Automated Security Analysis of Exposure Notification Systems

Kevin Morio[1] Ilkan Esiyok[1] Dennis Jackson[2] Robert Künnemann[1]

[1]*CISPA Helmholtz Center for Information Security*
[2]*Mozilla*

Abstract

We present the first formal analysis and comparison of the security of the two most widely deployed exposure notification systems: the ROBust and privacy-presERving proximity Tracing protocol (ROBERT) and the Google Apple Exposure Notification (GAEN) framework.

ROBERT is the most popular instalment of the centralised approach to exposure notification, in which the risk score is computed by a central server. GAEN, in contrast, follows the decentralised approach, where the user's phone calculates the risk. The relative merits of centralised and decentralised systems have proven to be a controversial question. The majority of the previous analyses have focused on the privacy implications of these systems, ours is the first formal analysis to evaluate the security of the deployed systems—the absence of false risk alerts.

We model the French deployment of ROBERT and the most widely deployed GAEN variant, Germany's Corona-Warn-App. We isolate the precise conditions under which these systems prevent false alerts. We determine exactly how an adversary can subvert the system via network and Bluetooth sniffing, database leakage or the compromise of phones, back-end systems and health authorities. We also investigate the security of the original specification of the DP3T protocol, in order to identify gaps between the proposed scheme and its ultimate deployment.

We find a total of 27 attack patterns, including many that distinguish the centralised from the decentralised approach, as well as attacks on the authorisation procedure that differentiate all three protocols. Our results suggest that ROBERT's centralised design is more vulnerable against both opportunistic and highly resourced attackers trying to perform mass-notification attacks.

1 Introduction

In response to the COVID-19 pandemic, digital exposure notification systems (ENS) have been designed, deployed and are now used by hundreds of millions of people around the world. These systems complement manual contact tracing efforts, providing automated notification to users that were potentially exposed to COVID-19 allowing them to take appropriate precautions.

Although various designs have been proposed, the systems which have seen real world deployment can be split into two families: centralised [12] or decentralised [31], depending on whether the risk calculation is performed on a central server or on the user's device. The merits of both approaches have been the subject of extensive debate, with much of the argument focused on potential privacy risks. Less attention, however, was paid to the comparative security of the two categories of systems.

In this work, we formally analyse the security of the two most widely deployed centralised and decentralised ENS: the ROBust and privacy-presERving proximity Tracing protocol (ROBERT), the leading centralised solution, and the Google Apple Exposure Notification framework (GAEN), which follows a decentralised design. Whilst ROBERT was proposed and developed as a cohesive whole, the GAEN framework leaves many security-critical components of the system open for the developers to implement. The countries that deployed applications based on the GAEN framework thus had to decide how to authenticate positive test results, how to disseminate this information to a central back end, and how to interoperate with other countries' back ends. Clearly, these design decisions critically impact the overall security of the system. Consequently, we focus on one of the most popular applications built with the GAEN framework, the Corona-Warn-App (CWA) developed in Germany, but also investigate the security of the DP3T system as originally specified. DP3T's specification informed the design of GAEN-based applications deployed in Austria, Belgium, Croatia, Ireland, Italy, the Netherlands, Portugal and Switzerland.

We carry out our formal analysis with Tamarin [23], a security protocol verifier. Our models include all the components of the deployed ENS including the user phones, a granular temporal and spatial model and the various back-end components and associated protocols. Our models allow for an unbounded number of users engaging in an unbounded number of sessions and allow the adversary to control user location and travel patterns, COVID-19 diagnostics and network communication. We further consider an attacker who can compromise any entity in the system, including users' phones, health authorities and national back ends. For each of the models, we ask the following questions:

a) Under what circumstances can an attacker pollute the ENS with forged information which will impact the risk computation?

b) Under what circumstances can an attacker mislead a user into believing they are at risk of infection?

By finding answers to these questions, we develop an ontology of possible attacks which systematises the result of much previous work claiming various attacks or proposing improvements without evidence (discussed in Section 2). The generality of our threat model allows us to compare the discussed systems with the required level of detail.

We discover attack patterns that are common across all systems, but also attacks that distinguish centralised and decentralised systems. This is fairly obvious in scenarios with back end compromise, but we also discover an important difference in the scalability of an attack when a phone of an infected user is under the adversary's control. While isolated false alarms can severely impact individuals, a large-scale attack can render the whole system unusable, severely impacting epidemic control efforts.

Independent of the architecture, the authentication schemes used in the three proposals give vastly different guarantees, which is largely because interactive schemes are hard to deploy at the necessary scale. We also find that federation increases the vulnerability to attacks on other countries' health authorities, with a stronger impact on centralised than decentralised systems. In total, we find 27 attack patterns across ROBERT, DP3T and the CWA, providing a detailed answer of the two previously posed questions.

Contributions This paper makes the following contributions: (a) The first formal security analysis of the most widely deployed ENS. (b) A systematic evaluation of the trade-offs between these systems w.r.t. a powerful shared threat model. (c) Novel techniques for modelling spatial and temporal aspects of these protocols. They are amenable for automated analysis and can be reused to model proposed refinements, extensions and designs.

Outline In Section 2, we discuss related work on the security of ENS. We then introduce ROBERT, DP3T and the CWA in Section 3. The modelling of the systems is discussed in Section 4 and their security properties in Section 5. The results of our analysis are presented in Section 6, limitations in Section 7. Finally, we conclude in Section 8.

2 Related Work

A substantial amount of previous work has proposed alternative ENS [11, 28, 29] and presented informal assessments of their properties [30, 18, 2, 32, 9, 5, 6]. However, only three [15, 22, 10] undertook a formal analysis of the security properties of ENS. In this section, we review these in detail, postponing a discussion of previously published attacks until we discuss our taxonomy in Section 6.

Danz et al. [15] investigate the security and privacy of the Temporary Contact Numbers Protocol and the early DP3T proposals, as well as the currently deployed GAEN framework. Their analysis is focused on the 'cryptographic core' of each proposal, specifically the Bluetooth proximity protocol. Unlike our work, they do not consider the security of the authorisation procedure for uploads and do not model the back end infrastructure or the spatial distribution of users. However, as they carry out their analysis by hand in the computational model, they are able to define and prove various privacy properties for the respective Bluetooth proximity protocols.

Kobeissi, Nicolas, and Tiwari [22] build a model of DP3T in order to demonstrate the flexibility and friendliness of their new protocol analysis tool. As the model was developed as an example, it only captures some parts of the protocol, limited to only three participants interacting in a prescribed pattern without an authorisation procedure.

Canetti et al. [10] propose two novel ENS and investigate their security and privacy. They formalise their security notions partly in the UC (universal composability) model and partly via game based definitions. Although their UC model provides a computational proof that the modelled systems meet their specified ideal functionality, the concrete implications of this ideal functionality are not explored and extracting the resulting security properties is a considerable challenge. Similar to Danz et al. [15], they model their proposed Bluetooth proximity protocols in detail, but leave the rest of their system, e.g. back end infrastructure and upload authorisation, abstract.

3 Exposure Notification Systems

In this section, we describe the design of ROBERT, GAEN, DP3T and the CWA.

We first introduce ROBERT, the centralised ENS designed by INRIA PRIVATICS and Fraunhofer AISEC [12]. ROBERT was proposed in April 2020 and deployed in the official French contact tracing application 'StopCovid' on 2 June 2020 (later rebranded to 'TousAntiCovid'). With over 12.3 million downloads, it is the most popular centralised ENS.

We then describe the GAEN framework which underlies many deployed decentralised systems. It is a library integrated with Google's Android OS and Apple's iOS to provide a Bluetooth proximity protocol. Next, we detail DP3T which predates the GAEN specification and is substantially similar in design, but also describes the various other components of ENS and guides the design of many European exposure notification applications. We also discuss the DP3T proposal for user upload authentication in detail. Finally, we elaborate on the design of the CWA, the official German exposure notification application, based on DP3T and the GAEN framework.

Terminology As we refer to the phone more frequently than to its owner, we will call a phone *infected* or *diagnosed* if its owner has been infected (or diagnosed) with COVID-19. The *contagious period* is the time period in which the phone's owner is assumed to be contagious. Ephemeral keys that are associated with this period are called *infected keys*. If two phones are close enough that one's beacon message reaches the other, they are *in proximity*.

3.1 ROBERT

In ROBERT, mobile phones continuously broadcast ephemeral identifiers and store the identifiers they receive from other users. If a user is diagnosed with COVID-19, they upload all the recorded ephemeral identifiers on their phone to a central back end, which is then able to link these ephemeral identifiers to the users that broadcast them, carry out a risk calculation and notify any impacted users.

For linking ephemeral identifiers to users, phones must first register with a central server, which assigns them a permanent identifier, gives them an authorisation key and stores an encryption key for them. The server then periodically generates batches of ephemeral Bluetooth identifiers (EBIDs) and transmits them to the phone.

In normal operation (Fig. 1), phones continuously broadcast their EBID for the current epoch and record any received EBIDs transmitted by other phones (1). Should a user test positive, the health authority provides them with a special authorisation code which they enter into their phone (2). The phone then transmits this code and all witnessed EBIDs during the contagious period to the central server (3). The server verifies the authorisation code with the help of the health authority (not shown). Knowing each user's encryption key, the server can decrypt the EBIDs and learn the permanent identifiers of each user who has been in proximity to the contagious user (4). After performing a risk calculation, the central server informs the impacted users of the risk (5), but does not disclose any information about the cause of the risk.

The specification explicitly leaves the authorisation procedure for step (2) open, but requires that 'only authorised and positive-tested users are allowed to upload'. We thus investigated the source code of the back-end server, submission code server and the REST API specification[1] and found that the submission code server produces *long codes* for health professionals with a default expiration time of 8 days and *short codes* with a default expiration time of 60 minutes. A positively tested user is given such a code in form of a QR code, which can be scanned to initiate the upload of EBIDs.

ROBERT also supports a federated deployment, in which different ROBERT deployments can interoperate with one another. In this case, all servers in the federation are provided with a federation key which is used to encrypt a country code for each EBID. When receiving EBIDs from an infectious user, the back end can always learn the country code of an EBID and forward the EBID to the relevant server for processing. We provide a more detailed description of registration, broadcast, authorisation, risk calculation and federation in Appendix A.

3.2 GAEN

In April 2020, Google and Apple announced a framework for decentralised ENS (substantially similar to previous proposals such as DP3T, PACT[2] and TCN[3]) which they deployed via their respective mobile phone operating systems. This framework implements the broadcast protocol and risk calculation, but leaves the other components such as the interaction with the health authority and between back ends to the developers of the country-specific contact tracing applications, such as the CWA and other DP3T implementations.

In the GAEN framework (Fig. 1), phones do not need to register with any central provider. Instead they generate a fresh secret key at the start of each day, called the Temporary Exposure Key (TEK), which is then used to generate ephemeral identifiers. Those are broadcast (1) and recorded by other nearby phones. If a user is diagnosed as infected, they will receive authorisation from a health authority (2) to upload their tracing keys to the system's back end (3). The back end validates the authorisation token and stores the keys (4). These can then be downloaded by other users of the system (5), who can use the information to evaluate whether they have been near the infected phone and if necessary, notify the user.

Unlike ROBERT, infected phones upload their own keys instead of payloads received from other phones. Further, the central server does not process user data and instead acts as a bulletin board for contagious keys which are distributed to all users. We provide a more detailed description of broadcast and risk calculation in Appendix A.

3.3 DP3T

Prior to the release of the GAEN framework, the DP3T ENS was proposed by a pan-European group of researchers [31] in March 2020. DP3T uses a Bluetooth protocol very similar to the GAEN framework, but, in contrast, also proposed a number of authorisation procedures.

[1] Available under `https://gitlab.inria.fr/stopcovid19`. Note that we cannot be sure (a) that this code actually runs on the server and (b) if system parameters like token validity deviate from their defaults.

[2] Privacy Automated Contact Tracing

[3] Temporary Contact Numbers

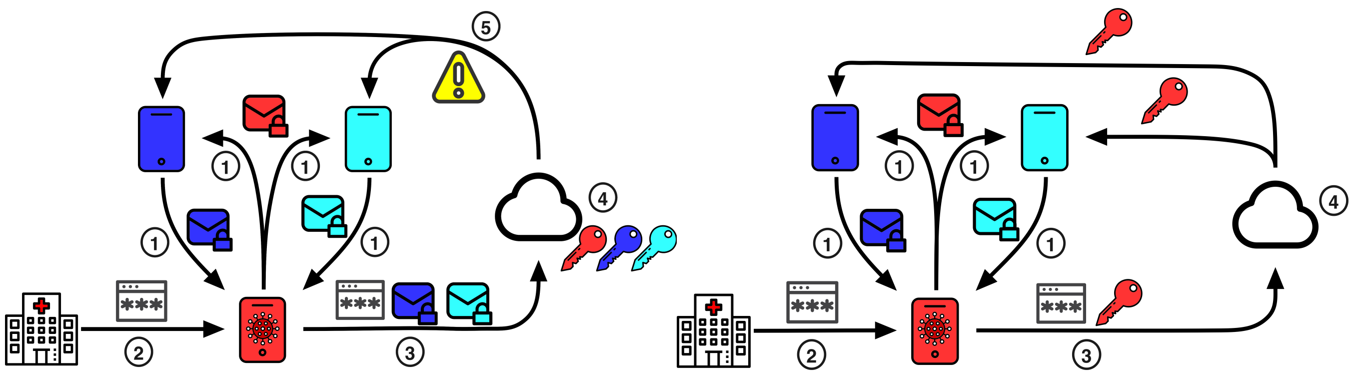

Figure 1: An overview of ROBERT (left) and the GAEN framework (right). The colour of the icons on an edge refers to the entity which originally generated that value.

In the GAEN framework, users who have been diagnosed with COVID-19 must upload their TEKs to a central server, which distributes them to all other users to begin the risk evaluation process. Consequently, the security of this upload procedure is critical to the overall system, as users could maliciously upload their TEKs to cause a false alert and trigger quarantine procedures for users who are not actually at risk.

In the vast majority of GAEN deployments, the user must be diagnosed as infected with COVID-19 by a health professional. The DP3T consortium proposed three different authorisation protocols, which detail how the user can authenticate with a health authority and use the resulting credential to upload their TEKs to the back end. Variants 1 and 2 are very similar to the ones used by the CWA which we describe in Section 3.4. We describe Variant 3, designed to have the strongest security properties, below.

Device Bound Authorisation Codes The three-party protocol between the user, health authority and back end is depicted in Fig. 2. The user generates a blinded commitment of their previous TEKs and transmits them (via the Internet or by displaying a QR code) to the health authority before being tested. Later, if the test is positive, the health authority provides the user with an authorisation code (a digital signature). The user can then transmit this authorisation code to the back end which checks if the code is both valid and recent. The user is assumed to share an authentic channel with the health authority and, separately, with the back end.

This protocol is claimed to ensure that only users who have tested positive can upload and that the keys they upload correspond to those on the device at testing time. Hence, any user wishing to upload malicious values (e.g. from another user) must tamper with their device before being tested. This is argued to significantly raise the cost of any attack on the system, as an attacker must coerce (persuade, bribe) individuals prior to their testing, rather than only targeting those who test positive.

Federation In the GAEN framework, each country must provide its own back end and national infrastructure. This raises the question of how to handle travellers between countries, who are typically a high-risk category for COVID-19 exposure. In DP3T, phones inform their 'home' back end about regions they frequently visit, which then forwards tracing data from these regions' back ends to the phone.

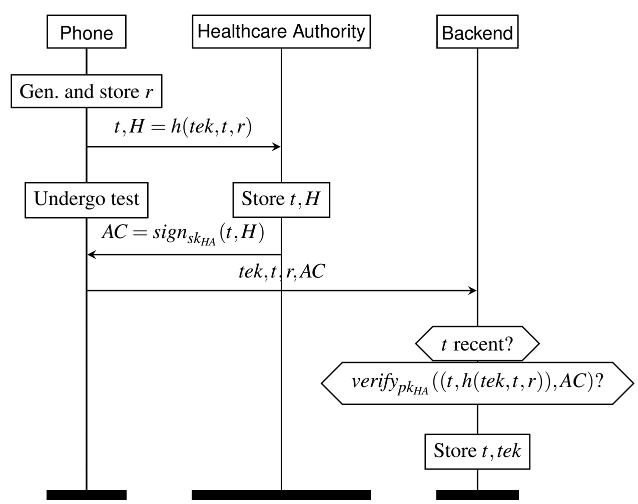

Figure 2: Authorisation procedure for Variant 3 of DP3T. Let t be the starting epoch number of the tek and r a random value. For each tek in the contagious period, an additional H and AC are exchanged, up to a maximum of 14.

3.4 CWA

The Corona-Warn-App was developed by SAP SE and Deutsche Telekom and is based on the GAEN framework. It uses a different authorisation scheme from DP3T. Users are issued a QR code when they visit a testing facility. They scan the code with their device, which automatically retrieves their test result. If the test is positive, they can request a TAN (transaction authentication number) that can be used once to upload the TEKs. As this process is vulnerable to human error, e.g. forgetting to scan the QR code or the medical facility not providing the code, a telephone authentication option (teleTAN) is offered as backup. These two authentication protocols are similar to DP3T variants 1 and 2. They are detailed further in Appendix C.

European Federation Gateway Service The European Federation Gateway Service [24] was developed to provide a solution across the EU. Each country's back end connects to a central database server via mutually authenticated TLS. See Appendix C for more details.

4 Formalisation

In this section, we discuss the scope of our formal models and the decisions we had to make. We have created models for ROBERT, DP3T and the CWA (using the GAEN framework) in Tamarin.[4] In cases where the specifications lack important details or are ambiguous, we examined the public source code of ROBERT and the CWA to inform our decisions. Limitations of our approach are discussed in more detail in Section 7.

Tamarin We perform our analysis with Tamarin [23], a protocol verification tool in the Dolev-Yao model. In this model, messages are described as abstract terms composed from function symbols that represent cryptographic primitives. The behaviour of these cryptographic primitives is specified by an equational theory on terms.

For example, a hash function is represented by function symbol h and the empty equational theory, essentially describing it as a random oracle. Symmetric encryption and decryption are typically written as the function symbols senc, sdec and the equation $\mathsf{sdec}(k, \mathsf{senc}(k,m)) = m$, which describes the behaviour of the decryption function. A term is either an atomic value, a variable or composed from other terms with a function symbol. Atomic values are either public names, which are known to the attacker and protocol, or fresh names, which are drawn uniquely and model keys and nonces.

Protocols are modelled as a combination of multiset rewrite rules (MSRs) and trace-based 'restrictions'. MSRs describe the possible actions in a protocol, such as sending a particular network message or updating local state.

[4]The models are available in the Tamarin repository.

Consider the following example:

```
[ Phone(id,t,k) ] --[ UseKey(id,k) ]-> [ Out(senc(k,t)) ]
```

$\mathsf{Phone}(\ldots)$, $\mathsf{UseKey}(\ldots)$ and $\mathsf{Out}(\ldots)$ are *facts* containing one or more *terms* (like $\mathsf{senc}(k,t)$). The state is modelled as a multiset of facts and is rewritten by subsequent application of MSRs. An MSR can be applied if the facts on the left-hand side ($\mathsf{Phone}(\ldots)$) are part of the state. When the MSR is applied, these facts are removed and substituted by the facts on the right-hand side ($\mathsf{Out}(\ldots)$). The transition is labelled with one or more events, also called action facts, in the middle of the rule ($\mathsf{UseKey}(\ldots)$).

The trace of a system is the sequence of events starting from an empty state. A set of default rules defines the adversary behaviour, incorporating the special facts Out and In for protocol output and input.

Security properties and the aforementioned restrictions are defined in the form of trace properties. These are specified in a first-order logic with two sorts, *temp* for time points, i.e. the position within a trace, and *msg* for terms. The atoms of this first-order logic are: $\perp$ for false, $t_1 \approx t_2$ for term equality, $i \lessdot j$ for time point ordering, $i \doteq j$ for time point equality and $F@i$, where F is an event and i is a variable of sort *temp*, for the appearance of F at index i. We use $i \neq j$ as a shorthand for $i \doteq j \implies \perp$. Restrictions (also specified via trace properties) allow for complex conditions to be imposed on the protocol execution, e.g. checking the validity of a signature or timestamp. A restriction is assumed to be true, while a security property is to be verified.

Scope Our models cover the entire ENS, from user devices and their interactions over Bluetooth, to their communication with the back-end servers and health authorities. There is no limit to the number of users in the system or the sessions they engage in. Similarly, there may be multiple back-end systems in operation, simultaneously and alongside their associated health authorities. The back-end systems may be in federation with one another. By lines of code, our models are (individually) among the largest models compared to Tamarin's model repository (see Appendix D).

Temporal Model Both ROBERT and GAEN divide time into two different size windows, typically on the order of 24 hours and 10 minutes. We dub the former 'days' and the latter 'epochs' for clarity. Any particular epoch is said to belong to a unique day. In GAEN, epochs are aligned with Unix time (i.e. seconds since 1 January 1970). ROBERT, however, aligns epochs with the starting time of each country's back-end services, which we model faithfully.

The temporal model is critical for our analysis and required several iterations to achieve the necessary performance. All three protocols have timestamps that occur in messages, requiring a distinction between timestamps and the time point they refer to. For example, a particular timestamp ('Monday') can refer to many time points during some period.

We tried and subsequently abandoned two approaches from the literature. The first models timestamps as integers, which can be represented in Tamarin via its associative and commutative operator '+'. Reasoning about equivalence is costly in this case. The second models a global clock which emits fresh values for timestamps as a protocol party. In both approaches, any message containing a timestamp could potentially leak information via that timestamp, as it is a priori a secret that could be used elsewhere, e.g. for encryption. Hence, Tamarin's backward analysis needs to perform a case distinction over the origin of that timestamp whenever considering message deduction.

Instead, we model timestamps as public names, avoiding costly deduction because public values are trivially known in Tamarin. Whenever a timestamp e is used in a rule (e.g. as the current epoch), the rule is annotated with an event (e.g. $Epoch(e)$), but the choice of e is arbitrary. Instead, we use restrictions to give events a meaning, e.g. for each two rule instances with $Epoch(e)$ for the same e, any third $Epoch(e')$ between those must have $e = e'$. Other axioms relate epochs to days or require that an order on timestamps implies an order on all corresponding time points. An interesting observation is that we do not need a complete characterisation of time; the protocols we consider require, e.g. non-repetition, but not actual monotonicity. We thus defined a set of sound but not complete axioms, some of which we will now elaborate on. We annotated our model's MSRs with the following events.

PClaimAtRisk(id_R, ts)**:** Phone id_R claims user was at risk at timestamp ts.

Day(ts)**, Epoch**(ts)**:** Whatever action is performed in this rule occurs on day/epoch ts

IsAt(id, p, ts)**:** Phone id is at place p at timestamp ts.

HAClaimInfected(id_I, ts_b, ts_e)**:** Health authority claims that the contagious period of the phone id_I is (ts_b, ts_e).

Within14Days(ts, ts')**:** Timestamps ts and ts' are 14 days apart.

At various points in each protocol, agents may check the ordering or distance between particular epochs and days. We define the following events that occur whenever a rule requires this check to be successful.

EarlierDay(ts, ts')**, EarlierDayEq**(ts, ts')**:** Timestamp for day ts is earlier (or equal to) ts'.

EarlierEpoch(ts, ts')**, EarlierEpochEq**(ts, ts')**:** Timestamp for epoch ts is earlier (or equal to) ts'.

These define the relation that timestamps impose on the protocol run. Their semantics are encoded via restrictions. Consider the following instance for DP3T, where days are the measure of time.

```
EarlierDayEq(ts1, ts2) <=>
  (All #t1 #t2. Day(ts1)@t1 & Day(ts2)@t2
      ==> ts1 = ts2 | (not (ts1 = ts2) & #t1 < #t2))
```

Whenever a timestamp is considered smaller than another somewhere else in the protocol, all rule instances need to be consistent with that order. Any action effectuated at the earlier timestamp must indeed happen before the latter action unless they are the same. The case distinction in the consequent of this restriction governs the structure of the proof. By explicitly negating the first case in the second disjunct, we prune unnecessary proof steps.

We, furthermore, require timestamps to never repeat, or more precisely, whenever they do repeat (i.e. two actions are effectuated on the same day) all actions in between must be assigned the same timestamp.

```
restriction timeDay:
"All d #t1 #t3. Day(d)@t1 & Day(d)@t3 & #t1 < #t3
     ==> (All o #t2. Day(o)@t2 & #t1 < #t2
                              & #t2 < #t3 ==> o = d)"
```

We also relate smaller units of time to larger units of time. For example, in DP3T this concerns intervals and days:

```
restriction timeEpochInDates:
"All e d o #t1 #t3. Interval(e)@t1 & Day(d)@t1
                  & Interval(e)@t3 & Day(o)@t3 ==> d = o"
```

Spatial Model In order to faithfully model the proximity protocol, we employ a granular spatial model in which locations have a unique tag.

The following rules define writing and reading a message `m` on the Bluetooth channel for the location `$place` at a day `$d` and epoch `$e`. Note that the `$` indicates a variable of sort public (a subsort of *msg*) meaning that the rule can be instantiated with arbitrary public names for these variables. The `!` indicates a permanent fact that once added persists and is thus not removed by other rules.

```
rule BLEwr:
    [ In(m) ]
  --[ Day($d), Epoch($e), BLEwr($d, $e, $place, m) ]->
    [ !SpaceTime($place, $d, $e, m) ]

rule BLErd:
    [ !SpaceTime($place, $d, $e, m) ]
  --[ Day($d), Epoch($e), BLErd($d, $e, $place, m) ]->
    [ Out(m) ]
```

The attacker can read and write from an arbitrary location at any time, but we record the BLEwr or BLErd event when they do. The honest protocol rules directly read from or write to `!SpaceTime`, but each rule that does is annotated with an event IsAt collecting the phone identifier, place, day and epoch. Two users `id,id'` are said to occupy the same location at the same time if they both visit the location during the same epoch, i.e. if

```
Ex p d e #t1 #t2. IsAt(id,  p, d, e)@t1
                & IsAt(id', p, d, e)@t2
```

Soundness of the Spatial Model In our spatial model, two users are in proximity to each other if they share the same location at the same point in time. Users in a particular location share a local Bluetooth channel. In this sense, locations can be thought as being overlapping circles whose size is determined by BLE's transmission radius. Users can also travel between locations in which case the timing and direction of their movement is controlled by the adversary.

As the location can be freely chosen, a phone can be at different locations at the same time (day or epoch). This means that proximity is not necessarily transitive, e.g. a phone P_1 can be in proximity to P_2 but not to P_3, even if P_3 is in proximity to P_2. As Bluetooth reception is far from robust, messages transmitted may not be received by other users at the same location. This is reflected since reception is controlled by the adversary. Note that proximity does not necessarily mean the users were close enough to transmit COVID-19, just close enough to potentially transmit Bluetooth packets.

Cryptographic Primitives In ROBERT, we model the Diffie-Hellman key exchange between the user's application and the back end with a prime order group.

For the symmetric encryption used in the ECC and EBID, we employ the standard primitives and the nonce-based encryption described in [16]. The MAC over the ciphertexts is represented using the standard symbolic authentication primitive. These are the most accurate symbolic models published for these types of primitives; however, this nonetheless entails some loss of accuracy. For example, as in ROBERT the MAC is truncated to 40 bits, the probability of successful forgery is not negligible, yet this cannot happen in the symbolic model.

In the CWA, we model key derivation using the standard primitives for HKDF. We treat the use of AES to unroll additional keys similarly. We do not model the Associated Encrypted Metadata, as it only contains auxiliary information for the risk calculation and is not authenticated.

5 Security Properties and Threat Model

Intuitively, the security desired of an ENS is easy to state:

> *The ENS should notify the user of risk if, and only if, the user was in proximity to an individual who was diagnosed by a health authority as being contagious at that time.*

However, this property becomes considerably more complex when we consider (a) what 'in proximity' and 'at the time' means w.r.t. the technical constraints within which the protocols have to operate, (b) how users and health authorities relate across countries, and (c) when we consider the impact of the compromise of devices, communication channels or infrastructure. In this section, we introduce and justify our security definitions as well as describe our systematisation of the attacker's capabilities.

Recall that we model the proximity of two users by providing each location with a unique identifier. We treat two users at the same location during the same epoch as being in proximity for the purposes of exposure notification. In practice, the ENS would take into account the precise duration of the exposure and other information, e.g. signal strength. But these are not relevant for our evaluation of the overall design of each system.

In each system, we have multiple designated health authorities (one per country) who can perform diagnostic tests on users. We associate each user with their phone, as we can only measure if phones are in proximity, not their users. Putting this together, we can formally state our soundness property:

Definition 1 (Soundness)**.** *Whenever an honest phone id_R reports to a user that they were in contact with an infected phone at time point t_c, then there is another phone id_I for which the following conditions are met* a) *the health authority (HA) determined that id_I was contagious in the time interval $[t_b, t_e]$,* b) *t_b and t_e are no more than 14 days apart,* c) *both were in proximity at time t_c,* d) *$t_c \in [t_b, t_e]$, and* e) *$id_I \neq id_R$.*

Condition *a)* expresses the integrity of the alarm. It requires a discretisation of time that is present in all three protocols. Condition *b)* ensures that the health authority's time interval remains in reasonable bounds. This is important in case of a compromised health authority. Condition *c)* will be expressed by discretising space similarly to time. In contrast to the discretisation of time, this is not due to the protocol, but a modelling choice. We say that two parties are in proximity, if they sent or received a message on the Bluetooth channel at time t and place p, where t and p are discrete values.

Condition *d)* links the time of proximity to the health authority's diagnoses. In DP3T, the health authority only determines the day of the (positive) test. Consequently, the condition is weakened in this case. Condition *e)* expresses that alarms cannot be caused by reflection attacks. Such an attack is not a big problem in itself—a risk notification to a user who has already been diagnosed as infected is presumably harmless, but may be confusing and potentially distort statistics about risk events. In Tamarin, soundness is expressed as follows (see Section 4 for details):

```
All idR instClose dayClose tsRisk #tRisk.
    PClaimAtRisk(idR, dayClose, instClose)@tRisk
  & Day(tsRisk)@tRisk ==>
(Ex idI place dayContag dayTest [#t1..t6].
    IsAt(idR, place, instClose)@t1
  & IsAt(idI, place, instClose)@t2
  & HAClaimInfected(idI, dayContag, dayTest)@t3
  & EarlierDayEq(dayContag, dayClose)@t4
  & EarlierDay(dayClose, dayTest)@t5
  & Within14Days(dayContag, dayTest)@t6
  & not (idR = idI))
```

Whilst the security of the authorisation protocol is implicitly captured by our soundness property, it is instructive to make it explicit, as it describes a common point of failure. In ENS using the GAEN framework, the authorisation is particularly sensitive, as it concerns the publication of a phone's secret key via the back end.

Definition 2 (Upload Authorisation for GAEN protocols)**.** *If a phone id_I generated a key k and the back end B_{cc} releases k, then id_I was diagnosed as infectious by the health authority.*

In ROBERT, the information disclosed does not originate from the infected phone, but was picked up by it.

Definition 3 (Upload Authorisation for ROBERT)**.** *If the back end B_{cc} accepts the upload of ephemerals, presumably recorded by a phone id_I, then id_I was diagnosed as infectious by the health authority.*

5.1 Evaluation Methodology

We seek to systematically characterise the space of possible attacks from an attacker endowed with a rich set of capabilities that we will detail below. As Tamarin is capable of automatically deducing attacks as well as proofs of their absence, we use it to guide our search by counterexamples, i.e. attacks. This technique was used in the past to compute strongest threat models under which a claim holds true [19] or compute protocol security hierarchies [7].

Note that both soundness and upload authorisation are implications of the form $A \implies B$, hence we begin by attempting to prove $A \implies B$ ('the property holds'). When Tamarin inevitably returns an attack, we (manually) identify an attack pattern P_1, characterising a class of similar attacks using the compromise events in Table 4 in the Appendix. We then extend the conclusion to include the possibility of the attack, i.e. $A \implies B \vee P_1$ ('the original property holds or the attacker followed P_1'). This is a weaker property and thus potentially provable. We repeat this process until no more attacks are discovered and Tamarin is able to prove a result of the form $A \implies B \vee P_1 \vee \cdots \vee P_n$. As we now have a formal proof that there is no attack against $A \implies B$ unless it matches one of the P_i, we obtain an exhaustive categorisation of all attacks (against this target property and within the bounds of our formal model).

Compare this with the classical approach, where a purportedly realistic threat model is fixed a priori. All too often, the literature considers only those threat models realistic for which either security can be proven or for which attacks can be mitigated by technical means, like modifying the protocol. For ROBERT, e.g. back end compromise would be excluded from the start, as there is no way to mitigate this attack.

Similar to a classical threat model, attack patterns give a conditional security statement: Their negation $(\neg P_1 \wedge \ldots \wedge \neg P_n)$ guarantees the property. If one thinks that $(\neg P_1 \wedge \ldots \wedge \neg P_n)$ is realistic, then the proof for $A \implies B \vee P_1 \vee \cdots \vee P_n$ gives assurance, just like a classical threat model.

This method has three advantages over classical threat models. First, attack patterns are more precise than threat models typically are: Instead of saying 'the attacker can compromise phones' they can say: 'the attacker can compromise one phone that was in proximity to the victim's phone in the critical time window.'[5] Second, when we consider the necessary condition for the property to hold, i.e. the above negation $(\neg P_1 \wedge \ldots \wedge \neg P_n)$, we obtain a negative statement about the attacker (the attacker cannot do X), whereas a threat model gives a positive statement (the attacker can [only] do Y). It is easier for the system operator to check if P_1 is really impossible than to ensure the threat model is not missing real-world possibilities. Third, if the attack patterns are fine-grained enough, their risk can be estimated, whereas the risk 'attack outside the threat model is possible' is harder to quantify.

5.2 Adversary Capabilities

We consider an arbitrary number of phones, with an arbitrary movement and broadcast pattern determined by the adversary. Key rotation is also under the control of the adversary. Any number of these phones may go through the testing procedure, begin to upload their keys or finish that process. Likewise, any combination of phones can download updates from their back-end server and process risk calculations. We support arbitrary interoperation, as there may be several back ends and health authorities whose regions of coverage may be disjoint or overlapping; however, honest phones interact only with the health authority they were registered with.

The adversary is capable of compromising any of the agents in the system, including the back end infrastructure of the ENS. They may also compromise a phone, including its key material and records of previously witnessed ephemerals. Compromising the back end grants the adversary access to the underlying database and the back end's signing key. Further, whilst the attacker is considered to have full control over messages sent over the Internet, we model the various secure channels between phones, back end and health authority faithfully. Unless the attacker compromises a device in a particular channel, they are unable to inject or edit messages over the channel, although they may delay or block such messages arbitrarily. In addition to compromising phones and using them to send ephemerals from particular locations, the adversary may also inject, read and edit Bluetooth messages. We allow phones and the adversary to be in multiple locations during the same epoch, to reflect the fact that our locations may be very small due to the small intended radius of the Bluetooth protocol. This allows the adversary to perform relay and replay attacks. We list all the adversary's capabilities in Table 4 in the Appendix.

[5] This is in contrast to prior work [19, 7], where attack patterns were limited to adversarial capabilities, which allowed automating their discovery. We purposefully explore the space by hand, adding information to the attack pattern until it is descriptive about the attack.

Table 1: Runtime and memory consumption

	ROBERT	DP3T	CWA
Verification time	14 h 8 min	1 h 48 min	1 h 3 min
Peak memory	43.15 GB	6.83 GB	7.67 GB
Proof steps			
Upload auth.	80	75	4271
Soundness	170,137	31,561	17,262

6 Analysis Results

Based on the evaluation methodology in the last section, we present a categorisation of all attacks (in our model) against soundness and upload authorisation in ROBERT, DP3T and the CWA.

We found 27 attack patterns in total. Table 3 provides a comparative summary of the attacks, listing the requirements for mounting attacks against upload authorisation and soundness. The executive overview in Table 2 provides an assessment of the potential impact (in terms of falsely alerted users) by attack vector. We present the attacks in full detail in the full version [25, Appendix D].

The verification of the models was performed on an Intel(R) Xeon(R) CPU E5-4650L workstation with four cores for each model. The verification time, peak memory consumption and proof steps are presented in Table 1. We see that the majority of the verification time was spent on soundness.

Table 3 and [25, Table 4] summarise attack patterns by target property. For example, the negation of upload authorisation in ROBERT (Definition 3) says that there is a phone id_I which the back end marks as infected, but that was not diagnosed as such by the health authority (similar for Definitions 1 and 2). As we can refer to the target phone id_I in attack patterns, we can even include a quantitative assessment. For instance, the attack pattern A1 permits *any* phone to share the QR code received by the health professional, but requires the *same* phone to upload their EBIDs. For example (simplified):

```
All id t #t1. MarkPositive(id, t)@t1 ==>
    (Ex dt #t3. TestPositive(id, dt)@t3)
// This was the target property. Now attack pattern 1:
  | (Ex idI ebid qr [..].
CorruptPhoneReceive(idI, <'infected', [..], qr>)@t1
CorruptPhoneSend(id, <'upload_hello', ebid, [..], qr>)@t2
) | ... // Other attack patterns
```

We can test this hypothesis by replacing id_I, which represents the infected phone sharing the QR code, with id, the target phone, in CorruptPhoneReceive. Phone identities in attack patterns are always existentially quantified, hence such substitutions strengthen the corresponding lemma. If Tamarin can prove this stronger lemma, we report the more specific attack pattern. If not, Tamarin provides an attack trace disproving the modified lemma. As the original lemma was proven, the attack trace must be the original attack pattern (otherwise, the original lemma would not be provable) but the substituted identity must be different from id_I (otherwise, the modified lemma would be provable). We manually inspect the attack to see if that phone's identity is otherwise related to id_R in particular and if the actions of this phone can be repeated arbitrarily often.

Besides phone identities, we applied this methodology to QR codes to find out whether a compromised back end is necessarily linked to the phone in question.

Previous research has identified many of these attack patterns before. Whilst in some cases, we do discover some new variations that widen their applicability, the differences are not substantial. Rather, we provably show that these attack patterns characterise all possible violations of the desired security properties and, consequently, rule out further attacks and allow a systematic comparison of the three protocols on a level playing field with a unified terminology. In these details, we find a number of surprises and previously unremarked differences in attack effectiveness.

6.1 Executive summary

In Table 2, we further condense these patterns to allow for a high-level comparison of the three designs.

The different effects of a back-end attacker are a direct and well-known result of the centralised architecture: In ROBERT, this attacker can simply send a risk notification, while for DP3T and the CWA, Bluetooth communication is required even when the back end is compromised. If the attack is not targeted, but aims at disrupting the service, then there is little difference, as we can assume almost any phone to have been in contact with at least one other phone.

The starkest difference is when a health authority is attacked. The unspecified upload procedure in ROBERT seems to have left a weak spot. A QR code captured in any country is enough to declare an arbitrary phone infected and thus trigger the alarm on a number of phones only limited by how many ephemerals the attacker can capture, be it by choosing the right phone to declare infected, i.e. one that was near many target victims, or by setting up an antenna.

While DP3T's specification achieves strong security properties, the CWA implements a weaker protocol.[6] The results of our end-to-end analysis on false risk notifications reveals intriguing economics should the attacker have access to several infected phones. If a group of pranksters wants to raise alarms and one of them can obtain a positive test result, then in DP3T, they can trigger warnings for everyone who met that person, in the CWA for whoever in the group has met the most people and in ROBERT for anyone that met anybody in that group.

The security of authorisation procedure governs how much freedom the attacker has in choosing the infected phone. If, in

[6] Note that some real world deployments of DP3T also chose a weaker authorisation scheme.

	Attack vector	ROBERT	DP3T	CWA
Back end	all-out	**all phones**	all phones in proximity to some other phone	
	targeted	any phone	all contacts of targeted phones	
Health authority	active	any phone	all contacts of targeted phone	
	passive (steal QR code)	**all contacts of arbitrary phone***	(phone-and-test-specific authentication codes)	
Infected phone(s)	modified phone	**contacts of the whole group**	contacts of phone	max. contacts among group mem.
	+ passive antenna	**all phones in reach**	*(see row 'modified phone')*	
	+ active antenna	*see row 'modified phone'*	**all phones in reach**	

(*) Affords attack vector "infected phone"

Table 2: Executive overview: maximum impact by attack vector and protocol. The effect with the strictly highest impact is bold if defined. Attack vectors are colour-coded by attack complexity and detection risk. The more red (greyscale: darker) the attack vector's background, the more realistic it is.

addition, the attacker has the knowledge to set up an antenna, then the severity of this vector becomes a numbers game.

The most effective attack for a lone individual on ROBERT is to obtain as many ephemerals as possible and upload them using a compromised phone or a stolen QR code (X4). Here, the attacker is passive on the Bluetooth channel and can affect as many phones as the upload limit and the ability to obtain ephemerals. Comparable attacks are possible in DP3T (Y4, Y5) and the CWA (Z2, Z3), where the attacker uses a disclosed or forged key for broadcasting ephemerals and maliciously uploads it. However, in DP3T and the CWA, the attacks require active transmission on the Bluetooth channel, which risks detection.

Risk Assessment While the centralised/decentralised dichotomy has an impact on back-end compromise, this attack vector requires considerable expertise and is risky for the attacker. It is much easier to attack the health authorities, because there are many of them and they are difficult to secure. In ROBERT, theft of QR codes can have an enormous impact and requires little expertise. Moreover, health authority's likely have more than one QR code at their disposal. By setting up a local instance of the ROBERT back end (based on the published source code and its default settings), we experimentally confirmed that a QR code sheet contains QR codes for 10 days, each with an 8-day validity.

Attacks via modified phones are even simpler to mount and, here again, the choice between centralised/decentralised comes to effect: If a group of people decided to disrupt ROBERT in a coordinated effort, they could easily collect ephemerals and route them to a single phone with modified software. The impact can be amplified with an antenna if the attacker has the expertise. On our ROBERT instance, we experimentally confirmed that it is sufficient to reliably receive ephemerals over a 17-minute period whereas for DP3T and the CWA, the attacker must reliably send for 10 minutes [9]. We consider the risk of a sustained attack with passive and thus hard-to-detect antennas to be significant and the potential impact high enough to disrupt the system, at least temporarily. An active antenna is, nevertheless, a threat if the attacker is risk-tolerant or intents a one-off attack.

6.2 Upload Authorisation

We will now discuss the attacks in more detail, starting with those against Definitions 2 and 3.

Infrastructure Compromise In order for the attacker to list n phones as infected, with all of the systems we studied, it suffices for the attacker to compromise a back end or health authority (A2, A3, A4, B1, B2, C1, C2). In this case, the attacker can send false positive test results or can 'sign-off' keys of their choice. Vulnerabilities of this nature occur in practice, for example, the Swiss contact tracing application 'SwissCOVID', which uses the same design as the CWA, failed to validate signatures made by the health authority [1]. Such vulnerabilities can have a much more serious impact on ROBERT, as we will see in the next section.

Device Compromise Of course, attacking back-end infrastructure is likely to be much more difficult than compromising other users' phones. ROBERT, DP3T and the CWA all achieve very different security properties in this regard. ROBERT achieves the weakest security property, where the compromise of any phone that recently has or is about to test positive (or otherwise obtains a valid QR code) allows the attacker to mark arbitrarily many devices as infected (A1, A2, A3, A4). This is in part because ROBERT, as specified, has no limitations on the number of ephemerals a user can upload.[7] In practice, an operator could review anomalously sized uploads, but the effective upper limit could not be lower than the total number of a user's contacts and is thus likely to be in the hundreds or thousands (e.g. a retail worker in a busy supermarket). Contrastingly, in DP3T and the CWA, a device

[7] We did indeed not find any limit on how many ephemerals can be uploaded in a single batch request in the server source code as published. In fact, our local tests showed that more than a million EBIDs can be uploaded to our ROBERT server.

Table 3: High-level comparison of the attack surface. We categorise the attack patterns from the full version [25, Appendix D]. Each bullet point summarises a common attack vector.

Effect	All	ROBERT	DP3T	CWA
Mark *n* phones infected	• Compromise back end or HA (A2, A3, A4, B1, B2, C1, C2)	• Compromise 1 phone at any time (A1) • Passively compromise HA (obt. QR code) (A2)	• Compromise *n* phones prior to testing (B3)	• Compromise *n* phones at any time (C1)
Raise alarm on *n* phones	• Relay Attacks (Victim phone ↔ infected phone) (X3, Y3, Z1)	• Observe *n* phones and mark *n* phones as infected (X1, X2, X4) • Compromise back end (X5, X6, X7). • Compromise registration phase* (X7)	• Replay Attack: Victim ↔ Positive (Y3) • Transmit to target and mark *n* phones as infected (Y1, Y2, Y4, Y5, Y6, Y7)	• Transmit to target and mark *n* phones as infected (Z2, Z3, Z4).

(*) The encoding of Definition 1 excludes the trivial attack where the phone is fully compromised.

is limited to uploading a small number of keys, typically 14, since devices only upload their own keys.

DP3T achieves a stronger security property than either ROBERT or the CWA, as the attacker must compromise a phone prior to the user getting tested (B3). This constitutes a substantial barrier to an attacker, since only a small percentage of tested people will actually test positive, meaning the attacker must compromise multiple devices in order to achieve a single malicious upload. The upload authorisation procedure of DP3T requires the user to commit to their device's keys in advance, preventing the attacker from inserting their own keys later. However, to our knowledge, this authorisation procedure has not been deployed in practice, most likely because it requires a closer integration between the contact tracing system and the health authority's system which may not be practical in all countries.

In the CWA, it is possible to mark people as infected by uploading TEKs of other users (C1). For example, if a TEK has been used by someone near a large group of people which is later uploaded by another positively tested compromised phone. In practice, however, it can be challenging to extract the TEKs managed by the GAEN framework.

Whilst modelling the CWA, we found that due to underspecification in the upload authorisation procedure, an attacker who learnt the *guid* belonging to a phone which had tested positive (e.g. by scanning the QR code sent to the user even after it had been used), could request an arbitrary number of TANs. We found that this attack had independently been reported to the CWA team [3]. Nevertheless, our analysis proves that enforcing a one-to-one relationship between TANs and registration tokens, as suggested in the initial report, prevents the attack.

6.3 Raise Alarm on *n* Phones

Escaping the Contagious Period We discovered a previously unreported attack on soundness in ROBERT: An attacker who receives a positive test result is able to upload ephemerals outside of their contagious period as determined by a health authority (X2). Whilst the attack is relatively minor in isolation, it allows a determined attacker to target victims over a longer period of time than they would otherwise be able to. For example, whilst the health authority may wish to only begin the tracing period for the previous three days, an attacker would be able to upload ephemerals for the previous two weeks. ROBERT is vulnerable to this attack, because the authorisation codes used do not identify the start of the contagious period. Instead, long codes are generated in advance in batches with a validity of 8 days. We also point out that this long validity period makes authorisation codes more valuable and thus incentivises their compromise. We have responsibly disclosed this vulnerability to the ROBERT team.

Relay and Replay Attacks All three protocols allow an attacker to carry out a relay attack in order to mark honest users as at risk (X3, Y3, Z1). Whilst the risk of these attacks was generally noted in the early analyses [2, 17], the impact of this attack on ROBERT is greater than previously realised.

In this type of attack, the attacker is present at two different locations at the same point in time and forwards messages between them. In ROBERT, the attacker must observe messages from victims and rebroadcast them near positive individuals. Conversely, in DP3T and the CWA, the direction is reversed: The attacker must broadcast near each victim. Comparatively, the attacker only needs to broadcast near one infected party in ROBERT.

Real world analysis by Baumgärtner et al. [9] has shown a substantial asymmetry between transmission and reception of the Bluetooth packets used in ROBERT, DP3T and the CWA. In particular, whilst their tests showed they could effectively receive the Bluetooth packets in real-world conditions (collecting around 20 packets a minute in Frankfurt's central train station), they struggled to transmit reliably at ranges of greater than 10 m with commodity equipment, e.g. without a dedicated antenna. Transmission is hampered by the fact that in DP3T and the CWA devices only actually wake up to receive messages for a short period of time in a 5-minute window.

This asymmetry makes large-scale relay attacks against ROBERT much more practical than against DP3T or the CWA. In ROBERT, the attacker simply has to acquire a single Bluetooth message from each device it wishes to target which a dedicated attacker can achieve over a large range.[8] As mentioned earlier, it is sufficient to reliably receive ephemerals over a 17-minute period which corresponds to 102 messages per target phone. Contrastingly, in DP3T and the CWA, the attacker must successfully inject multiple messages over a 10-minute period to each target. Further, the ROBERT attacker is entirely undetectable to its victims, compared to the need to mass-broadcast ephemerals to the DP3T and the CWA victims which may attract attention from local authorities and civil fines. A further distinction is that even when considering a selective attack, a ROBERT attacker may passively gather as many ephemerals as they wish and then decide who should be falsely marked at risk at the last moment. Contrastingly, a DP3T or CWA attacker must pre-commit exactly who they will attack by actively broadcasting ephemerals to them.

DP3T is vulnerable to an additional replay attack (Y3), previously reported by AISEC [2] in which the attacker can record an ephemeral and later replay it on the same day. DP3T ephemerals are not bound to specific timeslots, instead they are generated from a single day seed and are broadcast in random order. Similarly, whilst CWA's ephemeral identifiers are bound to a specific epoch interval, in practice user devices tolerate packets which may be as much as two hours outside of this interval, ostensibly for reliability when a phone's local time is incorrect. Contrastingly, ROBERT rejects ephemeral identifiers which are outside a narrow window of a couple of seconds, rendering it invulnerable to this attack.

Infected User Compromise Attacks Similar to relay attacks, the adversary may compromise an infected user (via phishing, malware, extortion, bribing or stealing their QR code) in an attempt to have other users alarmed. That is, unlike in a relay attack, the adversary has privileged access to a device or information belonging to an infected user.

In ROBERT, the attacker must first observe ephemeral broadcasts from the target users, as to inject records of these into the target user's device (X1, X2, X4). As in the relay case, the attacker only needs to have access to a single infected user in order to alarm many devices. With DP3T (Y1, Y2, Y4, Y5, Y6, Y7) and the CWA (Z2, Z3, Z4), the attacker must instead transmit to the target users, using ephemerals generated from the user's secret key.

These attacks are especially powerful in ROBERT since the attacker can entirely control how the Bluetooth contact is recorded. Contrastingly, in DP3T, the Bluetooth reception is still performed by an honest device, limiting the attacker to the capabilities of their hardware. That is, in ROBERT, an attacker could observe a small number of ephemerals in a 15-minute period (e.g. at the very limit of their reception range), but record the contact on the compromised device as a strong, nearby signal for the same duration. In DP3T and the CWA, the attacker would have to be much closer to be able to generate a strong enough signal for the user's device to accept it.

Non-Local Attacks on ROBERT For DP3T and the CWA, a user can only be falsely alarmed if the attacker is able to either inject Bluetooth messages into their vicinity or compromise a device in the victim's vicinity. This limited 'blast radius' imposes a substantial economic burden on an attacker wishing to carry out a mass-broadcast attack, as they must be able to inject Bluetooth messages into the proximity of each recipient. As we noted earlier, this comes with a variety of physical and operational challenges.

Contrastingly, ROBERT's blast radius is extremely large. An attacker who can compromise the back-end server can alert any and all users using that back end without restriction. Further, for users of federated ROBERT back ends, the attacker can still cause malicious alerts after observing them once. Consequently ROBERT is more vulnerable to catastrophic failure than DP3T and the CWA. We do not believe this property has previously been noted.

We also discovered a subtle risk for future implementation errors: If the back end's secrets become guessable, e.g. due to a bad key-generation algorithm, the attacker can trigger alarms without active control over the back end, e.g. without controlling the channels between back end and phones. At that point, the attacker can transform any previously received ephemeral, no matter how old, and convert it into a recent one. A single infected phone is enough to raise an alarm with any of those.

Federation Attacks on DP3T and the CWA We find a number of attacks on DP3T and the CWA when deployed in a federated setting according to their original specifications. Whilst we do not report all these attacks here for reasons of space, they arise due to ambiguous requirements regarding the release of the same uploaded key by multiple back ends. In both systems, the back end must only release a day key once it will no longer be accepted by user devices, for example by

[8] A first estimate targets a maximum attenuation of 92 dB. Assuming line of sight and an antenna that has 20 dB gained (e.g. a satellite dish) it is plausible to reach a distance of 5 km. For comparison, this is roughly the radius of the city of Paris (population 10 M). We are not aware of studies for BLE, but there are empirical results on 802.11b /n/ac [4], which operates on a similar frequency and 20 MHz bandwidth (instead of 1 MHz).

waiting until the end of the day the key is reported for. In the case of the CWA and other systems build on the GAEN framework, this necessitates waiting two hours after the end of the day, to account for the clock skew toleration.

In a federated setting, this is compounded since individual back ends may release the same key. Should any back end release a key 'early' prior to the expiry time of another back end, an attacker can rebroadcast this uploaded key until the later expiry time, triggering false notifications. A limited form of this attack was reported in [21] with a number of countries; however, we believe the first report [20] actually dates back to April 2020 by Hirschi. Our analysis shows that ensuring federated servers agree on the expiry date for a particular key provably prevents this class of attacks. This mitigation has been adopted by the European Federated Gateway Service.

User Compromise Another attack possible in any system is when the attacker simply compromises the phone of the victim. In this circumstance, in DP3T and the CWA they can inject a database record corresponding to a past exposure to an infectious person. In ROBERT, they can simply fake a positive exposure status response from the server. A similar style of attack was reported by [21] and dubbed 'time travel' attacks, as they focused on resetting the phone's clock and simultaneously injecting a Bluetooth message.

7 Limitations

Our formal analysis is carried out in the Dolev-Yao model where cryptographic primitives are abstracted, hence attacks against them are not covered. Moreover, time and space are usually not considered in this model. While our spatial model is reasonably close to reality, our model of time assumes all phones to have an accurate clock. Hence it cannot capture desynchronisation attacks (e.g. [27]). In our analysis, we focused on one aspect, the soundness of alarms, which ensures that every risk notification is justified by a risk event. However, we do not consider the privacy aspect or completeness in this work, the latter requiring that every risk event result in a notification,

We analysed the protocols according to their specification. We validated how QR codes are distributed in ROBERT on the published source code, because this part is underspecified and the impact of the attack depends crucially on how easy they are to obtain. The other attacks can be easily validated by specification, hence a correct implementation would also allow to mount them. However, we cannot be sure that the implementation does not introduce new vulnerabilities by diverging from the specification.

All models assume that a particular phone is registered in only one country and that there is only one back end per country. This is explicit in the DP3T and CWA specifications and implicit in ROBERT's.

Like the protocol specification, our models abstract both the health professionals and the infrastructure used to communicate with the patient into one party, the health authority. Our model considers only a single health authority per country, thus requiring each country's infrastructure to have a single key for signatures and to share all communication channels. While this is not strictly required by the specification, in practice each participating country's infrastructure is set up on the national level.

Moreover, we assume that a user tests positive only once. This is a requirement in the ROBERT specification, but the enforcement mechanism is left abstract. In DP3T and the CWA, the specification suggests that all cryptographic material specific to that device is deleted after testing positive.

We simplify the detailed risk calculation from the model to a yes/no decision, ignoring the associated encrypted metadata. Instead, we allow the adversary to arbitrarily determine the outcome of risk calculations, which is equivalent to assuming that the adversary entirely controls the metadata and risk calculation. Additional model-specific simplifications are detailed in Appendix D.2.

8 Conclusion

In this work, we employed formal verification techniques to systematically categorise existing and new attacks on the soundness of risk notifications in ENS. Although exposure notification protocols seem fairly simple, the details are of utmost importance. Besides complex temporal and spatial interactions between phones, cross-country coordination between health authorities, back ends and authentication providers need to be modelled. This and the design of fine-grained compromise rules caused our formal models to reach several thousand lines, pushing the practical limit of existing tooling.

Overall, a knowledgeable attacker or a dedicated group of pranksters could realistically damage a system with little risk of getting caught. We want to avoid an oversimplified takeaway message, so we refer to Section 6.1 for a half-page executive summary of the attacks we found.

Some of the discovered high-level attacks were known to the community (at least those for DP3T), albeit in a simpler form and without a precise understanding of their consequences and requirements. We found new attacks that affect possible secondary use cases or could become relevant due to implementation errors (consider, e.g. X7). We therefore hope that the code which runs on back-end servers remains open source and is continuously published for review.

This commendable practice points at an interesting research question: How can we make sure future versions do not add new attacks? Or that the code correctly implements the specification. Extracting protocols from source code is just one possible answer. Others are monitoring or testing approaches to compare new versions with formal models such as the ones we presented here.

References

[1] Fabian Aggeler. Missing signature validation of JWT when alg=none · Advisory · DP-3T/dp3t-sdk-backend. GitHub. July 30, 2020.

[2] Fraunhofer AISEC. Pandemic Contact Tracing Apps: DP-3T, PEPP-PT NTK, and ROBERT from a Privacy Perspective. Cryptology ePrint Archive, Paper 2020/489. 2020.

[3] An attacker can generate unlimited valid TANs · Issue #144 · corona-warn-app/cwa-verification-server. GitHub. June 7, 2020.

[4] Daniele Antonioli, Sandra Siby, and Nils Ole Tippenhauer. Practical Evaluation of Passive COTS Eavesdropping in 802.11b/n/Ac WLAN. In: *Cryptology and Network Security*. Cham, 2018, pp. 415–435.

[5] Benedikt Auerbach, Suvradip Chakraborty, Karen Klein, Guillermo Pascual-Perez, Krzysztof Pietrzak, Michael Walter, and Michelle Yeo. Inverse-Sybil Attacks in Automated Contact Tracing. In: *Topics in Cryptology – CT-RSA 2021*. Cham, 2021, pp. 399–421.

[6] Gennaro Avitabile, Daniele Friolo, and Ivan Visconti. Terrorist Attacks for Fake Exposure Notifications in Contact Tracing Systems. In: *Applied Cryptography and Network Security*. Cham, 2021, pp. 220–247.

[7] David Basin and Cas Cremers. Know Your Enemy: Compromising Adversaries in Protocol Analysis. In: *ACM Transactions on Information and System Security* 2 (Nov. 17, 2014), 7:1–7:31.

[8] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A Formal Analysis of 5G Authentication. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA, 2018, pp. 1383–1396.

[9] Lars Baumgärtner et al. Mind the GAP: Security & Privacy Risks of Contact Tracing Apps. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). Dec. 2020, pp. 458–467.

[10] Ran Canetti, Yael Tauman Kalai, Anna Lysyanskaya, Ronald L. Rivest, Adi Shamir, Emily Shen, Ari Trachtenberg, Mayank Varia, and Daniel J. Weitzner. Privacy-Preserving Automated Exposure Notification. Cryptology ePrint Archive, Paper 2020/863. 2020.

[11] Marco Casagrande, Mauro Conti, and Eleonora Losiouk. Contact Tracing Made Un-Relay-Able. In: *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*. New York, NY, USA, 2021, pp. 221–232.

[12] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. ROBERT: ROBust and Privacy-presERving Proximity Tracing. May 2020.

[13] Cas Cremers, Martin Dehnel-Wild, and Kevin Milner. Secure Authentication in the Grid: A Formal Analysis of DNP3: SAv5. In: *Computer Security – ESORICS 2017*. Cham, 2017, pp. 389–407.

[14] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication. In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016 IEEE Symposium on Security and Privacy (SP). May 2016, pp. 470–485.

[15] Noel Danz, Oliver Derwisch, Anja Lehmann, Wenzel Puenter, Marvin Stolle, and Joshua Ziemann. Provable Security Analysis of Decentralized Cryptographic Contact Tracing. Cryptology ePrint Archive, Paper 2020/1309. 2020.

[16] Alexander Dax, Robert Künnemann, Sven Tangermann, and Michael Backes. How to Wrap It up - A Formally Verified Proposal for the Use of Authenticated Wrapping in PKCS#11. In: *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. 2019 IEEE 32nd Computer Security Foundations Symposium (CSF). June 2019, pp. 62–6215.

[17] DP3T. Privacy and Security Attacks on Digital Proximity Tracing Systems. Apr. 21, 2020.

[18] DP3T. Upload Authorisation Analysis and Guidelines. Apr. 30, 2020.

[19] Guillaume Girol, Lucca Hirschi, Ralf Sasse, Dennis Jackson, Cas Cremers, and David Basin. A Spectral Analysis of Noise: A Comprehensive, Automated, Formal Analysis of Diffie-Hellman Protocols. In: 29th USENIX Security Symposium (USENIX Security 20). 2020, pp. 1857–1874.

[20] Lucca Hirschi. Lack of clock synchronization can be exploited by a weak adversary to trigger (targeted) false positive · Issue #14 · DP-3T/reference_implementation. GitHub. Apr. 16, 2020.

[21] Vincenzo Iovino, Serge Vaudenay, and Martin Vuagnoux. On the Effectiveness of Time Travel to Inject COVID-19 Alerts. In: *Topics in Cryptology – CT-RSA 2021*. Cham, 2021, pp. 422–443.

[22] Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. Verifpal: Cryptographic Protocol Analysis for the Real World. In: *Progress in Cryptology – INDOCRYPT 2020*. Cham, 2020, pp. 151–202.

[23] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: *Computer Aided Verification*. Berlin, Heidelberg, 2013, pp. 696–701.

[24] Martin Mitev and Hannes Rollin. eHealth Network: European Proximity Tracing. 2020.

[25] Kevin Morio, Ilkan Esiyok, Dennis Jackson, and Robert Künnemann. Automated Security Analysis of Exposure Notification Systems. Full version. 2022. arXiv: 2210.00649.

[26] Aleksi Peltonen, Ralf Sasse, and David Basin. A Comprehensive Formal Analysis of 5G Handover. In: *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA, June 28, 2021, pp. 1–12.

[27] Yarin Perry, Neta Rozen-Schiff, and Michael Schapira. A Devil of a Time: How Vulnerable Is NTP to Malicious Timeservers? In: *Proceedings 2021 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium. Virtual, 2021.

[28] Krzysztof Pietrzak. Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing. In: *Progress in Cryptology – INDOCRYPT 2020*. Cham, 2020, pp. 3–15.

[29] Benny Pinkas and Eyal Ronen. Hashomer–Privacy-Preserving Bluetooth Based Contact Tracing Scheme for Hamagen. In: *Real World Crypto and NDSS Corona-Def Workshop*. 2021.

[30] Leonie Reichert, Samuel Brack, and BjÖRN Scheuermann. A Survey of Automatic Contact Tracing Approaches Using Bluetooth Low Energy. In: *ACM Trans. Comput. Healthcare* 2 (Mar. 2021).

[31] Carmela Troncoso et al. Decentralized Privacy-Preserving Proximity Tracing. 2020.

[32] Serge Vaudenay. Centralized or Decentralized? The Contact Tracing Dilemma. 531. 2020.

[33] Stephan Wesemeyer, Christopher J.P. Newton, Helen Treharne, Liqun Chen, Ralf Sasse, and Jorden Whitefield. Formal Analysis and Implementation of a TPM 2.0-Based Direct Anonymous Attestation Scheme. In: *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. New York, NY, USA, 2020, pp. 784–798.

[34] Jorden Whitefield, Liqun Chen, Ralf Sasse, Steve Schneider, Helen Treharne, and Stephan Wesemeyer. A Symbolic Analysis of ECC-Based Direct Anonymous Attestation. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2019, pp. 127–141.

A Details on ROBERT

Registration The central server is set up with a country code CC_S, a server key K_S, a federation key K_{fed} and a registration key pair (sk_S, pk_S).

After the user downloads the application, the application connects back to the server over TLS and is provisioned with several parameters, such as pk_S, the number of minutes in an epoch and other configuration data. The application then generates a public and private key pair: (sk_A, pk_A). The application sends pk_A to the server.

The client and server compute their shared secret K_{SA} from each other's public key and then derive encryption and authentication keys for the application: K_A^{Enc}, K_A^{Auth}.

Broadcast Immediately after registration and periodically thereafter, the application will connect to the central server in order to obtain a list of encrypted Bluetooth identifiers (EBIDs) and encrypted country code identifiers (ECCs), which the server computes from its key K_S and the federation key K_{fed}.

$$EBID_{A,i} = \{i, id_A\}_{K_S} \quad ECC_{A,i} = \{CC_S, EBID_{A,i}\}_{K_{fed}}$$

where i is the epoch number. As i has a bit length of 24 and id_A of 40 $EBID_{A,i}$ is produced by a block cipher with a block size of 64 bits and a key size of 192 bits. In contrast, the encryption of $ECC_{A,i}$ uses $EBID_{A,i}$ as the initialisation vector in AES output feedback mode, to link both together.[9]

The server encrypts this list using key K_A^{Enc} and the application periodically retrieves it.

In each epoch, the phone concatenates $(EBID_{A,i}, ECC_{A,i})$ with the lowest 16 bits of the current timestamp T. Consequently T resets roughly every 18 hours. This payload is then authenticated with a MAC:

$$M_{A,i} = ECC_{A,i}, EBID_{A,i}, T$$
$$MAC_{A,i} = HMAC_{K_A^{Auth}}(M_{A,i})$$

The resulting pair of messages and MAC are then periodically broadcast. Any phone in proximity parses the message and checks whether the timestamp is within a configurable tolerance of the current time ('typically a few seconds' [12]). Note that T does not prevent inter-day replays, since the counter periodically resets.

Authorisation of Uploads When the health authority diagnoses the owner of P as contagious, a contagious period is determined. The health authority then proposes the owner to upload all ephemerals that it collected from other phones during this time period to the server, along with the time that it received them. The specification explicitly leaves the authorisation procedure open, but requires that 'only authorised

[9]The input (CC_S) is actually smaller than one block, so this amounts to applying AES to $EBID_{A,i}$ (with some padding) and xoring the first 8 bits with CC_S.

and positively tested users are allowed to upload'. We thus investigated the source code of the back-end server, submission code server and the REST API specification.ROBERT has *long codes* which are encoded as QR codes and distributed via the SIDEP platform, a centralised system for registration of test results and are valid for 8 days. Patients that are not confirmed via a SIDEP-connected laboratory or diagnosed as a probable case by a health professional obtain *short codes* which are valid for only 60 minutes. The SIDEP platform allows producing long codes in batch, hence the health professional can obtain long codes with starting dates up to 10 days in the future (and aforementioned 8 days of validity).

The phone obtains such a code from a health authority as a QR code and attaches it to the ephemerals that it wants to upload. The server (1) validates the token and (2) parses the ephemerals. For each transmitted ephemeral, the server (3) checks that the reception time and the 16-bit timestamp in the ephemeral roughly match, (4) decrypts $ECC_{A,i}$ and compares it with its own country code, forwarding the message to the right server, if necessary. Then, (5) $EBID_{A,i}$ is decrypted and parsed to (i, id_A), (6) id_A checked to belong to a registered phone, and (7) i compared with the claimed reception time. With id_A, (8) the emitting phone's key is retrieved and (9) used to validate the MAC. Finally, i is added to the database LEE_{id_A} as an epoch where id_A was exposed (LEE stands for list of exposed epochs).

As the identity assigned to each phone is in principle opaque, the ROBERT specification suggests that additional actions, such as the deployment of a mixnet, could be taken to prevent the server from learning the uploader's identity through metadata such as IP addresses. In practice, no such system has been deployed.

Risk Calculation The risk of exposure is computed by the back end upon request by the phone. This request contains the current epoch i, time t_{req} and $EBID_{A,i}$, $ECC_{A,i}$. The EBID and ECC values are used to authenticate the origin of this request, which is performed exactly with the same steps (1) to (8) as above.

If the authenticity of the origin is established, the server returns '1' in case the risk score exceeds some threshold, and '0' otherwise. The risk score is derived from this phone's (id's) list of exposed epochs LEE_{id} and metadata like reported signal strength and duration,

Federation Interoperability between back ends is achieved using the shared federation key K_{fed}. Each back end—there is one per country code CC_S —forwards the recorded EBID and its metadata on a secure channel to the back end identified by the encrypted country code $ECC_{A,i}$. The receiving back end then processes the record as normal.

B Details on GAEN

Broadcast The GAEN framework divides time into consecutively numbered 10 minute epochs since the start of Unix time. When the framework is first activated on a device, a fresh Temporary Exposure Key (TEK) is generated and tagged with the current epoch index: TEK_d. After the end of each 24-hour period, a fresh TEK is generated for use in that period and any TEKs older than 14 days are deleted.

In a particular 24-hour period with TEK_d, two further keys are derived using HKDF:

$$RPIK_d = HKDF(TEK_d, \text{``}\texttt{ENRPIK}\text{''})$$
$$AEMK_d = HKDF(TEK_d, \text{``}\texttt{ENAEMK}\text{''})$$

The RPIK and AEMK are then used to derive a Rolling Proximity Identifier (RPI) for each epoch j and some encrypted associated metadata:

$$RPI_{d,j} = AES_{128}(RPIK_d, j)$$
$$AEM_{d,j} = AES_{128}(AEMK_d, j, \text{``}\texttt{Metadata}\text{''})$$

The metadata contains the version of the framework used by the transmitter and the estimated transmission power. The device then periodically broadcasts these two values and records any ephemerals it receives from nearby devices, alongside the reception time and relative transmission strength.

Risk Calculation The framework handles risk calculation on behalf of the contact tracing application. Indeed, the framework (in conjunction with the operating system) actively protects the stored keys and recorded ephemerals. However, the contact tracing application can pass a list of newly uploaded keys to the framework which will then generate the resulting identifiers and check whether it has observed any. This matching process has a two-hour tolerance to allow for clock skew on the sending or receiving device.

As a further security measure, newly provided keys must be signed by the operator of the national back end. The signature public key is certified by Google/Apple.

C Details on the CWA

QR Code Based Upload Authorisation This protocol is depicted in Fig. 3. When a test is conducted by a health professional, the user scans a QR code with their phone which contains a randomly chosen *guid*. The phone hashes this *guid* and sends it to a verification server (VS) to obtain a *registration token regToken*. The verification server stores the hashed *guid* and hashed registration token *regToken*. The phone then polls in a regular interval for updates on its owner's test results by sending its registration token to the VS.

The hashed *guid* is also attached to the test sample analysed by the laboratory. The VS can thus request updates on a test result by sending the hashed *guid* to a test result server (TRS).

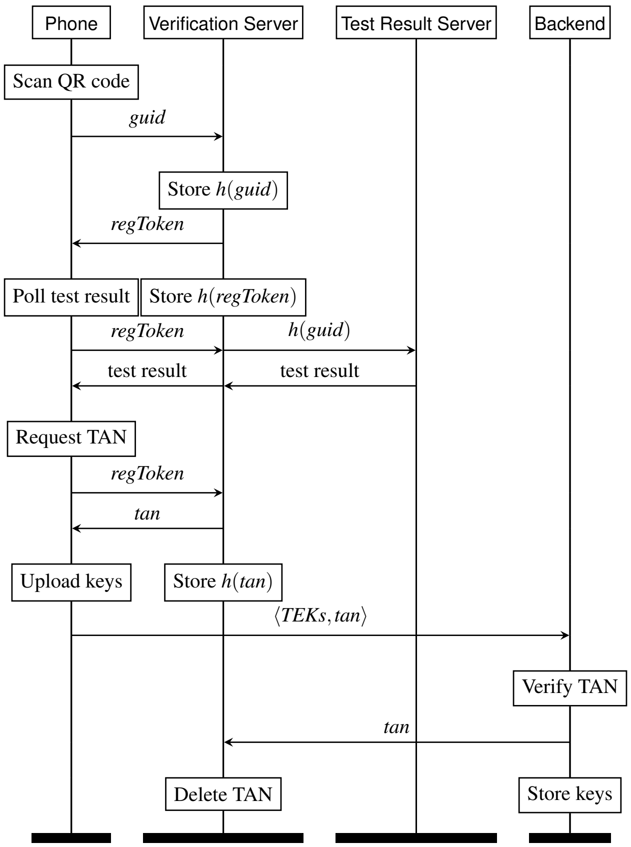

Figure 3: Authorisation procedure using a QR code for the CWA, where *guid*, *tan* and *regToken* are freshly generated.

The next time the phone polls, the verification server will forward the test result from the TRS.

If the test result is positive, the phone can request a TAN from the VS, which stores a hashed version of the TAN. The phone can then start the upload process by attaching the TAN to its TEKs. The back end will check whether the TAN is valid by contacting the VS in which case the hashed TAN is deleted from the VS.

Telephone Based Upload Authorisation As some testing facilities are not equipped to provide QR codes and some patients may fail to scan the QR code at the time, a backup authentication protocol is also offered. In this protocol, the user calls a medical hotline, is interviewed to ensure they recently had a positive test and receives a teleTAN directly over the phone which they enter into their application. The phone then contacts the verification server, provides the teleTAN and receives a registration code. The rest of the protocol proceeds as in the QR code variant.

European Federation Gateway Service In the GAEN framework, each country must provide its own back end and national infrastructure. This raises a question of how to handle travellers between countries, who are typically a high-risk category for COVID-19 exposure. The European Federation Gateway Service (EFGS) was developed by T-Systems to provide a solution across the EU. As detailed in [24], the EFGS architecture uses a central database server which every country's back end connects to via mutually authenticated TLS.

Users who are uploading their keys are expected to mark which countries they have been present during their contagious period. Their back end includes this information when it uploads their keys to the EFGS database. Periodically, the other back ends fetch all new keys from this database and build a list of contagious keys for each country in the federation. Users then periodically download all the keys corresponding to countries they have recently visited.

D Modelling Challenges

While the high-level description of the protocols is simple, we aimed for a detailed analysis that includes authentication procedures and federation, leading to a large model size (several thousand lines) and 12 person months of model development. Behind the models of the TPM [33], 5G [8, 26], DNP3 [13] and ISOIEC [34], the CWA model (1053 lines) appears to be the fifth-largest protocol model in Tamarin's model repository,[10] with the ROBERT and DP3T models being only slightly smaller (935 and 832 lines).

In contrast to traditional analyses, we developed a granular compromise analysis, which meant that certain simplifications were unavailable (e.g. combining the store of the health authority and the back end into one to avoid modelling their communication) and that Tamarin's analysis had to regard more edge cases. This, in turn, required us to write custom heuristics that prioritise resolving goals that help conclude from the occurrence of certain messages to which parties need to be compromised to produce this message.

We already described our evaluation methodology in Section 5.1, which we believe to be a novel approach to formal analysis, exploring the space of compromise scenarios rather than setting in a priori, which in this case was simply not possible, because they were not yet known. We exemplify the methodology below, but first discuss some modelling aspects.

D.1 Modelling Oracles

The oracles we developed are relatively simple in operation, but required manual inspection of the proof tree to be developed. The two main ideas are (a) prioritising constraints involving the adversary deducing a private key, as these typically rapidly lead to contradictions; and (b) deprioritise constraints involving the ordering of particular events, as those

[10] We count variants of the same protocol (e.g. there are two models for 5G [8, 26]) only once. Furthermore, there are models not contained in Tamarin's repository, e.g. TLS 1.3 [14].

Table 4: Adversarial Capabilities

Infrastructure	Capabilities	Events
Internet	Eavesdrop and inject messages	*built-in*
Bluetooth	Read and write message m at time t and location p	$\mathsf{BLErd}(t,p,m)$, $\mathsf{BLEwr}(t,p,m)$
Phone		$\mathsf{Corrupt}(\langle 'P', id\rangle, \star)$ with $\star =$
	Reveal long-term (ROBERT) or day key k (DP3T, CWA)	$\mathsf{CorruptPhoneKey}, k$
	Reveal witnessed ephemeral *eph*	$\mathsf{CorruptPhoneReceived}, eph$
	Send and receive message m to and from back end or HA	$\mathsf{CorruptPhone}(\mathsf{Send}\vert\mathsf{Receive}), m$
	Read and write testing state s (DP3T)	$\mathsf{CorruptPhoneTestDB}(\mathsf{Read}\vert\mathsf{Write}), s$
Back end		$\mathsf{Corrupt}(\langle 'B', cc\rangle, \star)$, with $\star =$
	Reveal long-term key k	$\mathsf{CorruptBState}, k$
	Receive message m from phone (ROBERT, DP3T)	$\mathsf{CorruptBReceive}, m$
	Send message m to VS (CWA) or phone (ROBERT)	$\mathsf{CorruptBSend}, m$
	Receive message m from VS (CWA)	$\mathsf{CorruptBReceiveFromVS}, m$
	Receive message m from phone (CWA)	$\mathsf{CorruptBReceiveFromPhone}, m$
	Reveal QR codes *qr* (ROBERT)	$\mathsf{CorruptQRList}, qr)$
	Reveal phone registration keys and identifer (ROBERT)	$\mathsf{CorruptBIDTable}, (k_1, k_2, id)$
	Reveal federation key k (ROBERT)	$\mathsf{CorruptBFederationKey}', k$
Verification server (VS)		$\mathsf{Corrupt}(\langle 'VS', cc\rangle, \star)$, with $\star =$
	Send and receive message m to and from back end (CWA)	$\mathsf{CorruptVS}(\mathsf{SendTo}\vert\mathsf{ReceiveFrom})\mathsf{TRSnB}, m$
	Send and receive message m to and from phone (CWA)	$\mathsf{CorruptVS}(\mathsf{SendTo}\vert\mathsf{ReceiveFrom})\mathsf{Phone}, m$
Health authority (HA)		$\mathsf{Corrupt}(\langle 'HA', cc\rangle, \star)$, with $\star =$
	Reveal long-term key k (DP3T)	$\mathsf{CorruptHAState}, k)$
	Send message m to phone (DP3T)	$\mathsf{CorruptHASend}, m)$

lead to additional cases and grow quadratically in the number of distinct time points, without progressing the analysis in terms of message deduction.

D.2 Modelling Limitations

ROBERT The ROBERT specification requires that the back end notifies a user that they are at risk only once. Our model reflects this requirement, although we are unsure if a practical system deployed at scale could successfully maintain this invariant.

DP3T In DP3T, we assume that uncompromised phones only generate a single TEK on a given day. In certain rare situations this may not be the case, but our analysis also covers compromised phones which may use multiple TEKs simultaneously. When phones upload their key material, we model this as multiple independent uploads rather than a single upload containing a list of keys. This simplifies the analysis but does not change the behaviour.

CWA We make similar assumptions as for the DP3T model. In the upload authorisation procedure, we model the verification server and the test result server as one entity, as in practice they are controlled by the same organisation. We also assume that an honest phone requests a TAN at most once. Dishonest phones may send multiple requests.