Modeling and analyzing the Corona-virus warning app with the Isabelle Infrastructure framework

Florian Kammüller and Bianca Lutz

Middlesex University London and
Technische Universität Berlin
f.kammueller@mdx.ac.uk|bialut@gmail.com

Abstract. We provide a model in the Isabelle Infrastructure framework of the recently published Corona-virus warning app. The app supports breaking infection chains by informing users whether they have been in close contact to an infected person. The app has a decentralized architecture that supports anonymity of users. We provide a formal model of the existing app with the Isabelle Infrastructure framework to show up some natural attacks in a very abstract model. We then use the security refinement process of the Isabelle Infrastructure framework to highlight how the use of continuously changing ephemeral ids improves the anonymity.

1 Introduction

The German Chancellor Angela Merkel has strongly supported the publication of the mobile phone Corona warning app by publicly proclaiming that the "Corona App deserves your trust" [1]. Many millions of mobile phone users in Germany have downloaded the app with 6 million on the first day. This app is one amongst many similar application that aim at the very important goal to "break infection chains" by providing timely information of users whether they have been exposed to close contact with a person that has been infected.

The Corona-virus warning app was a quite costly project but this was mainly due to the management of Telekom and SAP being in the driving seat. But the app has been designed with great attention on privacy: a distributed architecture [] has been adopted after a long and heated debate with supporters of a central architecture. The distributed architecture is based on a very clever distributed application design whereby users' phones are sending highly anonymized so called "Ephemeral IDs" at physical locations via the Bluetooth protocol. The app saves those IDs of people in close proximity. When at a later date an infected person reports his infection to a central server, the unique root ID is published and in the daily check all mobile phones connecting to the central server download the root IDs of infected people. Since the Ephemeral IDs can be mapped to the root ID all Ephemeral IDs that have been saved over the last 14 days allow users' phones to regularly check whether their user has been exposed to an

infected person and issue a warning to the user and recommendation to contact health authorities.

The Isabelle Infrastructure framework [10] allows modeling and analyzing architecture and scenarios including physical and logical entities, actors, and policies within the interactive theorem prover Isabelle supported with temporal logic, Kripke structures, and attack trees. It has been applied for example to Insider analysis in airplanes [11], privacy in IoT healthcare [3], and recently also to blockchain protocols [9].

The technical advantage of modeling an application in the Isabelle Insider framework lies in (a) having explicit representations of infrastructures, actors and policies in a formal model that (b) allows additional automated verification of security properties with CTL, Kripke structures and Attack Trees within the interactive theorem prover Isabelle.

The Corona-virus app has been produced based on a sophisticated security concept conceived by experts in the field, and has strong claims with regard to mathematical support:

"Such a design builds on strong, mathematically provable support for privacy and data protection goals [...]" **[TODO:[]**todoC][1]

However, there has been, as of yet, to our knowledge no formal verification been involved. Even if a "post-production" formal specification seems pointless, it allows to reveal weak points of the architecture, show that the measures that have been conceived are suitable to cover those weak points. Thereby, we believe that our current work is useful to increase the trust in the Corona-virus app necessary for its wide adoption which in turn is crucial for it to be efficient.

The contributions of this paper are (a) formal re-engineering of the Corona-virus warning app (b) providing an additional security and privacy analysis with interactive theorem proving certification of a novel view on the system architecture including actors, locations, and policies, (c) a formal definition of a security refinement process that allows to improve a system based on attacks found by the attack tree analysis and (d) an application of the refinement to improve security of the Corona-virus app specification.

In this paper, we first provide some background in Section 2: we give a detailed overview of the development history and relevant parts of the Corona-virus warning app (Section 2.1) We then introduce the Isabelle Infrastructure framework (Section 2.2). Next, we present our model (Section 3) and analysis of privacy and attacks (Section 4). The found attack on the first abstract specification motivates refinement. The formal definition of refinement for the Isabelle Infrastructure framework is introduced and illustrated on the Corona-virus warning app (Section 5) before drawing some conclusions (Section 7).

The formal model in the Isabelle insider framework is fully mechanized and proved in Isabelle (sources available [4]).

[1] **[TODO:[]**todoC]**BIB-REF**: DP-3T Whitepaper p. 2

2 Background and related work

2.1 DP-3T and PEPP-PT

This paper is mainly concerned with the architecture and protocols proposed by the Decentralized Privacy-Preserving Proximity Tracing project (DP-3T). The main reason to focus on this particular family of protocols is the *Exposure Notification Framework*, jointly published by Apple and Google, that adopts the main core principles of the DP-3T proposal. This API is not only used in the German Corona-virus warning app but has the potential of being widely adopted in future app developments that might emerge due to the reach of players like Apple and Google.

There are, however, competing architectures noteworthy, namely protocols developed under the roof of the *Pan-European Privacy-Preserving Proximity Tracing* project (PEPP-PT), e.g. PEPP-PT-ROBERT [**TODO:**[]todoC][2] and PEPP-PT-NTK[**TODO:**[]todoC][3].

Neither DP-3T nor PEPP-PT are synonymous for just one single protocol. Each project endorses different protocols with unique properties in terms of privacy and data protection. Yet, on a higher level of abstraction, it seems feasible to distinguish two basic architectures: protocols as endorsed by PEPP-PT might be characterized as centralized architectures whereas DP-3T-inspired protocols follow a (more) decentralized approach.[4]

Basic DP-3T protocol: How does it work? Upon installation, the app generates secret daily seeds to derive so called *Ephemeral IDs* (EphIDs) from them. EphIDs are generated locally with cryptographic methods and cannot be connected to one another but only reconstructed from the secret seed they were derived from.

During normal operation each client broadcasts its EphIDs via Bluetooth whilst scanning for EphIDs broadcasted by other devices in the vicinity. Collected EphIDs are stored locally along with associated meta-data such as signal attenuation and date. In DP-3T the contact information gathered is never shared but only evaluated locally (on the device).

If patients are tested positive on the Corona-virus, they are entitled to upload specific data to a central backend server. This data is accumulated by the backend server and redistributed to all clients regularly to provide the means for local risk scoring, i. e., determining whether collected EphIDs match those

[2] **[TODO:**[]todoC]**BIB-REF**: ROBERT github repository

[3] **[TODO:**[]todoC]**BIB-REF**: NTK ???; mentioned in "Response to Analysis of DP-3T"

[4] As we will see, DP-3T involves a central backend server. It is decentralized with regard to collection and evaluation of contact information: In centralized architectures the server provides a risk scoring services, whereas decentralized approaches rely on local risk assessment and, thus, do not need to share contact information with the backend.

broadcasted by now-confirmed Corona-virus patients during the last, e. g., 14, days.

In the most simple (and insecure) protocol proposed by DP-3T [TODO:[]todoC][5] this basically translates into publishing the daily seeds used to derive EphIDs from. Aside from additional security measures, like signing content the server provides as a general rule, the protocol implemented by ENF and, hence, CWA follows this low-cost design.[TODO:[]todoC][6] DP-3T proposes two other, more sophisticated protocols that improve privacy and data protection properties to different degrees but are more costly.

2.2 Isabelle Infrastructure framework

The Isabelle Infrastructure is built in the interactive generic theorem prover Isabelle/HOL [12]. As a framework, it supports formalization and proof of systems with actors and policies. It originally emerged from verification of insider threat scenarios but it soon became clear that the theoretical concepts, like temporal logic combined with Kripke structures and a generic notion of state transitions were very suitable to be combined with attack trees into a formal security engineering process [2] and framework [6].

Overview Figure 1 gives an overview of the Isabelle Infrastructure framework with its layers of object-logics – each level below embeds the one above showing the novel contribution of this paper in blue on the top.

Instantiation of Framework The formal model of the Corona-warning app uses the Isabelle Infrastructure framework instantiating it by reusing its concept of *actors* for users and smartphones whereby locations correspond to physical locations. The Ephemeral IDs, their sending and change is added to Infrastructures by slightly adapting the basic state type of infrastructure graphs and accordingly the semantic rules for the actions move, get, and put. The details of the newly adapted Infrastructure are presented in Section 3. Technically, an Isabelle theory file `Infrastructure.thy` builds on top of the theories for Kripke structures and CTL (`MC.thy`), attack trees (`AT.thy`), and security refinement (`Refinement.thy`). Thus all these concepts can be used to specify the formal model for the Corona-virus warning app, express relevant and interesting properties, and conduct interactive proofs (with the full support of the powerful and highly automated proof support of Isabelle). All Isabelle sources are available online [7].

[5] [TODO:[]todoC]**BIB-REF**: Low-cost decentralized proximity tracing; DP-3T Whitepaper p. 14ff

[6] [TODO:[]todoC]**BIB-REF**: CWA solution architecture (→ github:CWA); probably ENF specification

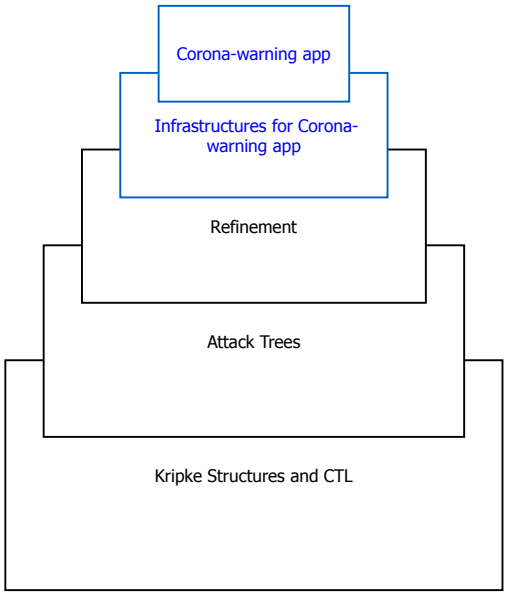

Fig. 1. Generic Isabelle Infrastructure framework applied to Corona-warning app.

Refinement An additional feature that has been integrated into the Isabelle Infrastructure framework motivated by security engineering formal specifications for IoT healthcare system is an extension of the formal specification process introducing refinement of Kripke structures [6,10]. It refines a system model based on a formal definition of a combination of trace refinement and structural refinement (or datatype refinement). The definition allows to prove property preservation results crucial for an iterative development process. The refinements of the system specification can be interleaved with attack analysis while security properties can be proved in Isabelle. In each iteration security qualities are accumulated while continuously attack trees scrutinize the design. One of the contributions of this paper is to explore different concepts of refinement: the formal expression of refinement, enables to pin down (i.e. exemplify) different concepts of refinement (data refinement, action refinement, trace refinement (aka spec refinement) and combinations thereof with concrete attack scenarios.

3 Model

4 Analysis

5 Refinement

Intuitively, a refinement changes some aspect of the type of the state, for example, replaces a data type by a richer datatype or restricts the behaviour of the

actors. The former is expressed directly by a mapping of datatypes, the latter is incorporated into the state transition relation of the Kripke structure that corresponds to the transformed model. In other words, we can encode a refinement within our framework as a relation on Kripke structures that is parameterized additionally by a function that maps the refined type to the abstract type. The direction "from refined to abstract" of this type mapping may seem curiously counter-intuitive. However, the actual refinement is given by the refinement that uses this function as an input. The refinement then refines an abstract to a more concrete system specification. The additional layer for the refinement can be formalized in Isabelle as a refinement relation $\sqsubseteq_{\mathcal{E}}$. The relation `mod_trans` is typed as a relation over triples – a function from a threefold Cartesian product to `bool`, the type containing true and false only. The type variables σ and σ' input to the type constructor `Kripke` represent the abstract state type and the concrete state type. Consequently, the middle element of the triples selected by the relation `mod_trans` is a function of type $\sigma' \Rightarrow \sigma$ mapping elements of the refined state to the abstract state. The expression in quotation marks after the type is again the infix syntax in Isabelle that allows the definition of mathematical notation instead of writing `mod_trans` in prefix manner. This nicer infix syntax is already used in the actual definition. Finally, the arrow $\Longrightarrow$ is the implication of Isabelle's meta-logic while $\longrightarrow$ is the one of the *object* logic HOL. They are logically equivalent but of different types: within a HOL formula P, for example, as below $\forall x.P \longrightarrow Q$, only the implication $\longrightarrow$ can be used.

```
mod_trans ::  (σ Kripke × (σ' ⇒ σ) × σ' Kripke)
              ⇒ bool                    ("_ ⊑(-) _")
 K ⊑ℰ K' ≡ ∀ s' ∈ states K'. ∀ s ∈ init K'.
           s →*σ' s' ⟶ ℰ(s) ∈ init K
           ∧ ℰ(s) →*σ ℰ(s')
```

The definition of `K` $\sqsubseteq_{\mathcal{E}}$ `K'` states that for any state s of the refined Kripke structure that can be reached by the state transition in zero or more steps from an initial state s_0 of the refined Kripke structure, the mapping $\mathcal{E}$ from the refined to the abstract model's state must preserve this reachability, i.e., the image of s_0 must also be an initial state and from there the image of s under $\mathcal{E}$ must be reached with 0 or n steps.

5.1 Property Preserving System Refinement

A first direct consequence of this definition is the following lemma where the operator $\triangleleft$ in $\mathcal{E}\triangleleft$`(init K')` represents function image, that is the set, $\{\mathcal{E}(x).x \in$ `init K'`$\}$.

lemma `init_ref: K` $\sqsubseteq_{\mathcal{E}}$ `K'` $\Longrightarrow \mathcal{E}\triangleleft$`(init K')` $\subseteq$ `init K`

A more prominent consequence of the definition of refinement is that of property preservation. Here, we show that refinement preserves the CTL property of $\mathsf{EF}s$ which means that a reachability property true in the refined model `K'` is already true in the abstract model. A state set s' represents a property in the predicate

transformer view of properties as sets of states. The additional condition on initial states ensures that we cannot "forget" them.

theorem `prop_pres`:
$$\texttt{K} \sqsubseteq_{\mathcal{E}} \texttt{K'} \implies \texttt{init K} \subseteq \mathcal{E}\triangleleft\texttt{(init K')} \implies \forall\ \texttt{s'} \in \texttt{Pow(states K')}.\ \texttt{K'} \vdash \mathsf{EF}\ \texttt{s'} \longrightarrow \texttt{K} \vdash \mathsf{EF}\ (\mathcal{E}\triangleleft\texttt{(s')})$$

It is remarkable, that our definition of refinement by Kripke structure refinement entails property preservation and makes it possible to prove this as a theorem in Isabelle once for all, i.e., as a meta-theorem. However, this is due to the fact that our generic definition of state transition allows to explicitly formalize such sophisticated concepts like reachability. For practical purposes, however, the proof obligation of showing that a specific refinement is in fact a refinement is rather complex justly because of the explicit use of the transitive closure of the state transition relation. In most cases, the refinement will be simpler. Therefore, we offer additional help by the following theorem that uses a stronger characterization of Kripke structure refinement and shows that our refinement follows from this.

theorem `strong_mt`:
$$\mathcal{E}\triangleleft\texttt{(init K')} \subseteq \texttt{init K} \wedge \texttt{s} \rightarrow_{\sigma'} \texttt{s'} \longrightarrow \mathcal{E}\texttt{(s)} \rightarrow_{\sigma} \mathcal{E}\texttt{(s')} \implies \texttt{K} \sqsubseteq_{\mathcal{E}} \texttt{K'}$$

This simpler characterization is in fact a stronger one: we could have $s \rightarrow_{\sigma'} s'$ in the refined Kripke structure `K'` and $\neg(\mathcal{E}(s) \rightarrow_{\sigma} \mathcal{E}(s'))$ but neither s nor s' are reachable from initial states in `K'`. For cases, where we want to have the simpler one-step proviso but still need reachability we provide a slightly weaker version of `strong_mt`.

theorem `strong_mt'`:
$$\mathcal{E}\triangleleft\texttt{(init K')} \subseteq \texttt{init K} \wedge (\exists\ \texttt{s0} \in \texttt{init K'}.\ \texttt{s0} \rightarrow^{*} \texttt{s}) \wedge \texttt{s} \rightarrow_{\sigma'} \texttt{s'} \longrightarrow \mathcal{E}\texttt{(s)} \rightarrow_{\sigma} \mathcal{E}\texttt{(s')} \implies \texttt{K} \sqsubseteq_{\mathcal{E}} \texttt{K'}$$

This idea of property preservation coincides with the classical idea of trace refinement as it is given in process algebras like CSP. In this view, the properties of a system are given by the set of its traces. Now, a refinement of the system is given by another system that has a subset of the traces of the former one. Although the principal idea is similar, we greatly extend it since our notion additionally incorporates refinement. Since we include a state map $\sigma' \Rightarrow \sigma$ in our refinement map, we additionally allow structural refinement: the state map generalizes the basic idea of trace refinement by traces corresponding to each other but allows additionally an exchange of data types. As we see in the application to the case study, the refinement steps may sometimes just specialize the traces: in this case the state map $\sigma' \Rightarrow \sigma$ is just identity.

In addition, as we can observe in the following example in the second refinement step, we also have a simple implicit version of *action refinement*. In an action refinement, traces may be refined by combining consecutive system events into atomic events thereby reducing traces. This will be detailed on the example below.

6 Refining the Specification

7 Conclusions

7.1 Protection goals of attacks

7.2 Summary and discussion of relevance of the approach

We can establish in our formal framework an attack that even a system using changing Ephemeral ids can be broken if the attacker physically follows a victim. This is a basic attack on anonymity: a user's connection between his Ephemeral Ids and personal details (Iphone MAC or name) is revealed to the attacker. The protection goal of privacy is thereby destroyed.

When establishing the attack we start from a simplistic scenario that does not use Ephemeral Ids but fixed ids. In this (over)simplified model the attack is established. We then define a formal Refinement calculus for the Isabelle Infrastructure framework to refine the attack to a system with Ephemeral Ids that change in fixed time intervals obfuscating the relationship between user and his pseudonym[7].

Now, the refinement shows that although the Ephemeral Ids change regularly the same attack that has been identified on the very abstract level (fixed ids) persists. The refinement allows refining the datatype (Id $\mapsto$ EphId) but also delivers the usual trace refinement (behaviours of the refined systems are a subset of the traces of the abstract system). This persistence of the attack precisely shows which part of the system behaviour is responsible for the attack. In a second refinement step using action refinement based on this insight from the (repeated) attack, we can exclude the dangerous attack trace[8].

Clearly, the abstract attack we establish is obvious on an informal level but the persistence of the attack on the refined system is less obvious. The remedy by a second refinement step is an evident restriction of the system behaviour which gives a clear specification of a system secured against this attack. The use of formal methods therefore lies not in the discovery of an obvious attack on a simplified system but in showing how a formal specification including security refinement can lead to a stepwise improvement that is accompanied by formal proof in the Isabelle Infrastructure framework. The solution to exclude the attack in the second refinement step binds the action move together with a put action. This shows that besides datatype refinement and trace refinement our Refinement calculus also entails action refinement. This action refinement is implemented implicitly by changing the effects of the actions in the semantic state transition relation. In future work, we could think about making the action refinement more explicit by considering a relationship between semantic

[7] We identify the smartphone and the user which might be also recognized by his appearance (face)

[8] Implication on the actual protection goals that are endangered by the attack on privacy are discussed in the larger context of the Corona-warning app (see previous Section)

rules or by refining the refinement notion to a more explicit layer of protocol steps – similar to what has been done in previous applications of the Isabelle Infrastructure framework for example to Auction protocols [8] or the Quantum Key Distribution [5].

The security refinement in general might seem pointless, as in the first step of refinement the attack persists and even though the second refinement gets rid of this specific attack, it doesn't exclude the reachability of the attack goal altogether (if the attacker Eve gets Bob on his own in a location she can map all used EphIDs to him). However, the refinement makes the system relatively more secure in that for a larger number of traces the abstract attack does not work anymore[9]. It is important to emphasize that security refinement is a cyclic process that improves security but does not usually terminate (like a loop) with a fixed point of 100% secure system. In a refined model new detail may give rise to new attack possibilities. These additional attacks can be identified using the Attack Tree calculus and trigger further refinement steps.

References

1. D. Bundesregierung. Die corona-warn-app: Unterstützt uns im kampf gegen corona, 2020. German government announcement and support of Coronavirus warning app.
2. CHIST-ERA. Success: Secure accessibility for the internet of things, 2016. http://www.chistera.eu/projects/success.
3. F. Kammüller. Attack trees in isabelle. In *20th International Conference on Information and Communications Security, ICICS2018*, volume 11149 of *LNCS*. Springer, 2018.
4. F. Kammüller. Isabelle infrastructure framework with iot healthcare s&p application, 2018. Available at `https://github.com/flokam/IsabelleAT`.
5. F. Kammüller. Attack trees in isabelle extended with probabilities for quantum cryptography. *Computer & Security*, 87, 2019.
6. F. Kammüller. Combining secure system design with risk assessment for iot healthcare systems. In *Workshop on Security, Privacy, and Trust in the IoT, SPTIoT'19, colocated with IEEE PerCom*. IEEE, 2019.
7. F. Kammüller. Isabelle infrastructure framework for ibc, 2020. Isabelle sources for IBC formalisation.
8. F. Kammüller, M. Kerber, and C. Probst. Insider threats for auctions: Formal modeling, proof, and certified code. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 8(1):44–78, 2017. Special Issue on Insider Threat Solutions - Moving from Concept to Reality.
9. F. Kammüller and U. Nestmann. Inter-blockchain protocols with the isabelle infrastructure framework. In *Formal Methods for Blockchain, 2nd Int. Workshop, colocated with CAV'20*, Open Access series in Informatics. Dagstuhl publishing, 2020. To appear.
10. F. Kammüller. A formal development cycle for security engineering in isabelle, 2020.
11. F. Kammüller and M. Kerber. Applying the isabelle insider framework to airplane security, 2020.

[9] Adding probabilities as in [5] enables quantifying this.

12. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002.