A Secure, Privacy-Preserving IoT Middleware Using Intel SGX

Pascal Gremaud
University of Fribourg
Fribourg, Switzerland
pascal.gremaud@unifr.ch

Arnaud Durand
University of Fribourg
Fribourg, Switzerland
arnaud.durand@unifr.ch

Jacques Pasquier
University of Fribourg
Fribourg, Switzerland
jacques.pasquier@unifr.ch

ABSTRACT

With Internet of Things (IoT) middleware solutions moving towards cloud computing, the problems of trust in cloud platforms and data privacy need to be solved. The emergence of Trusted Execution Environments (TEEs) opens new perspectives to increase security in cloud applications. We propose a privacy-preserving IoT middleware, using Intel Software Guard Extensions (SGX) to create a secure system on untrusted platforms. An encrypted index is used as a database and communication with the application is protected using asymmetric encryption. This set of measures allows our system to process events in an orchestration engine without revealing data to the hosting cloud platform.

ACM Classification Keywords

D.2.11 Software Architectures: Information hiding; E.3 Data Encryption: Public key cryptosystems

Author Keywords

IoT Middleware; Privacy-Aware Computing; Security; Trusted Execution Environments; Intel SGX

INTRODUCTION

The constant growth of the IoT introduced a set of challenges, several important ones being related to security. While most of them are being addressed, the privacy subdomain still suffers from multiple unsolved ones. A critical issue is the fact that any IoT middleware performing data processing automatically requires access to these data. As Yan et al. observed, "[...] IoT services are based on data process, analysis and mining. This fact actually greatly intrudes user privacy" [6]. As cloud computing is seen as a convenient solution for IoT middlewares, one can only assume the trustworthiness of the hosting platform regarding application data, without being given any actual proof of it. However, trust is not sufficient in this context and it is widely accepted that more research is needed on this topic [6][5][1]. To approach this problem, limited processing on encrypted data can be achieved using homomorphic encryption. Talos [4] is a privacy-preserving system storing Things data in a cloud database and allowing queries on encrypted data. While it solves the privacy issue, this system is not able to run an orchestration engine in the cloud itself and thus relies on clients for handling the application logic.

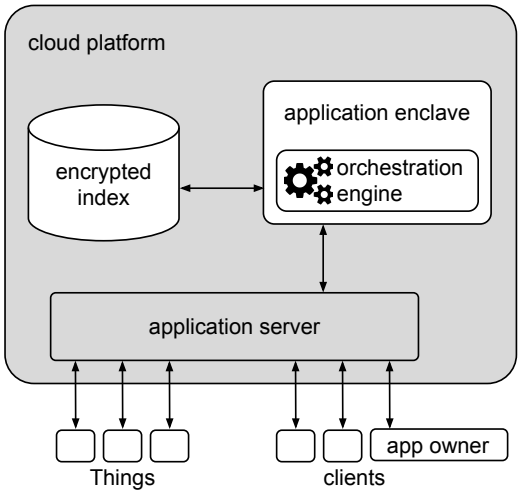

Figure 1. General architecture of the system. The grey area corresponds to untrusted components.

Since the last few years, several solutions to trust issues have been presented in the form of TEEs. Namely, Intel added a set of CPU instructions called SGX to their processors based on Skylake microarchitecture. These new instructions allow the creation of so-called enclaves, consisting of encrypted memory regions. The code and data inside an enclave are protected and cannot be accessed by any other processes, independently of their privilege level. This new technology enables secure computing on untrusted, remote platforms. Examples of using SGX to protect data execution include VC3 [3], a secure system for distributed MapReduce computations. The IoT middleware we propose uses SGX technology as a solution to privacy issues inherent to cloud computing.

SYSTEM ARCHITECTURE OVERVIEW

The components of our system architecture are shown in Fig. 1. The cloud platform hosts the application and is an untrusted part of the system. The enclave inside it contains an orchestration engine to interact with the Things and clients, which code and data are not accessible by the cloud platform. SGX was not designed to efficiently protect large portions of memory,

IoT '17 October 22–25, 2017, Linz, Austria

ACM ISBN 978-1-4503-5318-2/17/10.
DOI: https://doi.org/10.1145/3131542.3140258

this is why an encrypted index outside the enclave is used as a database. Its encryption key is provisioned by the application owner and stored inside the enclave. As a consequence, the index itself is considered trusted. Several secure indexes such as CryptDB [2] allow query execution over encrypted data and can be used to increase time performance for data retrieval. The index contains Things data as well as the orchestration engine data, as we do not necessarily considerate the application enclave as a long-lived component. This is also the reason why no information is stored solely inside the enclave. An application server is used as an interface between the application and the outside world. Being an untrusted component, all of the messages payloads it processes are encrypted. Things communicate with the application via the application server and thus need to be able to encrypt and decrypt messages they send and receive. Clients have the same role as Things while being oriented towards data consumption rather than data production. Finally, the application owner is a particular client whose main task is to instantiate the application and to create and manage the orchestration engine rules.

SECURE INITIALIZATION AND COMMUNICATION

All participants (the application, Things, and clients) use asymmetric encryption to communicate. The system itself is protocol-agnostic, most messaging protocols with asymmetric encryption support are suitable for implementation. Once the enclave is created by the untrusted platform, a remote attestation procedure is started using an attestation service provided by Intel, proving to the application owner that he is communicating with a genuine enclave containing the expected application code. In parallel, the untrusted platform transmits the public key of the application owner to the enclave. On attestation success, the application owner can establish a secure communication with the enclave and verify that he has been registered in it. The last step is the enclave receiving the key of the encrypted index from the application owner. At this point, the three essential trusted components have been initialized and the system is operational.

Once the setup has been completed, the concept of trusted and untrusted components is transparent to the participants. The application owner registers both Things and clients to the application. The public key of each of the registered participants is stored in the index and the public key of the enclave is shared with them. Finally, the application owner uploads the application logic (i.e. the orchestration engine rules) to the enclave. Alternatively, the entire index can be securely uploaded via the untrusted application server if an application is migrated from another platform.

At this point, it is necessary to understand that some data cannot be protected from the untrusted components. While the content of exchanged messages is inaccessible, it is still possible to infer information by analyzing the frequency of the messages, as well as the identity of the clients or Things sending (or receiving) them. One can, for instance, deduce a link between a Thing and a client if a message is systematically sent to a client after one is sent to the platform from a given Thing. Analyzing the message frequency, the untrusted components could also deduce whether a Thing is a simple logger using fixed time intervals or is communicating based on events. Even when considering such scenarios, the attack surface remains limited and the actual data is never revealed.

CONCLUSION AND FUTURE WORK

In this poster, we propose a secure, privacy-preserving IoT middleware using Intel SGX. By enabling secure computation in untrusted environments, such a system can solve important security issues inherent to cloud computing. More generally, TEEs offer impressive, new solutions to solve problems related to privacy and security in general.

A first prototype of the proposed architecture is being implemented with the Intel SGX Software Development Kit (SDK) for Linux, in conjunction with web technologies—Restful Web Services, JSON Web Encryption (JWE)—and shows the feasibility of the project. As next steps an extensive security analysis of the system would highlight its strengths and weaknesses and could open the way to other similar approaches for IoT middlewares. We will also produce performance tests to ensure that the overhead introduced by both SGX and the systematically encrypted communications does not prevent the system from being used as an actual middleware solution.

REFERENCES

1. Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. 2016. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems* 56 (2016), 684–700.
2. Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP '11)*. ACM, New York, NY, USA, 85–100.
3. Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy data analytics in the cloud using SGX. In *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 38–54.
4. Hossein Shafagh, Anwar Hithnawi, Andreas Droescher, Simon Duquennoy, and Wen Hu. 2015. Talos: Encrypted Query Processing for the Internet of Things. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*. ACM, New York, NY, USA, 197–210.
5. Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. 2015. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks* 76 (2015), 146–164.
6. Zheng Yan, Peng Zhang, and Athanasios V Vasilakos. 2014. A survey on trust management for Internet of Things. *Journal of network and computer applications* 42 (2014), 120–134.