Towards an Open Framework Leveraging a Trusted Execution Environment

Javier González[1] and Philippe Bonnet[1,2]

[1] IT University of Copenhagen, Denmark
{jgon,phbo}@itu.dk
[2] INRIA Paris Rocquencourt, France

Abstract. Sensor data is a core component of big data. The abundance of sensor data combined with advances in data integration and data mining entails a great opportunity to develop innovative applications. However, data about our movements, our energy consumption or our biometry are personal data that we should have full control over. Likewise, companies face a trade-off as the benefits of innovative services must be weighted against the risk of exposing data that reveal core internal processes. How to design a data platform that enables innovative data services and yet enforce access and usage control? The solutions proposed in the literature to this trade-off all involve some form of trusted execution environment, where data and processing is trusted and safe from corruption by users or attackers. The hardware that could support such trusted execution environments is however closed to the research community: OEMs disable security extensions from their development boards and the software handling these security extensions is not open. In this paper we present a framework that combines commercially available hardware and open source software. It can be used today by the research community as a trusted execution environment to investigate future big data platforms.

1 Introduction

The decrease in size, price and power consumption of large classes of sensors equipped with computation and communication capabilities is making sensor data a crucial component of big data systems [4]. The sheer availability of data about energy consumption in an office raises obvious questions: does constant employee monitoring improve productivity? Does it improve the quality of facility management? Does it improve energy efficiency? What if this data is in the possession of a direct competitor: can it be misused? The trading of personal data, maintained outside our control, for innovative data services is recognized as a significant problem by analysts and even by European legal organizations [4,6]. The problem is as similar for companies that are in a position to trade sensitive internal data for innovative services.

To address this increasing information security problem, the consensual solution is to develop a data platform that enables innovative data services (relying

G. Wang et al. (Eds.): CSS 2013, LNCS 8300, pp. 458–467, 2013.

on modern data integration, mining or clustering techniques) while allowing users to retain some form of control over who access this data (access control) and how (usage control). Put differently, we need a trusted data platform to support innovative services based on sensor data. In [8], solutions involving a trusted middleware layer are described. In [1], we proposed the vision of trusted cells, a decentralized data platform based on trusted execution environments embedded on personal data devices (set top boxes, smart phones or smart meters) at the edges of the Internet. These visions are based on the premise that trusted execution environments are actually available and can be programmed to enforce access and usage control policies. However, the hardware that could support such trusted execution environments has so far been unavailable to the research community: OEMs disable security extensions from their development boards and the software handling these security extensions is neither open nor widely available. Our experience is that it requires first hand information to know which security extensions (if any) are enabled in a given board, making it impossible for most developers to determine which are the security capabilities of the processors powering their own boards.

In this paper, we describe an open framework that combines hardware and open source software, and provides a trusted execution environment that is (i) readily available to the research and open source communities, (ii) fits well into well-known programming frameworks such as Linux and Android, and (iii) is rich enough to support the design of future big data platforms. We call this framework the Arm-Xilinx/OpenVirtualization framework as it combines secure hardware from Xilinx, based on a chip equipped with the TrustZone system from ARM, with the open source, secure operating system OpenVirtualization from Sierraware.

In the rest of the paper, we discuss the requirements for supporting usage control in big data platforms (Section 2), we then discuss what kind of framework is needed to support our vision of trusted cells(Section 3). Finally, we describe the ARM-Xilinx hardware platform and the OpenVirtualization software that constitute a very promising framework in this context (Section 4). We finally draw our conclusions in Section 5.

2 Enforcing a Usage Control Model

Once a user give away some data, she loses any form of control over it. The decision of sharing data is, so far, a discrete operation: either all or none of the data is shared. When dealing with sensitive data, mid points are sometimes achieved by means of external legal agreements, such as Nondisclosure Agreements (NDAs), licenses, and other terms of usage. A user might for example give her consent to some terms of usage when she gives away personal data for a given service (e.g., giving away location for a smart phone app or giving away energy consumption data for a social game aiming at improving energy efficiency). Legal actions are then possible if a non permitted data access or distribution is perpetrated. These legal actions are complex and possibly costly. But most importantly, they are

taken once the damage is already done. Furthermore, no legal agreement can prevent malicious attacks from being perpetrated against the devices storing sensitive data. Ideally, the data sharing process would enable two (or more) parties to negotiate a contract, which we call a usage model, defining who can access the shared data and how this data is used, while the underlying data platform ensures that the contract is met by all parties at all time. Put differently, the data platform enforces the usage model by preventing contract breaches either from the contract parties or from third parties. Now, we are faced with two core questions: (1) How does a usage control model looks like? and (2) what does it take to enforce it?

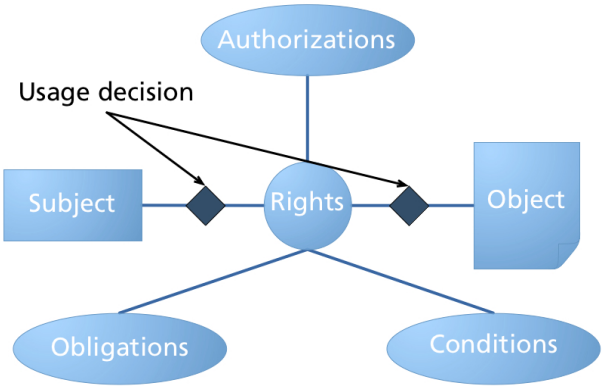

Fig. 1. $UCON_{ABC}$ model: the reference monitor enforces usage decisions (can a given subject apply a right on a given (set of) object(s)?) based on subject and object attributes, as well as authorizations, obligations and conditions.

Usage control models usually refer to $UCON_{ABC}$ [7]. In the $UCON_{ABC}$ model, subjects provide or consume data objects. Objects might contain secrets about identified subjects (the data producer, the data consumer, or possibly a third party). A subject accesses objects via a set of usage functions referred to as rights. A reference monitor is responsible for taking usage decisions based on the subject and object attributes as well as Authorization, oBligations and Conditions *A*BC. Authorizations are predicates that define whether a subject is authorized to hold a right; obligations are predicates that define the actions that a subject must take before or while it holds a rights; and conditions define predicates that must hold true about the environment in which the subject requires a given right.

What does it take to enforce a $UCON_{ABC}$ model? The data platform should implement a reference monitor and guarantee that there will be no access to data objects unless appropriate usage decisions (i.e., decisions that respect the contract negotiated by all parties) are taken. The basic idea is that the reference monitor is a software component that relies on hardware security features separating a secure world (where objects, attributes, as well as rights, authorizations

and conditions are securely stored, while usage decisions are securely executed) and a non secure world (where the rest of the processing takes place). There is today, to the best of our knowledge, no implementation of any $UCON_{ABC}$ model. In fact, there has not been, so far, any easily accessible framework that distinguishes secure and non secure worlds. We expect that the form of hardware security and rich operating system integration that we describe below will be a breakthrough for the research community and that it will enable experimentations with various forms of usage control models.

3 Implementing Trusted Cells

In [1], we proposed a model, called Trusted Cells, where a client side reference monitor is embedded on smart devices (including set top boxes, smart phones or smart sensors) at the edges of the Internet. With trusted cells, the infrastructure (communication, computing and storage) is untrusted, while the personal devices that data owners and consumers use to provide or access data are trusted. In the rest of the paper, we focus on Trusted Cells as a data platform implementing a $UCON_{ABC}$ model. We hereby encourage the community to define alternative data platforms.

The vision of the trusted cells is based on the separation of a trusted and a rich environment maintaining a high throughput between them[1]. How to separate a non trusted and a trusted environment? There are three options. The first one (O1), is to opt for a software solution, without hardware support. The second solution (O2) is to fully separate a secure device (where software and hardware are trusted), e.g., a secure token, and a non secure device, e.g., a PC in which the secure token is plugged. The third option (O3) is to consider a processor that propose a secure mode of execution and a non secure mode of execution.

Before we review these three options, we need to precise the types of attacks that trusted cells might be submitted to [9]. First, **Hack Attacks**, which are limited to software. Examples of hack attacks include viruses or malware. These attacks are normally triggered by an user approving the installation of a piece of software that then executes the attack. Second, **Shack Attacks**, which are low-budget hardware attacks. Attackers have physical access to the device, but they lack the knowledge or equipment to carry out an attack at an integrated circuit level (e.g., scanning I/Os, forcing pins, reprogramming memory devices). Third, **Lab Attacks**, which are comprehensive and invasive hardware attacks. Attackers have access to laboratory equipment and the knowledge to perform unlimited reverse engineering of a given device (e.g., reverse engineering a design, performing cryptographic key analysis). The assumption should be that a device can always be bypassed by a lab attack given enough time and budget.

[1] Note that a rich non secure world and a high throughput between secure and non secure worlds, will enable innovative applications and that there is always a trade-off between how rich the secure world can be and how easy it is to guarantee that the secure operations are indeed secure. Exploring this trade-off is a topic for future work

Now, who might be a third party attacker (in addition to the Subject and Object)? We adopt the taxonomy proposed by IBM [3], which distinguish between three classes of attackers. First, **intelligent outsiders**, i.e., remote attackers, who lack specific knowledge on the system. They might have access to moderately sophisticated equipment. These attackers try to reproduce software or simple hardware attacks published in the Internet by a technical expert, rather than attempt to create new ones. Second, **trained insiders**, that are technical experts, highly educated, with experience and access to sophisticated tools. They are considered trusted, possibly employed by the company developing the device subject of the attacks. Their knowledge of the system varies, but it can be assumed that they have access to the information describing it (e.g., secret information, detailed designs). They seek discovering an sharing new class attacks. Third, **funded organizations** that represent organizationally founded teams of trained attackers. They are capable of carrying out sophisticated attacks by means of advanced analysis tools. They are capable of designing new and innovative attacks exploiting the most insignificant weaknesses.

So, what kind of attacks do we envisage? The most probable attackers are (i) data owners that might want to alter their own data (e.g., customers could try to lower their energy consumption bill by altering the measurements stored in their own smart meters), (ii) subjects that access unauthorized objects (voluntarily or not), and (iii) third parties that intentionally extract (large volumes of) data objects (e.g., organized crime that plans to sell industrial information to a competitor or organize burglaries in a neighborhood at a time when all houses are likely to be empty, inferring this information from electricity usage).

The first solution (O1), only based on software is thus not appropriate. Note that today, all popular data platforms, are only based on software solutions. Note also, that the number of large scale break-ins and leaks are getting greater by the day [2] and their effects are beginning to get noticed by the population (e.g., in Denmark[3].

The second option (O2) relies on dedicated, tamper resistant processors extended with security and cryptographic features that take care of sensitive operations. For example, a dedicated processor can be used to perform cryptographic operations associated with the management of DRM digital certificates. The main characteristic of this approach is that security is provided by physical isolation. The dedicated processors are difficult to access physically, and their design and size makes it difficult for a sophisticated lab attack to succeed. Depending on the impediments to perform a lab attack, and the response of the platform to a potential successful attack, the platform can be described in terms of its level of tamper resistance. These processors can be used in combination with certified toolkits to heighten up the overall security of the platform (e.g., a Chip-and-PIN terminal). The main downside of this approach is that their level of security is sustained by a detriment in performance and functionality,

[2] http://www.indefenseofdata.com/data-breach-trends-stats/

[3] http://cphpost.dk/news/national/nation-œincreasingly-vulnerable-cyber-attack

making dedicated processors not eligible for the applications we are used to see in our phones or laptops. Examples of these dedicated, tamper-resistant processors are IBM CryptoCards[4]and ARM SecurCore Processors[5]. This class of solution might be appropriate for trusted cells, if the protection against lab attack is paramount.

The third solution (O3), where hardware is not tamper-resistant, still provides some level of hardware security together with a rich environment. For example, TrustZone is ARM's approach to bringing security and high performance together. It is tightly integrated into Cortex-A processors, making use of the AMBA AXI bus and specific TrustZone Intellectual Property (IP) blocks to extend throughout the system. This allows to secure peripherals such as memory, crypto blocks, keyboards or screen in runtime, impeding malicious software to intercept or alter communications and operations involving sensitive data. Even though ARM does not provide an implementation for TrustZone, they have worked together with Global Platform in defining a standard specification for trusted environments. The result is the Global Platform Trusted Execution Environment (TEE). From an architectural perspective, TrustZone can be conceived as a set of security extensions that enables a TEE running in parallel to a Rich Execution Environment (REE). High performance tasks are executed in the REE, while tasks that require an extra level of security are executed in the TEE. What is interesting about the REE - TEE separation is that it does not require a dedicated processor for the TEE. Each physical processor core provides two virtual cores, one considered Non-secure (Normal World) and the other Secure (Secure World), and a mechanism to context switch between them, known as the monitor mode. The implementation of the monitor mode relies on the so-called NS bit that is added to the bus transactions and to cache tags in the system. The NS bit is an addition to the AMBA3 AXI Advanced Peripheral Bus (APS), a peripheral bus that is attached to the system bus using an AXI-to-APB-bridge [2]. It is indirectly derived from the identity of the virtual core that performs a given instruction or memory access. Since the monitor is the most sensitive component of TrustZone, it is always handled by the Secure World. Also, since context switching is done by hardware, the overhead is minimal.

The main advantage we see in TrustZone is that it intrinsically supports security in high performance tasks involving sensitive data. Also, TrustZone being implemented in Cortex-A9 and Cortex-A15, which are the most popular processors for mobile platforms at the moment (e.g., Samsung Exynos and Nvidia Tegra series), gives us the advantage of developing for an already known and extensively deployed platform; not to mention the announced partnership between AMD and ARM that promises the incorporation of TrustZone-based processors in AMD chips[6] to be included in smartphones, set-top boxes and laptops.

[4] http://www-03.ibm.com/security/cryptocards/

[5] http://www.arm.com/products/processors/securcore/

[6] http://www.amd.com/us/press-releases/Pages/amd-strengthens-security-2012jun13.aspx

Finally, since TrustZone does not require a dedicated processor, it saves components, making it a cheap alternative to secure tokens.

TrustZone is not tamper-resistant, however. Lab attacks are out of the scope of the hardware protection provided by the TrustZone IP blocks and the AMBA AXI peripheral bus. This is not much of a problem for trusted cells, which constitute a decentralized data platform. Much more problematic is the secrecy surrounding it. TrustZone technology was first introduced in 2003 [5] and was officially presented in a press release by ARM in 2004[7]. Surprisingly though, 10 years later, TrustZone's market is still almost exclusive to Trusted Logic, Gemalto and Giesecke&Devrient (MobiCore), and it is difficult to find it mentioned in any research work besides the one by the Graz University of Technology[8]. All three world leading security companies monopolize the TrustZone market, driven by financial stakeholders such as Visa and MasterCard, by impeding 3rd parties - specially research oriented institutions - to make use of it by opposing to open implementations. The level of security provided by TrustZone is therefore virtually increased through obscurity: A lock unknown to locksmiths is world's most secure lock.

The lack of open APIs and libraries, in combination with Original Equipment Manufactures (OEM) such as Samsung or Nvidia, disabling the TrustZone security extensions for their development boards is a big impediment for the research community to get hold of the technology. At the time of this paper the only boards fully supporting the TrustZone technology are: Xilinx Zynq-7000 AP SoC ZC702, Nvidia Kayla DevKit (Tegra 3) and ARM Versatile Express. Our experience is that they are all subject to nondisclosure agreements (NDA), if TrustZone use is intended. Overcoming the closeness of TrustZone requires implementing, supporting and promoting standards, as well as making them available to an active, diverse community. This is the main contribution of this work. We are actively pushing for the expansion and distribution of an open source implementation of a TEE for the TrustZone security extensions. This involves bringing the parts together, dealing with licensing limitations and making source code and documentation accessible.

While this third option, based on TrustZone is attractive on paper, the secrecy surrounding it and the lack of open APIs and libraries is a huge barrier to its utilization in the context of trusted cells (or any other innovative data platform). We are working together with Xilinx and Sierraware to remove this barrier.

4 The ARM-Xilinx/OpenVirtualization Framework

Sierraware[9], an embedded virtualization company, developed Open Virtualization[10], which is the first open source alternative that leverages the security extensions present in ARM TrustZone. It is composed by (i) SierraVisor, a hypervisor

[7] http://www.arm.com/about/newsroom/5688.php
[8] http://www.iaik.tugraz.at
[9] http://www.sierraware.com
[10] http://www.openvirtualization.org

for ARM-based systems, and (ii) SierraTEE, a TEE for ARM TrustZone hardware security extensions. While the hypervisor is an interesting contribution, it is the TEE open source implementation that changes the game, since it opens the door for developers and researchers to using TrustZone. Open Virtualization's TEE implementation is compatible with Global Platform's TEE specification.

Open Virtualization was first released between 2011 and 2012. Since then, Sierraware has maintained an open source distribution of it. However, their focus has been in their commercial distribution. The main issue with this approach is the risk of contaminating one of the versions with the other. This is specially significant when it comes to IP blocks and other licensed software. As a consequence, releasing code for the open source version requires the overhead of it having to be audited by the different vendors, making the process tough and slow, and preventing code distribution via repositories. Also, maintaining a commercial product implies inevitably that publicly available documentation is limited and incomplete.

We are working together with Xilinx[11] and Sierraware to improve this situation. Our main contributions here are (i) facilitating the distribution of Open Virtualization by structuring it and making it available through a public service code repository, (ii) increasing the public knowledge of TrustZone by working with Xilinx in documenting the support of Open Virtualization for the Xilinx SoC ZC702, and as a consequence of this (iii) helping expanding the reach of TrustZone and Open Virtualization to the research community. The git repository containing Open Virtualization is available and can be used directly for the Xilinx ZC702 board[12]. We are at the moment working with Xilinx to create a wiki to complement the repository with documentation on both TrustZone and Open Virtualization. Our intention is to continue supporting this project and improve those components of Open Virtualization that are relevant to our research. Ideally, this would be the beginning of a community around Open Virtualization, where researchers and developers could contribute and build knowledge. We are also working towards making this git repository Sierraware's main vehicle for their open source distribution.

Let us take an example of how the ARM-Xilinx/OpenVirtualization framework could support Trusted Cells. Consider Servfos, a fictive company, that provides energy services based on data obtained from the smart sensors it is world renowned for. Their clients supply Servfos with data coming from the sensing infrastructures deployed in their buildings, and Servfos uses these data to give their clients all sorts of statistics and other information on their energy usage efficiency. This allows Servfos' clients to detect energy leakages and reduce their monthly expenses. Also, Servfos works with the local municipality in a contest for "The Greenest Firm in the Block". Those business that wish to participate agree on Servfos providing the municipality with a set of spatio-temporal aggregates on their energy consumption, which are then used for the contest and for some energy awareness games in different social networks. Finally, Servfos gives

[11] `http://www.xilinx.com`
[12] `https://github.com/javigon/OpenVirtualization`

their clients the possibility to participate in an European project that promotes energy efficiency in office buildings. While the concrete businesses the data is coming from is not relevant, it is required to access all data points from one random office in each participant's building for a whole month. For each service they provide, Servfos has a different contract with each of their clients, where all parties agree on who will access which data, and how it is going to be used. Servfos' clients are willing to participate in green initiatives, but they are concern about potential uncontrolled accesses to their sensitive data.

In order to support all the services that Servfos offers to their clients two things are needed: a policy model that is flexible enough as to allow the definition of complex access and usage policies (the contract), and an engine that enforces this contract. Derived from this, our specific challenges are: (i) defining an usage control model for sensor data in terms of UCON authorizations, obligations and conditions, (ii) identifying which components of a database system should be placed in a secure environment in order to enforce this usage control model, and finally (iii) providing an implementation for it. The ultimately goal is to have a framework (iv) that allows companies like Servfos to provide valuable services on top of sensor data belonging to individuals or organizations, in such a way that data owners can count on the access and usage control policies they define to be enforced. Such framework represents the data platform we have described in this paper and an implementation of the trusted cells.

Data flows between the trusted cell located in the client side and the one in the Servfos side. The trusted cells control the access to the sensor data according to the $UCON_{ABC}$ model. They guarantee that sensitive data is stored in secure memory and processed by a secure processor (TEE). High performance processes take place in a REE. To do so they make use of the TrustZone security extensions of the ARM Cortex-A9 processor powering the trusted cell. The software enabling the use of TrustZone is Open Virtualization: an open source implementation of Global Platform TEE's specification developed by Sierraware. The communication between the cells is secured by cryptographic keys that are stored in the Secure World.

5 Conclusion

In the process of designing and implementing a data platform for sensor data that allows data owners to share their data while retaining a form of access and usage control over it, we have encountered a challenge in finding suitable hardware to support it. The secrecy surrounding commercially available hardware platforms that leverage secure processing, and the lack of open implementations for them, limit their use, specially inside the research community. What is more, the industrial monopoly created around secure platforms such as TrustZone virtually increases their level of security through obscurity, representing a benefit for attackers that are smart enough to find unreported loopholes.

In this paper we present our efforts for changing this situation. We have worked together with Xilinx in bringing to the research community Open Virtualization: an open source implementation of a TrustZone TEE environment developed by Sierraware. The goal is that our efforts result in the establishment of a community around Open Virtualization and TrustZone, where researchers and developers can contribute and build knowledge. In the process of discussing the suitability of TrustZone for our research purposes, we have also given an overview of commercially available hardware platforms for trusted big data platforms.

Finally we have presented the roadmap for a data platform that supports the manipulation, analysis and sharing of large volumes of sensor data, while addressing the information security problem that this introduces. This data platform is the materialization of a trusted cell. It is the combination of a formal usage control model ($UCON_{ABC}$), a commercially available, extensively deployed hardware technology that leverages security at a low price (ARM TrustZone), and an open source implementation of Global Platform's TEE standard (Open Virtualization). This is a first step towards a decentralized data platform based on a TEE at the edges of the Internet that supports innovative services on sensor data.

References

1. Ancieaux, N., Bonnet, P., Bouganim, L., Nguyen, B., Popa, L.S., Pucheral, P.: Trusted cells: A sea change for personal data services. In: CIDR (2013)
2. Amba®, A.: axitm and acetm protocol specification. Technical report, ARM (2013)
3. Abrahan, D.G., Dolan, G.M., Double, G.P., Stevens, J.V.: Transaction security system. IBM Systems Journal 30(2), 206–229 (1991)
4. Gantz, J., Reinsel, D., Lee, R.: The digital universe in 2020: Big data, bigger digi tal shadows, and biggest growth in the far east. In: IDC (February 2013)
5. ImObersteg, G.: Arm trustzone extension delivers hardware security for next generation, opensystem, armpowered solutions. Intelligence 2, 6–12 (2003)
6. Katzenbeisser, S., Kursawe, K., Preneel, B., Sadeghi, A.-R.: Privacy and security in smart energy grids (dagstuhl seminar 11511). Dagstuhl Reports 1(12), 62–68 (2011)
7. Park, J., Sandhu, R.: The uconabc usage control model. ACM Trans. Inf. Syst. Secur. 7(1), 128–174 (2004)
8. IEEE Computer Society. Data engineering. Bulleting of the Technical Committee on Data Engineering 35(4) (2012)
9. ARM Security Technology. Buiding a secure system using trustzone technology. Technical report, ARM (2009)