Proceedings of the
2nd Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY),
September 25, 2020

On the Formalization of Decentralized Contact Tracing Protocols

P. A. Abdulla[1], M. F. Atig[2], G. Delzanno[3], M. Montali[4], and A. Sangnier[5]

[1]Uppsala University
[2]Uppsala University
[3]DIBRIS, University of Genova
[4]Free University of Bozen
[5]IRIF, Universitè Paris Denis Diderot

Abstract

We present a preliminary formalization based on transition systems of decentralized contact tracing protocols for smart devices equipped with Bluetooth trasmitters. In our model the behaviour of individual users, via their app, is modelled as a timed automata with a local unbounded memory. Protocol configurations consist of the current state of a shared server and a finite set of local states containing the states of individual users. The transition system models the interaction between devices in the same physical location and between a sigle device and the shared server. In the paper we address different research directions concerning semi-automated verification based on automated reasoning tools of the considered class of protocols, theoretical issues related to the expressiveness of the resulting class of formal models, and data-driven analysis of the logs collected on the server as well as on user devices.

1 Introduction

The goal of contact tracing is to make people aware of possible contacts with positively diagnosed people so that they should possibly have an infection test [16]. For pandemic diseases, reporting a positive diagnosis is mandatory. Contact tracing methods, e.g., smartphone apps, are aimed at investigating who could have been contaminated by a positively diagnosed person and alert them. After the COVID-19 pandemic, several protocols have been proposed as an automated support for this task. The Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) proposed both centralized and decentralized solutions. As a decentralized solution, they propose the Decentralized Privacy-Preserving Proximity Tracing (DP3T), and the specifications of ROBERT4 and NTK5. Both centralized and non-centralized systems require a central server for alerts and typically differ in the way ephemeral identifiers are generated. In general the system architecture is based on a smart app, a notification server as in Publish/Subscribe architectures, and a possibly trusted server. An important notion here is that of ephemeral identifiers, i.e., identifiers that are frequently changed either by a trusted authority/server or by the app itself in order to reduce the risk of a malicious use of private user data from third parties (e.g. the Paparazzi's attack [3, 16]). Ephemeral identifiers are kept locally in the smart app. In centralized versions of the protocol, ephemeral

identifiers are downloaded from a trusted server. In decentralized versions, ephemeral identifiers are created by the app. The smart app installed on the user device, e.g. the Immuni app in Italy, makes use of the Bluetooth interface to broadcast the currently valid ephemeral identifier. The communication range of the Bluetooth interface is tuned so as to reach the distance for a possible contagious contact. The app collects all identifiers received by other apps located in the same physical place and stores them locally. Identifiers whose temporal marks fall out of the incubation period are deleted from memory. When users are diagnosed to be infected, they can release a report to the shared server. The server is used to make the report (a log containing ephemeral identifiers) available to each user interested in checking potential contacts with infected users. This step can be implemented in several different ways depending on the system architecture and on the role of the authority in the whole process. One possibility is to send the report to a central log database. Users can then consult the database to check if they crossed their way with infected users, i.e., they share some ephemeral identifier with one of them.

Taking inspiration from [1, 10, 11, 12, 9, 8], in this extended abstract we present a preliminary formalisation of decentralized contact tracing protocols based on transition systems, a mathematical tool typically used in abstract semantics of concurrent and distributed systems. In our specification, users apps are modelled as timed automata with a local memory. System configurations consist then of the current shared server state (a sort of global memory) and a multiset of local states of user instances. In the first part of the paper we describe a mathematical presentation of contact tracing protocols aimed at introducing a formal language to describe protocol properties and possible verification procedure for the resulting models. In the last part we highlight possible future directions concerning both semi-automated verification and more theoretical issues.

2 Contact Tracing Systems

In this section we introduce a possible formalization of a decentralized contact tracing protocol. To specify the protocol rules and behavior (computation), we will use a transition system defined on configurations consisting of a set of states of individual users. User configurations contain a local memory needed to store the list of broadcasted and collected identifiers. System configurations contain a global memory that gathers user logs and a set of user states that corresponds to the current number of registered indivuals. To simplify the model, we fix the number of agents and simulate dynamic injection by randomly selecting the initial value of local clocks used to define validity of ephemeral identifiers (i.e. local clocks may have different values).

We will use the standard notation $\cup, \cap, \setminus, \subseteq, \subset$, and $\in$ to indicate set operations. Furthermore, we will use $\mathcal{P}(D)$ to indicate the powerset of D. For simplicity, for an element $e \in D$ and a set M, we will use $e \cup M$ as an abbreviation for $\{e\} \cup M$. Finally, we will use $\langle e_1, \ldots, e_n \rangle$ to denote tuples of elements in D. In the rest of the paper we assume that users are uniquely identified via a denumerable set U of identifiers (user identifiers assigned upon registration). Similarly, we assume that ephemeral identifiers are represented via a denumerable set E of labels. Both sets come equipped with standard comparison operators such as $=$ and $\neq$. Furthermore, we introduce a timestamp domain T needed to identify validity periods of ephemeral identifiers. For simplicity, we consider days as time units and model time elapsing transitions according to them. Elements in T are totally ordered and can be compared with equality $=$ and with the less than ordering $<$. As a convention, we add a special value $\perp$ (undefined time instant).

We fix a parameter $\delta > 0$ for defining validity periods of ephemeral identifiers. Namely, ephemeral identifiers are considered to be valid if and only if they have been generated in the period $[ct - \delta, ct]$, where ct is the current time. The parameter δ represents the estimated incubation period of the virus. Identifiers with time stamp $ct - \delta$ need not to be considered in the query step. We will also fix a second parameter γ with $0 < \gamma < \delta$ to denote the frequency for rotating ephemeral in the user app. Rotation is needed for security reasons, e.g., to reduce the risk of data leakage (visited locations, contact details). During the validity time of an ephemeral identifier, the same user can broadcast several different identifiers (δ/γ fresh identifiers). All these identifiers must be maintained in memory to be used when crosschecking them with identifiers of other users.

2.1 Transition System

Contact tracing systems are designed so as to be equipped with a shared (centralized or distributed) server that is used to propagate reports to all registered users (a sort of notification system as in Publish/Subscribe architectures). The server can be viewed as a (distributed) register in which reports of positive users are first collected and then made publicly available to other users for inquiries about their health status. To abstract away from possible implementations, we model the shared server as a global memory defined via the map $G : U \times E \to T$ such that $G(u, e)$ denotes the last time user u generated or received (depending on the type of adopted protocol) the ephemeral identifier e. We assume here that identifiers are unique and that they are generated in a given day (a timestamp included in the report).

Users (or, better, user apps) can be modelled as individual processes in a concurrent system. Processes interact by sending and receiving messages (identifiers) only when they are in the same physical location. For the sake of this presentation, we abstract away from physical locations and model physical contact using non-determinism, i.e., at each step we non-deterministically select the set of users that receive the identifier broadcasted by a given app. In a sense, a receiver group can be viewed as an abstract representation of a physical location. Users need to maintain in local memory the current ephemeral identifiers, updated every γ time units, as well as the sets of broadcasted and received identifiers. Based on all the above mentioned assumptions, user states are tuples of the following form: $\langle q, u, e, c, S, R\rangle$ where

- $q \in \{0, 1, 2\}$ is the user status: negative, possibly positive, and positive, respectively;
- $u \in U$ is the user id;
- $e \in E$ is the current valid ephemeral identifier that the user can broadcast to his/her neighbors;
- $c \in T$ is the current value of the timer used to update the identifier;
- $S : E \to T$ and $R : E \to T$ are mappings that represent the local memory, namely the set of sent and received identifiers by user U. Timestamps are used here for checking their validity.
  - If $S(e) > 0$, than $S(e)$ maintains the last time user u sent e;
  - if $R(e) > 0$, than $R(e)$ maintains the last time user u received e.

We will use C to denote the set of all possible user states defined on U and E, and $\mathcal{P}(C)$ the powerset of C. The global state of a contact tracing system can be defined as a tuple $\langle G, \Delta, t\rangle$, where G is the server state, $\Delta \in \mathcal{P}(C)$ is a set of user states, each one carrying a unique user identifier, and $t \in T$ is the current (global) time. We assume here that in the initial global state $t = 0$, user states contain distinct ephemeral identifiers, and local timers are initialized with random values to simulate injections of users at different times, and the local mappings S and R of user states are such that $S(x) = R(x) = \perp$ for all $x \in E$.

We now define a transition system to describe the operational semantics of contact tracing protocols. In this preliminary study, we abstract away from cryptographic operations and focus on data management steps. To simplify the definitions, for $\Delta \in \mathcal{P}(C)$, we will use $\Delta \Leftarrow x, t$ to denote the set obtained by updating every user state $\sigma = \langle q, u, e, c, S, R\rangle \in \Delta$ into the new user state $\sigma' = \langle q, u, e, c, S, R'\rangle$ s.t. $R'(x) = t$ and $R'(y) = R(y)$ for every $y \neq x$. Similarly, for $\sigma = \langle q, u, e, c, S, R\rangle \in C$ we will use $\sigma \Rightarrow x, t$ to denote the new user state $\langle q, u, e, c, S', R\rangle$ s.t. $S'(x) = t$ and $S'(y) = S(y)$ for $y \neq x$.

Let us first consider the phase in which a device broadcasts ephemeral identifiers to other devices in the same location. Let $\Delta \in \mathcal{P}(C)$ be the current set of user states. We non-deterministically single out the sender user $\sigma \in C$ and a group of receivers $\Pi \in \mathcal{P}(C)$, containing only users in state $q < 2$, by requiring that $\Delta = \sigma \cup \Pi \cup \Gamma$ for $\sigma \in \Delta$, $\Pi, \Gamma \in \mathcal{P}(C)$. The effect of the transitions is to update the state of sender and receivers:

$$\langle G, \sigma \cup \Pi \cup \Gamma, t\rangle \to \langle G, \sigma' \cup \Pi' \cup \Gamma, t\rangle$$

where $\sigma = \langle 0, u, e, c, S, R\rangle$, Π contains only user states with local state $q < 2$, $\sigma' = (\sigma \Rightarrow e, t)$, and $\Pi' = (\Pi \Leftarrow e, t)$.

Global and local memory can be reset from time to time as specified in the following three rules.

$$\langle G, \Delta, t\rangle \to \langle G', \Delta, t\rangle$$

where, for every $u \in U, e \in E$, G' is obtained by setting $G'(u,e) = \perp$ if $G(u,e) < t-\delta$, and $G'(u,e) = G(u,e)$ otherwise.

$$\langle G, \sigma \cup \Delta, t\rangle \to \langle G, \sigma' \cup \Delta, t\rangle$$

where $\sigma = \langle q, u, e, c, S, R\rangle$ and $\sigma' = \langle q, u, e, c, S', R'\rangle$ s.t., for every $e \in E$, $S'(e) = \perp$ if $S(e) < t - \delta$, and $S'(e) = S(e)$ otherwise, and $R'(e) = \perp$ if $R(e) < t - \delta$, and $R'(e) = R(e)$ otherwise.

We consider now the report phase in a protocol formulation in which users who are diagnosed as positive submit as a report a copy of the log of all broadcasted identifiers i.e. they copy S in the global memory G. Other formulations are possible, e.g., the report may consist of the mapping R instead of S.

$$\langle G, \sigma \cup \Delta, t\rangle \to \langle G', \sigma' \cup \Delta, t\rangle$$

where $\sigma = \langle 0, u, e, c, S, R\rangle$, $\sigma' = \langle 2, u, e, c, S, R\rangle$ and, $G'(u,x) = S(x)$ for every $x \in E$, and for every $v \neq u$ and $y \in E$, $G'(v,y) = G(v,y)$. We assume here that users in state 2 need to register to re-enter, i.e., return to state 0, into the system.

Let us now consider the query phase. In this phase users crosscheck the set of received identifiers with the global memory. If valid ephemeral identifiers are contained in the intersection with a time stamp compatible with the incubation period, than the user updates its state to 1.

$$\langle G, \sigma \cup \Delta, t\rangle \to \langle G, \sigma' \cup \Delta, t\rangle$$

where $\sigma = \langle 0, u, e, c, S, R\rangle$, there exists v, x s.t. $G(v,x) = R(x) = s$, $t - \delta \leq s \leq t$ and $\sigma' = \langle 1, u, e, c, S, R\rangle$. We assume here that users in state 1 need to register again into the system to return to state 0.

The final part of the specification is related to time elapsing steps and ephemeral identifiers update. We model this step by incrementing global time, local clocks, and controlling whether or not the validity time of identifiers expired.

$$\langle G, \Delta, t\rangle \to \langle G, \Delta', t+1\rangle$$

where Δ' is obtained from Δ as follows. Assume that $\Delta = \{\sigma_1, \ldots, \sigma_k\}$. Furthermore, let $F = \{x_1, x_2, \ldots, x_k\}$ be a set of fresh identifiers, i.e., such that they do not occur in Δ. We define $\Delta' = \{\sigma'_1, \ldots, \sigma'_k\}$ in such a way that for each $i : 1, \ldots, k$ if $\sigma_i = \langle q, u, e, c, S, R\rangle$, then

- if $q = 0$ then
  - if $c = \gamma - 1$, then $\sigma'_i = \langle q, u, x_i, 0, S, R\rangle$, i.e., we update the current ephemeral identifier with a fresh value;
  - otherwise if $c < \gamma - 1$, then $\sigma'_i = \langle q, u, e, c+1, S, R\rangle$, i.e., we update the local counter.
- if $q > 0$, then $\sigma'_i = \langle q, u, e, c, S, R\rangle$, i.e., we leave the user state unchanged.

A computation is a sequence of configurations $\sigma_0\sigma_1 \ldots$ that, starting from an initial configuration σ_0 with a certain number of users, contains configurations such that $\sigma_i \to \sigma_{i+1}$ for $i \geq 0$. The above described formal semantics can be extended in order to model different ways to handle fresh identifiers creation (via a shared server, etc) as well as to model explicitly physical locations and user movement from location to location. The proposed transitions system however can already by used as an abstract representation of a decentralized version of the protocol.

As an example, consider a configurations $\sigma_0 = \langle G, \Delta, 0\rangle$ in which Δ consists of three user states a, b, and c in their initial state (local state equal to 0, empty memory, distinct ephemeral identifiers i_a, i_b, and i_c respectively). We can then apply a broadcast step to a sender a and a non-deterministically selected subset of Δ, e.g. $\{b, c\}$. User a will update S_a s.t. $S_a(i_a) = 0$ whereas b, c will update their R memories s.t. $R_b(i_a) = R_c(i_a) = 0$. Starting from the new configuration σ_1, we can apply k time elapsing steps and reach configuration σ_2 with clock k. We can then apply another broadcast step to sender b and receiver a so that $S_b(i_b) = R_a(i_b) = k$. We can then apply a report step moving a to state 2 and copying its memory S to G, thus having $G(a, i_a) = 0$. In the query step user b will move to state 1, i.e., receives a notification by the server, only if $k \leq \delta$.

3 Verification and Data-driven Analysis

The formalization of contact-tracing protocols has several interesting applications. First of all, we plan to apply Automated Reasoning techniques based on SMT solvers in order to validate a parameterized formulations of decentralized versions of the protocol in the input language of Infinite-state Model Checkers such as Cubicle as discussed in previous works on distributed algorithhms [6], and Publish/Subscribe architectures [8]. In this setting it is possibile to exploit the combination of the Theory of Arrays and Arithmetics to finitely express and then validate automatically such a protocol for an arbitrary number of users. In this context the idea is to represent the state of individual users as cells of an unbounded arrays possibly containing a subfields unbounded arrays representing their local memory. Indeed, in Cubicle it is possible to express transition systems operating on multidimensional unbounded arrays. In the considered case study, the set of user states can be represented as unbounded tables indexed on process identifiers and, for each process, the local memory can be represented again as an unbounded table e.g. containing the list of sent or received identifiers. In tools such as Cubicle, and more in general in several other examples of infinite-state model checkers, symbolic search procedures can be applied in order to validate the protocol specification againts safety and restricted types of liveness properties. As an example of correcteness property, let us consider the following correspondence property (typical of cryptograohic protocols): if user u is in state 1, then there exists a positive user v that emitted an identifier in state t received by u at time t' s.t. $t - t' < \delta$ or sim. This property can be formulated as a coverability problem

Integrity properties can also be modelled by modeling malicious agents and equipping them with the possibility of sniffing data sent by the users, e.g., identifiers, reports, user credentials in order to obtain information on user position and contacts etc.

Concerning computability issues, it seems possible to define further abstractions or restricted versions of the proposed formalization in order to exploit decidability and complexity results for ad hoc networks (they model receivers groups via broadcast messages) [10, 11, 1], and asynchronous broadcast networks (they model local memories) [13]. Furthermore, it would be interesting to consider abstractions based on Population Protocols [14] a class of concurrent systems that could be used to model a pandemic behaviour in a population. Another interesting connection arises from the fact that this class of protocols appears to naturally enjoy various boundedness notions that have been studied towards the verification of data-aware business processes [5]. On the one hand, ephemeral identifiers are subject to expiration, and consequently are akin to fading fluents in [7], and go well with the notion of recency bound in [2]. On the other hand, while a user can in principle encounter unboundedly many other users over the whole timeline, it is reasonable to assume that only boundedly many will be met in a fixed timespan, a property resembling that of bound-by-resources in [15].

Data-driven analysis of the user and server logs are also an interesting research direction more related to more classical Artifical Intelligence techniques. In particular data mining and prediction techniques could be applied here in order to infer potential clusters and links between them as well as to extract prediction models for the diffusion of the virus as proposed by Yoshua Bengio in [4].

References

[1] P. A. Abdulla, G. D., O. Rezine, A. Sangnier, and R. Traverso. On the verification of timed ad hoc networks. In *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*, volume 6919 of *Lecture Notes in Computer Science*, pages 256–270. Springer, 2011.

[2] Parosh Aziz Abdulla, C. Aiswarya, Mohamed Faouzi Atig, Marco Montali, and Othmane Rezine. Recency-bounded verification of dynamic database-driven systems. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 195–210. ACM, 2016.

[3] G. Avitabile, V. Botta, V. Iovino, and I. Visconti. Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system. *IACR Cryptol. ePrint Arch.*, 2020:493, 2020.

[4] Y. Bengio. Peer-to-peer AI-tracing of COVID-19. https://yoshuabengio.org/2020/03/23/peer-to-peer-ai-tracing-of-covid-19.

[5] Diego Calvanese, Giuseppe De Giacomo, and Marco Montali. Foundations of data-aware process analysis: a database theory perspective. In Richard Hull and Wenfei Fan, editors, *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 1–12. ACM, 2013.

[6] S. Conchon, G. Delzanno, and A. Ferrando. Declarative parameterized verification of distributed protocols via the cubicle model checker. *Accepted for publication in Fund. Inf.*, 2020.

[7] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded situation calculus action theories. *Artif. Intell.*, 237:172–203, 2016.

[8] G. Delzanno. Towards the automated verification of publish/subscribe networks. In *Proceedings of the 1st Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, co-located with the 18th International Conference of the Italian Association for Artificial Intelligence, OVERLAY@AI*IA 2019, Rende, Italy, November 19-20, 2019*, volume 2509 of *CEUR Workshop Proceedings*, pages 35–40, 2019.

[9] G. Delzanno, A. Sangnier, and R. Traverso. Parameterized verification of broadcast networks of register automata. In *RP*, pages 109–121, 2013.

[10] G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, pages 313–327, 2010.

[11] G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 441–455, 2011.

[12] G. Delzanno, A. Sangnier, and G. Zavattaro. Verification of ad hoc networks with node and communication failures. In *FORTE/FMOODS'12*, volume 7273 of *LNCS*, pages 235–250. Springer, 2012.

[13] G. Delzanno and R. Traverso. Decidability and complexity results for verification of asynchronous broadcast networks. In *Language and Automata Theory and Applications - 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, volume 7810 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2013.

[14] J. Esparza, P.Ganty, J. Leroux, and R. Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017.

[15] Marco Montali and Andrey Rivkin. Model checking petri nets with names using data-centric dynamic systems. *Formal Asp. Comput.*, 28(4):615–641, 2016.

[16] Serge Vaudenay. Centralized or decentralized? the contact tracing dilemma. *IACR Cryptol. ePrint Arch.*, 2020:531, 2020.