Secure Upload Authorisation for Digital Proximity Tracing

The DP-3T Project

30 April 2020

Digital proximity tracing systems based on Bluetooth Low Energy (BLE) handshakes between personal smartphones are being widely considered as a tool to help health authorities and governments control the spread of the global SARS-CoV-2 outbreak. Many countries desire systems which require an infected user be diagnosed by a recognised medical authority before the contact tracing process can begin. This is in contrast to systems that rely purely on self-reporting, i.e. in which any user can report that they are infected.

In this document, we identify the requirements of a secure upload authorisation system, the security and privacy concerns it must address, and analyse a number of authorization procedures against these requirements. We present these proposals in the context of the DP-3T proximity tracing system, but we want to highlight that *these upload authorisation procedures are suitable for use in other systems, such as Google and Apple's exposure notification framework*[1].

[1] See Appendix II for details.

Executive summary

In this document study three proposals for authorisation procedures. We highlight the different tradeoffs in Table 1:

Proposal	User Experience	Operation by Healthcare Authority	Security and Privacy
1. Simple Auth Codes	Authorisation code provided remotely (phone call, SMS, email, etc.)	Minimal infrastructure required. Distribution of codes to officials could be electronic or paper based.	Weak security properties including risk of users forwarding authorisation code to others.
2. Activated Auth Codes	Authorisation code exchanged during testing procedure (transcription, QR code)	Central infrastructure required to issue or activate codes. A basic web application would suffice.	Achieves stronger security properties against technically unsophisticated misuse. Does not stop a determined attacker.
3. Data Bound Auth Codes	Authorisation code exchanged during testing procedure (transcription, QR code)	Infrastructure at each test facility required, in addition to central infrastructure.	Achieves the strongest security properties. Mitigates the misuse of codes after testing by even a sophisticated attacker.

Table 1. Comparison of the proposals presented in this document

In summary, all of the proposals presented here avoid brute force and denial of service attacks as part of their design. They also ensure that attacks on individual privacy are not inadvertently introduced. None of the proposals require users to perform difficult technical tasks. However, as can be seen in Table 1, there are difficult tradeoffs between the practical details and the security properties.

For healthcare systems with mature technological support and access to internet-connected devices during testing, we highly recommend procedure 2 or 3 as they exchange authorisation codes only through QR codes. If this is not feasible, design 1 has the lowest overhead, albeit with a greater risk of code misuse.

We discuss the usability, security and privacy properties of these proposals in full later in the document, including security properties which we do not think can be achieved by any reasonable authentication procedure.

We identify the following important principles that we believe are essential for a usable, secure and privacy preserving testing process:

- Individual's identifiers or other personal information should never be linked to information uploaded to the backend.

- The usability of the system for testers and users is paramount to a successful deployment. If the system is unavailable due to denial of service or unwieldy to use, it will hamper the adoption of contact tracing in general.

- The security risks and incentives around users misusing their authorisation codes or the likelihood of phishing attacks are difficult to predict in advance, and consequently care must be taken to mitigate the effectiveness of these attacks.

It is likely that system implementers will want to customise these proposals to suit their own needs. We highlight considerations for the practical implementation of these schemes in Appendix I. We also discuss how the above proposals could be adapted to suit Google and Apple's proposed exposure notification framework in Appendix II.

Contents

Executive summary **2**

Contents **4**

Design Requirements and Assumptions **5**

Current Test and Notification Process 5

Use case: the DP-3T Proximity Tracing System 6

Functional Requirements 6

Usability Goals **7**

Security and Privacy Concerns **8**

Threat model 8

Security Risks 9

Unauthorised Upload of Data to the Backend System (S1) 9

Denial of Service (S2) 10

Privacy Risks 10

Inference or Linking Attacks by the Healthcare Authority (P1) 10

Inference or Linking Attacks by the Backend (P2) 10

Security and Privacy Non-Goals 10

Manipulation of Device Information prior to Testing 10

Linkability of User Information between Colluding Healthcare Authority and Backend 11

Proposals for Authorisation Procedures **12**

Proposal 1: Simple Authorization Codes 12

User story 12

Step-by-step description 12

Usability Analysis 13

Security Analysis 14

Privacy Analysis 14

Summary 14

Proposal 2: Activated Authorisation Codes 14

User story 15

Step-by-step description 15

Usability Analysis 16

Security Analysis 16

Privacy Analysis 17

Summary 17

Proposal 3: Data-Bound Authorisation 18

User story 18
Step-by-Step Process 18
Usability Analysis 19
Security Analysis 20
Privacy Analysis 20
Summary 20
Appendix I: Considerations for Implementing Authorisation Codes 22
Appendix II : Adapting the Proposals to Google and Apple's Design 23
Proposal 1 - Simple Authorisation Codes 23
Proposal 2 - Activated Authorisation Codes 23
Proposal 3 - Device Bound Authorisation Codes 23
Appendix III : A Stronger Version of Proposal 3 25

Design Requirements and Assumptions

We first set out our understanding of typical testing and notification procedures. We then discuss the functional and epidemiological requirements we require for all of our proposals.

Current Test and Notification Process

In order to be tested a patient visits "the health authority" to undergo a diagnostic test. This "health authority" could be a clinic, a hospital, or a dedicated testing facility. We refer to the day of the visit as the time of the test `tTest` (Step 1, Figure CP). At this time, the health official and patient establish how the patient should be notified of the result.

The exact testing mechanism and the delay between test and its result vary between countries and test systems. We abstract this process as the health official dispatching the sample to a lab for analysis (Step 2, Figure CP), which sends the test results back to the health official (Step 3, Figure CP). If the test result is positive, it is important to establish when the patient's contagious period began. We refer to the (estimated) start of the contagious window of the infected patient as the ***onset date***, dubbed `tContag`.

The health official notifies the patient of the result (Step 4, Figure CP). Notification can be performed via phone, email, SMS, or any other communication channel. We assume that this message includes the test result (positive or negative) and if positive, other supplementary information such as a request to contact the health official to discuss their probable onset date, or advice on how the patient could determine this themselves.

If the health authority is also performing manual contact tracing, the health official will also attempt to identify other people who are likely to have contracted the virus from the patient (Step 5, Figure CP) and notify them (Step 6, Figure CP).

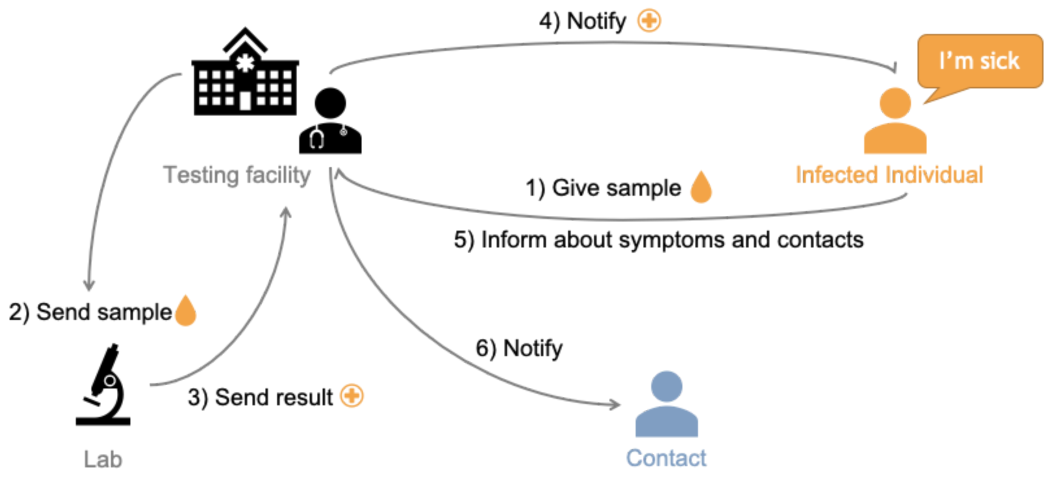

Figure CP: This flow diagram shows our abstraction of the current testing process

Use case: the DP-3T Proximity Tracing System

Throughout this document, we use the DP-3T proximity tracing system as a use case to illustrate the integration of authorization processes.

In the DP-3T low cost design, to enable decentralised proximity tracing, ***infected users publish a key*** SK_t as a ***compact representation of the ephemeral identifiers*** the infected user's phone has broadcast during the contagious period. This information allows other user's devices to check whether they had an encounter that might have led to an infection. This per-infected key (SK_t) and dates on which the user was believed to be contagious is the only information a user shares with other entities in the system.

The DP-3T low cost design is very similar to the current proposal by Google and Apple for exposure notification.[2] Consequently, all the designs we present in this document can be extended to support the Google and Apple design and we discuss this possibility further in Appendix II.

Functional Requirements

We now describe the functional requirements of the authorization process. We are aware that each country, or region, will have existing processes and systems in place to manage mass testing, to communicate between testing facilities and laboratories, and to inform patients. Consequently, we exclude proposals that require elaborate integration work

[2] Google and Apple (29 April 2020) *Exposure Notification: Cryptography Specification (v 1.2).* Available from https://www.apple.com/covid19/contacttracing.

with existing systems and focus on proposals that make minimal assumptions about the precise nature of the testing process.

To ensure the effectiveness of the contact tracing process, it is crucial that the beginning of the contagious window is determined as accurately as possible. If the selected onset date differs substantially from the actual onset data, it will result in false negatives (if too late) or false positives (if too early). Consequently, we treat the onset date as information which should be authenticated by the backend to ensure the patient does not make a mistake or attempt to mislead the system. However, optionally the onset date can be left for the patient to determine with only minor changes to these proposals. We also assume that onset time is calculated after a test is confirmed as positive. However, if the onset time is calculated during the testing procedure that is compatible with our proposals as well. We also assume that the earliest onset data is no more than 2 weeks in the past.

Some healthcare authorities may wish to offer anonymous testing where they do not record any identifying information, including contact details, about the user. Typically the user will contact the healthcare authority through a medium of their choice in order to acquire their test results. Consequently we require that the authorisation procedure does not rely on existing identifiers so as not to preclude the use of anonymous testing.

Usability Goals

In conversations with epidemiologists and national health authorities we have gathered a set of properties that a user-friendly system that has a high chance of adoption should provide. These are ***desirable properties*** that should not be read as strict requirements. It is likely that different healthcare authorities will choose to make different tradeoffs when deploying their systems, e.g. to align with manual tracing procedures or existing e-health solutions.

Ease-of-use for health authorities (U1)

Health authorities are placed under considerable time and resource pressure in order to deliver the maximal number of tests at the minimal cost. In particular, the availability of trained medical staff, volunteers and IT infrastructure is likely to be limited. Therefore care must be taken to ensure the authorisation procedure is as easy to operate as possible for healthcare authorities.

Where possible, the healthcare authority should not be required to operate a computer or mobile device ***during the testing procedure*** (**U1A**) and even when technology is available, reliable internet access should not be assumed (**U1B**). This is to enable the deployment of testing centers in locations with little infrastructure (e.g. parking lots).

The IT systems operated by the healthcare authority are likely to be difficult to reconfigure and subject to considerable regulation, consequently, the healthcare authority should not be required to store additional application specific information alongside a patient's healthcare or test record (**U1C**).

During the notification procedure for a positive diagnosis, the healthcare authority should not be required to operate a computer or other device (**U1D**) or have reliable internet access (**U1E**).

As significantly more users will undergo testing than actually testing positive, the labour and resources required during testing should be minimised, even if this leads to a moderate increase in labour during the notification procedure for positive users (**U1F**). This also means that (**U1A**, **U1B**) are higher priority goals than (**U1D**, **U1E**).

Finally, any additional information that needs to be exchanged between the healthcare authority and user should be as short as possible and easily transferred over a variety of communication mediums (phone, letter, SMS, etc) (**U1G**).

Ease-of-use for users (U2)
Users are in general not technically sophisticated and are unlikely to spend time troubleshooting if technical issues arise. They can also be unreliable when asked to record or provide information, as users undergoing testing typically have more pressing concerns than interacting with their contact tracing app.

In particular, the user should not be required to retain information, above and beyond what is required for the normal testing process, for lengthy periods of time (**U2A**). The user should not be required to carry out a technical process or difficult task, e.g. transcribe a long number from a telephone call (**U2B**).

The authorisation system should be easy to implement and operate (U3)
As the authorisation system must be integrated with each countries' existing medical infrastructure, the amount of development and integration work should be minimised (**U3A**). Furthermore, it should not require expensive resources (e.g. hardware, network communication) to operate (**U3B**).

Security and Privacy Concerns

An insecure upload authorisation procedure could jeopardise the privacy and security of the entire system. In this section, we provide a threat model and a list of risks that should be evaluated when considering any approach. We also explicitly describe properties that in our opinion cannot be provided by the authorisation procedure.

Threat model

Regular user. A typical user of the system who is assumed to be able to install and use the application by navigating its user interface (UI). They will exclusively look at information available via the app UI and interact with exposed settings.

Tech-savvy user (Blackhat/Whitehat hacker, NGOs, Academic researchers, etc.).
This user has access to the system via the mobile App. The user is assumed to be technically sophisticated and can decompile/modify the app. Has access to the source code of the various systems. They may interact with APIs and stored files with their own tool, unrestricted by any UI limitations imposed upon them.

Health authority learns about the outcome of a test for an individual and is trusted to authorise only those individuals who have truly tested positive to trigger contact tracing.
On the other hand, the health authority may try to link additional data of the system to the individual (e.g., to link the keys published by the backend to specific individuals that tested positive). We assume that the health authority does not collude with the backend.

Backend is trusted in publishing only the secret keys and onset dates of infected individuals that have been validated positive by the health authority. The backend has access to all the data uploaded by the app users as well as provided by the health authority.

Security Risks

*Unauthorised Upload of Data to the Backend System (**S1**)*

The backend system publishes a list of anonymous identifiers corresponding to infected individuals and the dates they became infectious. A malicious user may try to compromise the integrity of this data by trying to upload identifiers with an incorrect onset date or identifiers corresponding to uninfected individuals.

There are several ways in which an attacker might try to achieve this. Firstly, they might try to dishonestly obtain the credentials of an authorised user. For example, they might try to phish the user for their authorisation credentials (**S1A**), phish the healthcare authority into revealing a user's credentials (**S1B**) or perform a brute force attack on the backend system in order to recover valid credentials (**S1C**). We discuss considerations avoiding brute force attacks in Appendix I.

Secondly, the attacker may be a user who has been correctly authorised, but is willing to misuse their credentials. For example, they may try to use their credentials on another user's device in order to upload that user's information rather than their own (**S1D**). Alternatively, they may alter the information on their own device so that incorrect data is uploaded (**S1E**).

It is also possible for misbehaviour to occur at the healthcare authority, for example by issuing authorisation codes for personal benefit and whilst technological means cannot fully prevent this, they should attempt to mitigate it (**S1F**).

*Denial of Service (**S2**)*

A malicious party may try to deny use of the system to others. For example, a denial of service attack could be performed where the attacker overwhelms the backend system with spurious requests. This is particularly important when considering authorisation systems as rate limiting and other defensive measures may inadvertently make denial of service attacks easier.

Privacy Risks

*Inference or Linking Attacks by the Healthcare Authority (**P1**)*

Although the healthcare authority is necessarily trusted with confidential information about the patient’s medical condition, it is not desirable for the authorisation procedure to require the disclosure of additional information about the patient. For example, the healthcare authority should not be able to link data provided by the patient during authorisation or testing to the data uploaded by the patient to the backend.

Mitigating this risk ensures that the healthcare authority does not (unintentionally or otherwise) build a database recording ephemeral identifiers recorded or broadcast by the patient which could then be compromised or otherwise misused. Further, it prevents the healthcare authority from being able to infer the social graph of users undergoing testing or receiving positive diagnoses.

*Inference or Linking Attacks by the Backend (**P2**)*

The backend should learn as little information as is feasible about the user performing an upload. Consequently, the authorisation procedure should not allow the backend to learn any additional information about the patient's interaction with the healthcare authority.

Mitigating this risk ensures that the backend does not (unintentionally or otherwise) build a database linking identifiers to infectious users which could then be hacked or otherwise misused.

Security and Privacy Non-Goals

Manipulation of Device Information prior to Testing

When considering risk **S1** (Unauthorised Upload of Data to the Backend System), there are practical limitations on what the authorisation procedure can achieve. In particular, technological means alone cannot establish whether a particular device belongs to a particular user or guarantee the integrity of the data on it. For example, a regular malicious user may simply exchange phones with another party prior to testing, or arbitrarily alter the information stored on it.

Further, a tech-savvy malicious user may duplicate their secret keys in order to later cause false alarms. For example, they modify a number of devices to share the same secret key or exchange their own secret keys with another user's.

These problems cannot be solved by the authorisation procedure and we do not discuss them further in this document.

Linkability of User Information between Colluding Healthcare Authority and Backend

In our analysis we assume that the healthcare authority and backend do not collude in order to identify users. Although in principle we could imagine a complex system built around anonymous communication and credentials which might be able to partially mitigate this type of collusion, we are not aware of an existing system that would scale to the anticipated number of users required for effective contact tracing.

Proposals for Authorisation Procedures

We now present several different sample proposals for secure upload authorisation and explain how they integrate with existing test and notification processes. For each proposal, we explain what happens at each step in the process, describe the user story, and analyse its privacy and security risks and whether it provides the desired usability properties.

Proposal 1: Simple Authorization Codes

In this proposal, the backend generates authorization codes that permit patients to upload data. These codes may be generated in advance and periodically distributed to test facilities or can be requested on demand by healthcare officials.

The healthcare authority can also determine the onset dates for which a code is valid. For example, they may wish to engage in discussion with a user about their probable onset data and then issue an authorisation code linked to the agreed-upon date (simplifying data entry for the user). Or, due to time constraints, the healthcare authority might provide the user with guidance on determining the onset date, authorise a code for any time in the past two weeks, and leave the calculation of the onset date to the user.

After a positive test result, the health official provides an authorisation code to the infected individual, who enters it into their app. The app checks with the backend to ensure it is a valid authorisation and performs the upload of the relevant information.

User story

1) Test time
At testing time, the user visits the health authority to give a sample for analysis as usual. She does not have to perform any additional action.

2) Test results notification
The user receives an authorisation code along with her positive test result. This code could be communicated via phone, sms or whichever medium is used in the normal notification process. She enters the authorisation code into her phone and confirms the upload. Optionally, she is prompted to enter a valid date within the past 14 days as the onset date.

Step-by-step description

1) Setup
At setup time, the backend generates a list of authorization codes `[AC1, AC2, ...]` and sends them to the health authority. Codes are distributed to healthcare officials either in advance in batches (e.g. printed lists) or provided on demand through a web interface.

2) Test time
When a patient visits the health authority, she gives a sample for analysis and her contact information. There is no additional interaction between the health official and the user.

3) During analysis
No additional information must be stored in the system other than a patient's test record.

4) Test results notification
After the health authority receives a positive test result, an official can engage in discussion with the patient to determine a probable onset date and issue them an authorisation code valid for the date. Alternatively, the official may choose to issue them with an authorization code valid for a particular period of time and provide guidance on how the user can determine their onset date. The authorisation code is given to the user.

5) Upload
The patient enters the auth code `AC` into their smartphone. Optionally, if the contagious period is determined by the user, they also enter the first day of the contagious window `tContag`. The device transfers the tuple $(AC, tContag, SK_{tContag})$ to the backend. The backend checks whether the authorisation code is valid and adds $SK_{tContag}$ to the list of infected identifiers. The backend then removes `AC` from the list of valid codes.

Usability Analysis

(U1) Ease-of-use for health authorities
The proposal provides the lowest barrier to adoption by health authorities. It does not require a computer or internet access or record or store any additional application specific patient information (**U1A**, **U1B, U1C, U1D, U1E**). No additional steps are required during testing (**U1F**). Only a single code needs to be transferred between the healthcare authority and user (**U1G**). However, the code is likely to be transferred remotely (e.g. phone call) and will need to be a long sequence of letters and numbers due to security requirements.

(U2) Ease-of-use for users
The proposal does not require the user to remember any additional information, as the authorisation code is entered by the user immediately after it is issued to them (**U2A**). If authorisation codes are not bound to particular days, users will also need to enter an onset data, which requires additional effort.

However, the user will need to enter a long authorisation code, which might have been transcribed during a phone conversation. This is moderately technically difficult (**U2B**).

(U3) Ease of implementation and operation
There are minimal resource and integration requirements for this proposal. The healthcare authority could operate an entirely paper-based system or use web

technology where available. There is no significant development required, as the generation and authentication is handled by the backend (**U3A**). Furthermore, no expensive resources are required to operate the system (**U3B**).

Security Analysis

Provided that the authorisation code contains enough entropy[3], it is difficult for an attacker to brute force an authorisation code (**S1C**). However, as the code is not bound to the user's device, phishing attacks are possible (**S1A, S1B**), as well as misuse of credentials on or between devices (**S1D**, **S1E**). However, the short time window between the user being issued the credential and using it limits the time in which phishing can take place.

If codes are not bound to particular days, there is an increased risk that users alter the date they report according to their own preferences (**S1E**).

If codes are pregenerated, the healthcare authority must be careful to restrict access to the code sheets, otherwise an attacker would be able to falsely diagnose a number of individuals (**S1F**). A possible mitigation would be for the backend to provide an interface for the revocation of issued codes. Alternatively, if the codes are issued on demand misuse can be mitigated through audit logs and monitoring.

This proposal does not introduce any additional risks for denial of service (**S2**), provided that the rate limiting is not used as a mitigation for short authorisation codes.

Privacy Analysis

As the authorisation code is not linked with the user's information and the user does not transfer any information to the healthcare authority, there are no additional linkability attacks directly introduced by this proposal (**P1**, **P2**).

However, if the backend is trusted to generate and distribute the authorisation codes to different geographical regions or facilities, this could be used to infer information about the users of those codes. Consequently, any distribution should be handled by the healthcare authority.

Summary

This proposal has significant advantages in terms of simplicity, ease of operation for the healthcare authority, and privacy. However, it does require users to enter a code transcribed from another medium (e.g. the phone) which reduces usability and is prone to misuse (e.g. a user enters the code into someone else's device).

[3] We discuss this requirement further in Appendix I

Proposal 2: Activated Authorisation Codes

In this proposal, authorisation codes are exchanged at testing time, rather than at notification time. This ensures that the exchange is carried out in person, which reduces the chances of errors and leads to a faster notification procedure, at the cost of increased complexity during testing.

During the in person exchange, it is possible to have the authorisation code generated by the user's device and then recorded by the health official, or to have the health official provide an authorisation code from the backend to the user. The first variant is easiest if the health official has access to a device such as a barcode scanner or QR code reader. Alternatively, the user's device is likely to have a camera which can be used as a QR code reader to support the second variant.

User story

1) Test time
At testing time, the user either provides an inactive authorization code generated by her device to the healthcare official who records it, or the healthcare official provides a code that the user enters into their device. This exchange could be performed with a QR code or by manual transcription. The user is told that this code will later be activated if their test result is positive.

2) Test results notification
The user receives a notification about her positive test result. Optionally, she is asked to contact the health authority to determine the beginning of the contagious window. The authorisation code she has received at test time is activated by the health authority. She instructs her device to transmit the authorisation code to the backend and to upload her information.

Step-by-step description

1) Setup
At setup time, either the backend generates a list of inactive authorisation codes `[AC1, AC2, ...]` and sends them to the health authority. Alternatively, each user device generates an inactive authorisation code for their device. Health officials register for access to the web application that allows them to communicate with the backend.

2) Test time
When a patient visits the health authority, she gives a sample for analysis and her contact information. The health official and patient exchange **inactive authorisation code** `AC` and the code is attached to the test record.

3) During analysis

The health authority stores the patient's inactive authorisation code `AC` together with the patient's test record. The patient stores the inactive authorisation code `AC` in their device.

4) Test results notification
After the health authority receives a positive test result, it notifies the patient. Optionally, the notification includes a request to contact the health authority to receive guidance on how to determine the beginning of the contagious window `tContag`. Once the patient has contacted the health official, the health official determines `tContag` and activates `AC` for a short time window, for example a few hours, by sending `(AC,tContag)` to the backend. This binds the authorisation code to a specific date.

5) Upload
The patient instructs her device to begin the upload. The device transfers the tuple $(AC, SK_{tContag}, tContag)$ to the backend. The backend checks whether the authorisation code is valid and activated and that `tContag` matches the date received from the health authority. The backend adds $SK_{tContag}$ to the list of infected identifiers and removes `AC` from the list of valid codes.

Usability Analysis

(U1) Ease-of-use for health authorities
This proposal does not require the health authority to operate a device during the testing procedure or have internet access (**U1A**, **U1B**), but it does require additional application specific information to be stored along with a patient's test record (**U1C**). It also requires this to be done for every tested patient, rather than every infectious patient (**U1F**).

During the notification procedure, the healthcare authority must have access to a computer which can communicate with the backend in order to authorise the user's code (**U1D**,**U1E**). However, the information exchange is carried out in person rather than remotely which is beneficial for speed and accuracy of the transfer (**U1G**).

(U2) Ease-of-use for users
Users do not have to carry out a technical process or difficult task procedure (**U2B**). However, they must retain the exchanged identifier in their app until the result of the test is known, however this should be straightforward (**U2A**).

(U3) Ease of implementation and operation
This proposal requires the healthcare authority to contact the backend to mark particular codes as active before notifying a user, which requires a web app or similar integration (**U3A**). However, there would be relatively little load on this server as officials would only interact for a positive case (**U3B**).

Security Analysis

Provided the authorisation codes are long enough[4], this proposal does not introduce any avenue for a brute force attack (**S1C**). No protection is provided against phishing (**S1B**, **S1C**) as the user retains their credential for a long period of time during which they might be targeted.

Provided the user is not technologically savvy, the proposal does protect against inputting the code on a different device (**S1D**). If the code is generated by the healthcare authority, it no longer needs to be retained in a user visible manner. If it is generated by the app, it does not need to be editable at all.

However, a technologically savvy user can still edit the information on their own device or input the code on a different device (**S1D**, **S1E**). However, they cannot change their onset date if it has been fixed by the healthcare authority.

The proposal mitigates any misbehaviour at the health authority by ensuring that all code activations are tied to a particular infected patient (**S1F**).

The system does not introduce any additional denial of service vulnerabilities (**S2**).

Privacy Analysis

This proposal does not introduce any new linkability attacks (**P1**, **P2**) provided that any authorisation code stored on the user's device is deleted once it has been used to authorise an upload.

Summary

This proposal enjoys slightly stronger security properties against misbehaviour by non-technologically sophisticated users and the healthcare authority. However, it does require more record keeping by the healthcare authority both at the point of testing and in confirming a positive diagnosis.

[4] We discuss this requirement further in Appendix I

Proposal 3: Data-Bound Authorisation

In this proposal, a strong cryptographic binding is made between the data on the device presented by the user at testing time and later used to upload their test results. In short, individuals at test time commit to the data they will upload later if their test results are positive, so that they cannot decide to send other data after they have tested positive. The health authority authorises the upload of this data after they receive a positive test result.

User story

1) Test time
At testing time, the user visits the health authority to give a sample for analysis as usual. She accesses her smartphone which provides her test commitment in the form of a QR code or authentication string which she communicates to the tester.

2) Test results notification
The user is notified by the healthcare authority in the normal manner and instructs her device to upload her data. No additional code or authorisation is required.

Step-by-Step Process

1) Setup
During the system setup phase, an authentication key `C` is distributed to each health authority and the corresponding verification keys are sent to the backend. These codes should be kept confidential, but will be stored in a policy/setting dependent manner according to the healthcare authority. We leave the authentication mechanism abstract: the authentication key `C` could be a signing key pair with a certificate[5], or a symmetric key for a MAC.

2) Test time
At time `tTest`, the patient enters into her smartphone that she is taking a test today. Her device computes a random value r_{tPast} and hash:
$H_{tPast} = h(SK_{tPast}, tPast, r_{tPast}, \text{“test”})$

for `tPast = tTest - 14 days`. We choose `tPast` as 14 days before `tTest` to ensure that if the test comes back positive, the beginning of the infectious period falls within the time window `[tPast, tTest]`. The patient provides the generated hash to the tester. This could either be in form of a token that the patient communicates to the health official or a QR code that the health official scans from the patient’s smartphone.

3) During Analysis

[5] Note that this only requires a signature key for the entity that approves testers, which then can provide certificates for their signing keys, with the public key of the approving entity stored at the backend.

The health authority stores the time t and the commitment H_{tPast} alongside the patient's medical record. The patient's phone stores the tuple (r_{tPast}, tPast) locally.

4) Test Result Notification
After the health authority receives a positive test result, it sends the result via the usual channel to the patient. The notification includes a request to contact the health authority to receive guidance on how to determine the beginning of the contagious window tContag. Once the patient has phoned the health official, the health official determines tContag, which is necessarily between tPast and tTest. The health official computes:

$$AC = \text{auth}_C(\text{tContag}, \text{tPast}, H_{tPast}, \text{“postest”})$$

using their authentication key C. The health authority sends (AC, H_{tPast}, tContag) to the backend.

5) Upload
After receiving a positive test result, the patient instructs their phone to send to the backend the data necessary to trigger the decentralised proximity tracing procedure, i.e., they send

$$(SK_{tPast}, \text{tPast}, r_{tPast}, \text{tContag})$$

The backend receives this message and computes

$$H' = h(SK_{tPast}, \text{tPast}, r_{tPast}, \text{“test”}),$$

and looks in its database for a tuple (AC, H_{tPast}, tContag) such that $H' = H_{tK}$.

If such a tuple is found and $\text{ver}_C((\text{tContag}, \text{tPast}, H_{tPast}, \text{“postest”}), AC)$, the backend accepts the upload. Otherwise it rejects it.

If the upload is accepted, the backend computes the patient's secret key for the first day of the contagious period $SK_{tContag}$ from SK_{tPast} (by repeatedly applying the key forwarding function tContag-tPast times) and adds $SK_{tContag}$ to the list of infected identifiers. The backend removes the tuple (AC, H_{tPast}, tContag) from its list of valid tokens.

Usability Analysis

(U1) Ease-of-use for health authorities
The healthcare authority is required to operate a computer during testing, not necessarily, but ideally with internet access (**U1A**, **U1B**). The healthcare authority is also required to store additional information alongside a patient's test record (**U1C**).

During notification, access to a computer with internet access is required to authorise the upload of the patient's information (**U1D**, **U1E**).

The healthcare authority does have to perform additional labour during testing (**U1F**), but the only exchange of authorisation codes is carried out in person rather than remotely (**U1G**).

(U2) Ease-of-use for users
The user is not required to retain any information provided by the healthcare authority (**U2A**). The user does not have to perform any technical process or enter any codes (**U2B**).

(U3) Ease of implementation and operation
The authorisation system would require integration with the existing medical infrastructure both in testing facilities and in the notification workflow (**U3A**). There would also need to be a system for storing the tested patient's commitments which entails operational and hardware costs (**U3B**).

Security Analysis

Phishing is largely mitigated by this approach, the user's commitment code is transferred in person and uniquely identifies their data, so it is of little use to an attacker if revealed (**S1A**). Likewise, although the healthcare authority could be tricked into accepting a new commitment code, it does not matter if it is inadvertently revealed to an attacker (**S1B**). Brute force attacks are not possible (**S1C**).

Even a sophisticated user cannot upload another user's information or incorrect data, provided they only began to deviate from the protocol after the testing procedure was performed (**S1D**, **S1E**).

Misbehaviour at the healthcare authority is largely mitigated as it would require collaboration between both the testing facility and the individual performing the notification procedure (**S1F**).

No additional denial of service vulnerabilities are introduced (**S2**).

Privacy Analysis

Although this design involves the user transferring a commitment code which uniquely identifies her data to the healthcare authority, careful design ensures that the healthcare authority cannot later link this information to the uploaded data. In particular the random values r_{tPast} which are revealed only to the backend and not the healthcare authority, ensure that the healthcare authority cannot later check which user uploaded which keys (**P1**).

The backend does not learn any additional information about the patient's test or medical information (**P2**).

Summary

This proposal achieves the strongest security properties of the three that we have considered. In particular, it significantly limits any misbehaviour by a user after the testing procedure takes place. However, it does require a more complex implementation and usability tradeoffs compared to the earlier proposals.

Appendix I: Considerations for Implementing Authorisation Codes

In this document we have presented authorisation codes as a token that is eventually transmitted to the backend alongside the user's uploaded data. However, alternative implementations are possible. For example, the health authority may run their own web service where the user's application can exchange an authentication code for a signed token containing a unique random value. The backend can then accept any valid signature on a new unique value without having to directly interact with the health authority for authorisation code generation or use. This flow integrates well with common identity provision services such as OAuth or OpenID Connect.

The length of authorisation codes is ultimately an implementation detail, but it is important that these codes contain enough entropy to prevent brute force attacks by a malicious party. We recommend the use of codes containing at least 64 bits of entropy.

In systems in which an exceptionally large number of codes[6] are generated in advance (e.g. the first proposal) or where an exceptionally large population is served by one system, it is possible that additional entropy would be necessary to prevent guessing attacks over long periods of time.

If codes are written numerically, in order to aid transcription when the parties communicating do not share a native language, this would entail a minimal length of 20 numerical digits. It is possible that with the use of sophisticated rate limiting or CAPTCHA systems, these codes could be shortened.

As users are likely to make errors when entering authorisation codes, we highly recommend the use of check digits or other error correcting encodings. These digits would allow a device to locally confirm if an authorisation code had been correctly entered by a user. Note that any check digits do not count towards the length of the code for security purposes.

[6] E.g. hundreds of thousands (or more) active authorisation codes

Appendix II : Adapting the Proposals to Google and Apple's Design

Here we discuss how to adapt these proposals from DP3T's low cost design to the exposure tracing design proposed by Google and Apple.

Proposal 1 - Simple Authorisation Codes

No changes are required except that the user's device uploads the relevant day keys rather than the master key.

Proposal 2 - Activated Authorisation Codes

No changes are required except that the user's device uploads the relevant day keys rather than the master key.

Proposal 3 - Device Bound Authorisation Codes

This proposal requires minor changes. The main difference is that in Google and Apple's design (as well as in DP3T's unlinkable design), the backend does not receive and roll a single key for every infected user. It rather receives one key per epoch. We detail the changes in each step in the Step-by-Step process below.

2) Test time

Rather than a single hash, 14 seperate hashes are calculated and transferred: the device computes 14 random values $r_{tPast}, .., r_{tPast+13}$ and hashes:

H_{tPast} = h(SK_{tPast} , tPast , r_{tPast} , "test")

$H_{tPast+1}$ = h($SK_{tPast+1}$, tPast+1 , $r_{tPast+1}$, "test")

...

$H_{tPast+13}$ = h($SK_{tPast+13}$, tPast+13 , $r_{tPast+13}$, "test")

The patient provides the 14 generated hashes to the tester.

3) During Analysis

The health authority stores the time t and the commitments $(H_{tPast}, ..., H_{tPast+13})$ alongside the patient's medical record.
The patient's phone stores the tuple $(r_{tPast}, ..., r_{tPast+13},$ tPast) locally.

4) Test Result Notification

Once having determined the contagious window tContag, the authority computes a collection of authentication codes

$$AC_t = auth_C(t, H_t, \text{"postest"}), \text{ for } t \text{ in } [tContag, tPast+13]$$

using their authentication key $\mathtt{C}$. The health authority sends the tuples $\{(\mathtt{AC}_t, \mathtt{H}_t)\}_{t \text{ in } [\mathtt{tContag,tPast+13}]}$ to the backend.

5) Upload

After receiving a positive test result, the patient instructs their phone to send to the backend the data necessary to trigger the decentralised proximity tracing procedure, i.e., they send

$$(\mathtt{SK}_t,\ \mathtt{t},\ \mathtt{r}_t)\ \mathtt{for\ t\ in\ [tContag,tPast+13]}$$

For every such tuple, the backend computes

$$\mathtt{H}_t' = \mathtt{h}(\mathtt{SK}_t,\ \mathtt{t},\ \mathtt{r}_t,\ \text{“test”}),$$

and looks in its database for a tuple $(\mathtt{AC},\ \mathtt{H})$ such that $\mathtt{H}_k' = \mathtt{H}$.

If such a tuple is found and $\mathtt{ver}_C(\ (\mathtt{t},\ \mathtt{H}_t,\ \text{“postest”}),\ \mathtt{AC}_t)$, the backend accepts the upload and adds $\mathtt{SK}_t$ to the registry of infected. Otherwise it rejects it.

Note: in the above proposal, the use of 14 keys is an example. It can be adapted to support as many keys as is the epoch granularity chosen by the system which we present in Appendix III. Also, in the above proposal the user commits and uploads keys only until the epoch `tPast+13`. This leaves out the keys in the time window between the time of the test and the time when the upload occurs. If needed, this can be solved by letting the user preemptively commit to a few more keys in the future.

Appendix III : A Stronger Version of Proposal 3

In the Proposal 3 the backend learns a key for an epoch tPast that may be a few days older than the beginning of the contagious window tContag. Here we discuss an adaptation of Proposal 3 that allows a security-efficiency tradeoff. The idea is similar to the adaptation to Google and Apple's design discussed in Appendix II.

Below we detail the changes needed to the Step-by-Step process. The proposal is parametrized by an integer $1<=D<=14$: a small D gives more efficiency while a large D gives more security.

2) Test time

At time tTest, the patient enters into her smartphone that she is taking a test today. Her device computes D (D<=14) random values $r_{tPast}, .., r_{tPast+D-1}$ and hashes:

H_{tPast} = h(SK$_{tPast}$, tPast , r_{tPast} , "test")
$H_{tPast+1}$ = h(SK$_{tPast+1}$, tPast+1 , $r_{tPast+1}$, "test")
...
$H_{tPast+D-1}$ = h(SK$_{tPast+D-1}$, tPast+D-1, $r_{tPast+D-1}$, "test")

for tPast = tTest - 14 days. We choose tPast as 14 days before tTest to ensure that if the test comes back positive, the beginning of the infectious period falls within the time window [tPast, tTest]. The patient provides the generated hashes to the tester. This could either be in form of a token that the patient communicates to the health official or a QR code that the health official scans from the patient's smartphone. If D is small, code words or a string could also be used.

3) During Analysis

The health authority stores the time t and the commitments ($H_{tPast}, ..., H_{tPast+D-1}$) alongside the patient's medical record.
The patient's phone stores the tuple ($r_{tPast}, ..., r_{tPast+D-1}$, tPast) locally.

4) Test Result Notification

After the health authority receives a positive test result, it sends the result via the usual channel to the patient. The notification includes a request to contact the health authority to receive guidance on how to determine the beginning of the contagious window tContag. Once the patient has phoned the health official, the health official determines tContag .
The health official computes:

tMin = min(tContag, tPast+D-1)

recalling tPast = t - 14, and

AC = auth$_C$(tContag, tMin, H_{tMin}, "postest")

using their authentication code C.
The health authority sends (AC, H_{tMin}, tContag) to the backend.

5) Upload

After receiving a positive test result, the patient instructs their phone to send to the backend the data necessary to trigger the decentralised proximity tracing procedure, i.e., they compute

$$\texttt{tMin = min(tContag, tPast+D-1)}$$

and send

$$(\texttt{SK}_{\texttt{tMin}}, \texttt{ tMin, } \texttt{r}_{\texttt{tMin}}, \texttt{ tContag}).$$

The backend receives $(\texttt{SK}_{\texttt{tMin}}, \texttt{ tMin, } \texttt{r}_{\texttt{tMin}}, \texttt{ tContag})$, computes

$$\texttt{H'= h}(\texttt{SK}_{\texttt{tMin}}, \texttt{ tMin, } \texttt{r}_{\texttt{tMin}}, \texttt{ ``test''}),$$

and looks in its database for a tuple $(\texttt{AC, } \texttt{H}_{\texttt{tMin}}, \texttt{ tContag})$ such that $\texttt{H'= } \texttt{H}_{\texttt{tMin}}$.
If such a tuple is found and $\texttt{ver}_{\texttt{C}}\texttt{((tContag, tMin, } \texttt{H}_{\texttt{tMin}}, \texttt{ ``postest''), AC)}$, the backend accepts the upload. Otherwise it rejects it. If the upload is accepted, the backend computes the patient's secret key for the first day of the contagious period $\texttt{SK}_{\texttt{tContag}}$ from $\texttt{SK}_{\texttt{tMin}}$ (by repeatedly applying the key forwarding function $\texttt{tContag-tMin}$ times) and adds $\texttt{SK}_{\texttt{tContag}}$ to the registry of infected.