Trusted Operations On Mobile Phones

Hassaan Janjua
KU Leuven
HassaanAbdulKhaliq.Janjua@cs.kuleuven.be

Wouter Joosen
KU Leuven
Wouter.Joosen@cs.kuleuven.be

Sam Michiels
KU Leuven
sam.michiels@cs.kuleuven.be

Danny Hughes
KU Leuven
Danny.Hughes@cs.kuleuven.be

ABSTRACT

The widespread use of mobile devices has allowed the development of participatory sensing systems that capture various types of data using the existing sensors on mobile devices in order to upload the data to cloud based services for later use. Gathering data from such sources requires a mechanism to establish trust on the sensor data. For example an application may require a proof of authenticity of sensor readings originating from anonymous sources. Establishment of trust on the sensor data has been addressed in the literature. However, in many cases this sensor data needs to be preprocessed on the device itself before being uploaded to the target server. This processing could include resizing of images, hiding identifiable faces and sensitive data in images, anonymization of GPS data etc. while ensuring the chain of trust from data capture to the delivery of data to the consumer. There is a need for a framework that provides a means to implement arbitrary operations to be performed on trusted sensor data while guaranteeing the authenticity of the data. This paper presents the design and implementation of a framework that allows the capture of trusted sensor data, the development of trusted operations on sensor data, and provides a mechanism for performing predefined trusted operations on the sensor data such that the chain of trust is maintained. Evaluation shows that the performance of the proposed system is reasonable and that the trust guarantees are strong.

CCS CONCEPTS

• **Security and privacy → Trust frameworks**;

KEYWORDS

Trusted Execution Environment (TEE), Rich Execution Environment (REE), TrustZone, Secure World, Normal World, Trusted Application, Authenticity, Fidelity

MobiQuitous 2017, November 7–10, 2017, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/10.1145/3144457.3144502

ACM Reference Format:
Hassaan Janjua, Wouter Joosen, Sam Michiels, and Danny Hughes. 2017. Trusted Operations On Mobile Phones. In *the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3144457.3144502

1 INTRODUCTION

The idea of participatory sensing using mobile phones has been around for more than a decade [5]. It was inspired by the huge number of mobile devices that were already present at that time. Now after nearly 11 years the situation has become even more favorable for participatory sensing with more than 4 billion mobile phones and more than 2 billion smart-phones being used across the world [26]. The huge number of readily available mobile sensing and computing devices provide enormous opportunities to deploy widespread low cost sensing infrastructure for collecting data from participants who would voluntarily provide sensor data or even provide feedback in exchange for services or rewards [12].

Participatory sensing provides many advantages over dedicated sensing infrastructure. Since mobile phones and their communication infrastructure are already present, the marginal extra cost of sensor infrastructure deployment is only limited to the development of the mobile application that will be used to collect and relay the sensor data to a central server. Since participatory sensing could potentially assimilate anyone using a mobile phone it has the capacity to support much larger systems. Participatory sensing also creates the opportunity to use people as qualitative sensors that collect user feedback and opinions where subjective data makes more sense than objective values, a practical example of such system is Google Local Guide program [12].

Participatory sensor data collection often requires that some sort of trust is ensured for the data as it is not feasible to trust all participants. Fabricating false sensor data e.g. image data is quite straightforward by using off the shelf tools such as Photoshop [21] and Pixelgarde [22]. Furthermore a malicious user may use malware to compromise a smart phone's software stack to inject false data into the participatory sensing application. In many cases participants can be motivated to provide false or repetitive data in order to gain benefits or to simply disrupt the system with malicious intent. A real world example of such an exploit is Google Local Guide which relies on volunteers to upload pictures of the places that they visited and offers rewards in exchange, a malicious user can upload unrelated pictures and claim rewards from the service. Another real world example is joyn [14] community. The business model of joyn revolves around participatory sensing of data by volunteers in exchange for rewards such as discount coupons at

various market places. The more data volunteers collect, the more rewards they could achieve. This provides a clear motivation to report false data in order to achieve rewards.

The question of trustworthiness of participatory data was first raised in 2009 by Dua et al. [7]. Since then, many approaches for adding trust to the sensor data have been introduced [4, 8, 9, 16, 19, 20, 24]. These approaches only address the problem of trustworthiness of raw sensor data, but these approaches do not provide a practical solution because raw sensor data is rarely used in real world scenario. Many times this data needs to be preprocessed on the device itself in order to make it usable. [9] addressed the need for preprocessing sensor data while maintaining trust. But this approach only covers image and voice data, and is limited to non-rooted Android systems.

In our approach, trust assurance is achieved by placing the root of trust in the Trusted Execution Environment (TEE). TEE ensures trust by exposing only a limited interface to the outside world, and as a result reduces the attack surface. This reduced attack surface is relatively easy to maintain, and chances of introducing vulnerabilities that could potentially compromise the security of the system are significantly reduced. On the other hand we have a contradictory requirement of performing operations on sensor data inside the TEE. Our solution to this problem is to introduce a generic framework that provides the common functions for verifying and signing the data, and to provide a unified mechanism to implement trusted operations; as a result this framework can be rigorously tested and verified independently. This approach is useful because all signing and verification code is implemented only once for all types of sensor data operations, thus reducing the attack surface, as well as the amount of code to be verified, as compared to implementing the signing and verification for each sensor. Our framework is capable of performing arbitrary operations on sensor data repeatedly while keeping track of the performed operations maintaining the chain of trust.

The key contribution of this paper is a practical framework that not only supports the validation of sensor data integrity but also allows arbitrary operations to be performed on the data and ensures the chain of trust. Our framework also supports binding two or more sensor data sources together in a trustworthy manner. Our framework attaches timestamps to each sensor reading. This time stamp is later used to bind multiple sensor data together while maintaining trust for example, binding of GPS data with an image to create a trusted geotagged image, binding of a sound clip with GPS data to prove location of the sound etc.

The remainder of the paper is organized as follows. Section 2 describes the background on Trusted Execution environment, ARM TrustZone technology, and the available Secure zone operating systems. In section III we provide related work in trusted sensors data, and approaches for maintenance of trust in modified sensor data. Section IV describes a real world scenario with use cases that implement our solution. In section V we define the threat model that our system addresses and defines the extent of capabilities of an adversary who is trying introduce fabricated sensor data as original. Than we provide the implementation and evaluation of our framework in sections VI and VII respectively. Finally in section VIII we provide our conclusion.

2 BACKGROUND

A Trusted Execution Environment (TEE) is an isolated secure environment that provides processing, memory and storage capabilities that are protected from the operating system, which is termed as the Rich Execution Environment (REE). TEE requires hardware support to provide protection from the REE. Many embedded hardware solutions are available for implementing TEE from various vendors. The hardware solution provided by ARM is TrustZone [3], Intel provides Software Guard Extensions (SGX) [13], AMD provides AMD Secure Processor [2]. A TEE operating system manages the security hardware and provides a means to communicate between the TEE and REE. Many TEE operating systems are available as open source as well as proprietary software such as OP-TEE by Linaro [18], Trustonic [27], Qualcomm QSEE, T6 by TrustKernel [6], SecuriTEE by Hansol Secure [25]. GlobalPlatform addresses this heterogeneity in TEE by providing a standardized technical specifications, which facilitates development on various hardware platforms.

2.1 ARM TrustZone

ARM TrustZone is a hardware based execution environment isolation technology. TrustZone uses the concept of secure and non-secure worlds that are hardware separated, with non-secure software blocked from accessing secure resources directly. This separation of secure and non-secure worlds extends beyond the processor to encompass memory, software, bus transactions, interrupts and peripherals within an SoC. Both secure and non secure worlds have their own user mode and privileged mode, while the switching between the secure and non secure worlds is done via a special mode called the monitor mode. The approach to security in TrustZone is that the rich operating system runs in the non-secure world while a small and secure operating system runs in the secure world. The solution proposed in this paper relies on hardware based execution environment isolation technology. This dependence does not hinder with widespread deployment of our proposed solution because the said hardware based isolation technology is alreayd present in the majority of smart phones currently available in the market.

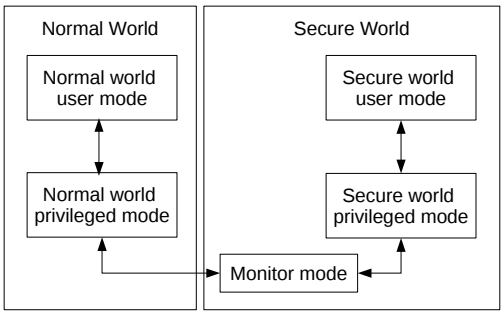

Figure 1: ARM TrustZone Secure and Normal Worlds

2.2 GlobalPlatform Trusted Execution Environment

GlobalPlatform is an international cross industry technical standardization body, it facilitates secure deployment and management of embedded applications. GlobalPlatform has standardized Trusted Execution Environment, which is a secure area in the main processor in an embedded device. TEE provides an isolated environment where sensitive data can be stored and processed. This isolated trusted environment is capable of providing end-to-end security by only allowing trusted and authenticated code to be executed in this environment. The TEE API specifications are independent of the underlying hardware implementation and provide a generalized set of APIs that can be used to implement hardware independent trusted applications. The framework presented in this paper utilizes the TEE internal APIs to implement a Trusted Application and TEE client APIs to communicate between the Secure and Normal worlds [17].

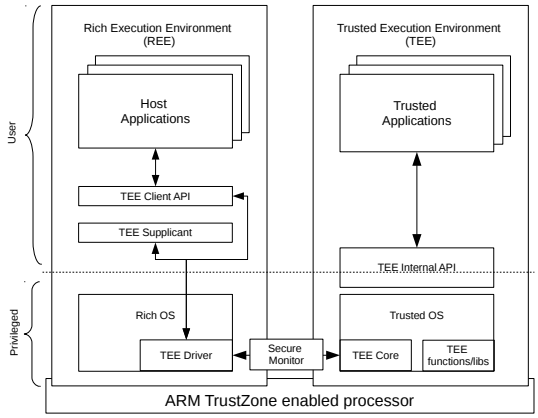

Figure 2: OP-TEE

2.3 Secure Operating System

A secure operating system runs in the secure world in parallel to the normal world operating system. It normally has a small footprint and a well defined communication mechanism with the normal world operating system. The smaller code base and limited communication interface reduces the attack surface of the system, making it difficult to compromise the security of the system. Global Platform has standardized TEE by providing API specifications for secure OS functionality. We have designed our framework to work around the Global Platform's TEE APIs so that the implementation is independent of any specific operating system.

OP-TEE is an open source secure operating system that complies with the GlobalPlatform TEE specifications. OP-TEE is ported to a number of TrustZone enabled ARM based platforms. OP-TEE is comprised of trusted operating system that runs in the secure world, normal world client side APIs and OP-TEE Linux drives. OP-TEE provides run time libraries and TEE internal APIs to support execution of Trusted Applications (TA) in secure world user mode. Normal world client side of OP-TEE provides TEE client APIs to supports communication with TA running in the secure world. The framework presented in this paper resides in the secure world where it is provided as a library that compiles with a trusted application. The framework implements TEE lifecycle events to provide an interface for the normal world.

3 RELATED WORK

The need for authenticating sensor data was quickly realized following the introduction of participatory sensing. The issue of attesting sensor data has been addressed using many approaches. One such approach is location proofs, which uses Wi-Fi access points to generate proof of position [23]. The relatively short range of the wireless infrastructure is used as a guarantee of location for the physical device. This technique is only possible due to the nature of location data, but it cannot be generalized to other sensors. Also this is not a strong proof of location against a capable adversary.

A more generalized approach is the use of dedicated Trusted Platform Module (TPM) to collect and authenticate sensor data. [7, 8] address the problem of establishing trust for the sensor data integrity using an external TPM. The approach proposes that an external TPM is used to sign the raw sensor data. Due to the dependence on the external hardware requirement, this solution is difficult to be deployed at a large scale.

Following the introduction of hardware based isolated environments such as ARM TrustZone [3] and Intel SGX/TXT [13] new approaches were proposed that utilized these hardware based isolated environments to add fidelity certificates to sensor data. The work done in [19] uses TrustZone to ensure the trustworthiness of the mobile sensor data. [19] introduces software abstractions for establishing trust and confidentiality of the sensor data. The framework conceived in this paper ensures trustworthiness of sensor data but does not address the usage of this data. Although the issue of operations on trusted sensor data is discussed in the context of providing differential privacy, the framework lacks support for generic operations on sensor data while maintaining trustworthiness. The proposed framework is an extension to this work in particular where we propose a comprehensive framework to not only acquire trusted sensor data but also provide mechanism to ensure trust after performing operations on that data.

[20] Provides a practical implementation of trusted operations on images. The framework uses the License verification library provided by Google [11] to validate the authenticity of the application itself, and implements Android keystore features provided on top of ARM TrustZone to store keys and perform signing operations. Although this technique utilizes the ARM TrustZone, it fails to exploit the full potential of the trusted execution environment, as a consequence it is vulnerable to attackers that have gained root privileges on the system. Malware installed on the system can hook the image capturing APIs and feed modified images to the system. In order to protect the system from such a threat model, we need to move the origin of data inside the TEE.

Executing arbitrary operations inside the TEE adds additional cost of certifying more code for the TEE and increases the attack surface for a potential adversary. We believe however, that including this code in TEE has more benefits than the risk it adds. [4] shows that it is feasible to implement image operations inside the secure

world of ARM TrustZone, but it utilizes this ARM TrustZone at the cloud server in order to provide isolated environment to perform image operations. Our work is the inverse of that, we propose the usage of TEE at the mobile client end to provide trust guarantees.

The work presented in [9] addresses the same problem for images and sound data as in the current paper but it implements a different approach and a lenient threat model which assumes that the sensor data cannot be modified at the device driver level. The approach taken by [9] is that it allows arbitrary modifications on the sensor data by any means, and later compares the modified data with the original and generates a signature after checking for meaning-preserving modifications. The weak link here is the mechanism of capturing full fidelity sensor data that relies on the Android security system. If we allow a more realistic threat model where the normal world operating system is compromisable, we cannot rely on the fidelity of the sensor data that was originally assumed to be trusted. Furthermore [9] only addresses the operations performed on image and sound data, but lacks generalized operations for all types of data. Although this technique proposes utilization of TrustZone in order to perform image operations, it does this to provide protection from malicious access rather than providing trust. Whereas the current paper proposes a scheme that provides trust guarantees even when the normal world operating system like Android has been completely compromised. This is achieved by maintaining the root of trust in the TEE.

4 SCENARIO

The paper is motivated by a real-world industrial project for Mobile Sensing seRvices for developing Geospatial IoT Applications (SeRGIo). SeRGIo aims to transform the role of a reliable mobile workforce (e.g. courier workers) into an intelligent human sensor network supported by an enabling mobile sensing framework for better understanding of citizens and cities.

Many service providers require their workers to move around a city on a daily basis for example courier workers, door to door salesmen, taxi drives etc. There is a huge potential of using this workforce in the field to collect a vast amount of data as well as to execute small activities such as taking pictures and collecting noise samples from around the city. Such service providers have potential to enter the market as a provider of citizen and city data for a variety of stakeholders.

These services can task their workforce to collect this data using sensors available on smart phones. This data will become much more valuable if guarantees for security, authenticity and trustworthiness of this data are provided to the customers. In order to realize these objectives we identified the following requirements that need to be fulfilled.

(1) Make sure that a given sensor data is indeed taken from the sensor at the specified time (Trust)
(2) Keep track of the operations performed on the sensor data and provide guarantees that only the said operations have been performed on the data.
(3) Merge trusted sensor data from multiple sensors to provide a derived data along with its own trust signature.

Since we utilize TEE as our platform to provide our solution, this choice imposes another requirement of small footprint and smaller attack surface. Additionally we do not want too much degradation in performance by providing trust guarantees. This leads us to the following requirements.

(1) Make sure that the overall footprint of the security code is minimal.
(2) Make sure that the trust guarantees does not introduce unacceptable performance overhead.

We fulfill these requirements by providing a framework to help implement capturing of the sensor data, calculate fidelity signatures, and implement operations on the sensor data while maintaining trust guarantees. This framework will abstract over the implementation details of calculating and verifying the trust signatures from the developer of these operations.

5 THREAT MODEL

We present the security claims of our framework with the help of multiple attacker models each having increasing control on the device. The goal of the adversary here is to inject fabricated sensor data into our system without being detected. On the other hand, our goal is to detect the fabricated data. Our system relies on the root of trust provided by the Trusted Execution Environment and the ability of the TEE to protect secret keys. This means that the system can be compromised if the security of TEE is breached or the secret key is stolen by means of a side channel attack. This kind of attacks are out of scope of this paper. Side channel attack is an open problem and is a topic of ongoing research with exploits and solutions emerging continuously. [28] [1]

5.1 Controlled Devices

We first analyze our system when an adversary is trying to falsify sensor data on a controlled device such as a device that is provided to the service provider employees and it is closely monitored. In this case the adversary only has control over the user mode application since any attempt to root an android device or to jail break an iOS device will be detected by the administrator. In this case the adversary has no control over the sensors and cannot introduce fabricated sensor data at the time of data acquisition. However the adversary can play a man in the middle attack by placing itself between the application and the back-end server, but in this case the server will be able to detect falsified data based on the signatures.

5.2 Bring Your Own Device (BYOD)

It is easy to acquire root privileges in Android or jailbreak an iOS device either using a malware or by users on purpose. We assume that in this case the adversary has full control over the normal world operating system, but does not have access to the TEE. In this scenario, the adversary again can only temper with the sensor data but does not have access to the secret keys and thus is unable to bypass the attestation mechanism.

6 ARCHITECTURE

The Trusted Operations framework is based on the GlobalPlatform TEE APIs [10]. GlobalPlatform Trusted Application specifications require each TA to implement a number of functions called the TA Interface. These functions are invoked by the TEE to notify events

during the lifecycle of a TA [10]. The Trusted Operations library implements the TA Interface and provides a single entry point as TO_Entry() to the TA. The TA can than register for the TA lifecycle events from within the TO_Entry function using event register functions provided by the Trusted Operations library. The sensor data operations are also registered with the framework as callbacks, these operations are called by the framework after verification of the sensor data, and the updated data is signed after performing the operation. Data acquisition from a sensor is implemented as an operation by the TA and the framework takes care of signing the original data before returning it to the REE. Trusted Operation framework is integrated with the TA in the form of a static library and all the operations and the lifecycle callbacks are statically bound with the TA. As shown in Figure 3, the Trusted Operations library is statically linked with the Trusted application while the operations that are to be performed on the sensor data are implemented in the TA independent of the trust establishment operations.

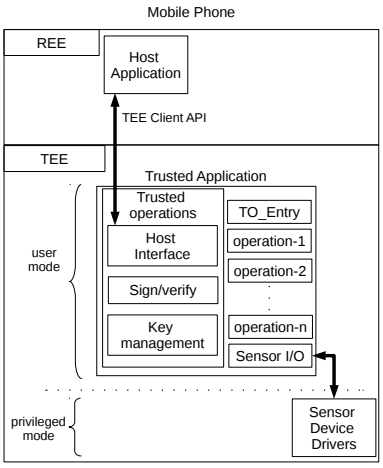

Figure 3: Trusted Operations Library

Trusted Operations are invoked from the REE using the GlobalPlatform TEE client API. In order to keep the root of trust in the TEE, the sensor data is always acquired using a trusted operation call. The Trusted Operation framework handles the signing of the data and attaches the Trust header to the sensor data. The trust header contains a cryptographic signature that is generated using a device specific private key, a list of operations that have been performed on the sensor data and a timestamp to track the sensor data acquisition time. The timestamp is based on the TEE-controlled real-time clock. This ensures that the TrustedData timestamp is independent of the REE time.

Initialization of asymmetric cryptographic keys and and distribution of the public key is an important step in the trust establishment. We suggest that the key initialization should be handled by an administer that has physical access to the device. The asymmetric keys are generated within the TEE and the public key is exported along with an identifier of the device. This public key is later to be used for the verification of sensor data that has originated from this device, while the private key is stored in secure storage within the device.

The TrustedOperations framework is responsible for signing the sensor data at data acquisition as well as after each operation that updated the sensor data. The framework also verifies the sensor data before performing the designated operation. Figure 4 illustrates the sequence of events that take place in the lifecycle of trusted data.

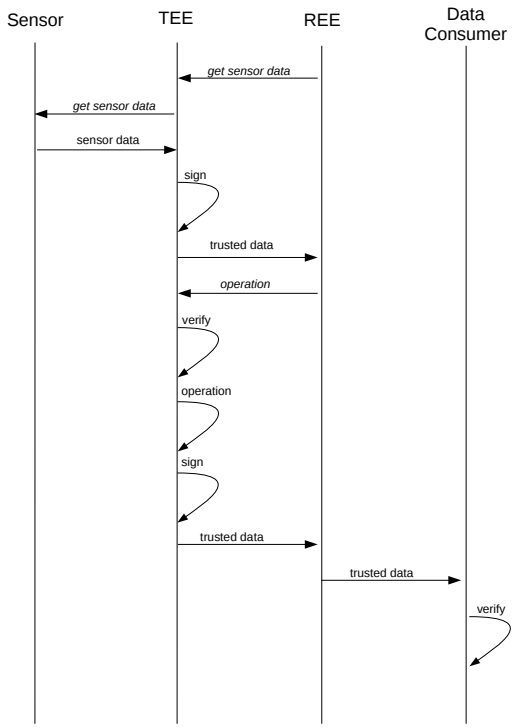

Figure 4: Trusted Sensor Data Life Cycle

7 IMPLEMENTATION

We implemented the Trusted Operations framework proof of concept using OP-TEE running on Raspberry Pi 3. OP-TEE is open source and implements Global Platform TEE and has been ported to many platforms suitable for academic and research purposes, this makes it an ideal choice for implementing the proof of concept Trusted Operations framework. The Raspberry Pi 3 has an ARM Cortex-A53 as its main processor which implements Arm TrustZone [3]. Raspberry Pi 3 processor provides ARM TrustZone exception states which are sufficient for implementing a functional proof of concept TEE and trusted applications. However, Raspberry Pi 3 lacks secure boot, memory, peripherals or other secure functions required by secure storage. Due to these limitations a system implemented on Raspberry Pi 3 is only suitable for non-commercial applications. We used the TEE development setup created by Sequitur Labs [15] for the prototype implementation of the Trusted operations framework.

OP-TEE supports statically linked as well as dynamically loadable Trusted Application (TA). We implemented the POC using dynamically loadable ELF. The client side of the framework utilizes the GlobalPlatform APIs directly. The only requirement from the client side is that the input and output data is preceded by the Trust header as shown in Figure 5.

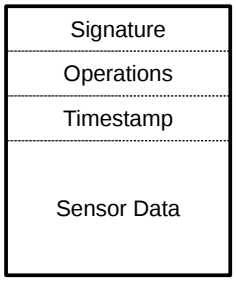

Figure 5: Trust Header

The TA execution starts with the TO_Entry(). This function is called at the TA creation time. The TA can register TA lifecycle callbacks as well as the Trusted Operations from within TO_Entry(). In order to automate the signature generation and verification the position of the input and out parameters is fixed by the framework. The first two parameters are reserved for input data, this data always has the Trust Header with a valid signature, the third parameter is optional and can be of any type while the fourth parameter is reserved for the output. The output also has the Trust Header attached. The trusted operation function has the following signature.

```
TEE_Result to_trusted_operation(
    void        *sess_ctx,
    uint32_t    cmd_id,
    uint32_t    param_types,
    TEE_Param   params[4]);
```

The first parameter sess_ctx is the pointer to the session context that was created at the time of session creation by the TA, cmd_id is the code against which the trusted operation was registered, param_types contains the types of the four parameters encoded in a 32 bit word, lastly params are the parameter for the trusted operation. Sensor data should be passed to the operation using the first and second parameter and must contain the Trust Header, when used as input parameter the type of the parameter must be 'TEE_PARAM_TYPE_MEMREF_INPUT' the third parameter is optional and can have any type while the last parameter is used for output of the operation and must be of type 'TEE_PARAM_TYPE_MEMREF_INOUT' when used as operation. The restriction on the types allow the framework to verify the input data before the operation and to sign the data after the operation.

The trusted operations framework uses the TEE API to calculate the signature and verify the trusted data. The signature is generated using the TEE internal API function TEE_AsymmetricSignDigest, conversely the sensor data is verified using the TEE internal API function TEE_AsymmetricVerifyDigest. The algorithm and size of has is left up to the used of the framework to decide.

8 EVALUATION

We used our framework to implement trusted operations for camera input. The trusted operations are taken from a real world scenario where an employee takes pictures of a designated area and sends them to a data server. The data server first verifies that the pictures were taken on the specified place, at the specified time and the fidelity of the picture is verifiable after any transformations have been applied on the image such resize, or encoding in a compressed format.

We implemented trusted operations for images that used trusted operations framework to show the effectiveness of the framework in segregating the trust establishment logic from the actual implementation of the operations.

We ported the stb image library [https://github.com/nothings/stb] and the TinyJPEG [https://github.com/serge-rgb/TinyJPEG] for OP-TEE to implement the image operations. Since the OP-TEE does not provide a full set of libc and libm libraries, we also ported a subset of the newlib libm in order to provide math functions for the image processing library. [https://sourceware.org/newlib/libm.html].

We simulated image capture from camera by reading the image from a file. The capture image operation reads the image and passes it to the REE using trusted operations framework. Before passing the image to the client the framework attaches a signature with the image data along with capture time and an empty operations list. This signature is later used to verify the fidelity of the data. Second operations is the Resizing the image, the operation is implemented using the stb image processing library. Encode and decode operations are implemented using the TinyJPEG library. All operations are implemented independent of the fidelity checking. The trusted operations framework performs signature verification before the operation and signature generation after calling the actual operation.

To evaluate our system, we implemented a chain of three operations on image data. The input is simulated using an image fie containing raw RGB data. The first operation is an image capturing operation that reads the image data and returns it back to the client. Than we performed a chain of operations that represents a natural flow of operations that a typical street surveillance system would perform. The system captures the image, resizes it to a smaller size and than encodes it to jpeg. This image along with its trust header will be sent to the data server which will first verify its fidelity and than store it in its database.

We computed the latency of calculating the hash, verifying and signing the data, using various sizes of data. As shown in Figure 6 the system incurs a constant overhead of 44 milliseconds for signing and 33 milliseconds for verification until 10KB and increases linearly with the size of data afterwards. Evaluation results in [19] show that the average time of a sensor reading on an ARM TrustZone takes more than 80 milliseconds. The time delay incurred by sensor read is greater than the time it takes to sign the sensor data therefore we conclude that fidelity validation is feasible at sensor data acquisition.

An other interesting observation is that the time it takes in context switch from REE to TEE and back is negligible as compared to the time it takes to perform a typical image operation. In ARM

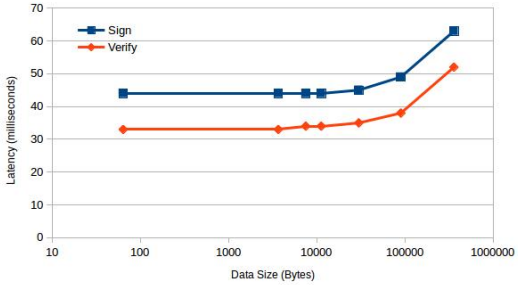

Figure 6: Sign/Verify Latency

TrustZone both secure and normal worlds share the same processing resources, the context switch cost is the only overhead that is incurred by performing these operations in the TEE.

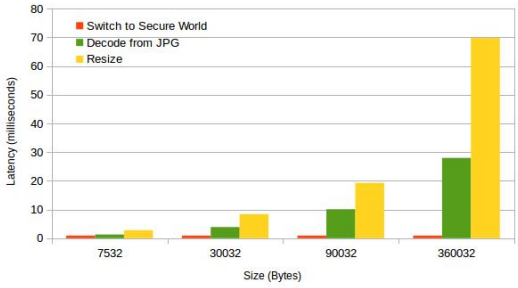

Figure 7: Operations vs Context Switch

Sensor data capture and the operations performed on the sensor data are typically triggered as a result of user interaction. In this scenario the overhead introduced by Trusted Operations is the combined cost of context switch, verification and signing. This overhead is less than 120 milliseconds for a data of size 300KB. We argue that this overhead is an acceptable delay in this scenario.

9 FUTURE WORK

We plan to extend the capability of the framework to capture trustworthy data from external sensors that are connected to a mobile phone and to perform trusted operations on the external data in the same manner. Our research will be focused on medical devices that are attached to a mobile phone and the trusted operations framework will be used to perform operations on the collected data. This paper suggests that the key generation and public key transportation is performed in a controlled environment where an administrator manually triggers key generation and extract the public key in order to register the device. An other approach is to use a third party certification authority to provide digital certificates. In order to avoid manual key exchange or third party certification authority we plan to implement public key distribution infrastructure based on distributed block-chain to create a network of trusted mobile devices. This key distribution mechanism will eliminate the need of a centralized key management entity and will enable installation of the trusted operations framework independent of a trusted administrative body.

10 CONCLUSION

This paper presents a framework that facilitates implementation of Trusted Operations on sensor data. The framework abstracts out the implementation details of signing and verifying sensor data and provides a clean way of implementing arbitrary operations that are to be performed on the sensor data, while the sensor data acquisition, signature generation and verification are handled by the framework itself. This approach factors out the common code for performing trusted operations, thus reduces the footprint of the overall code and provides opportunity to religiously test and verify the common code independent of the actual operations implementation. In the end the paper presents the performance evaluation of the framework and its example usage for image operations implemented in OP-TEE running on a Raspberry Pi 3.

11 ACKNOWLEDGMENT

This research is partially funded by the research fund KU Leuven, the ICON-SeRGIo project and the HI^2-DT/Universal4Mobile project.

REFERENCES

[1] Na-Young Ahn and Dong Hoon Lee. 2017. Countermeasure against Side-Channel Attack in Shared Memory of TrustZone. *arXiv preprint arXiv:1705.08279* (2017).
[2] AMD. 2017. AMD Secure Technology. (2017). http://www.amd.com/en-us/innovations/software-technologies/security
[3] ARM. 2017. ARM TrustZone. (2017). https://www.arm.com/products/security-on-arm/trustzone
[4] Tiago Brito, Nuno O Duarte, and Nuno Santos. 2016. ARM TrustZone for Secure Image Processing on the Cloud. In *Reliable Distributed Systems Workshops (SRDSW), 2016 IEEE 35th Symposium on*. IEEE, 37–42.
[5] Jeffrey A Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya Ramanathan, Sasank Reddy, and Mani B Srivastava. 2006. Participatory sensing. *Center for Embedded Network Sensing* (2006).
[6] Shanghai Pingbo Information Technology Limited Company. 2017. Trustkernel. (2017). https://www.trustkernel.com/en/
[7] Akshay Dua, Nirupama Bulusu, Wu-Chang Feng, and Wen Hu. 2009. Towards trustworthy participatory sensing. In *Proceedings of the 4th USENIX conference on Hot topics in security*. USENIX Association, 8–8.
[8] Peter Gilbert, Landon P Cox, Jaeyeon Jung, and David Wetherall. 2010. Toward trustworthy mobile sensing. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*. ACM, 31–36.
[9] Peter Gilbert, Jaeyeon Jung, Kyungmin Lee, Henry Qin, Daniel Sharkey, Anmol Sheth, and Landon P Cox. 2011. Youprove: authenticity and fidelity in mobile sensing. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 176–189.
[10] GlobalPlatform. 2017. GlobalPlatform Trusted Execution Environment (TEE). (2017). https://www.globalplatform.org/specificationsdevice.asp
[11] Google. 2017. Google Application Licensing Overview. (2017). http://developer.android.com/google/play/licensing/overview.html
[12] Google. 2017. Google Local Guides. (2017). https://maps.google.com/localguides
[13] Intel. 2017. Intel SGX. (2017). https://software.intel.com/en-us/sgx
[14] joyn. 2017. joyn. (2017). https://www.joyn.be/en
[15] Sequitur Labs. 2017. Raspberry Pi. (2017). https://www.sequiturlabs.com/media_portfolio/sequitur-labs-collaborates-with-linaro-to-lower-barriers-to-iot-security-education-for-raspberry-p
[16] Wenhao Li, Haibo Li, Haibo Chen, and Yubin Xia. 2015. Adattester: Secure online mobile advertisement attestation using trustzone. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 75–88.
[17] Linaro. 2017. OP-TEE - Documentation. (2017). https://open-tee.github.io/documentation/
[18] Linaro. 2017. OP-TEE - Linaro Wiki. (2017). https://wiki.linaro.org/WorkingGroups/Security/OP-TEE
[19] He Liu, Stefan Saroiu, Alec Wolman, and Himanshu Raj. 2012. Software abstractions for trusted sensors. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 365–378.
[20] Kwstantinos Papadamou, Riginos Samaras, and Michael Sirivianos. 2016. Ensuring the Authenticity and Fidelity of Captured Photos Using Trusted Execution

and Mobile Application Licensing Capabilities. In *Availability, Reliability and Security (ARES), 2016 11th International Conference on*. IEEE, 706–714.
[21] Photoshop. 2017. Photoshop. (2017). http://www.photoshop.com/
[22] Pixelgarde. 2017. pixelgarde. (2017). http://pixelgarde.com/
[23] Stefan Saroiu and Alec Wolman. 2009. Enabling new mobile applications with location proofs. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*. ACM, 3.
[24] Stefan Saroiu and Alec Wolman. 2010. I am a sensor, and I approve this message. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*. ACM, 37–42.
[25] Hansol Secure. 2017. SecuriTEE. (2017). http://www.sola-cia.com/en/securiTee/product.asp
[26] Statista. 2017. Number of smartphone users worldwide. (2017). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
[27] Trustonic. 2017. Trustonic. (2017). https://www.trustonic.com/
[28] Ning Zhang, Kun Sun, Deborah Shands, Wenjing Lou, and Y Thomas Hou. 2016. TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices. *IACR Cryptology ePrint Archive* 2016 (2016), 980.