BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 69(4), 2021, Article number: e137126
DOI: 10.24425/bpasts.2021.137126

Extensions for Apple-Google exposure notification mechanism

Adam BOBOWSKI, Jacek CICHOŃ, and Mirosław KUTYŁOWSKI *

Wrocław University of Science and Technology, Wybrzeże Stanisława Wyspiańskiego 27, 50-370 Wrocław, Poland

Abstract. We analyze the Google-Apple exposure notification mechanism designed by the Apple-Google consortium and deployed on a large number of Corona-warn apps. At the time of designing it, the most important issue was time-to-market and strict compliance with the privacy protection rules of GDPR. This resulted in a plain but elegant scheme with a high level of privacy protection. In this paper we go into details and propose some extensions of the original design addressing practical issues. Firstly, we point to the danger of a malicious cryptographic random number generator (CRNG) and resulting possibility of unrestricted user tracing. We propose an update that enables verification of unlinkability of pseudonymous identifiers directly by the user. Secondly, we show how to solve the problem of verifying the "same household" situation justifying exempts from distancing rules. We present a solution with MIN-sketches based on rolling proximity identifiers from the Apple-Google scheme. Thirdly, we examine the strategies for revealing temporary exposure keys. We have detected some unexpected phenomena regarding the number of keys for unbalanced binary trees of a small size. These observations may be used in case that the size of the lists of diagnosis keys has to be optimized.

Key words: contact tracing; exposure notification; privacy; verifiability; temporary exposure key; rolling proximity identifier; diagnosis key; data sketch; Jaccard similarity.

1. INTRODUCTION

1.1. Social distancing and exposure notification.

Contact tracing is one of the fundamental organizational measures used to fight infectious diseases. Firstly, it may enable an early treatment of an infected person, even before the first symptoms occur. Secondly, one can isolate a potentially infected person to prevent a further spread of the disease. This is particularly important in case of diseases that are difficult to diagnose and/or when an infection can remain unnoticed or misclassified long enough to miss the *golden window period* in a treatment.

Unfortunately, manual contact tracing became quite complex due to growing mobility of the population. In case of epidemics, it turns out that traditional manual procedures quickly lag behind the urgent needs due to a limited manpower and the amount of work to be done.

Another problem is that, as in case of Covid-19, an infected person may be infectious before any symptoms occur. Thereby, when a patient A turns out to be positive, then it should be necessary not only to quarantine every person B that has been in contact with A in a time interval $[t-\delta_1, t]$ for the current time t and δ_1 determined for this particular disease. Also every person C that has been in contact with B in time interval $[t-\delta_2, t]$ should quarantine. The difference $\delta_1 - \delta_2$ is the time that elapses after a person becomes infected till the moment when this person becomes infectious. This situation was one of the initial motivations for developing contact tracing mechanisms, as manual tracing turned out to be too slow to find these secondary contacts on time. The current Apple-Google exposure notification mechanism may enable detection of these potential infection chains.

In practice, a great problem is also a lack of awareness resulting in lack of cooperation with the health authorities as well as hazardous behavior. In some other cases, selfish persons with mild course of the disease do not quarantine and infect other people. Last but not least, reluctant cooperation may be caused by a fundamental lack of trust as well as controversial behavior of health authorities.

Improving the detection ratio of infected persons is crucial for the dynamics of the epidemic. Even if not all infection cases are recognized, once the replication ratio R_0 falls below 1, then the epidemics is gradually dying out.

1.2. Automatic exposure notification.

Organizational means may help to speed up contact tracing. For example, there might be an obligation to leave contact data when visiting places such as restaurants. Unfortunately, all such manual methods have limited effectiveness, are slow and costly to process.

Another line of speeding up contact tracing is to take advantage of electronic payment systems such as WeChat. As most payments in China for everyday chores go through WeChat and similar platforms integrated with communicators and social media, it is easy to trace back some of the contacts in a highly efficient way. For example, as in many cases tickets for public transportation are bought online via these platforms, one can relatively easily derive many contacts in public transportation. Unfortunately, in countries with online payments based on bank card and credit card cashless payments, the system cannot sup-

*e-mail: miroslaw.kutylowski@pwr.edu.pl

Manuscript submitted 2020-12-31, revised 2021-03-08, initially accepted for publication 2021-03-08, published in August 2021

port such functionalities without substantial technical changes as well as solving complex legal issues related to personal data processing.

The main breakthrough for automatic contact tracing was developing exposure notification systems based on Bluetooth Low Energy (BLE) protocol running on smartphones. In such a system each smartphone broadcasts pseudorandom identifiers and some encrypted metadata to its neighbors in a close proximity. These data are stored by the receivers and retained for a possible alert. The main idea is that if data over the BLE link have been received, then we may suppose that the smartphones' holders have been in a close proximity enabling a disease transmission.

After testing positive, there are two alternative ways to trace the contacts:

option 1: the identifiers received and stored by the device of the infected person correspond to persons that have been in contact – these persons must be warned. This requires deanonymization of pseudorandom identifiers of the persons to be warned.

option 2: the identifiers sent by the infected person are revealed. Then a device that has stored any of these identifiers in the past issues a warning to its holder.

Option 1. This approach has been implemented quite fast in Singapore within the Trace Together Program [1]. Similar approaches have been also promoted in an early period of the pandemic in Europe. Later it has been abandoned there due to heavy criticism regarding privacy protection. Indeed, since it is necessary to deanonymize the pseudorandom identifiers by health authorities, the trapdoor information of all users must be retained. Therefore contact tracing may enable creation of an efficient country-wide surveillance system run by state authorities. Interestingly, this approach was supported at some moment by German Robert-Koch-Institut, despite that Germany was one of leaders in pushing forward personal data protection rules in the EU.

In Singapore the tracing app has been augmented by a system of contact tracing wearables. In contrast to the apps, the wearables cannot connect to internet – they support only local communication. In all cases, if a person is tested positive, then the list of identifiers received and stored in the device is returned to the Singaporean health authority. At this moment the health authority has to derive the source of these identifiers. This is possible since the pseudorandom temporal identifiers are created by encrypting the user ID with a private key held by the Singaporean Ministry of Health. So the authority can decrypt a pseudorandom identifier and get the user ID. In turn, the real ID is recovered by a query to to a protected database, where links between the real ID's and the user ID's are stored.

Option 2. For the second approach the role of central authorities is limited – they provide necessary data to the users, but themselves are unaware of the contacts occurred. In case of a positive diagnosis the smartphone of an infected person uploads relevant data to a so-called *Diagnosis Server*. This server periodically creates a list of *diagnosis keys* originating from infected users in the relevant period. The lists of diagnosis keys are regularly downloaded by the apps on the smartphones participating in the system. A diagnosis key enables recomputing all pseudorandom identifiers sent by the smartphone of an infected person in the relevant period. So, after downloading the list of diagnosis keys an app can derive the corresponding pseudorandom identifiers and compare them with the identifiers received over the BLE protocol and stored internally. If a common entry or entries are detected, then an exposure notification is issued by the app to the smartphone holder (and possibly to a health authority).

Apart from recomputing the pseudorandom identifiers of a given period, a diagnosis key enables deriving the decryption keys for the encrypted metadata sent together with the identifiers. These metadata could be used to provide additional epidemic information. They can be also used to guard against attacks such as replaying the same identifiers at a different place.

A serious disadvantage of this approach is the dependence on participants' cooperation. On the other hand, it impedes the creation of a global surveillance system where all personal contacts are recorded and available for data processing.

1.3. Apple-Google contact tracing mechanism.

A major step in creating contact tracing solution was the platform developed jointly by Apple and Google. The crucial feature of this solution is implementing critical functions in the operating system and not on the application level. Therefore, one can provide more opportunities for creating customized applications based on a common well designed and tested platform. Moreover, Apple and Google may admit to their app stores only *official* apps for contact tracing and restrict the use of BLE signaling otherwise.

The main assumption of the architecture developed by Apple-Google is relying on user's self-control and not on any central authority. They aimed to provide a pragmatic platform that would generate a substantial amount of data for an early warning, but on the other hand that would follow the principles of privacy protection and data minimality of GDPR [2] – as far as it seems to be possible.

The Apple-Google platform has been used by the Corona-Warn-App consortium in Germany [3]. Compared to Singapore, it did not prove to be effective enough. The problem was a substantially smaller percentage of citizens using it (in both countries using contact tracing products is voluntary).

1.4. DP3-T.

The solution philosophy for Apple-Google contact tracing is shared with the Decentralized Privacy-Preserving Proximity Tracing (DP3-T) open source project run by a number of European universities and research institutions (see the DP3-T white-paper [4]). Their goal was to develop a privacy-preserving solution together with sound arguments regarding resilience to different attack endangering users' privacy. DP3-T describes three decentralized exposure notification mechanisms:

- **low-cost:** this is a simplified version of the Apple-Google mechanism described in Section 1.5, the crucial difference is

that the daily seeds are derived in a hash chain and therefore it suffices to report one key per diagnosed user,
- **unlinkable:** for each (short) epoch a new seed is chosen at random; while reporting a diagnosed person, a Cuckoo filter is used instead of an explicit list of seeds,
- **hybrid:** it is a compromise between both approaches– a seed may be shared by a number of pseudorandom identifiers, as in the case of the Apple-Google platform.

1.5. Apple-Google cryptographic details.

As we shall focus on the Apple-Google platform, we recall the details of this concept [5].

1.5.1. Broadcasting identifiers.

- The time is divided into 10 minute periods. For each period we assign an index *ENIntervalNumber* based on UNIX Epoch time as a 32-bit little endian value:

$$ENIntervalNumber(timestamp) = timestamp/(60 \cdot 10). \quad (1)$$

  In turn, 144 periods (which correspond to 24 hours) make a *TEKRollingPeriod*.
- For each *TEKRollingPeriod* a user's smartphone generates a separate *Temporary Exposure Key* (TEK). The 16-byte TEK key is computed by a cryptographic random number generator: $tek_i = \mathrm{CRNG}(16)$. No further details are specified.
- A TEK key is used to derive two other keys associated to the same *TEKRollingPeriod*. The first one is a *Rolling Proximity Identifier Key* (RPIK):

$$RPIK_i = \mathrm{HKDF}(tek_i, \mathrm{NULL}, \mathrm{UTF8}(\text{“}EN\text{-}RPIK\text{”}), 16),$$

  where HKDF stands for ”*the HKDF function as defined by IETF RFC 5869, using the SHA-256 hash function*”.
- The second key derived from a TEK is a so-called *Associated Encrypted Metadata Key*:

$$\mathrm{AEMK}_i = \mathrm{HKDF}(tek_i, \mathrm{NULL}, \mathrm{UTF8}(\text{“EN-AEMK”}), 16). \quad (2)$$

- During a 10 minutes period the smartphone is using a randomized BLE MAC address. During this period it broadcasts a pseudorandom *Rolling Proximity Identifier* together with the *Associated Encrypted Metadata*. The rolling proximity identifier $rpi_{i,j}$ for the *TEKRollingPeriod* i and a period j is computed as follows:

$$rpi_{i,j} = \mathrm{AES}(RPIK_i, PaddedData_j), \quad (3)$$

  where $PaddedData_j$ is the following 16-byte string:

$$\mathrm{UTF8}(\text{“EN-RPI”}), 0x000000000000, ENIntervalNumber(j),$$

  where $ENIntervalNumber(j)$ encodes the time when the period j starts according to the formula (1). Together with $rpi_{i,j}$ the smartphone sends *Associated Encrypted Metadata* corresponding to the *TEKRollingPeriod* i and the period j, which is an AES counter mode ciphertext:

$$\mathrm{AESCTR}(\mathrm{AEMK}_i, rpi_{i,j}, Metadata).$$

1.5.2. Recording contacts.

- Each active device collects the pseudorandom identifiers received over the BLE channel. The distance between devices can be estimated by the strength of the BLE signal. Only the identifiers corresponding to strong signals are stored, as the rest is likely to correspond to a safe distance. Similarly, if an identifier is not received repeatedly for a number of times, then it is ignored as the contact is likely to be too short for an infection.
- If according to the policies mentioned above an identifier $rpi_{i,j}$ should not be ignored, then it is stored together with the time of receipt and the corresponding AESCTR ciphertext.

1.5.3. Reporting positive diagnosis.

- When a user is diagnosed positive, he or she should use the app on the smartphone to send a warning to the Diagnosis Center. Entering an authorization code obtained from the health authority should be required in order to avoid false warnings (e.g. preventing an attack aiming to quarantine the people from the personal surrounding – like project team members). The warning contains the TEK keys corresponding to all days where the smartphone's holder has been potentially infectious (e.g. 14 TEK keys). For each TEK key its *TEKRollingPeriod* is provided as well.
- The Diagnosis Server aggregates all data received to one list of diagnosis keys and makes it available for downloading.
- The user's app periodically downloads the current list of diagnosis keys. For each TEK key from the list the app derives the corresponding rolling proximity identifiers according to the formula (3). The results are compared with the rolling proximity identifiers received by the smartphone over the BLE channel in the reported time. A certain discrepancy between the time of key creation and scanning is allowed. If a match is found, then the AESCTR ciphertext may be decrypted with the key derived with the formula (2). Finally the app may issue an *exposure notification*.

The white paper [5] does not specify what kind of data (if any) is contained in the ciphertext AESCTR, it only indicates that this is an non-authenticated encryption mode and the plaintext should be separately validated.

How the exposure notification is used is a different issue. One option would be to leave the decision to the smartphone owner, who voluntarily quarantines and informs a health authority. Another option is that the app automatically sends a notification to a health authority revealing in particular the identifier of the smartphone. Additional mechanisms might be used when not only a contact person of an infected person must be warned but also each contact of the contact person.

2. CRNG AND PRIVACY PROTECTION BY APPLE-GOOGLE SCHEME

2.1. CRNG and strong privacy.

The Apple-Google and DP3-T designs refer to generating random numbers as kind of a golden bullet of privacy protection. Namely, one should take into account an advanced adversary that can break the cryptographic schemes used to compute the rolling proximity identifiers. Such an adversary may exist

now (e.g., the most powerful security agencies not sharing all research results with the public), or may emerge in near future due to advances in cryptographic technologies. Potentially, such an adversary might be able to derive all values used to compute the rolling proximity identifiers (that is, the internal state of a device broadcasting BLE signals) and then recompute the identifiers sent at different times. Thereby the allegedly unlinkable identifiers would become linkable.

Estimating resilience of cryptographic schemes is always based on currently available knowledge on cryptanalytic state-of-the-art. We neither know for sure what are the capabilities of the most powerful cryptanalysts now nor we can predict the technical progress in the future. As the BLE signals can be systematically captured and stored for a future use, users' location privacy may be eventually violated. So unless there are some effective countermeasures implemented, it may be hard to deny that deploying contact tracing may have a side effect of creating a large scale surveillance system.

However, if rolling proximity identifiers are created from random numbers chosen anew each day, then linking between different days becomes impossible, regardless of the progress of cryptanalytical techniques. Therefore, despite the necessity to provide more data related to a diagnosed person, the proposed schemes refresh the cryptographic seeds at random. While the *low-cost* design from DP3-T is not involving a CRNG, there is a clear recommendation in [4] to move to the *unlinkable* or at least to the *hybrid* design, where the cryptographic material is refreshed frequently. According to the Apple-Google design, for each day a fresh random seed is chosen. This prevents linking the rolling proximity identifiers of the same device when two different days are concerned no matter how powerful the adversary is.

2.2. CRNG reality and tracing threats.

A solution based on a CRNG looks nice in theory, but in practice it may become a treacherous trapdoor. Namely, according to the current good practices, a CRNG should be implemented as a pseudorandom random number generator (PRNG) device using a (potentially imperfect) source of entropy and a secret seed. The seed and the entropy input are processed by one-way functions so that some output bits are generated as well as an update to the internal state of the PRNG.

A quite realistic threat is that the PRNG is pretending to work in this way, but in fact it is deterministic and ignores the entropy input. For standard designs of CRNG it is easy to build such PRNG. If the attacker can determine the initial seed of the PRNG, then he will be able to derive all random bits used by the device. On the other hand, an observer having no access to the internal state cannot determine whether the output comes from a genuine CRNG or from a PRNG ignoring the entropy. So it is necessary to conclude that from the technical point of view the CRNG is a particularly vulnerable element of the system.

2.3. Defense.

An audit of the app's code and of the random number generator from the operating system may not convince every user. Some of them may be afraid of the manufacturers colluding with the auditors and supervision authorities in order to create a large scale surveillance system. A good approach against such fears is *local verifiability* – procedures enabling a user to check privacy safeguards by himself.

The scheme proposed by Apple-Google can be easily supplemented with such privacy safeguards verifiable by a user:

- The user inputs a *user seed* u to his device D. As a rule, the user has to generate u outside D in order to remain independent from D. The user seed should have enough entropy to prevent guessing u by the attacker.
- In parallel to deriving the TEK keys, the device computes *user rolling modifiers*. Namely, for the jth period of a TEKRollingPeriod i it computes a 16 byte modifier
$$m_{i,j} = C\big(C(i,u),j\big),$$
where C is a cryptographic random number generator.
- The definition of rolling proximity identifiers changes to
$$rpi_{i,j} = \mathrm{Hash}\big(m_{i,j}, \mathrm{AES}\big(RPIK_i, PaddedData_j\big)\big). \tag{4}$$
- In case of a positive diagnosis, apart from the TEK keys, the device D should upload also the seed $C(i,u)$ for each day concerned.
- Recomputing the rolling proximity identifiers corresponding to infected persons is based on the equation (4).

Note that an adversary that corrupts the CRNG cannot link two rolling proximity identifiers if a Correlated Input Secure functions Hash and C are used [7]. Moreover, he cannot even link the keys $C(i,u)$ and $C(i',u)$ from the list of diagnosed keys for $i' \neq i$. Roughly speaking, a function F is correlated input secure, if for given inputs α, β and values h_1, h_2 it is infeasible to decide whether there is a parameter u such that $h_1 = F\big(C_1(u),\alpha\big)$ and $h_2 = F\big(C_2(u),\beta\big)$, where C_1 and C_2 are some simple circuits given as part of the input. This is exactly the problem a powerful adversary aims to solve: for two rolling proximity identifiers r_1, r_2 the adversary can compute the candidate arguments $\mathrm{AES}\big(RPIK_i, PaddedData_j\big)$ and $\mathrm{AES}\big(RPIK_{i'}, PaddedData_{j'}\big)$ and then ask if there are matching parameters $m_{i,j}$, $m_{i',j'}$ obtained via PRNG from the same seed such that

$$\begin{aligned} r_1 &= \mathrm{Hash}\big(m_{i,j}, \mathrm{AES}\big(RPIK_i, PaddedData_j\big)\big), \\ r_2 &= \mathrm{Hash}\big(m_{i',j'}, \mathrm{AES}\big(RPIK_{i'}, PaddedData_{j'}\big)\big). \end{aligned}$$

The user can check that the computation of the rolling proximity identifiers involved the modifiers in the way described. The verification procedure for a given day i is as follows:

1. the user records the rolling proximity identifiers of the own device D (preferably, some other computing device should be used),
2. the user chooses at random $j_1, \ldots, j_k$ and inputs them to D,
3. D returns the values
$$rpi'_{i,j} = \mathrm{AES}\big(RPIK_i, PaddedData_j\big),$$
for $j = j_1, \ldots, j_k$,
4. the user recomputes $m_{i,j}$ from u and checks whether
$$rpi_{i,j} = \mathrm{Hash}\big(m_{i,j}, rpi'_{i,j}\big),$$
for $j = j_1, \ldots, j_k$.

Note that probing a device does not create any additional threat in case of a malicious user running an attack $\mathscr{A}$ to derive the internal state of his own device. Indeed, if D is running the original protocol, then the user can create the input for $\mathscr{A}$ by setting a seed u and recomputing the rolling proximity identifiers according to (4). Obviously, then one can emulate the attack $\mathscr{A}$ on the modified rolling proximity identifiers and the data received from the verification procedure.

3. SOCIAL DISTANCING SMART MANAGEMENT

In this section we show that on top of the contact tracing mechanism one can build other functionalities that might be useful for management of social distancing. The goal is to maximize the effect of distancing and at the same time provide tools that enable to lift certain restrictions.

3.1. Same-epidemic-situation evidence mechanism.

Our goal in this subsection is to create an electronic evidence for two people of being in the same household or otherwise staying long enough in a close proximity, so that one can assume that these people are already in the same epidemic situation. In this case, they can be allowed to stay in a close proximity regardless of the social distancing rules. However, in order to secure against misusing this exempt, the case should be easy to verify in a reliable way.

As in some situations unconditional social distancing rules do not contribute to safety but increase the cost and burden of social distancing, some intermediate solutions have been adopted. For instance, in Germany in April 2020 it was forbidden to meet in public places in groups of more than 2 people. At the same time, there was no limit for people living in the same household.

This simplified approach has many disadvantages, the main are discrepancy between the formal status and the real situation as well as limited verifiability that are necessary if this policy has to be really enforced (especially in countries, where residence registration procedures do not reflect the real situation, e.g., in Poland).

We propose a simple mechanism for an app based on the Apple-Google contact tracing mechanism. The key component is a probabilistic sketch data structure equivalent to the popular MinHash schema (see [8]). This data structure has been designed for somewhat different purposes, however, it is well suited for our purposes. For a given stream of data $\{S\}_i$ it holds a fixed number of registers, say $R_1, \ldots R_k$. Each register is initially empty, and later it may store exactly one element of the stream. Assume that an element e_i arrives. Then we compute the values

$$w_1 = \mathrm{Hash}_1(e_i), \ldots, w_k = \mathrm{Hash}_k(e_i),$$

where $\mathrm{Hash}_1, \ldots, \mathrm{Hash}_k$ are independent cryptographic hash functions. Then for $i \leq k$ the current contents e of R_i is replaced with e_i if and only if $w_i < \mathrm{Hash}_i(e)$. For the later discussion, the hash values will be treated as binary numbers from the interval $[0, 1)$.

Building up data sketches. For the purpose of the same epidemic situation evidence each app holds a sketch based on the received rolling proximity identifiers. There are separate sketches – one per day. A sketch is erased once it is older than t_s days, where t_s is a parameter determined depending on the policy. Also the number k of registers in a sketch is a design parameter.

The received rolling proximity identifiers are inserted into a sketch in the following way:

- let r be the rolling proximity identifier received, let $r_{i,j}$ be the rolling proximity identifier computed by the app for broadcasting at this moment,
- the item to be inserted in the sketch is

$$f\big(r, r_{i,j}\big) = \mathrm{fgrprt}\Big(\mathrm{Sort}\big(r, r_{i,j}\big)\Big). \tag{5}$$

where fgrprt is for instance a cryptographic hash truncated to a relevant number of bits.

Note that if a device A is in a proximity of device B, then A and B receive mutually their rolling proximity identifiers. Since the rolling proximity identifiers change at A and B at the same time (up to small differences resulting from hardware discrepancies), A and B attempt to insert the same entry computed according to (5) to their sketches.

Properties of the sketches. For a while let us consider a single register of a sketch for a single day.

Note that if the devices A and B stay in a close proximity for a longer time, then their apps systematically try to enter the same entries into their data sketches. Whether a pair (r, r') will finally result in an entry placed in a sketch depends very much on the value of $f_i(r, r')$. As the sketches of A and B are influenced by rolling proximity identifiers sent by third parties, it may happen that $f_i(r, r')$ is small enough to be inserted into one of the sketches, while it looses against an entry already stored in the other sketch. Observe that such an entry does not originate from a contact between A and B. However, if $\mathrm{Hash}(r, r')$ is small, then it is likely that it will be inserted in both sketches.

Let $\big[R_1^A, \ldots, R_k^A\big]$ and $\big[R_1^B, \ldots, R_k^B\big]$ be the vectors of sketches of devices A and B. Let

$$s(A, B) = \frac{1}{k} \cdot \big|\big\{i : R_i^A = R_i^B\big\}\big|.$$

Then $s(A, B)$ is an approximation of the Jaccard similarity between the sets of all values $f(r, r')$ computed by both devices. Recall that the Jaccard similarity between finite sets X and Y is defined as

$$\frac{X \cap Y}{X \cup Y}.$$

Assume for instance that that a fraction p_A of all pairs composed by A correspond to the contacts with B. Similarly, assume that a fraction p_B of the contacts of B is with A (as the total number of contacts of A and B may differ, in general p_A need not to be equal to p_B). Then the Jaccard similarity between the observed sets is

$$\frac{p_A p_B}{p_A + p_B - p_A p_B}.$$

So, if $p_A, p_B > 1/2$, then $s(A, B) > 1/3$ with very high probability. From this formula we also deduce that if $p_A = p_B = \varepsilon \approx 0$, then $s(A, B) \approx 1/2\,\varepsilon$.

Note that we may take the parameter values such as $k = 256$ and the fingerprints consisting of 40 bits. Then the resulting sketch for one day will have the size of only 10 kB. If the sketches are kept for 14 days, then it will result in only 140 kB memory usage. So definitely, there is room for even larger choice of parameters taking into account the people that regularly meet a bigger group of people than just a few family members.

Inspection. The last procedure to be described is the case when the persons holding devices A and B are controlled whether they are in the same epidemic situation. The simplest solution would be to present the sketches by A and B to the controller. However, this is not the right solution due to the principle of data minimality: the controller gets an estimation of the number of rolling proximity identifiers received (see Section 3.2).

Any scheme for showing similarity of the sketches without revealing them would work. For example the following steps may be executed by A and B and a controller C:

1. A, B and C choose collectively a random element κ,
2. A and B transform their sketches using the element κ; namely, for an entry e the transformed value is $\mathrm{Hash}(e, \kappa)$,
3. A and B present the transformed sketches to C over separate encrypted channels,
4. C checks what is the fraction of common entries in the received transformed sketches. This fraction is computed separately for each daily sketch. Thereby the controller learns Jaccard similarity of the sketches and makes a decision based on the current policy.

3.2. Intensity of contacts.

As a side product, the data sketch\-es introduced in Section 3.1 can be used for estimating the number of contacts of a device. This is quite important since the infection risk depends very much on the number of people one has met in a given period. Namely, let us consider once again the sequence $w_1, \ldots, w_k$ of hashes computed from values stored in registers $R_1, \ldots R_k$. Then, for each $i = 1, \ldots, k$ the number w_1 is a realization of a random variable with distribution $\min\{U_1, \ldots, U_n\}$, where U_i are independent random variables with the uniform distribution on the interval $[0, 1]$ and n is the number of distinct contacts. Therefore the density function of the random variable w_i is given by the formula $f(x) = n(1-x)^{n-1}$. The standard maximum likelihood estimator method yields the following estimator of n:

$$\hat{n} = \frac{-k}{\sum_{i=1}^{k} \ln(1 - w_i)}$$

$\hat{n}$ is a consistent estimator of n, that is, it converges in probability to the true value n, as the sample size k goes to infinity. Fig. 1 presents the results of the experiments carried out to test the precision of this estimator. The figure shows the values of the quotient $\hat{n}/n$ for n from 1 to 1000 with fixed $k = 128$. For each n, the experiment was repeated 10 times. Let us notice that the precision of the estimator $\hat{n}$ increases with the increase of the k parameter. The conducted experiments show that for $k \geq 128$ the accuracy of this estimator is of 25% order, which should be sufficient for our needs. In these experiments, less than 1% of the results differed from n by more than 25%.

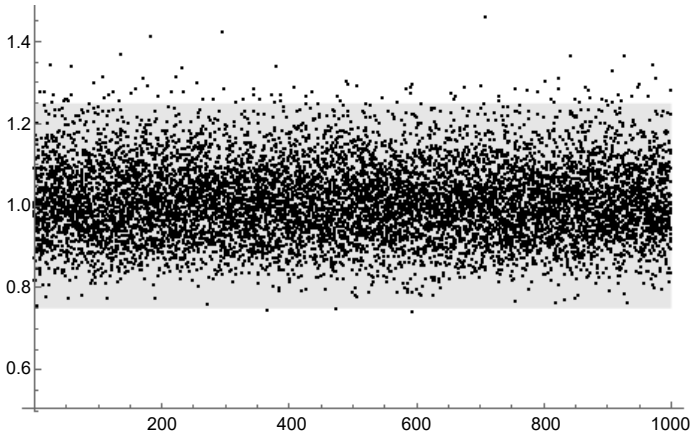

Fig. 1. Plot of experimental values of $\hat{n}/n$ for n from 1 to 1000 where $k = 128$. The gray area indicates values between 0.75 and 1.25

A data sketch has the advantage that the decision to change the sketch is memoryless and depends only on the current rolling proximity identifiers. Hence, it is not necessary to remember if a given pair (r, r') has been already used.

3.3. Quarantine.

A potentially infected person might be obliged to quarantine. Typically this means the requirement not to leave a fixed place. However, the essential target is different: the number of contacts with this person should be reduced to what is really necessary. So in case of COVID-19, there is no reason to ban leaving home as long as the person concerned avoids coming to a closer distance with others.

Fortunately, contact tracing platform makes creation of such a system relatively simple. However, for supervision of a quarantined person a second device H (e.g., a smartphone) provided by the health authority is necessary. The solution is based on the following assumptions concerning a person P in quarantine:

1. contact tracing is enabled on D and H including creation of the data sketches according to Section 3.1,
2. D is kept by the person P all the time,
3. D is not shielded in a Faraday cage,
4. device H remains in the range of D all the time.

If these assumptions are met, then it suffices to run the control procedure described in Section 3.1 for the devices D and H. The sketches should store no entries other than the common ones for D and H except for maybe a few accidental entries. Fulfilling the assumptions can be enforced in the following way:

ad 1 and 4 – A logging mechanism prevents P from any operation on H, i.e. switching it off. On the other hand, enabling contact tracing by D is witnessed by H. Namely, the sketches created by H must contain entries corresponding to D. In order to make cheating harder, H will send and receive rolling proximity identifiers rarely, at unpredictable moments.

A failure to receive a rolling proximity identifier witnesses a misbehavior.

ad 2 – Verification can be based on (automatic) calls to D at random moments that must be answered in person by P. Moreover, one can deploy an automatic trigger that starts audio recording if the voice characteristics do not match the known characteristics of the voice of P.

ad 3 – Receiving rolling proximity identifiers of third persons may be prevented by shielding all signals by a Faraday cage. To ensure shielding H as well, both devices might be inside the cage. However, H may monitor the signal strength. As the person in quarantine might be obliged to keep D and H at some minimal distance, an attempt to cheat would require some skills to provide appropriate Faraday cage – small enough to carry and reducing the signal strength between D and H. While this is technically possible, most users have no sophisticated engineering experience to cheat in this way.

4. MANAGING THE DAILY KEYS

According to the Apple-Google design, TEK keys of a diagnosed person form a certain period have to be uploaded to the Diagnosis Center. The number of these keys equals the number of days when this person could be infectious. In case of COVID-19 this number was set to 14, however it may change according to new statistics and virus mutations.

One of the consequences of a separate TEK key for each day is that an exposed user cannot determine whether the contacts with an infected person on days i and j (as indicated by the diagnosis keys and the stored rolling proximity identifiers) can be attributed to one or two different persons.

This approach might be justified by the need of privacy protection. However, this argument can be challenged: for a given TEK key τ a person P can see for which 10-minute periods the rolling proximity identifiers derived from τ has been received by P. This typically indicates who might have been the infectious person. On the other hand, from the epidemic point of view the probability of disease transmission could be different depending on whether two TEK keys from two different days correspond to the same person or to two different persons. On the other hand, the separate TEK per day policy eases description of the scheme.

Reducing the blacklist size. In case of the DP3-T low-cost solution it suffices to show a single TEK key on the list of diagnosis keys per person. This is a clear advantage against the Apple-Google mechanism, but the price paid is that leaking a single TEK key means lack of privacy from the day of this TEK key. However, there is a compromise between two approaches. Namely, the time can be divided into k day periods, called TEK-periods. For each TEK-period we choose at random a master key m. Then the TEK keys are derived in the same way as in case of the low-cost DP3-T: the ith TEK key tek_i for $i > 1$ is defined as $tek_i = \mathrm{Hash}(tek_{i-1})$, while $tek_1 = m$.

Let us assume that the *infection* period for which the TEK keys must be derived consists of q days. In general this period need not be contained in a single TEK-period. If $k \geq q$, then each infection period is contained in one or two TEK-periods. Then it suffices to provide the tek_i key from the first period for the day when the infection period starts, and the tek_1 key for the second TEK-period.

The last approach has one disadvantage: all TEK keys from the second period are revealed, including those that are not in the infection period concerned. There is a simple remedy for that, however the price to pay is the number of keys to be transmitted to the diagnosis center. Namely, for each TEK-period we choose m and m' at random. Then we compute $f_0 = m$ and $f_{i+1} = \mathrm{Hash}(f_i)$ for $i > 1$, $b_k = m'$ and $b_i = \mathrm{Hash}(b_{i+1})$ for $i < k$. Finally, $tek_i = \mathrm{Hash}(f_i, b_i)$.

Observe that now 4 keys have to be posted on the list of diagnosis keys per one diagnosed person. This is still much better than $k = 14$ in case of the Apple-Google mechanism. However, there is a subtle problem. It may happen that within an infection period a given person spent first a days in region A and then b days in region B. In this case we would like to partition the infection period in two intervals (corresponding to stay in, respectively, region A and B) and provide the diagnosis keys corresponding to each interval in the regional diagnosis key lists. The solution presented above enables linking these key on the diagnosis lists, while Apple-Google solution is immune against such problems.

A standard approach that can be applied is creating a binary tree with k leaves. In such a tree we put a master secret m in the root and compute the labels downwards: for a node with a label z, the label of its right son is $\mathrm{Hash}(z, 0)$ and the label of its left son is $\mathrm{Hash}(z, 1)$. The labels of k leaves are the daily TEK keys. If this is a balanced binary tree for k equal to power of two, then in order to reveal the labels of a suffix or a prefix of the list of leaves, it suffices to present at most $\log k$ labels. As two trees may be involved for a given infection period, this results in an upper bound of $2\log k$ labels.

The things become more complicated when we are talking about k's that are not powers of 2 and if we are interested exclusively in small values of k. In this case the tree cannot be fully balanced. The question we state is what is the number of labels (on average) that one has to show given different choices for k and different choices to balance the tree. We examine the situation by a exhaustive search through the set of trees.

The first set of experiments concerns the strategy where we attempt to balance the tree as much as possible. That is, if a (sub)tree T should have an odd number of leaves ℓ, then the first $\frac{\ell-1}{2}$ leaves are assigned to the left subtree and $\frac{\ell+1}{2}$ leaves are assigned to the right subtree.

Table 1 presents the experimental results for this case. The entries in the table denote the average number of labels that have to be shown to reconstruct the TEK keys of a given infection period of length q. Each row of the table corresponds to a fixed value of k while for a column the value q is fixed.

Table 2 shows the results where we do not attempt to balance the tree and consider arbitrary trees. Moreover, we are not examining the average case but the worst case.

As we can see from the tables, there are some intriguing phenomena that would be hard to predict. For instance, having

Table 1

Average number of keys required for covering infection period of length q: balanced trees strategy

number of leaves	the length of the infection period q:																	
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$k = 2$:	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10.0
$k = 3$:	1.33	2.0	2.33	2.67	3.0	3.33	3.67	4.0	4.33	4.67	5.0	5.33	5.67	6.0	6.33	6.67	7.0	7.33
$k = 4$:	1.5	2.0	2.25	2.5	2.75	3.0	3.25	3.5	3.75	4.0	4.25	4.5	4.75	5.0	5.25	5.5	5.75	6.0
$k = 5$:	1.4	2.0	2.4	2.6	2.8	3.0	3.2	3.4	3.6	3.8	4.0	4.2	4.4	4.6	4.8	5.0	5.2	5.4
$k = 6$:	1.5	2.0	2.33	2.5	2.83	3.0	3.17	3.33	3.5	3.67	3.83	4.0	4.17	4.33	4.5	4.67	4.83	5.0
$k = 7$:	1.43	2.0	2.0	2.43	2.71	3.0	3.14	3.29	3.43	3.57	3.71	3.86	4.0	4.14	4.29	4.43	4.57	4.71
$k = 8$:	1.5	2.0	2.25	2.5	2.88	3.0	3.12	3.25	3.38	3.5	3.62	3.75	3.88	4.0	4.12	4.25	4.38	4.5
$k = 9$:	1.44	2.0	2.44	2.67	3.0	3.22	3.22	3.33	3.44	3.56	3.67	3.78	3.89	4.0	4.11	4.22	4.33	4.44
$k = 10$:	1.5	1.9	2.5	2.6	2.8	3.1	3.3	3.4	3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	4.3	4.4
$k = 11$:	1.45	2.0	2.27	2.55	2.73	2.91	3.27	3.45	3.55	3.64	3.73	3.82	3.91	4.0	4.09	4.18	4.27	4.36
$k = 12$:	1.5	2.0	2.25	2.58	2.83	3.0	3.17	3.5	3.58	3.58	3.75	3.83	3.92	4.0	4.08	4.17	4.25	4.33
$k = 13$:	1.46	2.0	1.92	2.38	2.77	2.92	3.08	3.38	3.62	3.69	3.69	3.85	3.92	4.0	4.08	4.15	4.23	4.31
$k = 14$:	1.5	2.0	2.0	2.43	2.79	3.0	3.14	3.36	3.57	3.57	3.71	3.79	3.93	4.0	4.07	4.14	4.21	4.29
$k = 15$:	1.47	2.0	2.2	2.4	2.8	3.0	2.93	3.13	3.33	3.6	3.67	3.8	3.87	4.0	4.07	4.13	4.2	4.27
$k = 16$:	1.5	2.0	2.25	2.38	2.81	3.0	3.12	3.25	3.5	3.62	3.81	3.88	4.0	4.0	4.06	4.12	4.19	4.25
$k = 17$:	1.47	2.0	2.47	2.59	2.88	3.18	3.35	3.41	3.59	3.76	3.94	4.06	4.12	4.18	4.12	4.18	4.24	4.29
$k = 18$:	1.5	2.0	2.5	2.61	2.83	3.17	3.28	3.33	3.44	3.67	3.83	4.06	4.11	4.22	4.28	4.22	4.28	4.33
$k = 19$:	1.47	1.95	2.53	2.58	2.63	3.0	3.26	3.42	3.47	3.63	3.84	3.95	4.16	4.21	4.32	4.37	4.32	4.37
$k = 20$:	1.5	2.0	2.5	2.6	2.7	2.95	3.25	3.45	3.5	3.6	3.8	3.95	4.05	4.2	4.25	4.35	4.4	4.4

Table 2

The maximum number of labels to be shown for a given number of leaves k in a tree and infection period q. The results concern the optimal tree found

number of leaves	the length of the infection period q:																		
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$k = 2$:	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11
$k = 3$:	2	3	3	3	4	4	4	5	5	5	6	6	6	7	7	7	8	8	8
$k = 4$:	2	2	3	3	4	3	4	4	5	4	5	5	6	5	6	6	7	6	7
$k = 5$:	2	3	3	4	4	4	5	4	5	5	5	6	5	6	6	6	7	6	7
$k = 6$:	2	3	3	3	4	4	4	5	5	4	5	5	5	6	6	5	6	6	6
$k = 7$:	2	2	3	3	3	4	4	5	4	5	5	4	5	5	6	5	6	6	5
$k = 8$:	2	2	3	3	4	3	4	4	5	4	5	5	6	4	5	5	6	5	6
$k = 9$:	2	3	3	4	4	5	4	5	5	5	6	5	6	6	7	5	6	6	6
$k = 10$:	2	3	3	4	4	4	5	4	5	5	5	6	5	6	6	6	7	5	6
$k = 11$:	2	3	3	3	4	4	5	5	4	5	5	5	6	6	5	6	6	6	7
$k = 12$:	2	3	3	3	4	4	4	5	5	4	5	5	5	6	6	5	6	6	6
$k = 13$:	2	2	3	3	4	4	4	4	5	5	4	5	5	6	6	6	6	6	6
$k = 14$:	2	2	3	3	3	4	4	5	4	5	5	4	5	5	6	5	6	6	5
$k = 15$:	2	2	3	3	4	3	4	4	5	5	5	6	4	5	5	6	5	6	6
$k = 16$:	2	2	3	3	4	3	4	4	5	4	5	5	6	4	5	5	6	5	6
$k = 17$:	2	3	3	4	4	5	4	5	5	6	5	6	6	7	5	6	6	6	7
$k = 18$:	2	3	3	4	4	5	4	5	5	5	6	5	6	6	7	5	6	6	6
$k = 19$:	2	3	3	4	4	4	5	4	5	5	6	6	6	6	6	7	5	6	6
$k = 20$:	2	3	3	4	4	4	5	4	5	5	5	6	5	6	6	6	7	5	6

the infection period $q = 14$, the right choice for k is 15. This holds also for balanced trees and the average value instead of the worst case value. Perhaps even more surprising is the fact that in this case the number of labels in the worst case is the same as number of labels in every case, for the scheme with forward and backward keys f_i, b_i. Of course, the average case is much better than 4.

5. CONCLUSIONS

The general idea of Apple-Google platform for exposure notification is elegant, simple, transparent and therefore likely to be widely accepted by average users. Effectiveness of such tools has been proven in some countries, where initial frantic search for the infection chains has been replaced by a systematic data acquisition taking advantage of possibilities given by wearable user devices. Despite high density of population in countries such as Singapore strict controls have proved to be extremely helpful in reducing the number of infections.

We have shown that there are still unexplored additional potentials in the Apple-Google platform. Our main contribution is a possibility to automatically verify that two persons stay together in the same household or are in a continuous contact due to other reason like professional activities. In this case there should be no restriction for them to meet and stay together in other places.

The issue not discussed in depth so far are potential functionalities based on the *Associated Encrypted Metadata*. As already mentioned, they can be used against relay attacks, but presumably there are many other opportunities. The key is to find the most relevant choice given a limited bit size available.

REFERENCES

[1] Ministry of Health and Government Technology Agency (GovTech), *Trace Together Programme*, [Online]. Available: https://www.tracetogether.gov.sg.

[2] The European Parliament and the Council of the European Union: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/ec (General Data Protection Regulation). Official Journal of the European Union, L119.1, 4.5.2016.

[3] Corona-Warn-App Consortium, [Online]. Available: https://www.coronawarn.app/en/.

[4] Carmela Troncoso *et. al*, "Decentralized Privacy-Preserving Proximity Tracing," [Online]. Available: https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf.

[5] Apple & Google, "Exposure Notification Cryptography Specification," [Online]. Available: https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.2.pdf?1.

[6] D. Shumow and N. Ferguson, "On the Possibility of a Back Door in the NIST SP800-90 Dual Ec Prng," [Online]. Available: http://rump2007.cr.yp.to/15-shumow.pdf.

[7] V. Goyal, A. O'Neill, and V. Rao, "Correlated-input secure hash functions," *Theory of Cryptography Conference (TCC)*, 2011, pp. 182–200.

[8] A.Z. Broder, "On the resemblance and containment of documents," *Proceedings. Compression and Complexity of SEQUENCES 1997, Italy*, 1997, pp. 21–29.