deal with. Ways to mitigate this have been invented – VPNs and proxies, for example. However, it is rather trivial to trace you back given enough time – you just have to go through every proxy and it will eventually lead to

you. That is, given enough time and effort, the request and the response can be traced back to you with a 100% certainty.

Well, we can do better! The purpose of this paper is to a describe a method through which it is impossible to decisively conclude what data belongs to whom! By introducing this ambiguity, anonymity is augmented to the highest possible degree.

Package Deliveries and Greedy Customers

Your spouse's birthday is impending and you still haven't got a gift for them. You hop online and, knowing their sporty nature, you find this incredible smart watch with all the bleeding-edge tech available. Of course, such a grave matter cannot be made public, so you do everything in your power to keep this a secret. You order the watch, but instead of telling them to deliver it directly to your home, you state that you want it delivered to a courier's office.

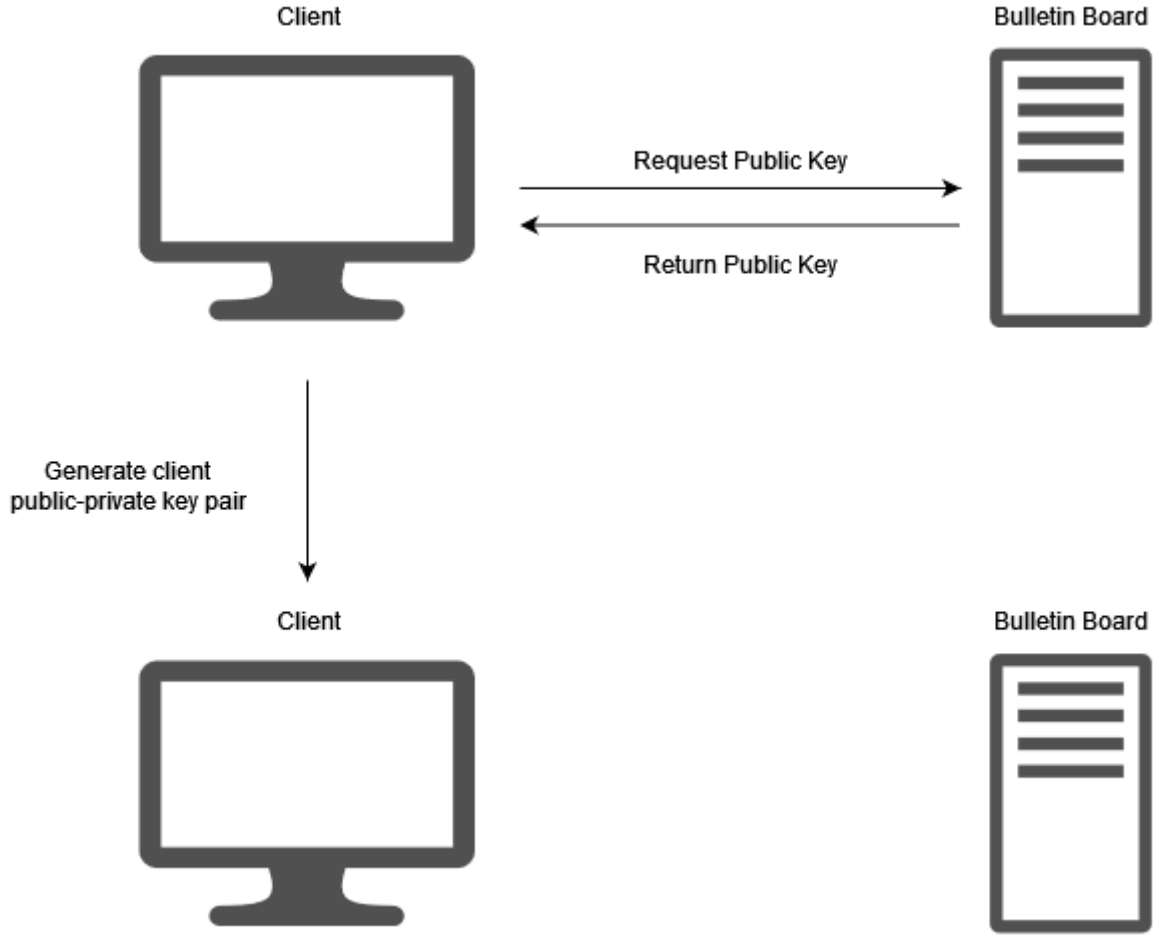
A few days go by and you go to the office to check if your package has been delivered. You enter and instead of going to the counter and asking for your package, you simply go to the inventory at the back, pick up *all* packages which had been delivered that day and simply stride off. Well, well, well, at the cost of being extremely greedy—mission achieved! Since you never gave the company you ordered the watch from your real address, they cannot trace the watch back to you, only to the courier's office. Moreover, the courier's office are unable to say which package you went there for, since you took all of them! All they know for sure is that you *visited* the courier's office, but they aren't cognisant of the purpose of your visit.

The *greedy deliveries* technique described in this paper is pretty much exactly the same (although even a bit more secure, since the contents of the packages are unknown).

In this system, however, courier's offices are called *bulletin boards*. A bulletin board (BB) serves as intermediary between you and your destination. It bears many similarities with a VPN, but also has a huge difference.

The initial stage of any request made via a greedy delivery is to request the public key from the bulletin board. Once you have the public key, you need to generate a 128-bit *request identifier (RQI)* from a uniform distribution. Next, you need to generate your own private-public key pair,

which will comprise the client public and the client private key. From now on, whenever you want to request a data from some source, it will work similarly to a VPN. First, you need to put the response identifier in your request. Following a new line, you need to add your client public key. Insert yet another new line and put your actual request here. Finally, you encrypt the entire message with the BB public key and send it to the Bulletin Board. It is paramount that you *substitute* your sender IP with the IP of the bulletin board, so that the source and destination IP in the request are both the bulletin board's. A BB should never respond to requests whose source and destination IP don't point to its IP address.

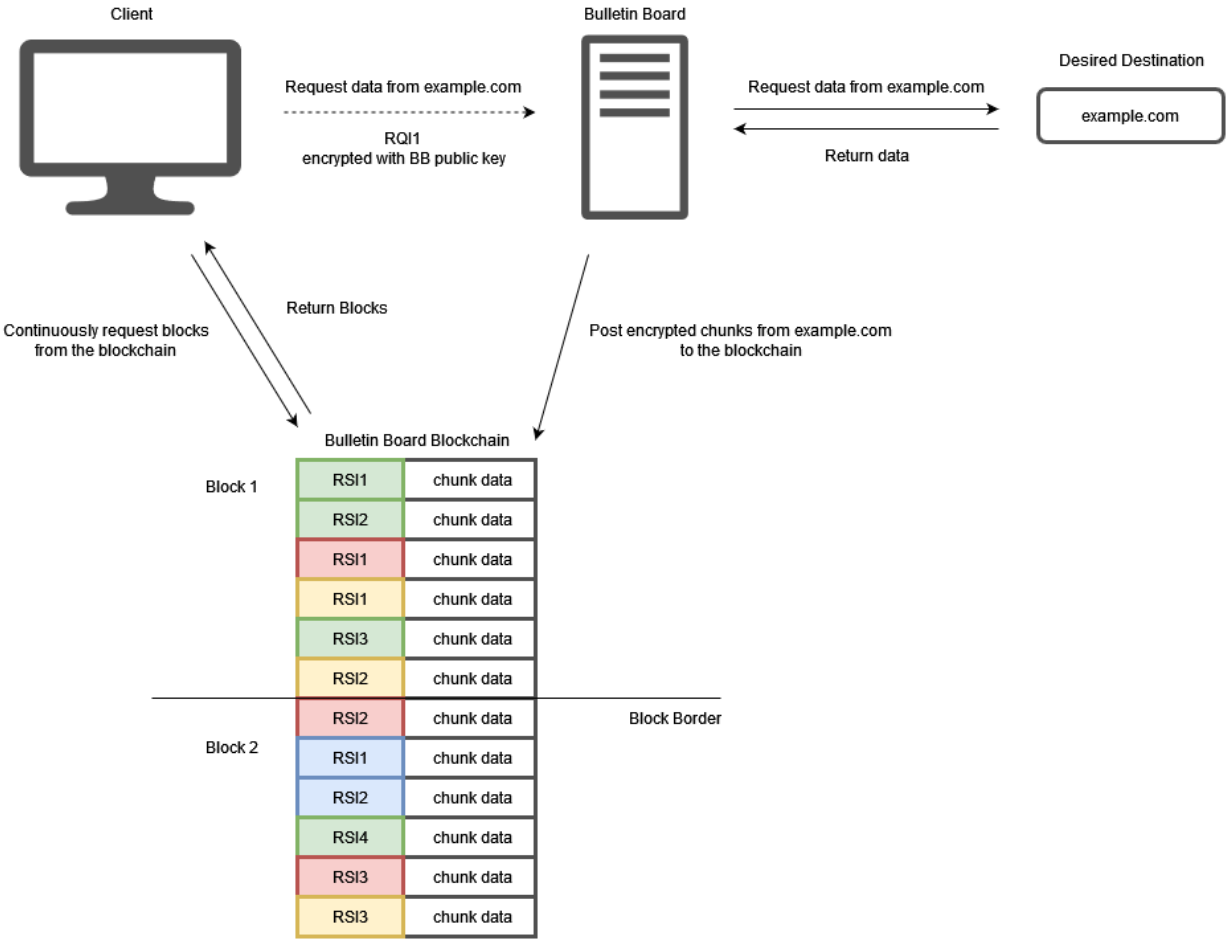


When your request arrives at the BB, it decrypts it via its private key and forwards it to its real destination. When the real destination answers with the requested info, the answer is segmented by the BB in fixed-length chunks. The first chunk from the response is prepended with a number, denoting how many the size of the actual response in chunks. These chunks are then encrypted with the client public key.

An algorithm, such as a CSRNG, is used to generate a sequence of 128-bit numbers, all based on the RQI in the initial request. These 128-bit numbers are called *response identifiers (RSIs)* and are assigned to each chunk from the response in the order they were generated and the order in which the chunks arrived. The first chunk gets matched with the first RSI, the second chunk with the second RSI and so on. These pairs are then pushed to the BB's blockchain in the order of the chunks' arrival. This, however, is true for the responses of all the BB's clients – they are all pushed onto the same blockchain. A block on this blockchain contains a fixed number of chunk-RSI pairs and is only available on the bulletin board for a limited time, after which the BB no longer serves it and deletes it.

Meanwhile, clients are constantly downloading the new blocks as they appear on the BB's blockchain. The client has in advance generated the same sequence of RSIs based on their RQI and is now actively searching through the blocks on the blockchain to find those chunk-RSI pairs which have RSIs matching those generated by the client. The client then decrypts those chunks with their client private key and recover the original response from their desired destination.

On the following diagram, RSIs that come from the same RQI are colored in the same way as a visual aid. In reality, it is not possible to distinguish which RSI comes from which RQI or even if two RSIs come from the same RQI.


Client
Bulletin Board
Desired Destination
Request data from example.com
RQI1
encrypted with BB public key
Request data from example.com
Return data
example.com
Return Blocks
Continuously request blocks
from the blockchain
Post encrypted chunks from example.com
to the blockchain
Bulletin Board Blockchain
Block 1
RSI1
chunk data
RSI2
chunk data
RSI1
chunk data
RSI1
chunk data
RSI3
chunk data
RSI2
chunk data
RSI2
chunk data
Block Border
Block 2
RSI1
chunk data
RSI2
chunk data
RSI4
chunk data
RSI3
chunk data
RSI3
chunk data