Secure Stream Processing for Medical Data

Carlos Segarra[1], Enric Muntané[1], Mathieu Lemay[1], Valerio Schiavoni[2], and Ricard Delgado-Gonzalo[1]

Abstract—**Medical data belongs to whom it produces it. In an increasing manner, this data is usually processed in unauthorized third-party clouds that should never have the opportunity to access it. Moreover, recent data protection regulations (*e.g.*, GDPR) pave the way towards the development of privacy-preserving processing techniques. In this paper, we present a proof of concept of a streaming IoT architecture that securely processes cardiac data in the cloud combining trusted hardware and Spark. The additional security guarantees come with no changes to the application's code in the server. We tested the system with a database containing ECGs from wearable devices comprised of 8 healthy males performing a standardized range of in-lab physical activities (*e.g.*, run, walk, bike). We show that, when compared with standard SPARK STREAMING, the addition of privacy comes at the cost of doubling the execution time.**

I. INTRODUCTION

Personalized health and medicine has the potential of being the next revolution in healthcare. It is also referred as P4 medicine (Predictive, Preventive, Personalized, and Participatory), and it provides the opportunity to benefit from more targeted and effective diagnoses and treatments [1]. To implement this, larger amounts of data and complex processing pipelines are gradually being deployed, what generally leads to offloading computation to third-party cloud providers. When the data-in-motion are vital signs, protecting user's privacy becomes a topic of crucial importance. Furthermore, recent data protection regulations (*e.g.*, GDPR [2], [3]) stress the importance of protecting sensitive information against malicious attackers or untrusted cloud providers.

The current state of the art for privacy-preserving computation falls in two big categories: homomorphic encryption or trusted hardware. Homomorphic encryption (HE) is a cryptographic scheme that allows evaluating algorithms over encrypted data without having to decrypt it [4]. In spite of reducing the trusted computing base (TCB) to zero on the remote side, HE frameworks are currently prohibitively slow [5]. On the other hand, we have Trusted Execution Environments (TEE). A TEE is an isolated area of a processor that grants *confidentiality* and *integrity* to the information therein contained. TEEs are nowadays available in commodity CPUs, namely ARM TRUSTZONE and INTEL® SGX. The INTEL® SOFTWARE GUARD EXTENSIONS (SGX) is a set of instructions and memory access changes added to the Intel® architecture. These instructions enable applications to create hardware-protected areas in their application memory address space called enclaves [6].

In this paper we present a real-time privacy-preserving streaming platform for performing cloud computing on live cardiac data that runs unmodified Spark applications in a distributed environment, performs critical parts of the computation inside enclaves, and provides end-to-end protection for users' data. Without any modification to their applications' source code, potential users would instantly benefit of privacy-preserving computing relying on Intel SGX technology with limited impact on the performance. The main contributions of this paper are: (i) a proof of concept for a privacy-preserving streaming platform for medical data, and (ii) a study on the overhead introduced by privacy-preserving processing techniques.

The paper is organized as follows. In Section I, we introduce the problem we are facing and summarize the proposed solution. In Section II, we describe our use case. In Section III, we describe the architecture of the solution. Its components and materials are introduced in Section IV. The experiments and results obtained are presented in Section V. Lastly, in Section VI, we expose our main conclusions and propose further lines of research.

II. MOTIVATION

To illustrate that the solution presented is feasible and ready to be deployed, we apply it to a current industry challenge: the processing of cardiac data in the cloud. Our use case contemplates a scenario where multiple sensors are monitoring the cardiac activity of different users, see Figure 2. There are two main systems used by fitness enthusiasts for monitoring heart activity: electrocardiograms (ECG) and photoplethysmograms (PPG). ECG-based systems measure the heart's electrical activity over time and

Fig. 1. Schematic representation of an ECG signal showing three normal beats. A normal electrocardiogram can be broken down in three waves: a *P wave* corresponding to the depolarization of the atria, a *QRS complex* corresponding to the depolarization of the ventricles and a *T wave* corresponding to the repolarization of the ventricle [7]. From an ECG the sensor extracts and streams the R-peaks' timestamp and the time elapsed between them.

[1]Carlos Segarra, Enric Muntané, Mathieu Lemay, and Ricard Delgado-Gonzalo are with the Swiss Center for Electronics and Microtechnology (CSEM), Neuchâtel, Switzerland `carlos.segarra@csem.ch`

[2]Valerio Schiavoni is with the Université de Neuchâtel, Neuchâtel, Switzerland `valerio.schiavoni@unine.ch`

Fig. 2. Schematic representation of the proposed architecture. (Left) Client-server workflow. An arbitrary number of clients composed of a sensor and a gateway communicate with a remote untrusted server with INTEL SGX via a FileSystem Interface. The adversarial model assumes a privileged attacker inside the machine. (Right) Client package breakdown. Each client is, in practice, made up of five different components: a `producer` and `consumer` service that interact with the remote end, a `eclipse-mqtt` message broker to distribute the newly generated samples and a `mqtt-subscriber` to process them, and lastly a `sensor` that can either be a separate piece of hardware or an artificial DOCKER service.

it is the chosen method by chest-based sensors [8]. PPG-based systems measure the variation of blood volume over time using LEDs and photodiodes. Although less precise, PPGs are the chosen technique by all wrist-based cardiac monitoring sensors [9].

In our case, we focus on the analysis of the Heart Rate Variability (HRV) [10], that is, the analysis of the variation in the time intervals between heartbeats (a.k.a. RR intervals). The HRV is of utmost importance since it has been shown to be a predictor for myocardial infarction [11], [12]. We assume that the extraction of these values is performed by the sensor. Figure 1 depicts a detailed ECG and the information streamed from the sensor.

III. ARCHITECTURE

A. System Description

The proposed architecture follows a client-server scheme. As depicted in Figure 2, the system is composed of a remote server located in the cluster (or cloud) with access to INTEL® SGX and a set of clients distributed among different locations. In our use-case, each client has two components: a sensor that monitors cardiac data from a user and streams the RR intervals together with their timestamps, and a gateway that aggregates samples and interacts with the remote end. We impose no restrictions on the type of sensor and, our current gateway implementation could run, for instance, on a RASPBERRY PI or a SmartPhone.

On the server, we deployed a modified version of APACHE SPARK that exploits SGX called SGX-SPARK. SPARK [13] is a cluster-computing framework used to develop distributed applications. We implemented the HRV algorithms in Spark's binding for SCALA. In particular, we implemented the

SDNN temporal analysis and the spectral analysis (HF, LF, VLF) [14].

SGX-SPARK [15] is a framework that wraps SCALA's compiler and enables unmodified Spark applications to run critical parts of the processing inside enclaves. In particular, RR intervals are always processed inside Intel SGX.

The client package is divided in four main components: (1) a sensor that streams samples at an approximate rate of 1 to 3 Hz (60 to 180 beats per minute); (2) a MQTT [1] message broker; (3) a subscriber on the gateway that processes new samples; and (4) a producer and consumer that interact with the remote end. The link between the gateway and the server is established over SFTP and between 230 and 690 bytes are transferred per second. In our implementation, we are using Eclipse Mosquitto [2] as open source MQTT broker.

B. Threat Model and Known Vulnerabilities

Since the communication between the gateway and the server is kept protected by encrypting the data and using a secure transfer protocol (SFTP), our threat model is equivalent to SGX's. We assumes the system software to be completely untrusted (this includes privileged software such as operating systems, BIOS or virtual machine monitors) [6]. To verify the integrity of the code stored in the server, and that the code is really running in an enclave, Intel provides a remote attestation protocol. Once attested, the enclave can be trusted. INTEL® SGX (in particular the memory encryption engine, MEE [16]) is not designed to be an oblivious RAM, this is, it does not hide memory access patterns. Thus, an adversary could perform traffic analysis attacks. Other

---

[1] https://mqtt.org/
[2] https://mosquitto.org/

Fig. 3. Comparison of the maximum number of clients served in parallel and the maximum input load between the proposed architecture and SPARK STREAMING. (Left) Proportional bar representation of the values at which each system becomes unstable: it can not process the input received during the past 10 seconds in under 10 seconds. (Right) Summary table of the same values.

vulnerabilities could be exploited using side-channel [17] and speculative execution attacks [18], [19]. The proposed architecture assumes the client package to be completely trusted. Protecting it lies out of the scope of the project. However, an approach using ARM TRUSTZONE and OP-TEE [3] could provide similar security guarantees.

IV. MATERIALS

The proposed architecture has two main components: a server side equipped with INTEL SGX, and a client package with a sensor and a gateway. The system's assessment is also twofold. Firstly, we run it with data captured from real users to prove that it works in a real case scenario. Secondly, we perform stress tests with artificially generated data and evaluate the overhead introduced by privacy-preserving execution. We are interested in showing that the proposed architecture is also competitive in other scenarios with more stressful workloads.

A. Hardware

On the server side we use an Intel® Xeon® CPU E3-1270 v6 @ 3.80 GHz with 8 cores, 64 GiB RAM, and enclave mode enabled. It is based on UBUNTU 16.04 LTS (kernel 4.19.0-41900-generic) to support Intel SGX Driver 2.0. For SGX-SPARK we use a development version provided by the LSDS group.

B. Sensors and data

The used database is obtained from CSEM's proprietary wrist located sensors and chest-located dry electrodes [20]. In particular, cardiac data is obtained from eight healthy males following a standardized protocol in which they perform a range of physical activities from sedentary to vigorous [21]. We also augmented the database with simulated data with equivalent statistical moments than the the latter, adapting to the demands in workload of the evaluation stress test.

V. EVALUATION

To evaluate the system we perform a set of different experiments to assess particular characteristics of the proposed architecture and how do they compare with a normal execution of SPARK STREAMING, *i.e.* without INTEL SGX. We are specially interested in measuring the overhead introduced by privacy-preserving computations. This is, how do enclaves affect the system's latency, or delay, and how it compares in terms of clients that the platform can serve simultaneously.

Each experiment consists of 20 minutes of stream processing, generating a result sample, or batch, every 10 seconds. We are interested in measuring the average batch processing time and the memory footprint, that is, how many resources is the job consuming. We consider that a configuration is unstable if the average processing time exceeds 10 seconds.

This is, the streaming platform can not process the amount of data received during the last 10 seconds within the next ten seconds. We simulate data overload via two mechanisms: a single client sending a lot of information per second and many clients being processed simultaneously. Figure 3 summarizes the results obtained for our architecture and SPARK STREAMING. In both simulations, Spark is configured to run one master (or driver) process and one worker process with 2 GB of allocated memory.

For the first set of experiments, we deploy a variable number of clients, each streaming samples at approximately 1 Hz, and measure at which number each configuration becomes unstable. The system is able to handle 110 clients simultaneously whilst SPARK STREAMING is able to handle 225 users in parallel.

The second set of experiments is performed with a single client, increasing the samples per second the client sends until each system becomes unstable. The system is able to process up to 16 kB per second whilst SPARK STREAMING manages workloads of up to 32 kB per second.

To put it in a nutshell, the system seamlessly performs computations on untrusted clouds without compromising user's data with the constraint of approximately halving the system performance. This result is, per se, competitive with other privacy-preserving computing frameworks [22]

but is a major improvement, in fact a novelty, for privacy-preserving computing frameworks since it is transparent to the programmer of the final application.

VI. CONCLUSION

In this paper, we have presented a proof of concept of a streaming platform that grants executions on remote, untrusted, servers or clouds with data and code confidentiality and integrity. It provides end-to-end protection transparently to the developer since it runs unmodified APACHE SPARK applications inside INTEL SGX's enclaves.

We have quantified the impact on overall system performance when protecting health sensitive data from an untrusted cloud provider. More precisely, when performing an HRV analysis, it halves the maximum supported workload and the maximum number of clients the system can process simultaneously. We consider the system to be mature enough to be introduced in a production environment, since it complies with current data protection regulations whilst still maintaining a reasonable performance.

A major further step in reducing the Trusted Computing Base of the overall system would be providing additional protection to the client package. Given the success of enclaves, we suggest considering trusted hardware designed for smaller embedded devices such as ARM TRUSTZONE in combination with a TEE-TEE point to point secure transport link such as TALOS [23].

REFERENCES

[1] G. P. Cumming. Connecting & collaborating - Healthcare for the 21st century. In *Proceedings of the Second European Workshop on Practical Aspects of Health Informatics [PAHI], Trondheim, Norway*, 2014.

[2] P. Voigt and A. Von dem Bussche. *The EU General Data Protection Regulation (GDPR)*. Springer, Cham, 2017.

[3] The European Parliment and the Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L119:1–88, May 2016.

[4] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on theory of computing (STOC'09)*, pages 169–178, 2009.

[5] C. Göttel, R. Pires, I. Rocha, S. Vaucher, P. Felber, M. Pasin, and V. Schiavoni. Security, performance and energy trade-offs of hardware-assisted memory protection mechanisms. In *Proceedings of the IEEE 37th Symposium on Reliable Distributed Systems (SRDS'18)*, pages 133–142, October 2018.

[6] 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 (HASP '13)*, pages 1–8, June 2013.

[7] L. S. Lilly. *Pathophysiology of heart disease: A collaborative project of medical students and faculty*. Lippincott Williams & Wilkins, 2001.

[8] T. Tamura and W. Chen. *Seamless Healthcare Monitoring: Advancements in Wearable, Attachable, and Invisible Devices*. Springer, 2018.

[9] J. Parak, A. Tarniceriu, Ph. Renevey, M. Bertschi, R. Delgado-Gonzalo, and I. Korhonen. Evaluation of the beat-to-beat detection accuracy of PulseOn wearable optical heart rate monitor. In *Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'15)*, pages 8099–8102, 2015.

[10] A. Camm, M. Malik, J. Bigger, G. Breithardt, S. Cerutti, R. Cohen, Ph. Coumel, E. Fallen, H. Kennedy, and R. E. Kleiger. Heart rate variability: Standards of measurement, physiological interpretation and clinical use. Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology. *Circulation*, 93(5):1043–1065, 1996.

[11] R. E. Kleiger, J. P. Miller, J. Th. Bigger, and A. J. Moss. Decreased heart rate variability and its association with increased mortality after acute myocardial infarction. *The American Journal of Cardiology*, 59(4):256–262, 1987.

[12] J. Th. Bigger, J. L. Fleiss, R. C. Steinman, L. M. Rolnitzky, R. E. Kleiger, and J. N. Rottman. Frequency domain measures of heart period variability and mortality after myocardial infarction. *Circulation*, 85:164–171, February 1992.

[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10)*, pages 1–7, 2010.

[14] F. Shaffer and J. P. Ginsberg. An overview of heart rate variability metrics and norms. *Frontiers in Public Health*, 5:1–17, September 2017.

[15] F. Kelbert. D3.2 SecureCloud: Specification and Implementation of Reusable Secure Microservices, December 2017.

[16] S. Gueron. A memory encryption engine suitable for general purpose processors. *IACR Cryptology ePrint Archive*, pages 1–14, February 2016.

[17] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard. Malware guard extension: Using SGX to conceal cache attacks. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (CoRR'17)*, pages 3–24, 2017.

[18] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *Proceedings of the 27th USENIX Security Symposium (USENIX Security 18)*, pages 991–1008, 2018.

[19] O. Weisse, J. Van Bulck, M. Minkin, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, R. Strackx, Th. F. Wenisch, and Y. Yarom. Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution. *Technical report*, 2018.

[20] O. Chételat, D. Ferrario, M. Proença, J.-A. Porchet, A. Falhi, O. Grossenbacher, R. Delgado-Gonzalo, N. Della Ricca, and C. Sartori. Clinical validation of LTMS-S: A wearable system for vital signs monitoring. In *Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'15)*, pages 3125–3128, 2015.

[21] R. Delgado-Gonzalo, Ph. Renevey, E. M. Calvo, J. Solà, C. Lanting, M. Bertschi, and M. Lemay. Human ernergy expenditure models: Beyond state-of-the-art commercialized embedded algorithms. In *Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management*, pages 3–14, 2014.

[22] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI'17)*, pages 283–298, March 2017.

[23] P.-L. Aublin, F. Kelbert, D. O'Keeffe, D. Muthukumaran, Ch. Priebe, J. Lind, R. Krahn, Ch. Fetzer, D. Eyers, and P. Pietzuch. TaLoS: Secure and transparent TLS termination inside SGX enclaves. *Imperial College London, Tech. Rep*, 5, 2017.