Contact Tracing on Blockchain

rev 0.1

Alessio Ferone, Antonio Della Porta, Luigi Borchia,
Giovanni Volpicelli
parthenopeblockchaingroup@gmail.com

April 24, 2020

In order to fight the SARS-CoV-2 current outbreak it's necessary to improve the notification process to subjects potentially at risk of contagion, by making it faster and easier. For this purpose many **contact tracing** systems are being exploited and requested by nations that are fighting the pandemic. The idea behind digital contact tracing is to trace contacts between individuals by using BLE (Bluetooth Low Energy) technology to calculate the estimated physical distance between devices. Collected data is then decentrally processed and only used to notify people who have been in contact with infects. Contact tracing does not aim to share personal data or track infected subjects.

Many models presented until now (DP-3T, PEPP-PT, Apple/Google system) work with a centralized backend as intermediary between the system peers. In this document is presented an alternative model which uses a blockchain system as backend technology, exploiting the **smart contracts** concept. We believe that such system could bring new ways to avoid the fragmentation that will result in case of many contact tracing applications releases.

Contributions to the document or the project are welcome.

Summary

In this document a contact tracing model with the following characteristics will be explaiend:

- **Privacy protection** for the infected subject and who has been in contact with him
- **Backend transparency** and auditing of the software that handles the notification for possible infects.

The only goal of the system is to notify individuals that have been in contact with people that have tested positive to the virus by the **sanitary system**.

The system **does not** aim to:

- Track the infected subject
- Reconstruct his social interactions

Participation to the proposed system is intended to be voluntary from both infected and not infected citizens.

System overview

The proposed system expose only the minimum necessary informations to the public. All data is stored on decentralized platforms, which are less likely to suffer attacks than centralized notification and storing systems.
The protocol is divided into three different phases:

(1) **Temporary IDs broadcast and reception**: users generate temporary IDs that are sent to nearby devices through BLE. At the same time users listen for temporary IDs from other close devices, which are stored on a local database; these IDs will be then analyzed to check for potentially contagious contacts.

(2) **Infection notification**: a user diagnosed with the virus will receive a confirmation code that will allow the publication on blockchain of a **compact form** which represents the temporary IDs set used during the period in which he was contagious.

(3) **Events reception from blockchain**: users will listen for events triggered by the blockchain; these events will signal new infections. At that point, it will be necessary to download the compact form released by the infected user and check in the local database for any matches between the local IDs and the ones reported by the infected user.

The system can be splitted in two parts:

- **Contact tracer (off-chain)**, which handles user's contacts tracing. This part of the system consists of a smartphone application that exploits BLE (Bluetooth Low Energy) technology and some of its characteristics for anonymous IDs broadcasting and reception.

- **Smart contract for infection notification (on-chain)**, which handles infects data reception and data notification to the system users. This part of the system consists of *smart contracts* released on blockchain systems based on EVM (Ethereum, Hyperledger Burrow, Quorum).

In the following section there will be described the technlogies used in the system and their details.

Temporary IDs and BLE (Bluetooth Low Energy)

Apple and Google recently published the specifications of their protocol. The released framework does not allow to modify key handling, so we believe it will soon become the standard for contact tracing on mobile devices.
The protocol introduces a new concept of pseudorandom IDs called *Rolling Proximity Identifiers* [2]. The device that executes the protocol will broadcast the RPIs through BLE (Bluetooth Low Energy) technology [1] and at the same time it will listen for RPIs from nearby devices, building in this way a contact history. RPIs are released as a 20 bytes payload, where the first 16 bytes represent the ID [1].

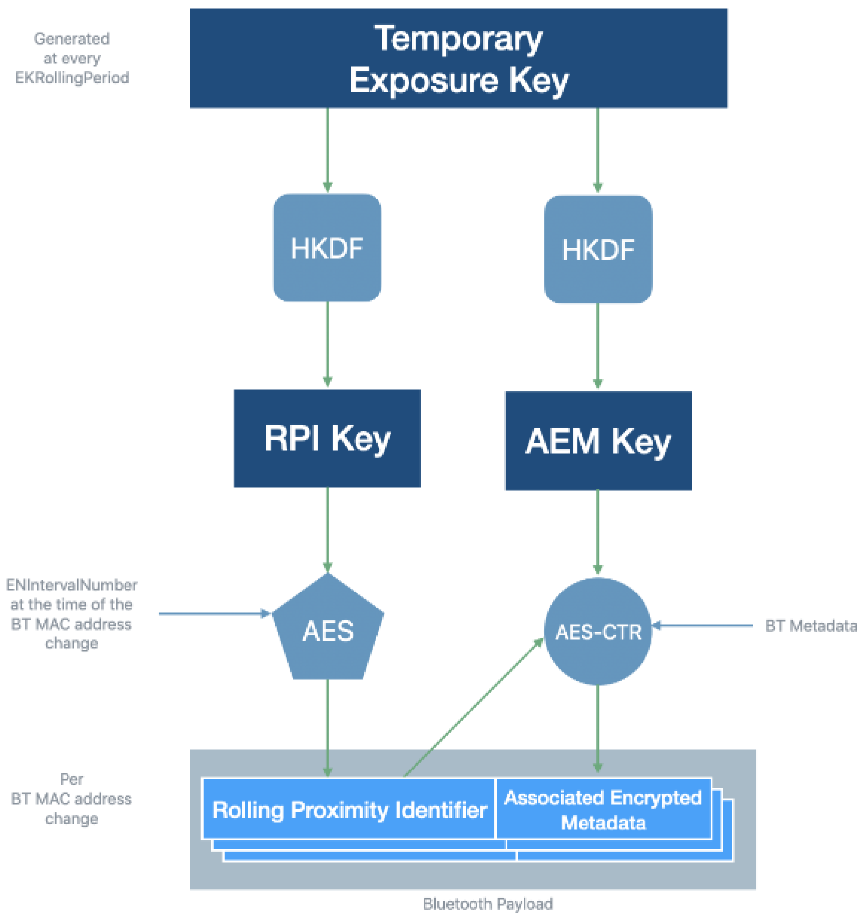
Generated at every EKRollingPeriod
Temporary Exposure Key
HKDF
HKDF
RPI Key
AEM Key
ENIntervalNumber at the time of the BT MAC address change
AES
AES-CTR
BT Metadata
Per BT MAC address change
Rolling Proximity Identifier
Associated Encrypted Metadata
Bluetooth Payload

Figure 1: Key management in Apple and Google's protocol [2]

Cuckoo Filters

A *cuckoo filter* is a probabilistic data structure that allows to verify if an element is included in a set. It's possible to get false positives, but no false negatives. It's a hash table where collisions are handled through *partial-key cuckoo hashing*, a mechanism based on the *cuckoo hashing* method. It's possible to maintain a false positive rate stable at 0.03% by setting a number of bits per entry in the table of 7 bits and the table load at 95.5%, while it's possible to maintain a ≈ 0.001% rate by using 16 bits for the table entries [4]. With *cuckoo filters* it's possible to have a compact representation of the RPI set generated by an infected.

Smart Contracts

Smart contracts allow definition of conditional transactions and blockchain-structured data storage. They are distributedly executed on the network provided by the blockchain on which they're defined. Also, they are publicly inspectable, both in code and stored data.
Smart contracts provide a tool for building a **public** and **transparent** back-end service. In the proposed model, keys compact forms notified by an infected are stored by a smart contract, which will have to emit events where details about new infections are specified.

Cuckoo filters size and *gasLimit*

One limit of using *smart contracts* as backend service is the maximum size of data transferable with a single transaction, which is defined by the *gasLimit*. Cost for data transfer through transactions are defined as follows:

- 68 *gas* for each byte different from 0
- 4 *gas* for each byte equal to 0

At the moment of this document writing, *gasLimit* for Ethereum public network is 10 millions, which would allow to transfer 146 kB. Usage of data structures such as *cuckoo filters* allows to efficiently represent many RPIs. It would in fact be possible to represent ≈ 20000 RPIs using 65 kB.

Protocol details

Contact tracing The contact tracing protocol works in a distributed way through an application that will use the framework made available by Apple and Google. Like previously said, RPIs generated by the user will be broadcasted through BLE (Bluetooth Low Energy), setting the device UUID as

the corresponding RPI. The UIID will be rewritten when the temporary RPI expires, on the release of the following temporary RPI.
At the same time users will listen for nearby devices from which they will catch the emitted RPIs. These RPIs will be stored in a local database for 14 days.

Infection notification When a user undergoes the epidemiological test, he will receive a r code composed of 32 random bytes. The $KEC256(r)$ hash will then be signaled to the *smart contract*, which will keep it inactive until a specific activation request.
In case of positive result of the test, the user will be notified by the predisposed body, that will have activated the previously agreed r code. If the user decides to notify the positive result of the test to the network, the application will build a *cuckoo filter* with all the RPIs generated during the infectious period. The filter will then be serialized and stored by the *smart contract*, using the r code as confirmation code. So, the application will have to execute a transaction that calls the *smart contract* with arguments `(filter, r)`.

Reception of events generated by the blockchain When a new reporting of positivity arrives, the *smart contract* will emit a serialized event such as $(timestamp, filter)$, that will be written inside the block in which the transaction will be included. The event will contain the timestamp of the reporting moment and the serialized form of the *cuckoo filter*.
Devices will have to query the blockchain using events in order to receive these informations, and then rebuild the corresponding *cuckoo filters* from their serialized representation; once this step is done, they will be able to check the local database for any contact at risk. If a match is found, the user will be notified.

Considerations on the infrastructure

The model defined in this documentation is based on the usage of a blockchain system, that could be represented by the main Ethereum network or an its own privately consortium instance. Many Ethereum-based projects allow the implementation of a permissioned blockchain system (Quorum, Hyperledger Burrow, Besu) [8, 6, 3].

Using *smart contracts* for collection and notification of new infects brings the need to distribute native blockchain system value for transactions executions to smart contracts of the system, making impossible its usage without the necessary funds. It's possible to avoid this limit by using **GSN** (Gas Station Network) [5]. Usage of this system allows to delegate the transaction payment to a third party (called *Relayer*) with a decentralized solution that allows the protocol participants to control the submission of new transactions to the network.

Protocol analysis

Security

Below the phases that make up the protocol will be discussed:

(1) During the **contact tracing** phase, all the possible vulnerabilities regard the possibility for a malicious actor to collect the emitted temporary RPIs through tools suitable for scanning the surrounding area.

(2) During the **infection notification** phase, a malicious actor could obtain the r code and notify wrong or malicious informations. Brute force attacks against the r code are limited by the characteristics of the chosen hashing algorithm.

(3) During the **blockchain-generated events reception** phase, the user could download misleading data caused by an r code obtained by a malicious actor. It's possible to mitigate the attack by linking the r code to a defined Ethereum address that belongs to the user.

(4) A malicious actor could collect informations about interations between a user and the smart contract (or the *Relayer*) through traffic analysis techniques. It's however possible to mitigate this kind of attack by using proxy servers and *dummt packets* to anonymize the requests [7].

***Rolling Proximity Identifiers* usage** In the presented model there are used RPIs rather than *Temporary Exposure Keys* introduced with the Apple/Google model [2] to not expose the user to possible analysis based on the generated RPIs. However, an additional technical study about this argument is still in discussion.

Scalability

The decentralized nature of the contact tracing mechanism via BLE makes it scalable as it allows to avoid bottlenecks.
However there are some considerations to make for every individual who uses the application in terms of space occupied in memory and used by the network:

- During contact tracing, each registered record occupies 16 bytes in memory.
- Per la segnalazione o la ricezione di un nuovo contagio i dati utilizzati dipendono dalla dimensione dei *cuckoo filters* generati. Tuttavia i *cuckoo filters* consentono di rappresentare i RPI generati dall'utente mantenendo una dimensione contenuta. For notification or reception of a new infection, necessary data amount depends on the generated *cuckoo filters* size. *Cuckoo filters* allow to represent the RPIs generated by the user maintaining a small size.

References

[1] Google Apple. *Exposure Notification - Bluetooth Specification*. 2020. URL: https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.1.pdf.

[2] Google Apple. *Exposure Notification - Cryptography Specification*. 2020. URL: https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.1.pdf.

[3] *Besu*. URL: https://www.hyperledger.org/projects/besu.

[4] Bin Fan et al. "Cuckoo Filter: Practically Better Than Bloom". In: *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*. CoNEXT '14. Sydney, Australia: Association for Computing Machinery, 2014, 75–88. ISBN: 9781450332798. DOI: 10.1145/2674005.2674994. URL: https://doi.org/10.1145/2674005.2674994.

[5] *GSN - Gas Station Network*. URL: https://www.opengsn.org/.

[6] *Hyperledger Burrow*. URL: https://www.hyperledger.org/projects/hyperledger-burrow.

[7] The DP-3T Project. *Privacy and Security Evaluation of Digital Proximity Tracing Systems*. URL: https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf.

[8] *Quorum*. URL: https://www.goquorum.com/.

[9] Dr. Gavin Wood. "Ethereum: A secure decentralized generalised transaction ledger". In: (2019). URL: https://ethereum.github.io/yellowpaper/paper.pdf.