A Secure User-Centred Healthcare System: Design and Verification

Eduard Baranov[1][0000−0002−7357−705X], Juliana Bowles[2][0000−0002−5918−9114], Thomas Given-Wilson[1][0000−0001−8700−2671], Axel Legay[1][0000−0003−2287−8925], and Thais Webber[2][0000−0002−8091−6021]

[1] Computer Science and Engineering Department, Université Catholique de Louvain
Louvain-la-Neuve, Belgium
{eduard.baranov,thomas.given-wilson,axel.legay}@uclouvain.be
[2] School of Computer Science, University of St Andrews
St Andrews KY16 9SX, UK
{jkfb,tcwds}@st-andrews.ac.uk

Abstract. With ever increasing amounts of travel, it is essential to have access to a patient's medical data from different sources including many jurisdictions. The Serums project addresses this goal by creating a healthcare sharing system that places privacy and security aspects at the center. This raises significant challenges to both maintain privacy and security of medical data and to allow for sharing and access. To address these strict requirements the Serums system design is supported by formal methods where design decisions are modelled and checked to meet safety and security properties. We report an experience in support of the system design with formal modelling with the Uppaal tool and analysis with exhaustive and statistical model checking. Results show that statistical model checking being a simulation-based technique can significantly improve feasibility of analysis while providing support for design decisions to ensure privacy and security.

Keywords: healthcare, data sharing, privacy, security, design verification, formal modelling

1 Introduction

Improving patient care has become a priority across European healthcare providers and government establishments [19]. There is an increasing movement towards fulfilling patients' rights to securely access and share their health data across borders [27]. For example, when traveling abroad a patient may require a specialist to follow up their ongoing treatment; the patient may even experience some kind of medical emergency. Such data sharing can help healthcare professionals and organisations to improve their care services to patients in terms of efficiency, effectiveness, and enhanced decision making [10]. However, a healthcare data sharing system must ensure data protection and security and be in line with the European General Data Protection Regulation[3] (GDPR).

[3] Information on GDPR can be found at https://gdpr-info.eu/

The EU Horizon 2020 research project Serums[4] aims to increase healthcare provision in Europe through the proposal of a secure and transparent data sharing platform able to ensure privacy when accessing patient data [23]. Serums is grounded in two major pillars. First, the application of innovative techniques and emergent technologies like Blockchain and Data Lake to increase reliability and resilience against cyber-attacks. Second, to promote user trust in the safe and secure operation of the system in hospitals and clinics. The literature points out that Blockchain and smart contracts have great potential for enabling secure medical data access to healthcare parties [30, 35].

The core of the Serums Smart Health Central System (SHCS), which is the focus of this paper, involves several software components interacting with each other to provide fine-grained access control with audit trail to the patients' medical information stored in multiple data vaults.

A mechanism to customise access control over medical records is put in place between Blockchain and Data Lake technologies to allow authorised users to access medical data on demand, following data access rules predetermined by patients and healthcare organisations within the system. The data exchange format provided by Data Lake in Serums is the Smart Patient Health Record (SPHR), which provides metadata information linking the patients to subsets of medical data distributed across different hospital databases. Access rules to the SPHR are part of the smart contracts that can be customised by users in the Blockchain.

Serums provides the means in which patients can create access rules over personal records to healthcare professionals as well as healthcare organisations (i.e., administrators) can manage its users and specific rules to establish the boundaries for accessing patient data. For instance, the system allows authorised users to define *who* and *when* exactly *what* parts of medical records can be accessed. The Serums SHCS aims to maintain the system's security with reduced likelihood of privacy breaches. Due to the sensitive area that concerns Serums, verifying its platform is an important step in the system development cycle of the project [23]. Especially, one must prove that the architecture design choices ensure meeting strict privacy and security requirements. Such validation shall be done at design time as per request of the GDPR.

In this paper, we focus on this validation objective using formal methods, i.e., by applying approaches that work on a formal representation of both the system and the requirements under validation. Such representation is language agnostic which allows us to concentrate on the requirements to be verified. Another advantage of formal representation is that this naturally offers a clear semantics, which are particularly useful to improve the system in case of bug detection. Formal methods have been used in multiple projects to support the analysis of systems and their design, e.g. [8, 13, 21, 24, 26]. One of the commonly used representations for systems is transition systems [4] where system behaviour is modelled with a set of states and a relation between them describing how the system can change states. In formal methods, requirements are represented with

[4] For more information refer to `www.serums-h2020.org`

temporal logic [5]. Such logic is a temporal extension of the classical Boolean logic that permits the validation of a hypothesis on sequences of transitions.

Among existing formal methods that can be applied to transition systems and temporal logic, there is *Model checking* (MC) [14, 32]. MC offers an exhaustive exploration of the state space of the system. Contrary to software testing, MC guarantees that any behaviour of the model satisfies the property. Unfortunately, albeit the approach has been widely deployed, it is still subject to the so-called state space explosion problem: state space of non-trivial systems can be extremely large making exhaustive exploration infeasible. To avoid these issues, several authors have proposed *Statistical Model Checking* (SMC) [22, 28, 29, 34] as a compromise between testing and MC. The core idea of SMC is to run many simulations on which a property is checked and to use a statistical algorithm to decide the probability of the property to be satisfied with a selected degree of confidence. SMC has been broadly applied in different areas and projects including [6, 7, 18, 25, 36, 38].

The contribution of the paper is to provide a formal representation of the Serums platform and to validate Serums requirements on the formal representation. The SMC approach has been implemented in a wide range of tools (a comparison can be found in [3]), among which we have selected the Uppaal toolset [9, 16, 1] that includes an SMC engine. Uppaal supports both MC and SMC, is efficient, and has been used in many projects, e.g. [20, 31, 33]. The Serums platform in Uppaal is represented with stochastic timed automata, i.e., transition systems equipped with both timed and stochastic information [17]. The timing constraints are currently not being used, though the tool support offers a possibility to extend the model and to consider time-dependent properties in the future. Our model is parametrisable, scalable, and modular in such a way that we offer a library of automata to represent and hence duplicate at will each piece of the system. This aspect allows easy incorporation of any conceptual change that may occur in the project and simulation of real-life situations.

We first verified a set of safety and reachability properties on increasing model size (obtained by increasing the number of users of the model). This allowed us to show that the system behaves correctly in non-trivial instantiations with many users. While MC can only be applied to a significantly simplified model, SMC can verify the full model. In a second step, we proposed a transition-system based representation of an attacker, i.e., a malicious user that would try to have access to Serums Data Lake without going through the central system. In case of success, such attacker could get access to private data of others which would compromise the security of the entire design. With our model we were able to show that the first version of the platform integration was indeed subject to such attack. By using Uppaal, we were able to correct our model and hence the corresponding concrete design.

The structure of the paper is as follows. Section 2 introduces Uppaal tool and related concepts. Section 3 overviews the Serums platform architecture detail specifying components and interactions in the form of a workflow. Section 4 presents the model created with the Uppaal tool that is used for the evaluation

of the Serums platform design. Subsection 4.1 demonstrates the model checking process considering reachability and safety properties and providing a comparison between MC and SMC approaches. Section 5 introduces a security aspect to the verification process exploring the resilience of the design to attacks and the necessity of security consideration at design stage. Section 6 highlights the applicability and the need for SMC in cases like Serums to ensure the system reaches the expected behaviour providing privacy and security to end-users.

2 Background

Systems in Uppaal are modelled as a set of timed automata interacting with each other via channels controlling the synchronisation of transitions of several automata and shared variables. Uppaal also provides a mechanism to model multiple automata with identical behaviour - a template that can have parameters and be instantiated any number of times for the simulation. Templates provide means to analyse different scenarios with various number of automata.

For the properties, Uppaal provides a query language based on a simplified version of Timed Computation Tree Logic (TCTL). Temporal operators require a property to hold in either all execution paths, denoted with A, or in at least one execution path, denoted with E. In addition, operators have different modalities quantifying over specific paths. For example, $A\Box p$ requires proposition p to hold in all states of all execution paths and $E\Diamond p$ requires p to hold in at least one state of at least one execution path. Note that Uppaal does not allow nesting of formulas involving temporal operators, i.e. temporal operator can only be the outermost operator in the formula, therefore formulas like $E\Diamond p$ && $E\Diamond q$ shall be separated into 2 queries for p and q respectively.

Contrary to MC exhaustively exploring the state space, SMC is based on the idea of performing large number of system executions and monitoring the desired property on the executions. For non-deterministic systems, each execution would be different, thus multiple executions would explore various parts of the system behaviour. Being a simulation-based approach, SMC is known to be less time and memory consuming than MC. Uppaal SMC [16] is an extension of Uppaal to perform Statistical Model Checking. The extension works with stochastic timed automata, adds probabilistic choice between enabled transitions and probability distribution for time delays. For the interaction between automata only broadcast channels are allowed to be used to ensure components be non-blocking.

For the queries, Uppaal SMC uses an extension of Metric Interval Temporal Logic (MITL). Note that property check with SMC engine is performed on finite traces unlike in original Uppaal tool, therefore linear time is considered instead of branching. Basic temporal operators in Uppaal SMC are $\Box p$ and $\Diamond p$ checking that p holds in all or at least one state of the trace respectively. There are different types of SMC queries supported by Uppaal, the following ones are used further in the paper. The first type computes a probability of a property to be satisfied and is specified with $Pr[\# \leq N]\ F$, where F is a property specified with MITL, N is the maximal trace length and $\#$ indicates that we consider

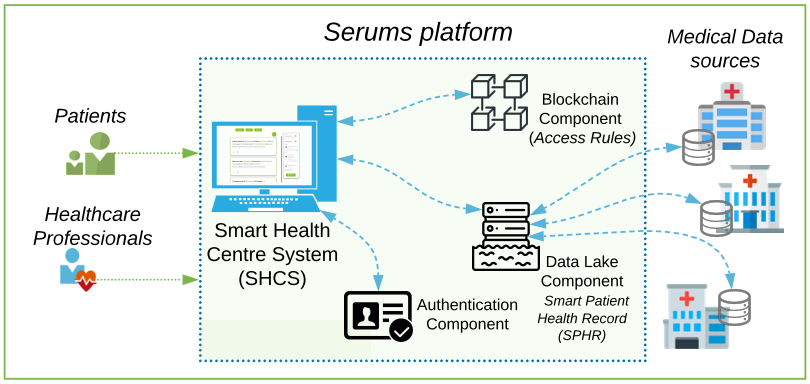

Fig. 1: An instance of the Serums platform with local Blockchain, Data Lake, and Authentication Components.

number of transitions in the trace length instead of time. The result of such query would be an interval $[x - \epsilon, x + \epsilon]$ with a confidence α, where ϵ and α are selected parameters. Another query type checks a hypothesis that probability of a property to be satisfied is above a given threshold: $Pr[\# \leq N]\ F \geq p_0$, where F and N are defined as above. For each query UPPAAL SMC builds a monitor that can check the property during the simulation, thus avoiding creation of a full simulation trace if the property can be decided on the first few steps. For the details of the monitoring, we address the reader to [12].

3 Serums System Design

The Serums project [23] addresses the need to securely share medical data to allow healthcare provision across different healthcare providers. For a patient visiting a new hospital, e.g., after the relocation to a different city or country, there shall be a simple way to access the patient's medical history as it is essential to provide effective healthcare. In contention with data sharing, the GDPR requires that the data shall be under the control of the patient: a patient's consent and approval are necessary for data access. The project goal is to create a platform for accessing and transferring these medical records in a secure and privacy-preserving manner among parties [23] that also meet regulatory requirements.

The overall architecture of the proposed platform is shown in Fig. 1. Serums platform architecture consists of several components interacting with a Smart Health Centre System (SHCS) [37]. The SHCS is a front-end that interacts with users (patients and professionals) and it is connected directly to other architectural components [11]. The Authentication component, which is central to check users' credentials, enables users' requests placed to other core components, like Blockchain and Data Lake. The Data Lake component is responsible for performing on-demand data acquisition and data processing: it can retrieve data

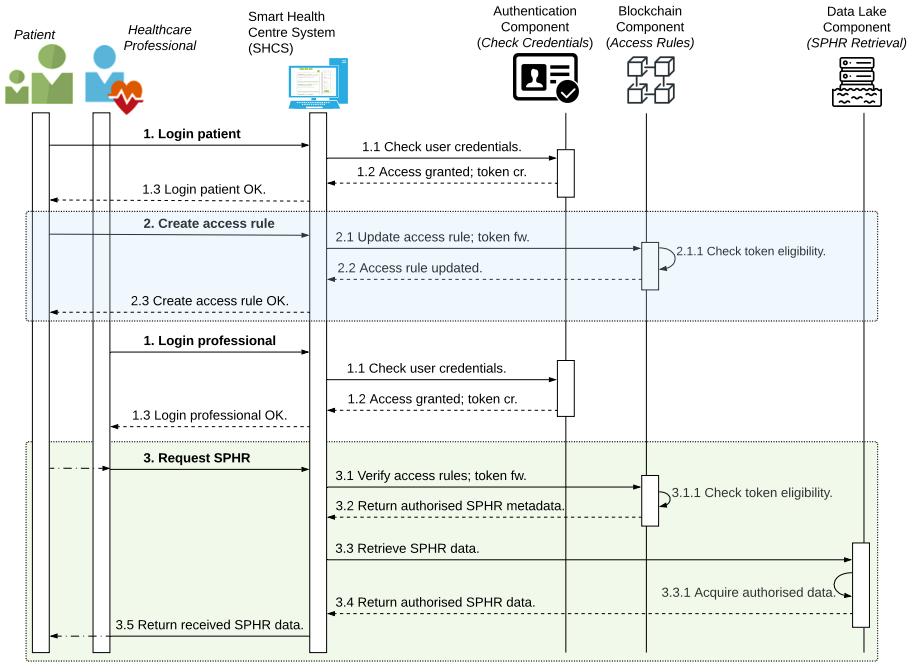

Fig. 2: Serums sequence diagram design showing the workflow for the main functionalities in the system.

from different sources and create a Smart Patient Health Record (SPHR) [10] structuring medical data as metadata. Upon request a patient's data can be securely retrieved and visualised by a given authorised healthcare professional.

Patients can control the data sharing with fine-grained access rules stored in smart contracts in the Blockchain component. The smart contracts contain access rules defining the access granted or denied to individuals, to parts of the patient's record, and for a given period of time. For example, a patient can make general information such as *name* and *blood type* be available to all medical personnel while specific *test results* shall be visible to the treating doctor only.

The main workflow of the Serums system is shown in Fig. 2. At first step (1) a patient tries to connect to the Serums SHCS and the request (1.1) is processed by the Authentication component. Authentication is done via JSON Web Token (JWT): a user after successful login receives a token that identifies him in all subsequent requests. At the next step (2) the patient can create a data access rule, for example, allowing a given healthcare professional to see his treatment history. This rule is added to the Blockchain (2.1), which first checks the forwarded access token for eligibility (2.1.1). At any further time, a healthcare professional can login (similarly via authentication component) and choose to request patient's record (3). The request is sent to Blockchain (3.1) which confirms the access token

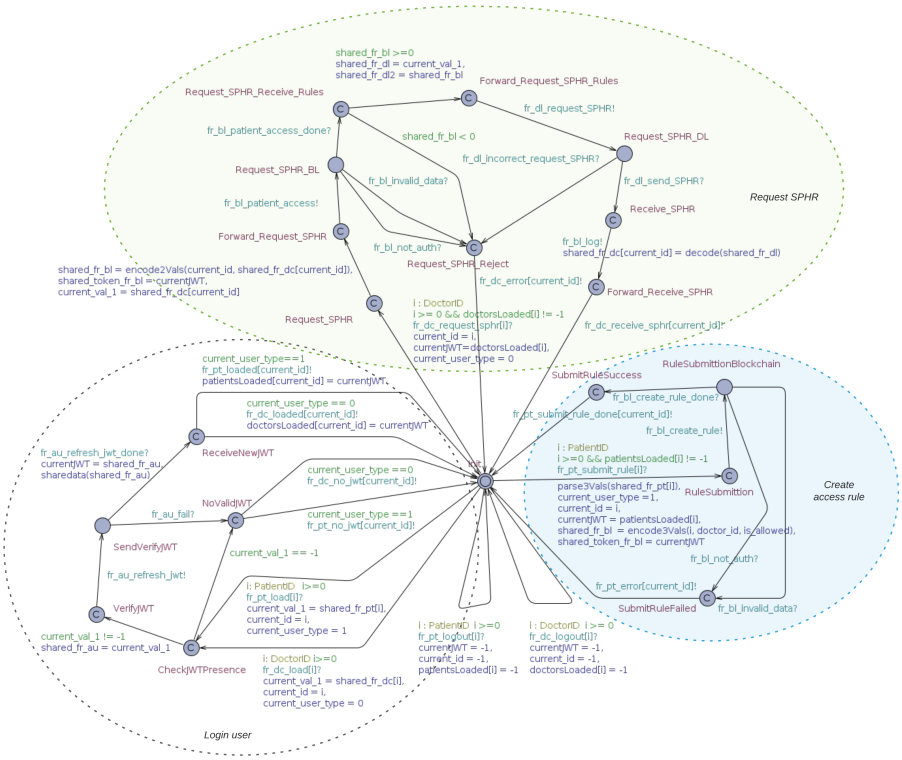

Fig. 3: SHCS component model.

eligibility (3.1.1) and afterwards returns to the SHCS the authorised metadata (3.2) that can be retrieved to the user according to the current rules in place for him. The SHCS then requests the medical data retrieval to the Data Lake component (3.3), which acquires data from varied data sources (3.3.1) and return the SPHR data to the requester in the front-end (3.4).

4 Serums System Modelling and Verification

Formal model being developed during design time can vastly improve safety and security of the resulted system. Building models of well-defined design decisions is straightforward and these models can then be analysed or verified using techniques such as MC and SMC. Given such a set of requirements, MC and SMC can provide insight on which decisions provide more guarantees.

For the verification of the Serums system design we are building a model within the Uppaal tool and using it to check properties and to test different design options. In the figures automata are represented with a graph where nodes are states of the system (their names are labelled with maroon) and edges are

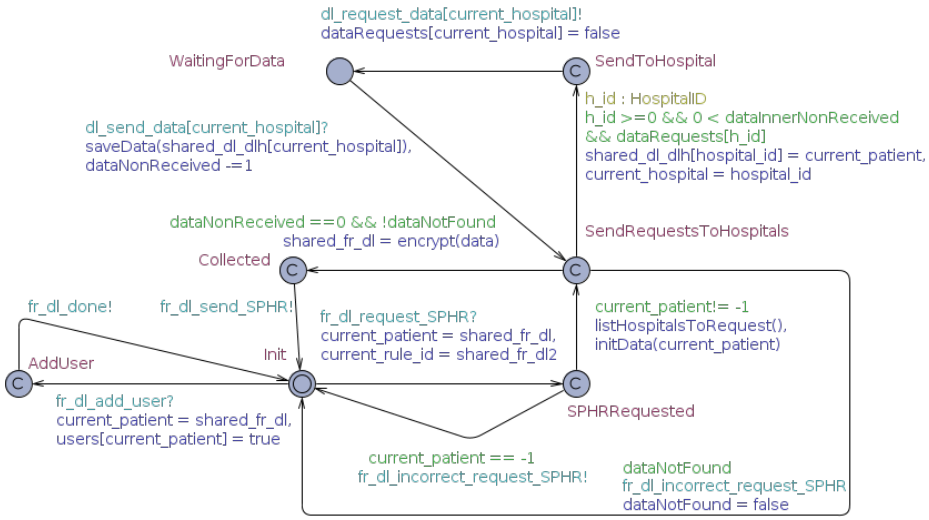

Fig. 4: Data Lake component model.

transitions defining how the system change states. Transitions have optional labels: tan label defines local variables to be used in other labels; green label is a guard controlling transition availability; turquoise label specifies synchronisation channel; blue label describe variables updates. Committed states marked with symbol `C` provide an additional control over system execution: if at least one automaton is in committed state then next transition shall be from one of such states. Detailed description of Uppaal models can be found in [9].

The model follows the Serums platform (Fig. 1) design and involves one or several automata for each component. The SHCS automaton shown in Fig. 3 is the central component interacting with users and other Serums components. Automata modelling patients and healthcare professionals are quite similar: they first can try to login to the Serums SHCS which includes interaction with the front-end Authentication automaton and then send one of the requests depicted in the sequence diagram in Fig. 2. In the model and properties, the term *doctor* is used in place of healthcare professional.

Login requests are initialised by users via interaction with SHCS that transfers them to the Authentication component modelled with two automata. The first automaton interacts with users during login and sign-up procedures and the second serves as a back-end simply responding to requests. Upon successful login, the back-end automaton generates an abstracted JWT that is shared with the user. The JWT is included in all subsequent user requests.

To create and modify access rules, user interacts with SHCS which forwards the request to the Blockchain automaton. Currently, we do not check properties of the Blockchain technology or smart contracts. Therefore, we abstract the Blockchain as a matrix storing data access rules. Whenever a patient creates or modifies access rule for a doctor, a corresponding cell of the matrix is updated.

Requests for medical data are performed in two steps. At first, SHCS interacts with the Blockchain automaton checking the access. If the access is granted, SHCS interacts with the Data Lake modelled with two types of automata. A single automaton shown in Fig. 4 represents the central part of the Data Lake: it receives a request for a patient's SPHR, collects it from local hospitals' databases and transfers the combined data. In addition, there is a set of automata modelling local data storage in multiple hospitals; the data can be updated by the hospital and been requested by the central Data Lake automaton.

4.1 Verification: Model Checking vs Statistical Model Checking

We are now ready to analyse properties of the model with UPPAAL. In order to show that the system behaves correctly with many users, we would vary the number of users by changing the number of instantiated doctor and patient automata during the analysis. For the paper, we consider two properties described hereafter.

1. Reachability property: doctors are able to receive an SPHR, in particular for any doctor there exists an execution path where the doctor can receive an SPHR of some patient.
2. Safety property: a doctor receives a patient's SPHR only if there is an access rule allowing the doctor to do so at the moment of request.

UPPAAL encoding of the reachability property consists of a set of queries; each query concerns a single doctor. The queries are $E\Diamond(d_i.\texttt{SPHRReceived})$, where d_i is the i^{th} doctor and `SPHRReceived` is a state of doctor's automaton in which SPHR is received. We consider a property satisfied if all queries are satisfied, the property checking time is computed as a sum of query times, and the memory consumption is the maximal consumption among all queries. Note that in the query we check presence of at least one execution path reaching `SPHRReceived` state rather than all paths; indeed one potential execution is all patients blocking some doctor forever.

Committed states in the model forbid parallel execution of SPHR requests and access rule modifications ensuring that no rule can be added or modified during SPHR request. Therefore, the safety property can be checked by the presence of [rules in the blockchain allowing access to the doctor at the state where the doctor receives SPHR. It must hold for any doctor at any point of time. The property is specified with the formula $A\Box\ \forall d\!:\!Doctor\ (d.\texttt{SPHRReceived} \implies blockchain.rules[d][sphr.patient] = \texttt{ALLOW})$.

In our experiments we are interested in a verification result as well as in time and memory required to perform them. Properties have been checked on a laptop with i7-8650U CPU and 16Gb of RAM.

Exhaustive MC is known to be afflicted by state space explosion [15]. On our model, MC quickly runs out of available RAM. One option to perform MC is to simplify the model: we only keep automata and transitions directly related to request SPHR routine (cf. Fig. 2) and remove everything else. In particular, we keep:

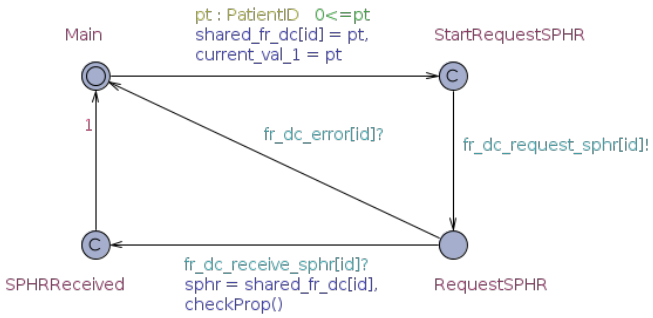

Fig. 5: Simplified automaton of a doctor.

1. automata for doctors, assuming them to be logged in and having an access token (i.e. all transitions related to login and to sign up are removed) shown in Fig. 5;
2. SHCS keeping only top part of the Fig. 3 (Request SPHR area);
3. Blockchain processing a single request to check access rules with assumption that rules are already created by patients and cannot be changed ensuring that for each doctor there is at least 1 patient providing access to the data;
4. Data Lake with an assumption that all patients' data in the model is accessible directly from central automaton without requesting storage at the hospitals.

The model checking results are shown in Fig. 6. Reachability property queries are evaluated within milliseconds: there are only a few available execution paths, and the desired state is reachable in less than 20 transitions. The safety property requires checking all the states of the model and, since the number of states grows exponentially with the number of doctors, there was not enough RAM to check the model with 8 doctors[5].

SMC provides an alternative to the exhaustive method since the simulation does not require computation of full state space while high confidence can be achieved without large time expenses [22, 28]. Memory consumption is also low since states that are not visited during the simulation are not generated. Encoding of properties is adapted as follows. For the reachability property we compute the probability of the state to be reached on traces of bounded length. In the original property we were checking the existence of at least one path, therefore for SMC we consider the property to be satisfied if the resulting probability is not close to 0. Note that the results of the SMC query provide more information: it also provides an estimation of the number of paths visiting the state.

The safety property has been checked with a hypothesis testing: H0 states that the property is satisfied with a probability above a selected threshold θ and

[5] It is possible to simplify the property by checking it separately for each doctor, however the simplification doesn't affect RAM consumption while single doctor check takes almost the same time as the check for all doctors.

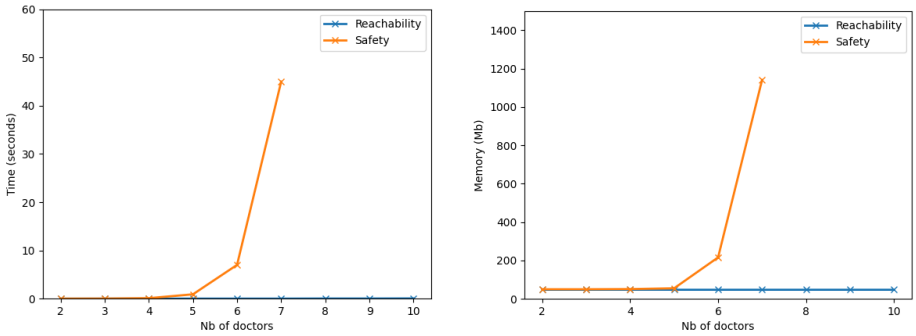

Fig. 6: Model checking time and memory with respect to the number of doctors on the simplified model. Safety property for 8 doctors runs out of RAM after 10 minutes execution without providing the result.

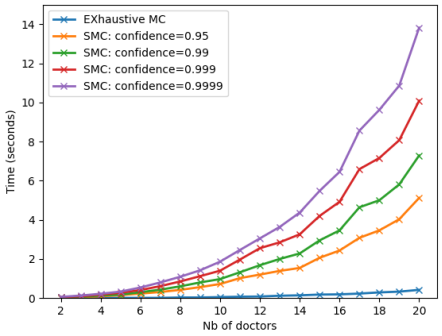

Fig. 7: SMC time on a simplified model for the reachability property with different confidence levels.

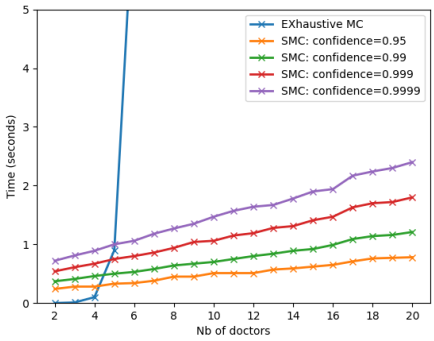

Fig. 8: SMC time on a simplified model for the safety property with different confidence levels.

H1 that the probability is below. An indifference region around θ covers the cases when it is not possible to select one of the hypotheses.

To compare results with the exhaustive model checking we used the same model and the following parameters for the SMC: traces are bounded with $1,000$ transitions, ϵ value for approximation interval for reachability property is 0.01, threshold for safety hypothesis is 0.99 with indifference region $[0.985, 0.995]$, confidence levels are 0.95, 0.99, 0.999, and 0.9999. The SMC engine is also able to estimate the number of data requests done within traces of given length, for $1,000$ transitions the expected value is 75.

The results of SMC evaluation are shown on Fig. 7 and Fig. 8. Reachability property is computed slower than with exhaustive model checking due to fact that evaluation is not stopped after finding a single trace reaching the desired state but exploring multiple traces until the probability is computed with the desired confidence. Confidence level 0.95 requires almost 200 simulations and

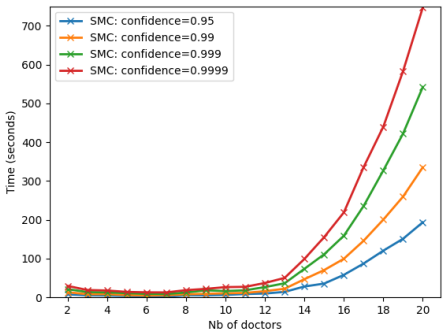

Fig. 9: SMC time on a complete Serums model for the reachability property.

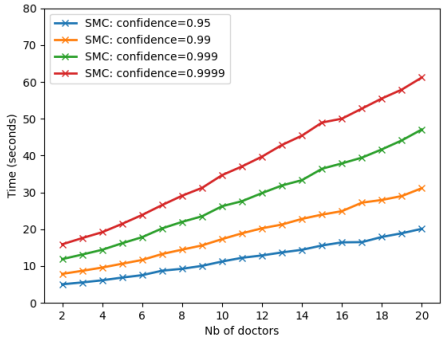

Fig. 10: SMC time on a complete Serums model for the safety property.

level 0.9999 requires more than 500 simulations. Safety property can be validated with SMC within 3 seconds even with 20 doctors, while the exhaustive method requires more than 40 seconds for 7 doctors and 1Gb of RAM. All SMC checks required just 50Mb of RAM.

While MC fails to check properties on a complete Serums model, SMC due to low memory requirements can provide results on the complete model. It involves patients modifying access rules at runtime; therefore, doctors could have access to the data during a limited interval of time and there is no guarantees that a doctor would have any access granted. Since the complete model is much larger, we raised the trace length to $10,000$; and expected number of SPHR requests to 450.

The plots presented in Fig. 9 and Fig. 10 show the computation times for both properties. Again, 50Mb of RAM was sufficient to calculate the results. Safety property can be evaluated within 1 minute even for a 0.9999 confidence. For the reachability property, we report time for a single query for one doctor. This property requires a large number of simulations to provide the satisfaction probability interval with the desired confidence due to high randomness of the system (the SPHR can be received only after a patient creates a rule allowing that). For 20 doctors, the confidence level of 0.9999 requires more than $20,000$ simulations, thus taking about 12 minutes.

Fig. 11 and Fig. 12 show the average time needed for a single simulation on simplified and complete models respectively computed as a total verification time divided by the number of simulations. Unsurprisingly, simulation time is independent from the desired level of confidence. Simulations for the reachability property are on average faster than for safety: the former ones can often be decided after few simulation steps while the latter ones (in case of being satisfied) requires simulating steps until the trace length bound. Note that the complexity of the model does not have a big impact on the simulation time: switching from a simplified model to a much more complex complete model and raising trace length by a factor of 10 affects the simulation time by a factor of 20.

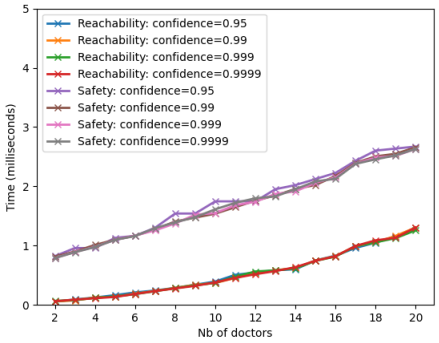

Fig. 11: SMC average time per simulation on a simplified model.

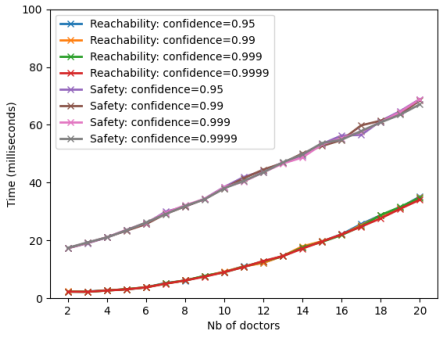

Fig. 12: SMC average time per simulation on a complete model.

5 Adding Security Requirements

Systems working in the real world should consider not only reachability and safety properties but security properties as well. Security properties shall check correctness of system behaviour in the presence of attackers. One cannot expect that users would follow the behaviour expected by the developers and the system must prevent all threatening interactions. Consideration of security properties shall be done at the design stage since vulnerabilities can be introduced there. In this section we show a simple attack vector and illustrate how formal modelling can help with detection of vulnerabilities.

The modular structure of the Serums system enables a distributed setup. In this case the components' API becomes available to the attackers. A problem arises with API of the Data Lake component. Considering an SPHR retrieval request (step 3.3 in Fig. 2), the only information received by the Data Lake is metadata about which part of patient's data shall be collected. The information of the original requester is not provided.

The existence of the Serums formal model allows us to validate whether this decision and the presence of an attacker can violate some of the properties. The modifications applied to the model are the following. Firstly, we model an attacker that imitates a doctor, but sending a request to the Data Lake instead of SHCS system. Metadata is not taken from the Blockchain rule but generated by an attacker. Secondly, we add a transition to the Data Lake automaton to receive request from the attacker (like receiving request from SHCS) and a transition to send the reply back. The modification has been done on both complete Serums model and its simplified version.

Exhaustive MC performed on the modified simplified model shows that the safety property defined in Subsection 4.1 is violated. There is a possibility to receive patient data without the stored rule allowing this. Indeed, an attacker can fabricate metadata granting unrestricted access to the patient's record and request all the data allowed by this fabricated metadata. SMC performed on

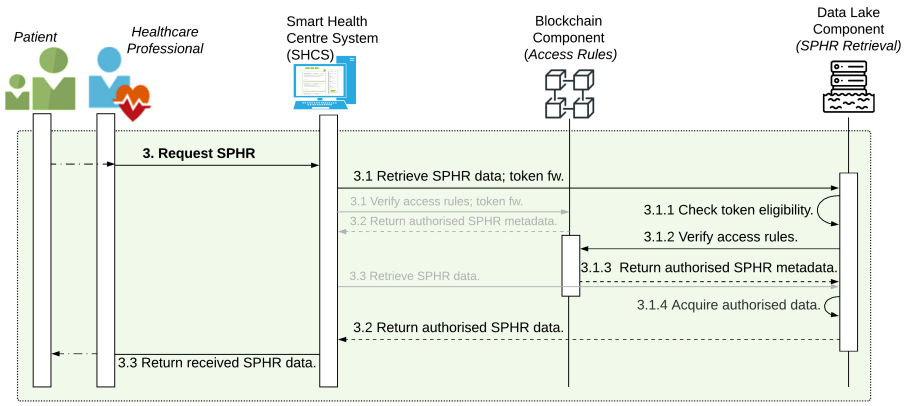

Fig. 13: Sequence diagram with alternate design for the SPHR data request functionality in the system.

both versions of the modified model rejects the hypothesis that the property is satisfied with probability above 0.99, i.e., there exist simulations on which the property is violated. This conclusion was drawn in less than 10 simulations.

The presence of this attack required us to modify the design of the system. The second proposed version of this functionality has a different behaviour and is illustrated with bold arrows in Fig. 13. Arrows depicted in light grey represent the previous interactions among components to make clear the system changes towards increased security. In this new version, the SHCS component directly sends all information to the Data Lake (3.1) and it is the Data Lake's responsibility to interact with the Blockchain and collect the data access rules (3.1.2). The Data Lake receives a patient id, a doctor id, and a doctor's access token and checks that the data is requested for the owner of the access token.

The Serums model has been updated to respect the new design flow. An attacker in this model can try to send any patient and any doctor id, however we assume that attacker cannot steal or forge someone else's access token. Verification of the properties with both exhaustive and statistical model checking showed that the properties are satisfied even with the presence of an attacker. The reachability property shows that the attacker can still receive some patient's data, however the safety property guarantees that the access to such data has been granted. Thus, we can conclude that the new design prevents the considered attack and covers the vulnerability.

Under the assumption that components' API is available from the outside world and that the request source cannot be identified, the model checking shows that the first version of the design has a vulnerability breaching the data confidentiality. For the second design the considered attack vector does not work. Thorough exploration of the design with the formal modelling allows us to find issues at the design time.

6 Conclusion

The Serums project aims to provide more secure smart health care provision with reduced potential for data breaches, and significantly improved patient trust and safety. This paper shows that formal methods and SMC in particular can provide significant support for architectural design decisions to ensure data sharing among healthcare providers with privacy and security.

Modelling of real-world projects that have multiple complex components results in large formal models on which exhaustive MC is infeasible. Only extraction of the behaviour related to a specific property results in a model small enough to be verified. Extraction requires manual work; in addition, it loses the ability to detect problems caused by interaction of different parts of the system. Alternatively, SMC can provide results with a high level of confidence on a complete model, exploring interactions between different functionalities and is capable of finding design flaws.

Future work includes two main directions. Verification of the Serums design to satisfy the remaining requirements and resilience against other attack vectors, including modelling and verification of smart contracts used in the blockchain model based on the approach from [2]. The other is the evaluation of the effectiveness of this proposed design on real-world healthcare environments provided by the Use Case partners in the project.

Acknowledgements. This research is funded by the EU H2020 project SERUMS (grant 826278). We thank Matthew Banton from the University of St Andrews for comments that greatly improved the platform security properties and Serums partners from Accenture and Sopra Steria for their help on the architectural diagrams design.

References

1. Uppaal. `http://www.uppaal.org/`
2. Abdellatif, T., Brousmiche, K.L.: Formal verification of smart contracts based on users and blockchain behaviors models. In: 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS). pp. 1–5. IEEE (2018)
3. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Transactions on Modeling and Computer Simulation (TOMACS) **28**(1), 1–39 (2018)
4. Arnold, A.: Finite transition systems - semantics of communicating systems. Prentice Hall international series in computer science, Prentice Hall (1994)
5. Baier, C., Katoen, J.P.: Principles of model checking. MIT press (2008)
6. Baranov, E., Given-Wilson, T., Legay, A.: Improving secure and robust patient service delivery. In: Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles. pp. 404–418. Springer (2020)
7. Basu, A., Bensalem, S., Bozga, M., Delahaye, B., Legay, A.: Statistical abstraction and model-checking of large heterogeneous systems. International Journal on Software Tools for Technology Transfer **14**(1), 53–72 (2012)

8. ter Beek, M.H., Borälv, A., Fantechi, A., Ferrari, A., Gnesi, S., Löfving, C., Mazzanti, F.: Adopting formal methods in an industrial setting: the railways case. In: International Symposium on Formal Methods. pp. 762–772. Springer (2019)
9. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Formal methods for the design of real-time systems. pp. 200–236. Springer (2004)
10. Bowles, J., Mendoza-Santana, J., Vermeulen, A.F., Webber, T., Blackledge, E.: Integrating healthcare data for enhanced citizen-centred care and analytics. Studies in Health Tech. & Inf. **275**, 17–21 (2020)
11. Bowles, J., Mendoza-Santana, J., Webber, T.: Interacting with next-generation smart patient-centric healthcare systems. In: UMAP'20 Adjunct: Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization. pp. 192–193 (July 2020)
12. Bulychev, P., David, A., Larsen, K.G., Legay, A., Li, G., Poulsen, D.B., Stainer, A.: Monitor-based statistical model checking for weighted metric temporal logic. In: International Conference on Logic for Programming Artificial Intelligence and Reasoning. pp. 168–182. Springer (2012)
13. Cerone, A., Elbegbayan, N.: Model-checking driven design of interactive systems. Electronic Notes in Theoretical Computer Science **183**, 3–20 (2007)
14. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Workshop on Logic of Programs. pp. 52–71. Springer (1981)
15. Clarke, E.M., Klieber, W., Novacek, M., Zuliani, P.: Model Checking and the State Explosion Problem, pp. 1–30. Springer (2012)
16. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. International Journal on Software Tools for Technology Transfer **17**(4), 397–415 (jan 2015)
17. David, A., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B., van Vliet, J., Wang, Z.: Statistical model checking for networks of priced timed automata. In: Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011. LNCS, vol. 6919, pp. 80–96. Springer (2011)
18. Ellen, C., Gerwinn, S., Fränzle, M.: Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains. International Journal on Software Tools for Technology Transfer **17**(4), 485–504 (2015)
19. Gavrilov, G., Vlahu-Gjorgievska, E., Trajkovik, V.: Healthcare data warehouse system supporting cross-border interoperability. Health informatics journal **26**(2), 1321–1332 (2020)
20. Gu, R., Enoiu, E., Seceleanu, C.: Tamaa: Uppaal-based mission planning for autonomous agents. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. pp. 1624–1633 (2020)
21. Harrison, M.D., Masci, P., Campos, J.C.: Formal modelling as a component of user centred design. In: Federation of International Conferences on Software Technologies: Applications and Foundations. pp. 274–289. Springer (2018)
22. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Implementations, LNCS, vol. 2937, pp. 73–84. Springer Berlin Heidelberg (2004)
23. Janjic, V., Bowles, J., Vermeulen, A., et al.: The serums tool-chain: Ensuring security and privacy of medical data in smart patient-centric healthcare systems. In: 2019 IEEE Int. Conf. on Big Data. pp. 2726–2735 (December 2019)
24. Jetley, R., Iyer, S.P., Jones, P.: A formal methods approach to medical device review. Computer **39**(4), 61–67 (2006)

25. Kalajdzic, K., Jégourel, C., Lukina, A., Bartocci, E., Legay, A., Smolka, S.A., Grosu, R.: Feedback control for statistical model checking of cyber-physical systems. In: International Symposium on Leveraging Applications of Formal Methods. pp. 46–61. Springer (2016)
26. Kwiatkowska, M., Lea-Banks, H., Mereacre, A., Paoletti, N.: Formal modelling and validation of rate-adaptive pacemakers. In: 2014 IEEE International Conference on Healthcare Informatics. pp. 23–32. IEEE (2014)
27. Larrucea, X., Moffie, M., Asaf, S., Santamaria, I.: Towards a gdpr compliant way to secure european cross border healthcare industry 4.0. Computer Standards & Interfaces **69**, 103408 (2020)
28. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: International Conference on Runtime Verification. pp. 122–135. Springer (2010)
29. Legay, A., Lukina, A., Traonouez, L.M., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Computing and Software Science, pp. 478–504. Springer (2019)
30. McGhin, T., Choo, K.K.R., Liu, C.Z., He, D.: Blockchain in healthcare applications: Research challenges and opportunities. Journal of Network and Computer Applications **135**, 62–75 (2019)
31. Mercaldo, F., Martinelli, F., Santone, A.: Real-time scada attack detection by means of formal methods. In: 2019 IEEE 28th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE). pp. 231–236. IEEE (2019)
32. Queille, J.P., Sifakis, J.: Specification and verification of concurrent systems in cesar. In: International Symposium on programming. pp. 337–351. Springer (1982)
33. Ravn, A.P., Srba, J., Vighio, S.: Modelling and verification of web services business activity protocol. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 357–371. Springer (2011)
34. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: 17th International Conference on Computer Aided Verification, LNCS, vol. 3576, pp. 266–280. Springer Berlin Heidelberg (2005)
35. Tanwar, S., Parekh, K., Evans, R.: Blockchain-based electronic healthcare record system for healthcare 4.0 applications. Journal of Information Security and Applications **50**, 102407 (2020)
36. Ter Beek, M.H., Legay, A., Lafuente, A.L., Vandin, A.: A framework for quantitative modeling and analysis of highly (re) configurable systems. IEEE Transactions on Software Engineering **46**(3), 321–345 (2018)
37. Webber, T., Mendoza-Santana, J., Vermeulen, A., Bowles, J.: Designing a patient-centric system for secure exchanges of medical data. In: Int. Conf. on Computational Science and Applications (ICCSA 2020). LNCS, vol. 12254, pp. 598–614. Springer (2020)
38. Zuliani, P.: Statistical model checking for biological applications. International Journal on Software Tools for Technology Transfer **17**(4), 527–536 (2015)