Hello and welcome to my explanation of the DIMY protocol

**What is DIMY?**

DIMY is a new type of contact tracing protocol, a protocol in which smartphone based digital apps exchange BLE. The smartphones act as clients, communicating with each other to record contact, while sending close contact information to a backend server, which is maintained by health organisations. When a user becomes diagnosed, they can request a list of contacts from the server. The difference between most contact tracing apps and DIMY is the key technologies used, as seen in Figure 1. DIMY makes use of a combination of Diffie-Hellman key exchange in combination with elliptic curve cryptography, Shamir secret sharing, bloom filters and blockchain technology. Each component of the protocol makes use of these technologies in a way to ensure very high efficiency, anonymity, and security. Additionally, it also uses a hybrid architecture, meaning not all the workload is placed on the server, the user side does most of the calculations while the server performs risk analysis and notifications. Figure 2 shows a brief overview of the DIMY protocol architecture, which will be explained in depth later in this presentation.

**Components of the DIMY protocol**

The DIMY protocol can be broken up into 5 components: The first is generating identifiers, which are the IDs transmitted between client devices. Using elliptic curve Diffie-Hellman, these IDs are used to represent the close contact between two or more people. The second component is focused on advertising these identifiers securely between users. For this Shamir secret sharing is used in combination with hash functions, to ensure that any attackers who try to capture the identifiers must first capture the right amount of shares and then decrypt them to obtain the IDs which would take significant time, due to the use of Diffie-Hellman. So even over the insecure BLE broadcasting channel, secrecy can be achieved. The third component is storing encounter information which is done using bloom filters. These bloom filters efficiently and securely store IDs by using a bit array, one for each day, to a maximum of 21 days which is the incubation period for covid. These 21 daily bloom filters are then combined, as part of the fourth and fifth components, to prepare for uploading to the blockchain, where smart contracts are used to process them in 2 ways. The first is if queries were made to check for risk of infection which is the contact verification process. Here the combined bloom filter is uploaded and a message is returned to and from the blockchain informing the user if they are at risk of COVID or not. The second is if an infection was declared and the close contact list, which is the combined bloom filter, was uploaded to be stored on the blockchain. These 5 components will now be discussed in further depth

**Generating and Advertising Identifiers**

“Ephemeral” identifiers are used to distinguish between devices, and are generated every 30 minutes. Their calculation is done using elliptic curve Diffie-Hellman, by first generating a random number and then using this as input to the elliptic curve to generate a 128-bit key, which is our ephemeral identifier. Each device generates its own “EphID”, however it does not directly broadcast this. Instead each device will employ Shamir secret sharing to split the ID into 30 equal parts, broadcasting 1 share every minute. According to Shamir secret sharing protocol, a device needs at least 15 separate shares of the ephemeral ID to reconstruct it, hence a device must be in close contact with another device for at least 15 minutes before it can reconstruct the ephemeral ID of that device. Let us say that device 1 gets the EphID from device 2. Once this happens, the random number generated, used in the formation of the device 2’s reconstructed ID, can be extracted. This is used by device 1, along with device 1’s random number, to obtain a new ID called the encounter ID. This encounter ID is what is stored and used as the proof that two devices were in close contact. One problem encountered was when a device 1 met device 2 while it was transmitting the last few shares of the previous EphID. When this happened, device 1 would receive the last few shares of EphID 1, but would then start receiving the first few shares of EphID 2, since a new EphID is generated every 30 mins, and thus would not be able to reconstruct either EphID, despite being in contact for more than 15 minutes. To solve this problem, BLE messages were advertised with 2 EphIDs, as shown in Figure 3, allowing for EphID’s to rollover after the epoch period. The use of elliptic curve Diffie-Hellman ensures the data cannot be used arbitrarily, since it is computationally expensive to reverse and it is combined with Shamir secret sharing. There is also no feasible way in which they can reconstruct the ID even if they have heard all the messages.

**Storing, Uploading and Verifying Encounter Identifiers**

After the Encounter ID has been found, it is stored in a bloom filter and then immediately deleted, which does two important things. The first is it prevents info leakage at the client level AND back end blockchain since the bloom filter will just be a bit array. They second is it reduces storage requirements. All the Encounter IDs for a day are stored in one “daily bloom filter” also known as a DBF, to a maximum of 21 DBFs for 21 days.

Smart contracts, which are pieces of code that are executed when conditions are met, are used to do 3 things in this Hyperledger permissioned blockchain. The first is when the user declares that they have been infected with COVID, the Health Authority issues an access token to the user, which is taken from the blockchain, issued by a smart contract. This token allows the user access to the permissioned blockchain. This access is used to upload a “combined bloom filter”, or CBF, again done by a smart contract, which is a combination of all 21 DBFs. The last use case for the smart contract is when a user wants to check if they met a COVID positive user. In this case, the users’ device will combine all 21 DBFs into a query bloom filter, or QBF. This is then uploaded to the blockchain, but for this an access token is not required. A smart contract completes the check for matches between the QBF and CBF, and regardless of the returned result, the QBF is deleted from the blockchain.

A REST API, which allows for a common platform between the differently programmed client side and backend blockchain sides to communicate with each other, is used as the middle man when uploading to the blockchain. Figure 4 to the right shows that the REST API is not used when the health authority issues access tokens, but only for uploading the CBF and QBF.

The permissioned blockchain, modelled after the Hyperledger Fabric blockchain, allows for organisations to only allow access to the blockchain under specific circumstances but does not actually handle any operations or storage. This means this blockchain ensures data integrity, transparency of operations and decentralised data storage.

**Conclusion**

DIMY is a protocol that combines concepts of elliptic curve Diffie hellman, bloom filters, shamir secret sharing and blockchain technology to create a contact tracing protocol that works efficiently, anonymously, and securely. Each of the components discussed previously incorporates one of these concepts. By using a hybrid architecture type for the protocol, the server does not do everything, instead it just assumes the role of risk analysis/notification processing. The client handles calculations such as CBF, QBF, EphID generation, secret reconstruction etc.

The figure to the right shows the general information flow in the DIMY protocol. User devices send and receive Shamir secret pieces, while also actively looking to reconstruct and generate the encounter IDs, which it places into bloom filters. The filters are then combined and sent to the blockchain back end, via HTTPS, where CBFs are stored if the individual was infected and QBFs are checked against the stored CBFs for possible infection. This encapsulates all of the inner workings of the DIMY protocol.