Exploiting new CPU Extensions for Secure Exchange of eHealth Data at the EU level

Luigi Coppolino
University of Naples 'Parthenope'
Naples, Italy
luigi.coppolino@uniparthenope.it

Salvatore D'Antonio
University of Naples 'Parthenope'
Naples, Italy
salvatore.dantonio@uniparthenope.it

Giovanni Mazzeo
University of Naples 'Parthenope'
Naples, Italy
giovanni.mazzeo@uniparthenope.it

Luigi Romano
University of Naples 'Parthenope'
Naples, Italy
luigi.romano@uniparthenope.it

Luigi Sgaglione
University of Naples 'Parthenope'
Naples, Italy
luigi.sgaglione@uniparthenope.it

***Abstract*—Cross-border health care requires that secure mechanisms for patient data exchange among distinct eHealth infrastructures be implemented. OpenNCP is a major initiative for achieving interoperability of eHealth data among European Member States. It is an Open Source implementation of a broker-based solution that enables the exchange of clinical data among countries having different languages and regulations. It provides some level of protection – using common security technologies (e.g., TLS) – but it has not been designed with the specific goal of achieving high levels of security, and therefore it is vulnerable to more subtle attacks, such as those by privileged users and/or software. In this paper we discuss how the new extension of COTS processors – namely *Software Guard eXtension (SGX)* – can be exploited to implement effective mechanisms against this specific category of attacks, which is particularly challenging. We present a general approach to harden systems, and discuss in detail how we implemented it in the context of OpenNCP. Also importantly, we evaluate the performance degradation induced by SGX.**

***Keywords*-eHealth Security, Intel SGX, Trusted Execution**

I. Introduction

The need for interoperable health services able to handle different languages, formats, and regulations among European member states led to *OpenNCP* [1]. This framework – currently adopted by 17 countries – allows to establish a set of shared practices with respect to patient data exchange between national infrastructures and to create a cross-border health information network through which clinical data can be exchanged. It is based on a federated architecture where national brokers are used for inter-country data exchange.
Security – in particular, *confidentiality* – of medical documents transferred from one country to another is extremely important not only for patient's privacy, but also for his/her safety. Standard hardening mechanisms are used in *OpenNCP*, e.g., Circle of Trust legal agreements, end-to-end secure communications (IPSec+SSL/TLS), and PKI services. However, since its main design objective was interoperability (as opposed to security), it is vulnerable to more subtle and advanced attacks, such as those perpetrated by users and/or software with higher privileges. For example, medical data is at risk when data is processed at the broker node, i.e., when it crosses the boundary between two distinct countries (and thus, security domains). A typical processing activity is when a document has to be translated to the language and format of the destination country. In that case, since content analysis is needed, data must be in plain text. This results in a vulnerability window, which exposes data in the host node to attacks by users and/or softwre with high privileges. As an example, an employee having privileged access to the server, may access memory contents during the processing phase, and steal - or even alter - patients' sensitive information.

In this context, the KONFIDO[2] has developed effective mechanisms to enhance the security of clinical data exchanged via *OpenNCP*. In this paper, we describe the work conducted to perform sensitive *OpenNCP* functions into the *Trusted Execution Environment (TEE)* of new Intel's CPU extension, namely *Software Guard Extension (SGX)*. Intel SGX – released in 2016 – allows for the creation of protected areas of memory, which are inaccessible even for privileged users. These TEEs, namely *Secure Enclaves*, provide protection of code and data residing inside through encryption and integrity checks of their memory range, directly performed by the CPU. SGX can be considered as a "reverse sandbox", i.e., it protects applications from the untrusted outside system, including the OS.
The idea is to leverage SGX for two different purposes: *i*) perform the transformation process into a secure enclave, thus hardening the elaboration of clinical data; *ii*) establish trusted communication channels between national broker nodes through the remote attestation mechanism provided by SGX, which allows to prove the trustworthiness of a software running in a specific enclave.
We analyzed two approaches to leverage the features offered by SGX. In fact, a technical challenge to be addressed regards the integration of C/C++ based TEEs into the Java Virtual Machine (JVM)-based environment of the OpenNCP framework. Following Brenner et *al.* [3], our approach has

been to leverage a Java Native Interface (JNI) in order to run sensitive code of OpenNCP within SGX enclaves by creating a *SGX-JNI Bridge*. This solution requires security-critical functions be rewritten in C/C++ and linked to the Java application through a well-defined interface.
The usage of SGX inevitably introduces an overhead in the system. Hence, we conducted a performance evaluation campaign with the goal of estimating the impact of using our *SGX-Secured* OpenNCP for clinical data exchange. We pursued an evaluation strategy based on combined use of modeling and prototype-based measurements. A *Queuing Model* of the transformation process was defined and filled with performance measures taken during the execution of our developed solution. In this way, we could define the best system setup supporting peaks of transformation requests.
Overall, the contribution provided by this paper is three-fold: *i*) It identifies a security vulnerability affecting the interoperability of a eHealth platform that is increasingly adopted in EU countries as cross-border infrastructure for medical data exchange; *ii*) It provides a security solution allowing to protect the infrastructure from attacks exploiting the identified vulnerability. This is not limited to eHealth data exchange, but – more in general – it may contribute to enhance the security of other infrastructures where data exchange has challenging security requirements (e.g., *eJustice*); *iii*) It evaluates the impact in terms of performance of the proposed technology and performs a possible system dimensioning by determining sustainable data rates in real-world scenarios.

The remainder of this paper is organized as follows. In Section II we provide background of Intel SGX and OpenNCP. Then, in Section III the threat model is defined. Section IV surveys the approaches available to integrate Intel SGX with *OpenNCP*. Afterward, in Section V, we provide details on the implementation of our solution. An evaluation of this solution is then reported in Section VI. This will be followed by VII where related works are overviewed. Finally, Section VIII provides concluding remarks.

II. Background

A. Intel SGX

Principles: The 7th generation of Intel's CPUs has been equipped with an innovative secure extension to the Instruction Set Architecture (ISA), namely Software Guard eXtension (SGX) [4]: a TEE based on a mechanism of "reverse sandbox" in which sensitive processes' address space is protected – at CPU level – even against OS. The idea behind is to protect selected code and data from disclosure or modification through the use of secure enclaves, i.e., address regions whose content is protected – via encryption and hashing – from any software outside the enclave, included privileged ones. Only the enclave code can access any part of the address space, except those areas belonging to other enclaves. The boundary between enclave and non-enclave sections is governed by the processor that blocks any access attempt from unauthorized processes. An interface – defined in a domain-specific C language – is declared by the programmer to establish entry points, i.e., calls to/from an enclave (namely *ecalls* and *ocalls*).

Remote Attestation: SGX dramatically enhances security of data residing in untrusted environments. Applications using SGX may be spanned over remote nodes. In such a situation it may be necessary to create secure enclave-enclave channels to exchange data without breaking the *circle of trust*. SGX therefore provides a mechanism, namely *Remote Attestation (RA)*, which enables service providers to provision applications, and to know with confidence that their secrets are properly protected [5]. The idea of RA is to prove – via a third remote entity – the goodness of a software running in a specific enclave. Such enclave must convince the other enclave – with which it is communicating – that it has a valid measurement hash, is running in a secure environment and has not been altered. The mutual verification is performed using a processor key, which is accessible only by a special enclave known as *Quoting Enclave*. SGX allows both *intra-attestation* and *inter-attestation* services, i.e., the procedure of RA performed between two enclaves residing on the same host or in different hosts, respectively. The remote attestation service builds a secure channel between the enclaves by performing a *Diffie-Hellman* key exchange.

B. OpenNCP

Overview: OpenNCP is meant to enable shared eHealth practices – with respect to patient data exchange – across European member countries. Besides supporting the correct data flow, goal of *OpenNCP* is to ensure the fulfillment of interoperability and legal requirements. *OpenNCP* provides a number of interoperable services, which enable national and regional eHealth platforms to set up cross-border health information networks. *OpenNCP* exchanges *Electronic Health Record (EHR)* composed of *Patient Summary (PS)* [6] and *ePrescriptions* [7]. The first contains patients' clinical data like personal information, insurance, allergies, vaccinations, previous treatments, and other important information, while the latter consists of electronic prescriptions of medicines that could be dispensed in the patient's home country and used abroad, in a Member State, in pharmacies. The PS is the document more exchanged by European countries. For this reason, in this work, we focus on the PS elaboration and transmission. Each operation of PS delivery entails a transformation of the document at the two communications sides. More precisely: *i*) *Transcoding*, i.e., the modification of the original document in a pivot OpenNCP-specific format; *ii*) *Translation*, i.e., the conversion of the pivoted document in the country's destination language
Figure 1 depicts the transformation process. A specific code (from the Danish code table) maps to an *OpenNCP* code. All

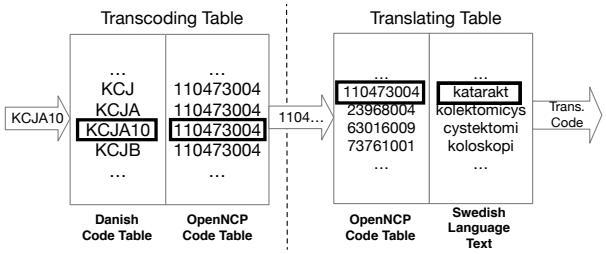


Figure 1: Patient Summary Transformation

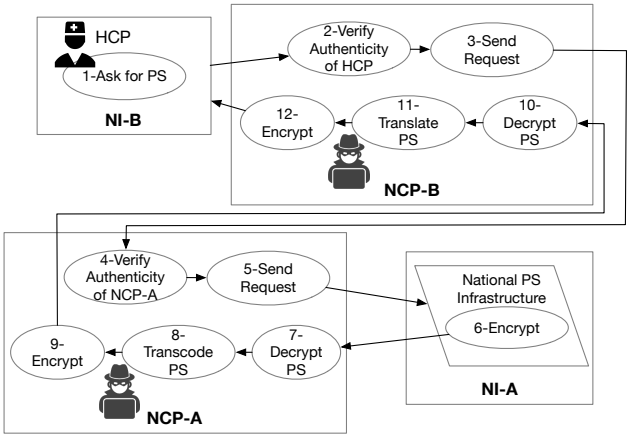


Figure 2: *OpenNCP* Dataflow Example

codes of the same disease category (i.e., KCJ to KCJB99) map to the same value in the *OpenNCP* code table. The transcoded term is sent to the other side where it gets translated in the destination language.

The confidentiality of PS data is particularly important. For this reason, the framework protects it via consolidated security mechanisms, which are Circle of Trust legal agreements, end-to-end secure communications (IPSec+SSL) and PKI infrastructure. However, this may not be enough. Before digging into the details of the reasons why the above-mentioned solutions do not provide a full security coverage, it is worth presenting the typical *OpenNCP* dataflow.

Dataflow: Let us assume a patient – an individual from country A – requests healthcare in a foreign country. The Health Care Professional (HCP) – a qualified individual from country B, e.g., a doctor – takes care of the request and needs the PS to perform the medical treatment. The provision of the PS to the HCP is realized as follows (see Figure 2):

- The HCP in country B uses his usual point-of-care system and requests (1) access to the Patient's health record stored in the National Infrastructure (NI) of country A.
- The request is authenticated (2) and sent to the National Contact Point (NCP) (3) of country B. The NCP is the most important unit that enables the exchange of data with other countries' NCP. It acts as a gateway for each country's National Infrastructure.
- The NCP-A verifies the identity of NCP-B (4) and forwards the request (5) to the NI-A, which returns a signed and encrypted PS (6). The clinical data than must be decrypted (7) to *transcode* entries (8) and then re-encrypted (9) before being transmitted to the NCP-B.
- The PS arrives to NCP-B. It is decrypted (10), integrity-verified, and *translated* (11) in the language of country B. Finally, the transformed PS is re-encrypted (12) and delivered to the HCP that made the request.

III. Risk Analysis and Threat Model

Clinical documents are exposed to confidentiality and integrity attacks during the transformation in NCPs. Particularly, this happens every time the PS crosses the border between two security-decoupled contexts. That is, when the NCP-A receives an encrypted PS from the NI-A and needs to decrypt, transcode, and re-encrypt the PS before sending it to the correspondent NCP-B. Or, when the NCP-B receives the PS from the NCP-A and needs to decrypt, transform, and re-encrypt the document before sending it back to the HCP. An example attack could be the following:

Clinical Data Tampering/Eavesdropping: a hostile user who gained access to the NCP host machine and escalated root privileges, or even a malevolent NCP administrator having full access to the system, dump the physical memory device (i.e., */dev/mem*) of the *OpenNCP* software responsible for document transformation [8][9]. They *alter* sensitive fields of the eHealth document to, e.g., hide critical allergies and so physically hurt the targeted patient. Another possibility is that they *steal* allergies information and then poison food with the particular case of allergy.

In this work we aim at protecting both *confidentiality* and *integrity* of clinical data subjected to the transformation process within NCP hosts. The spearhead of our solution is the capability of countering powerful adversaries having *root* privileges in the system. This is ensured by the TEE of Intel SGX. Our *SGX-Secured* OpenNCP ensures that exploited OS kernel services (running at *ring0*) cannot violate sensitive data. The malevolent user cannot intercept eHealth documents neither by performing L3 Cache side-channel attacks. On the contrary, she can violate data when resides in L1-L2 Cache like the case of *Spectre/Meltdown* attack [10]. Moreover, our *SGX-Secured* OpenNCP cannot protect against *side-channel* attacks [11] able to exploit timing and page faults [12], physical attacks that will unpackage a CPU, or any programming bugs in supporting libraries, which may lead to well-known attacks on SGX, like synchronization attacks [13]. Furthermore, attacks to the system availability, such as DoS attacks are outside of the scope of our work.

IV. Approaches to Integrate SGX and *OpenNCP*

Intel SGX allows writing secure enclaves in C/C++ language, while OpenNCP is written in Java and, therefore,

needs a Java Virtual Machine (JVM) to be executed. Two approaches for protecting *OpenNCP* sensitive functions with SGX are explored in the following.

A. Running a JVM in SGX Enclaves

One possibility consists in running a JVM in SGX enclaves in order to support the secure execution of *OpenNCP* sensitive functions. We investigated the possibility of porting an open source lightweight JVM, namely JamVM[1], compliant with the Java 8 specification, into the enclave. This is preferred to others – e.g., *Avian*[2] – since it is less complex. But, on the other hand, it does not provide the Just-In-Time (JIT) compiler, which has a certain performance impact since the JIT can reduce the runtime of Java programs substantially. However, JIT-less JVMs like JamVM have a smaller memory overhead and a faster startup time. Both properties are important when dealing with Intel SGX since the amount of protected memory available for enclaves is restricted. Usually, the JamVM depends on a number of libraries, which are not all needed in our use case:

- [Needed] *zlib*, a compression library
- [Not-Needed] *libdl*, a dynamic linking library
- [Needed] *libm*, a library providing math functions
- [Not-Needed] *librt*, a realtime extension library
- [Needed] *libpthread*, the POSIX thread library
- [Needed] *libc*, the C standard library containing needed functions for memory management and system calls

While some libraries (i.e. zlib and libm) have no dependencies and therefore can be efficiently moved into the enclave, others (i.e., libpthread and libc) heavily rely on system calls – forbidden in the enclave – and consequently entails several transitions in/out of the TEE that inevitably introduces overhead. Unfortunately, the approach of using the lightweight JamVM within enclaves is not feasible. In fact, we discovered that the TCB – even if smaller than the *HotSpot* one – is still too big, which violates one of the main security principles of TEE. Furthermore, we benchmarked a web server running on the solely *JamVM* and measured an extremely large overhead. This increases more ($\sim 30\%$) when SGX security mechanisms are put in place.

B. SGX-JNI Bridge

A different strategy is to run Java *OpenNCP* software outside the enclave, but uses a Java Native Interface (JNI) bridge to interact with code from within the enclave. The communication channel would be secured and give access to the security features of the enclave. The idea is to substitute sensitive Java functions of *OpenNCP* with new C/C++ classes and methods to be executed in the enclave. Therefore, an appropriate set of equivalent Java classes is developed. Methods implemented by the library are declared using the native keyword and left unimplemented in the Java code. This approach is proven to be one of the most effective as also shown by Brenneret *al.* [3].
A number of stub functions will be generated for the JNI wrapper: the wrapper, a chunk of C code that contains a specially named function to implement each of the classes native methods, and the generator is what gives us the names of these functions. We developed the JNI wrapper, implementing each function by converting its Java type arguments into appropriate C or C++ types, calling the native library, and converting the results back again.
We preferred this solution for three main reasons: *i*) The effort required to develop the bridge and port the sensitive code from Java to C++ is less than porting an entire Java Virtual Machine into the SGX enclave; *ii*) The TCB in this case is kept small, only the necessary code is ported in the enclave; *iii*) Finally, the performance of the application secured with this methodology are much better, the only source of overhead comes from SGX context switches.

V. The KONFIDO SGX-secured *OpenNCP*

We extended two components of *OpenNCP*, i.e., the *Transformation Manager (TM)* and the *Security Manager (SM)*, with the *SGX-JNI Bridge*. The goal was to move security-critical operations – i.e., encryption/decryption, signature verification, and document *transformation* – into the SGX enclaves. Such a porting activity firstly requires the identification of what runs inside and outside of an enclave, i.e., the definition of the SGX interface. The second step consists of the specification of libraries required to perform functions of interest and, therefore, the way of porting them. It is fundamental to identify main internal and external functions called by the TM and SM components.
Figure 3 depicts the sequence diagram of PS *transcoding* that clearly highlights those functions. In particular, the TM starts the transformation as soon as the SM decrypts the PS and the signature is verified. The *Workflow Manager (WM)* unit performs a *toPivot()* call by passing the PS in XML format as input. Then, after the validation of the Clinical Document Architecture (CDA)[3] – a document markup standard that specifies the structure and semantics of clinical documents – the TM starts the *transcoding* operation by iteratively asking the transcoded version of each original code to the *Terminology Service Access Manager (TSAM)* that queries a database to retrieve transcoded terminologies. Once all the codes are converted, the pivoted document is returned to the WM, encrypted and signed by the SM, and transmitted to the NCP counterpart, as seen in II-B. It is worth noting that SM, WM, and TM run locally in a NCP node, while the TSAM is a remote entity and exchanges the PS codes.
The question now is: *which partitioning solution can provide*

[1] http://jamvm.sourceforge.net/
[2] https://github.com/ReadyTalk/avian
[3] http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7

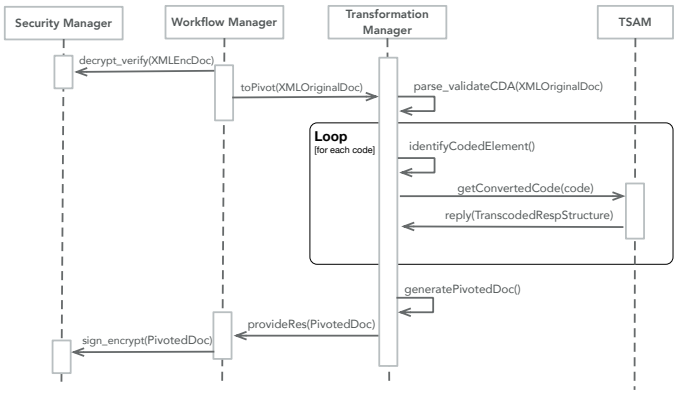


Figure 3: Sequence diagram of Patient Summary *transcoding*

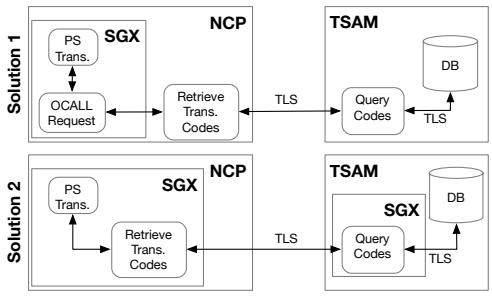


Figure 4: *SGX-Secured* OpenNCP solutions comparison

a reasonable trade-off between security and performance? It is important to identify which code belongs in the enclave, which code belongs outside of it, and how the untrusted and trusted parts interact. Depending on this choice, the SGX interface and the TCB – usually inversely proportional – vary in size. The partitioning may prove costly in terms of performance if program execution must frequently cross the enclave boundary interface. In our case, the idea is to access the enclave via the *SGX-JNI Bridge* with one *ecall* by giving in input the signed and encrypted document. Then, everything happens inside the enclave. That is, the SGX-SM takes care of verifying the signature and decrypting the file, while the SGX-TM deals with the document transformation.

A new version of the Security Manager was implemented, which executes within the enclave. In this way, the PS is never exposed unencrypted to the untrusted world. The SM uses SGX trusted libraries to carry out security operations. These implement AES-GCM algorithm to encrypt and MAC-protect the data. As far as the key management is concerned, we abandon the use of a Java Key Store (JKS) by moving keys within the SGX enclaves.

We identified two possible ways of leveraging SGX for the TM. These mainly differ for the level of security ensured and for their complexity. In both cases the effective transformation takes place within the TEE. In this regard, it must be pointed out that the transformation process can take place only if SGX is equipped with a XML parser able to perform the document correctness verification and the detection of codes to be transformed. We opted for a lightweight open source library, namely *TinyXML-2*[4], which has all the needed functionality and helps keep the TCB small. The library was modified to enable a correct linkage with SGX enclave code. The two approaches (Figure 4) are presented:

Solution 1: Execute the PS transformation within SGX and exit the enclave (*ocall*) to externally query the TSAM database. In this case, the possibility that the TSAM node is compromised and returns altered codes is not taken into account. In fact, while *OpenNCP* already guards the communication through consolidated techniques, it does not consider the risk of attacks at the TSAM node-level. Therefore, on one hand, this solution ensures *confidentiality* and *integrity* of the PS as the transformation process is shielded by SGX. On the other hand, it does not provide protection against attacks carried out in the TSAM node. An attacker, in fact, may modify the database running on the TSAM host and return modified codes. The SGX enclave interface includes three *ecalls* to initialize the JNI bridge, translate and transcode a PS document. Furthermore, the interface exposes an *ocall* function for requesting codes to the TSAM node. In this regard, instead of iterating over PS codes – as the original TM does – and for each of them starting a request, we preferred to create a buffer of queries in order to reduce the number of context switches between enclave and non-enclave worlds.
A new sequence diagram can be defined (Figure 5). The WM calls the *toPivot()* function, now exposed by the *SGX-JNI Bridge*. When this happens, an *ecall* is executed and the computation moves within the enclave. Here, the SGX-TM asks the SGX-SM to decrypt and verify the XML document. Afterward, it performs the same sequence of operations – as explained before – in order to *transcode* the PS. The only difference lies in the TSAM request, which in this case is buffered in order to reduce the number of *ocalls*. When the pivoted document is generated, the SGX-TM asks the SGX-SM to encrypt and sign the file before returning it.

Solution 2: We extend *Solution 1* by providing protection even for the TSAM server that, in this case, needs to be equipped with SGX hardware. The idea is to enforce an enclave-terminated TLS communication between NCP and TSAM, where *remote attestation* (seen in II-A) is used to verify the TSAM enclave validity. In this way, requests of translated codes do not require exiting the enclave, resulting in higher security. The database connection is also TLS protected but in this case it terminates within the enclave. *Data-at-rest* is kept encrypted and the key is saved within SGX. Encryption and decryption of database data are performed within the TEE.

[4]http://leethomason.github.io/tinyxml2/

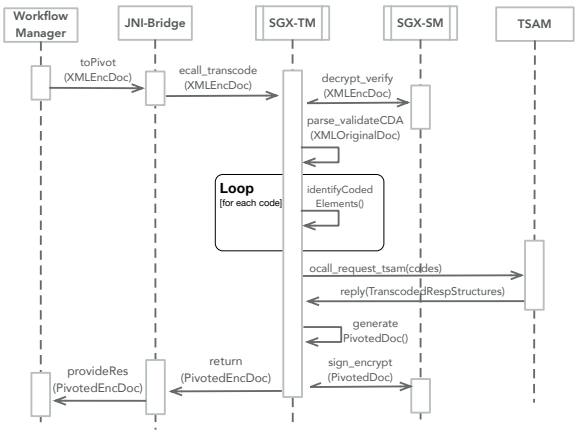


Figure 5: SGX-Enabled Patient Summary *transcoding*

VI. Evaluation

It is fundamental to analyze the impact on *response times* when the transformation process is performed within SGX enclaves. The reduced throughput resulting from the degradation of performance, and the latency in responses to the HCP, are indeed of major interest since the NCP – where the transformation takes place – receives demands of PSs from all hospitals in a country.

We adopted an approach based on combined use of modeling and prototype-based measurements. The strategy of using realistic values in the analytical model appears to be the most effective one. Since we are interested to the evaluation of sustainable PS request rates and to dimensioning the system in terms of number of SGX-enabled servers necessary to deal with pick rate of requests, we decided to model our system as an *M/M/s* queue. We will thus calculate the *waiting time* of PSs to be processed and compare results coming from the execution of three TM implementations, i.e., the default and the SGX ones, with and without RA.

1) Model of Patient Summary Transformation: The *M/M/s* queuing model must be characterized. In our case study, the *Arrival Process* – i.e., how PSs arrive to the server – is random at times $t_1, t_2, ..., t_j$. The inter arrival times $\tau = t_j - t_{j-1}$ are typically described in terms of the average time between two successive arrivals. The random arrivals in the time interval $(0, t]$ have a *Poisson* distribution. In many applications, like web server analysis, the independence property of the Poisson distribution does not reflect real life conditions. In our case study, instead, this is a correct assumption as PS requests – arriving with rate λ – are generally not correlated among them. The probability of PS requests in $(0, t]$ will be given by: $Pr(a(t) = n) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$

The *Service Mechanism*, i.e., how the PS transformation is made, was modeled with an *Exponential* distribution $f(\mu) = \mu^n e^{-\mu}$ with service rate (μ). The number of servers s running the transformation is a parameterizable value. We start with only one host and – based on the estimations – we tune such a parameter.

Finally, the *Queue Characteristics*, i.e., the dispatching discipline of PSs transformations. The algorithm we used is the First Come First Served (FCFS).

2) Selected Performance Metrics: As already said, the evaluation wants to measure time overhead due to usage of SGX within *OpenNCP* and so evaluate how many servers may be needed to support PS transformation requests. To this end, the output metrics we are interested in are:

- *Utilization Factor (%)* - The percentage of time that one or more servers are busy: $U = \rho/s = \lambda/s$
- The probability that there are n PSs in the system.

$$P_n = \begin{cases} \frac{\rho^n}{n!} P_0, & \text{for } n \leq 0 \\ \frac{\rho^n}{s! s^{n-s}} P_0, & \text{for } n > s \end{cases}$$

  Where P_0 is the probability that there are no PSs in the system, i.e., there is one or more servers idle:

$$P_0 = (\sum_{i=0}^{s-1} \frac{\rho^i}{i!} + \frac{\rho^s}{s!}(\frac{s\mu}{s\mu - \lambda}))^{-1}$$

- *Waiting Delay (s)* - The average time that a PS spends in the queue waiting for the transformation: $W_q = L_q/\lambda$. Where L_q is the average number of PSs waiting for the service:

$$L_q = \frac{P_0 \lambda \mu \rho^{s+1}}{(s-1)!(s\mu - \lambda)^2}$$

- *Overall Delay (s)* - The average time that a PS spends in the system in waiting and being served: $W = W_q + 1/\mu = L/\lambda$. Where L is the average number of PSs in the system: $L = W\lambda = L_q + \rho$

3) Experimental Testbed: The testbed used for the evaluation consists of two SGX-enabled machines. One residing in our laboratory in Italy, the other in Germany, belonging to a *Metal as a Service (MaaS)* deployment offering cloud style provisioning for physical servers. In the testbed scenario, the PS requests are performed from Germany to the Italian NCP. This, in turn, queries to the TSAM running in Germany for the codes transformation. Such a testbed is able to reproduce real conditions of a PS request procedures.

The machine hosted in Germany has the following specifications: an Intel Xeon E3-1270 v5 CPU with 4 cores @ 3.6 GHz with SGX extension enabled, 8 hyper-threads (2 per core), and 8 MB cache. The host has 64 GB of memory and runs Ubuntu 14.04.4 LTS with Linux kernel version 4.2.

The machine hosted in Italy has, instead, the following specifications: an Intel Core i7-6700 CPU with 4 cores @ 3.4GHz with SGX extension enabled, 8 hyper-threads (2 per core), and 8 MB cache. The host has 32GB of memory and runs Ubuntu 16.04 LTS with Linux kernel version 4.1.

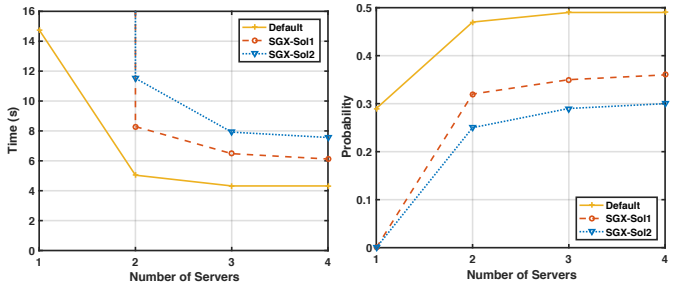
4) Estimation of Model Parameters: According to the guidelines released by EU [6], Patient Summary data includes patient administrative data and patient clinical data. The first includes fields like: national healthcare ID, personal

information, insurance number. The latter could comprise: alerts (e.g., allergies), medical history (e.g., vaccinations), medical problems, pregnancy history. These fields belong to an XML document having $\sim 2k$ lines and maximum depth equal to 15. In order to define the *Service Rate*, we measured the elapsed time between the submission of a transformation request and the return of results. We focused on the *transcoding* process and the following averaged results were obtained:

$$E[exectime] = \begin{cases} 4.3s, & \text{Default} \\ 6.2s, & \text{SGX - Sol. 1} \\ 7.3s, & \text{SGX - Sol. 2} \end{cases} \Longrightarrow \mu = \begin{cases} 837PS/hr, & \text{Default} \\ 580PS/hr, & \text{SGX - Sol. 1} \\ 493PS/hr, & \text{SGX - Sol. 2} \end{cases} \quad (1)$$

The *Arrival Rate* can strongly influence the system response time. We estimated the frequency of requests in a worst case condition, i.e., the NCP of a specific EU country receives requests from all its citizens staying abroad in Europe, and from all EU immigrants residing in that country. The Eurostat [14] report released in 2015 says that the country having more EU immigrants is Germany for a total of $4.17M$. At the same time the number of German citizens living abroad in other EU countries is equal to $928k$. Hence, the total number of potential NCP requests is $5.1M$. If we consider the worst-case scenario – i.e., everyone demands for PSs – the resulting rate is: $\lambda = 591PS/hr$.

5) Results: Tables Ia, Ib, Ic report numerical results of the evaluation conducted on the three TM versions, i.e., default, SGX-Solution 1, and SGX-Solution 2, respectively. The first evident result lies in the possibility of using one server only for the default transformation since in that case $\mu > \lambda$. In the other two cases, instead, $\mu < \lambda$ and therefore with $s = 1$ the waiting times inevitably diverges to infinite. However, the one server solution does not provide good results for the default TM. In fact – as also graphically reported in figure 6a – we observe that the estimated overall delay W is non negligible ($14.76s$), which may be not acceptable for eHealth requirements. For this reason, increasing the number of servers seems a reasonable choice even in that case. In this way, all parameters reach near-optimal values. We also notice that It would not make sense to further increase the number of hosts as there would be no significant differences. Another remarkable result regards the probability P_0 that there are no PS waiting in the system, i.e., that the system is idle. As figure 6b shows, the default version jumps from 0.29 to 0.47 by varying of just one unit the number of servers. Moreover, we also observe from table Ia that even the waiting time W_q drastically decrease down to $0.72s$.
The two reported graphs clearly highlight the impact of SGX on the TM. It is not surprising that all the measured metrics get worse, in particular for the RA-based version. Tables Ib and Ic demonstrate that using two servers is not the best solution, not only for the high waiting times but also for the utilization factor U, which is $> 40\%$ that is not good. Hence, in this two cases, using three hosts could certainly enhance such a metric.



(a) (b)

Figure 6: Overall delay of PSs in the system (6a) and probability that the system is idle (6b)

	$s=1$	$s=2$	$s=3$	$s=4$
U	70.6	35.3	23.5	17.6
P_0	0.29	0.47	0.49	0.49
W_q	10.44	0.72	0.07	0.008
W	14.76	5.04	4.32	4.32

(a)

	$s=1$	$s=2$	$s=3$	$s=4$
U	∞	50.9	33.9	25.4
P_0	0	0.32	0.35	0.36
W_q	∞	2.17	0.29	0.04
W	∞	8.29	6.48	6.12

(b)

	$s=1$	$s=2$	$s=3$	$s=4$
U	∞	59.9	39.9	29.9
P_0	0	0.25	0.29	0.30
W_q	∞	4.09	0.57	0.09
W	∞	11.52	7.92	7.56

(c)

Table I: Queuing analysis results for PS transformation using Default (Ia), SGX-Sol. 1 (Ib), and SGX-Sol. 2 (Ic) versions

What is also interesting to evaluate is the probability that there are n PSs in the system. We observed a strong difference between the default version and the SGX ones, in particular for the probability that there is only one PS. In that case we measured values of 0.48, 0.31, 0.26, respectively. At the same time, as n grows, differences decrease.

VII. Related Work

A number of research works investigated security solutions in the field of eHealth security. Some of these [15][16][17] used well-known techniques (e.g., PKI, RBAC, TLS) to securely transmit documents, enforce access control to the data, or protect *data-at-rest*. Others [18][19], discussed approaches targeting eHealth systems in the *Internet of Things* and *Cloud* domains. For example, Soceanu et *al.* [15] evaluated the applicability and feasibility of attribute-based access control (ABAC) policies, specified in XACML, a standard access control decision model specified by OASIS. Or more, Filkins et *al.* [17] provided guidelines based on proven security techniques to secure at different levels – i.e., service, transmission, and endpoint – data managed in eHealth infrastructures. Suciu et *al.* [18] analyzed the usage of existing methods for securely integrating big data processing with cloud systems based on Remote Telemetry Units (RTUs) for a cloud eHealth architectures.

Overall, no previous works provided an approach to protect clinical data processing against powerful privileged attackers. In our work, EHRs are always encrypted out of the CPU during computations and, thanks to SGX, neither *root* users can steal sensitive patients information.

VIII. Conclusions

In this paper, we have presented an approach based on the usage of commodity CPU extensions for securing eHealth data from attacks perpetrated by privileged users or software. Our proposed solution uses the hardware-assisted trusted execution environment of Intel Software Guard eXtension to harden clinical data exchanged within an OpenNCP infrastructure: a framework acting as a broker in cross border eHealth data exchange between European Countries. We have discussed how such deployment scenario is vulnerable to high privileged attackers able to gain access to the broker's machine and how SGX can address them. We have demonstrated that in spite of the overhead due to our solution, the SGX enhanced version of *OpenNCP* is still suitable to deal with realistic data loads.

Acknowledgments

This project received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreement No 727528 (KONFIDO). Authors are grateful to the SERECA consortium for the experimental testbed availability.

References

[1] M. Fonseca, K. Karkaletsis, I. A. Cruz, A. Berler, and I. C. Oliveira, "Openncp: a novel framework to foster cross-border e-health services," in *Digital Healthcare Empowering Europeans - Proceedings of MIE2015, Madrid Spain, 27-29 May, 2015*, 2015, pp. 617–621. [Online]. Available: https://doi.org/10.3233/978-1-61499-512-8-617

[2] M. Staffa, L. Sgaglione, G. Mazzeo, L. Coppolino, S. D'Antonio, L. Romano, E. Gelenbe, O. Stan, S. Carpov, E. Grivas, P. Campegiani, L. Castaldo, K. Votis, V. Koutkias, and I. Komnios, "An openncp-based solution for secure ehealth data exchange," *Journal of Network and Computer Applications*, vol. 116, pp. 65 – 85, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518301747

[3] S. Brenner, T. Hundt, G. Mazzeo, and R. Kapitza, *Secure Cloud Micro Services Using Intel SGX*. Cham: Springer International Publishing, 2017, pp. 177–191. [Online]. Available: https://doi.org/10.1007/978-3-319-59665-5_13

[4] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '13. New York, NY, USA: ACM, 2013, pp. 10:1–10:1. [Online]. Available: http://doi.acm.org/10.1145/2487726.2488368

[5] I. Anati, S. Gueron, S. P. Johnson, and V. R. Scarlata, "Innovative technology for cpu based attestation and sealing."

[6] "Eu guidelines on patient summary dataset," https://ec.europa.eu/health//sites/health/files/ehealth/docs/guidelines_patient_summary_en.pdf.

[7] "Eu guidelines on eprescription dataset," https://ec.europa.eu/health//sites/health/files/ehealth/docs/guidelines_patient_summary_en.pdf.

[8] D. V. G. A. G. D. P. Perazzo, C. Vallati, "Implementation of a wormhole attack against a rpl network: Challenges and effects," in *14th Annual Conference on Wireless On-Demand Network Systems and Services*, 2018, pp. 95–102.

[9] L. Coppolino, S. DAntonio, G. Mazzeo, and L. Romano, "Cloud security: Emerging threats and current solutions," *Computers & Electrical Engineering*, vol. 59, pp. 126 – 140, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045790616300544

[10] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," *CoRR*, vol. abs/1801.01203, 2018. [Online]. Available: http://arxiv.org/abs/1801.01203

[11] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel sgx," in *Proceedings of the 10th European Workshop on Systems Security*, ser. EuroSec'17. New York, NY, USA: ACM, 2017, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/3065913.3065915

[12] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, ser. SP '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 640–656. [Online]. Available: http://dx.doi.org/10.1109/SP.2015.45

[13] N. Weichbrodt, A. Kurmus, P. Pietzuch, and R. Kapitza, *AsyncShock: Exploiting Synchronisation Bugs in Intel SGX Enclaves*. Cham: Springer International Publishing, 2016, pp. 440–457. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-45744-4_22

[14] "Eurostat report on european immigrants," http://ec.europa.eu/eurostat/statistics-explained/index.php/People_in_the_EU_%E2%80%93_statistics_on_origin_of_residents.

[15] A. Soceanu, M. Vasylenko, A. Egner, and T. Muntean, "Managing the privacy and security of ehealth data," in *2015 20th International Conference on Control Systems and Computer Science*, May 2015, pp. 439–446.

[16] R. Charanya, S. Nithya, and N. Manikandan, "Attribute based encryption for secure sharing of e-health data," *IOP Conference Series: Materials Science and Engineering*, vol. 263, no. 4, p. 042030, 2017. [Online]. Available: http://stacks.iop.org/1757-899X/263/i=4/a=042030

[17] F. et al., "Privacy and security in the era of digital health: what should translational researchers know and do about it?" *American Journal of Translational Research*, vol. 8, 2016.

[18] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, and O. Fratu, "Big data, internet of things and cloud convergence – an architecture for secure e-health applications," *Journal of Medical Systems*, vol. 39, no. 11, p. 141, Sep 2015. [Online]. Available: https://doi.org/10.1007/s10916-015-0327-y

[19] H. Abie and I. Balasingham, "Risk-based adaptive security for smart iot in ehealth," in *Proceedings of the 7th International Conference on Body Area Networks*, ser. BodyNets '12. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 269–275. [Online]. Available: http://dl.acm.org/citation.cfm?id=2442691.2442752