Make Remote Forensic Investigations Forensic Again: Increasing the Evidential Value of Remote Forensic Investigations

Marcel Busch, Florian Nicolai, Fabian Fleischer, Christian Rückert,
Christoph Safferling, and Felix Freiling

Friedrich-Alexander University Erlangen-Nuremberg, Erlangen, Germany
{marcel.busch,florian.nicolai,fabian.fleischer,
christian.rueckert,christoph.safferling,felix.freiling}@fau.de

Abstract. Due to the increasing use of encrypted communication and anonymous services, many countries introduced new regulations that allow law enforcement to perform *remote forensic investigations*. During such investigations, law enforcement agencies secretly obtain remote access to a suspect's computer to search for and collect evidence, including full copies of the (unencrypted) communication data. In this paper, we argue that the evidential value of the acquired evidence can be substantially increased by two technical methods: (1) employing integrity verification techniques offered by secure hardware, and (2) exfiltrating the decryption key of encrypted communication only in order to decrypt communication obtained by lawful interception. To prove the practicality of both methods, we design and implement TEE-BI, a solution for Trusted Execution Environment-based introspection. We deploy TEE-BI on an Android-based hardware platform featuring an ARM TrustZone and demonstrate the stealthy extraction of Secure Sockets Layer encryption keys from an Android userland application. We evaluate the effectiveness, performance, and compatibility of our prototype and argue that it provides a much higher level of evidential value than (the known) existing remote forensic software systems.

Keywords: Remote Forensic Investigation · ARM TrustZone · Principle of Proportionality · Evidential Value · Translation Table Introspection · Android.

1 Introduction

Due to end-to-end encrypted connections for many Internet services (*i.e.,* web browsing or instant messaging) as well as strong anonymization provided by services like Tor, law enforcement is struggling to prevent "going dark" on cybercrime. One general approach to counter the increasing abilities of cybercriminals is to perform *remote forensic investigations* [55]. In many countries, this term is used to describe the practice of using hacking techniques to access and install government spyware on the end device of a suspect [22]. Given a government spyware on the target system, most investigative tasks are substantially simplified. For example, by bypassing anonymization services, law enforcement can identify the IP address and, thereby, often the location of a target machine. In a further practice, known as *remote forensic search*, law enforcement

uses spyware to covertly search and acquire evidence directly from the system. Moreover, encrypted communication can be intercepted *before* it is encrypted, then stored and exfiltrated to law enforcement. In analogy to remote forensic searches, we call this technique *remote forensic telecommunication surveillance*. While law enforcement agencies do not publicly report a lot about these practices, those cases of government spyware that have been publicized [20,54] confirm the above observations.

There are several downsides with the current practices. Firstly, it is a severe practical problem to get government spyware onto the target device without the device owner noticing. In practice, this is often achieved by social engineering (*i.e.,* tricking the user into installing the spyware under false presumptions) [39] or physical access to the device in combination with a root exploit or knowledge of access credentials [20]. Obviously, both approaches are not generally applicable, especially if the device owner is a cybercriminal fully aware of these tactics. In addition to the practical problems, these methods pose significant risks to the security of the Internet (*i.e.,* reducing the functions of computers by bypassing security techniques or the proliferation of zero-day exploits).

The second downside is a general problem with the evidential value of the data obtained through such remote forensic searches. The legal dangers do not necessarily lie in the expressiveness of the data obtained but rather in the unreliability of the method since a defendant can always claim that law enforcement "manipulated the computer" or "planted" the evidence. This claim is hard to refute since it is difficult to prove which version of the software actually was running on the remote computer at a particular point in time and whether it did exactly what it should have done.

The third problem of remote forensic investigations pertains to using them for surveillance of encrypted communication. Since the current technique of remote forensic telecommunication surveillance described above must acquire the content of the communication *before* it is encrypted, there is a small window of uncertainty of whether and when the encrypted communication is sent over the network. This is especially relevant in countries such as Germany, that have introduced regulations that may use remote forensic software for telecommunication surveillance as long as the police has access to only the content of the communication that is finally sent over the physical wire (often called *source wiretapping* or *source telecommunication surveillance* [23,47]). Given the above uncertainties, government spyware is inclined to overapproximate communication data and thereby fails to comply with legal requirements.

To state it clearly and upfront: This paper does not focus on the first problem, *i.e.,* it is *not* about how to secretly inject spyware onto mobile phones. We simply assume that governments can enforce the addition of some functionality on a mobile device, *e.g.,* by cooperating with the manufacturer of the device. Instead, we are concerned with the second and third problem, *i.e.,* the question of *how to design tools for remote forensic investigations that comply with general legal principles*.

1.1 Contributions

In this paper, we study design variants of methods for remote forensic investigations. As mentioned above, we do *not* focus on how to install the spyware on the phone but rather assume a government/manufacturer collaborative approach for targeted communication

interception that inserts government software into the general trusted computing base of mobile devices. Our contributions are as follows:

1. We argue that remote forensic software should be embedded into the trusted computing base of end devices that support guarantees rooted in secure hardware. Such techniques exist in many contexts in the form of *Trusted Execution Environments (TEEs)*, most notably with *ARM TrustZone (TZ)* on many mobile devices. Having a hardware-based root of trust makes it much easier to argue that the software was running as specified.
2. We discuss a novel design variant of remote forensic telecommunication surveillance: The central idea of our solution is to exfiltrate only the minimal amount of information from the end device – the cryptographic key – to open the cryptographic shield "on the wire" and to proceed with (classical) wiretapping. We argue that this technique is legally the least intrusive method for remote forensic telecommunication surveillance of encrypted traffic and therefore is the only method that satisfies general legal principles.
3. We design and implement TEE-BI, a TEE-based introspection framework for Android devices that (1) uses ARM TZ to guarantee and attest the integrity of a software, and that (2) can reliably and stealthily extract encryption keys from running applications to allow data to be decrypted in transit.
4. To show TEE-BI's effectiveness, we evaluate our prototype by extracting SSL crypto material from an SSL encrypted echo server on a real hardware platform using a recent Android version (Android 9). The chosen example is universal since it makes use of the `boringssl` library which is the default crypto library used by applications on the Android platform.

1.2 Roadmap

We begin with a discussion of the legal view on remote forensic investigations in Section 2 and give some background on the used technologies in Section 3. The design and implementation of our solution are described in Sections 4 and 5, followed by the evaluation in Section 6. We discuss related work in Section 7 and then conclude in Section 8.

2 Legal Perspective

Like all investigative measures, remote forensic investigations also require regulation. We first give an overview of international legislation regulating the use of government spyware and then discuss the legal requirements that must be met by investigative techniques. We will later use these requirements to evaluate and compare our solution.

2.1 Legislation Allowing Remote Forensic Investigations

Many countries now have established legal provisions to install spyware on end devices of citizens and therefore conduct remote forensic investigations. A recent overview of

the European Union [22] on the practices in several countries concludes that many countries, including most member states of the EU, now explicitly permit and regulate "hacking by law enforcement" [22], i.e., remote access to computers by spyware as a criminal investigative measure. In Germany, the Federal Code of Criminal Procedure was amended in this way just recently. [23]. In many other countries, remote forensic investigations are carried out within the bounds of existing laws which can be problematic [18].

For example, in the UK, the "Investigatory Powers Bill" enacted in 2016 allows law enforcement to "interfere" with targeted electronic equipment to retrieve data and communication [22, p. 104], and in the United States an amendment to Rule 41 of the Federal Criminal Procedure Rule offers law enforcement "remote access" to data [22, p. 121]. A new regulation in Germany even distinguishes between two forms of remote forensic investigation: a general "remote forensic search" ("Online-Durchsuchung") and a more specific "remote forensic telecommunication surveillance" ("Quellen-TKÜ") [23,47]. Roughly speaking, the latter is a restriction of the former case, but only to telecommunication data, *i.e.,* data that could be intercepted via wiretapping if encryption would be turned off.

2.2 Principle of Proportionality (Europe)

The proportionality principle is common to most domestic legal systems in Europe and is indeed a central figure of EU law itself and routinely applied by the European Court of Human Rights [17, p. 10-14] [15]. It is also an integral part of other legal systems like Canada [9] or Hongkong [11]. If authorities infringe upon a citizen's fundamental right, the infringement must not be immoderate in relation to the aim pursued by the government. The *proportionality test* consists of the following steps [4] [17, p. 16-17]: At first, an actual infringement of rights by a governmental act has to be stated, which is given for remote forensic telecommunication surveillance. The infringed right is the right to freedom of telecommunication. It is granted in national law, *e.g.,* Art. 10 Basic Law for the Federal Republic of Germany, but also in supranational law as in Art. 8 ECHR (European Convention of Human Rights) and Art. 7 CFR (Charta of Fundamental Rights of the European Union) [43]. Therein any form of undisclosed communication between natural and legal persons is protected from intervention by any government authority [30]. Subsequently, we have to examine if the intrusion stands in proportion to the governmental interest, in this case the interest in effective criminal investigation and prosecution, which is decided by the following: (1) The governmental aim has to be legitimate, which applies to criminal investigation and prosecution, and therefore to gaining access to telecommunication data (legitimacy). (2) The infringement must be suitable to pursue that legitimate aim (suitability). Remote forensic telecommunication surveillance can help to pursue that aim by gaining information for the investigation through telecommunication data. (3) The chosen measure has to be the least intrusive on individual rights amongst the equally suitable (necessity). We will argue later that due to the above stated smaller infringement, given a widespread possibility of application for our novel TEE-BI technique, the currently used method would be considered as not necessary in the sense of proportionality, and therefore is not justified. (4) Moreover, the infringement must be proportional to the government's gain of advantage for

the investigation (proportionality strictu sensu). This is a specific consideration for every individual case, which cannot be generalized. In our case, this step would not be taken, since the current method, given TEE-BI, already fails on the level of step 3, i.e. the infringement is already stated as not legitimate.

2.3 Balancing Test (US)

The American legal system also compares the level of infringement to the importance of the governmental aim, not by applying the principle of proportionality, but by performing the *balancing test*. The balancing test is divided into two steps: After stating the infringement, it is directly examined for being properly balanced to the governmental interest [17, p. 17-18]. The freedom of (digital) communication is guaranteed by the Fourth Amendment [27], which is infringed by remote forensic telecommunication surveillance. This infringement must be in balance to the governmental interest, therefore reasonable. While the Supreme Court stated that being reasonable is not invariably bound to being the least intrusive method [8,48], a more intrusive way of performing investigative actions is not immune to being declared unreasonable, thus unbalanced, in view of a less intrusive method. All of these aspects taken into consideration during the proportionality test, e.g., necessity, can, even if not explicitly named as such, be part of the balancing test [17, p. 17-23]. Thus, the current method of remote forensic telecommunication surveillance is not immune to being regarded as unjustified. In addition, it is, in any case, desirable to apply methods that are less intrusive into fundamental rights such as the Fourth Amendment.

2.4 Requirements for Digital Evidence

The value of evidence in criminal proceedings is determined by a set of general requirements that have been discussed widely in the literature. Such requirements have to be also met for digital evidence to have (stronger) evidential value. We now list the three most important ones for the present work [24]:

- Authenticity: The investigated data must be authentic to be of value for a criminal procedure. That applies to all electronically stored information involved [5]. The analysis, procedure, outcomes, and limitations must be explainable in a way, that the evidential value of the data can be assessed during the trial [40], so it can be assured the data was not altered.
- Reliability: Digital evidence has to be based on accurate and scientifically verified technologies, which have to be reliable themselves to make sure the evidence can be part of the (free) consideration of evidence and, in pre-trial stages, to establish a necessary degree of suspicion (e.g. US: “reasonable suspicion” (Terry vs. Ohio, 392 U.S. 1, 1968); Germany: “sufficient factual indication” (§152 (2) GER-CCP)) [24, p. 5].
- Verifiability: To ensure that the evidence acquisition can be followed up by the parties of the criminal procedure, the measure of collecting the data and information has to be repeatable and reproducible to prevent an extreme decrease of evidential

value. With due regard to recommendations of the US National Institute of Standards and Technology (NIST) there were already established further definitions of those criteria and how they can be met [38,24]

We will use these requirements later in the evaluation to compare TEE-BI's approach to current practices in remote forensic investigations.

3 ARMv8-A TrustZone Preliminaries

ARM chipsets are dominant on modern mobile devices, which usually incorporate the ARMv8 Instruction Set Architecture (ISA) [12] and support for ARM TrustZone [13]. A typical software architecture based on these hardware features consists of the logical components depicted in Figure 1. This architecture is achieved by partitioning the System-on-Chip (SoC)'s hardware components, especially the CPU and the RAM, and assigning them to either of two states, the Normal World (NW) or the Secure World (SW), as illustrated in Figure 2.

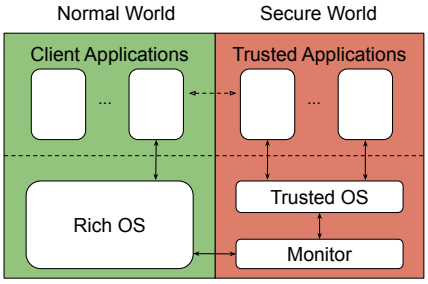

Fig. 1: ARMv8-A systems with ARM TrustZone support split the software architecture into a Normal World and a Secure World execution context.

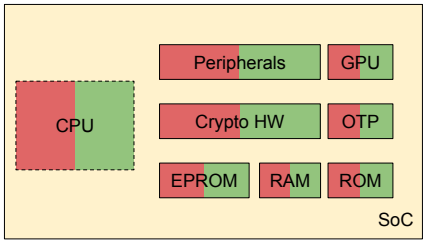

Fig. 2: ARM TrustZone allows for the partitioning of hardware components into a secure or non-secure state.

Most mobile devices use the NW to host feature-rich software, *i.e.,* a Rich Operating System (RichOS) and Client Applications (CAs), whereas security-critical tasks are carried out in the SW using a Trusted Operating System (TrustedOS) and Trusted Applications (TAs). On Android devices, depedent on the Original Equipment Manufacturer (OEM), we can find different TrustedOSs and TAs, like fingerprint unlock or face identification unlock. Therefore, NW and SW are also referred to as Rich Execution Environment (REE) and TEE, respectively. A context switch between these two execution contexts usually happens by entering the Monitor.

The two operating systems, RichOS and TrustedOS, have their own page tables and, consequently, do not share the same view of their virtual address spaces. As can be seen from this architecture, both contexts can run apps that are isolated from each other and the operating system using the means of virtual memory.

Given these architectural components, it is important to note that the process of initializing such a system is crucial for the security of the entire system. OEMs refer to this process as *Secure Boot* [19] and engineered the boot process to only load correctly signed and verified components forming the Trusted Computing Base (TCB). This TCB usually includes software executed in the SW.

TEE-BI is based on the architectural primitives given by ARMv8-A. Those are the same primitives vendors like Qualcomm, Samsung, or Huawei base their system designs on. Thus, the hardware requirements needed for a system like TEE-BI, are already present on millions of devices used by consumers these days.

4 Design

Trusted Execution Environments (TEEs) have become commonplace on modern mobile devices [35,42,44,45,10]. They are the foundation for our disk encryption, manage the access to security-sensitive peripherals like the fingerprint sensor, and decrypt our Netflix 4K streams on-thy-fly. More advanced features implemented within TEEs are real-time kernel integrity monitoring [14] and trusted user interfaces [33]. All of these features are available today and part of the mobile devices you can buy at your neighborhood electronics store.

Using this versatile technology, we are capable of performing a vast array of actions from a trusted context. Of course, when talking about this trusted context, we refer to the trust relationship you are committing to by deciding for a certain vendor. In this section, we suggest adding another party to this trusted context, which is *your government*. We propose TEE-BI, a TEE-backed system to exfiltrate only the minimal amount of information from a suspect's end device – the cryptographic key – to open the cryptographic shield "on the wire".

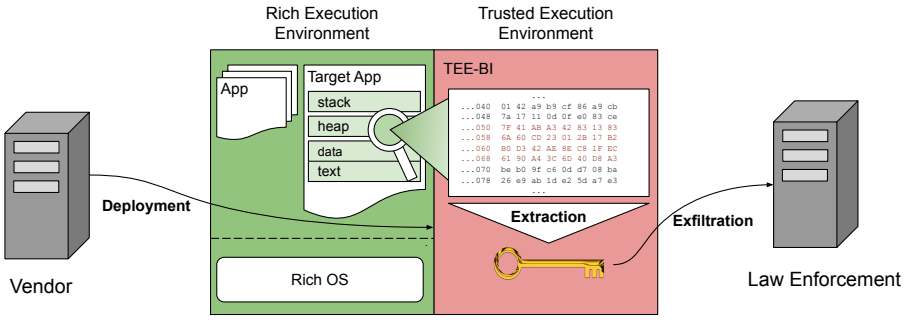

Fig. 3: TEE-BI is capable of stealthily exfiltrating cryptographic material from a target application using TEE-based introspection.

In the following sections, we elaborate on the three steps of TEE-BI. First, we discuss deployment variants of TEE-BI. Second, we explain the process of targeted cryp-

tographic key extraction. Third, we examine different ways of exfiltrating the extracted keys. Figure 3 illustrates these steps and serves as an overview.

4.1 Deployment

The goal of the deployment process is to load TEE-BI reliably into the TEE using the most direct way possible and guarantee its integrity and confidentiality during this process.

For this research, we reviewed major commercially used TEEs and found that the capabilities of dynamically loading code and guaranteeing its integrity are commonly available features. Samsung [45], Qualcomm [42], as well as Huawei [35], use signed executables and a verifying loader to run trusted code in their TEEs exclusively. In addition to the verification of executables, Huawei also employs a decryption step that is based on asymmetric cryptography in order to provide confidentiality guarantees. We assume these capabilities of a "secure loader" to guarantee TEE-BI's integrity and confidentiality since they are already used on millions of devices.

To reliably deliver TEE-BI to the target device, we suggest two different schemes, depending on the capabilities of the TEE. For both schemes, the underlying thought is that every mobile device connects back to update servers or other backends managed by the device vendor, which can be leveraged to deliver TEE-BI to the target device.

In the first scheme, the TEE has direct access to the networking interface of the device. The TEE can temporarily claim exclusive access to the networking interface, which is a technique already used for other peripherals, for instance, to accomplish trusted user interfaces [44,28]. For trusted user interfaces, the framebuffer, as well as the touch screen sensors, are temporarily taken over by the TEE from the REE. In our scenario, this peripheral would be the networking interface.

The second scheme, which is less complex and widely deployed, is the usage of the existing networking software stack of the RichOS. In fact, for this scheme, GlobalPlatform (GP), a standardization body for TEEs, already provides a detailed specification on how to integrate and implement the network access from a TEE context [26].

The deployment of TEE-BI requires vendors to load and execute third-party code in their TEEs. This use case is already part of major TEE platforms. For instance, Widevine [29] is a content protection framework for media that consists of components running inside of the TEEs of major mobile device vendors (*i.e.,* Samsung and Google devices). Considering the fact that all telecommunication providers in almost all countries can be forced to cooperate with law enforcement by regulations, it seems clear that similar regulations should at least in principle be possible in many countries.

4.2 Extraction

The process of data extraction from the TEE context has two steps. First, the target application has to be localized, and second, the relevant information has to be found and extracted. Both of these steps are making use of TEE-based introspection, meaning, TEE-BI leverages its unrestricted access to the main memory in order to reconstruct the virtual address space of the target process running within the REE and applies a set of heuristics to locate the information of interest.

The RichOS (*i.e.,* Android uses the Linux kernel) maintains a list of tasks corresponding to userland processes. This list of tasks is referenced from the RichOS. To get access to this list of tasks, TEE-BI exploits the fact that it has access to the translation tables of the REE and, therefore, can reconstruct the virtual address spaces of the RichOS and its userland applications.

To locate the target application, TEE-BI traverses this list of tasks and looks for indicators identifying the target application. In the simplest case, such an indicator is the name of the target application, which is part of its task list entry.

After finding the target application, TEE-BI takes the base of its first level page table from the task list entry, to identify all memory regions belonging to this application.

Given all the pages of the target app process, TEE-BI can reconstruct its entire virtual address space and extract any data from it. The process of finding the relevant information from a virtual address space of an application is implementation-specific. We use a set of heuristics to identify and extract cryptographic key material from the `boringssl` library [2], the predominantly used cryptography library on Android systems.

4.3 Exfiltration

As it is the case for the deployment of TEE-BI, we need a reliable way to communicate with a remote party (*e.g.,* law enforcement servers in Figure 3) for effective exfiltration. This communication can happen by leveraging one of the two schemes described above in Section 4.1: (1) temporary take-over of a network interface by the TEE, or (2) leveraging the REE networking software stack. The outgoing connection from the TEE to the law enforcement server must be cryptographically secured. After successful exfiltration of the key material, law enforcement entities can decrypt the traffic that they already captured via established methods [37,21].

5 Implementation

We implemented TEE-BI on a LeMaker HiKey 620 development board, which is one of the reference boards for prototyping Android systems [1]. This board is equipped with a Hisilicon Kirin 620 SoC (Cortex-A53 Octa-core 64-bit) implementing the ARMv8-A ISA. Our REE software stack consists of Android 9, which is based on a 4.14 Linux kernel. In the TEE, we run OP-TEE 3.4.0 [6]. Based on this setup, we implement TEE-BI to perform TZ-based introspection for fine granular data extraction.

5.1 Physical Memory Access

TEE-BI's core is implemented as an Open Portable Trusted Execution Environment (OP-TEE) Pseudo Trusted Application (PTA). PTAs, in contrast to TAs, run in the TrustedOS and, have access to privileged functions (*i.e.,* mapping arbitrary physical memory). Access to these functions is necessary in order to map physical memory used by the REE into the virtual address space of TEE-BI.

OP-TEE offers an interface to map physical memory pages as part of its shared memory implementation. For TEE-BI, we leverage this interface and add auxiliary functions to access physical memory in various ways conveniently. In particular, we add primitives for eight-byte and one-byte reads, as well as for reading null-byte terminated strings. Listing 1.1 shows the interface we use to access the physical memory.

```
uint64_t read_64(paddr_t paddr);
uint8_t read_8(paddr_t paddr);
void read_str(paddr_t paddr, char *buf, size_t buf_size);
```

Listing 1.1: TEE-BI's primitives for physical memory access.

5.2 REE Virtual Address Resolution

TEE-BI needs capabilities to resolve virtual addresses from the REE to physical addresses in order to traverse data structures used by the RichOS and its userland applications. Virtual address translation on ARMv8 is carried out using the interplay between the Memory Management Unit (MMU) and the Translation Table Base Registers (TTBRs) (*e.g.,* `ttbr0_el1` and `ttbr1_el1`).

On Linux, the highest bit of a given virtual address decides which of the two TTBRs is used. Having a memory layout with 4KB pages and a 4 level translation table, virtual userspace addresses reside in the range of

`0x0000000000000000` to `0x0000ffffffffffff`

and the kernel virtual address space is located within

`0xffff000000000000` to `0xffffffffffffffff`.

Therefore, page table walks for userspace are based on `ttbr0_el1`, and page table walks for the kernel are based on `ttbr1_el1`.

For TEE-BI, this means we have to differentiate virtual addresses from the RichOS and its userland to correctly resolve their underlying physical addresses. In order to resolve a userland virtual address, we implemented the four-level page table walk of 4KB pages in software.

Based on these resolution mechanisms, TEE-BI provides the same memory access functions as earlier introduced (see Figure 1.2), just for retrieving data for REE virtual addresses.

```
uint64_t read_64_virt(vaddr_t vaddr);
uint8_t read_8_virt(vaddr_t vaddr);
void read_str_virt(vaddr_t vaddr, char *buf, size_t buf_size);
```

Listing 1.2: TEE-BI's primitives for REE virtual memory access.

5.3 Task List Traversal

As discussed in Section 4.2, our entry point to the RichOS's process management is a task list. In the Linux kernel, the entry point to this list is the initial task (`init_task`). TEE-BI performs a task traversal as illustrated in Figure 4. We traverse the process tree by following children and sibling references until we find our target process. By doing this, we need to translate kernel virtual addresses to physical addresses, as explained

above. Our search criterion to identify the target is its `task_struct`'s `comm` field, which contains the name of the process. TEE-BI offers a modular and extendable design to allow developers to implement more advanced heuristics to identify the target process, if necessary.

Task structures within Linux have an `mm_struct` member describing their memory. As part of this struct, we can find the `pgd` member, which is the virtual address of the first level page table (see Figure 4). Since the layout of `task_struct` and `mm_struct` can change between kernels, TEE-BI provides configuration options for these parameters. Overall we use five different offsets for the traversal:

`off_children`: Offset of children list pointer in the `task_struct`
`off_siblings`: Offset of `siblings` member in the `task_struct`
`off_comm`: Offset of `comm` member in the `task_struct`
`off_mm`: Offset of `mm` member in `task_struct`
`off_mm_pgd`: Offset of `pgd` member in the `mm_struct`

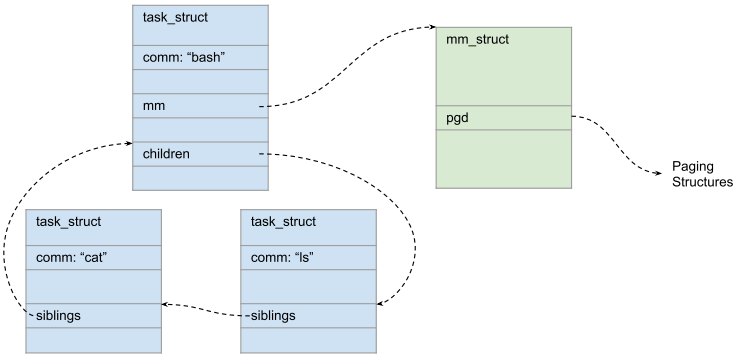

Fig. 4: TEE-BI traverses the process tree to find the target process.

5.4 Page Table Traversal

Having the entry point to the target process' paging structures, it is possible to traverse its virtual address space. Since the page tables hold physical addresses, TEE-BI uses the physical memory access functions for the traversal.

The modular design of TEE-BI allows developers to specify a callback function for each memory page (see Listing 1.3). The callback function allows for modularization and extendability of memory area handling. Using this technique, each use case can have its own callback function. This may be a simple search for strings or regular expressions but could also be a more complex extraction of nested data structures.

```
1 static uint64_t
2     handle_memory_area(paddr_t pgd, vaddr_t vaddr, uint64_t len);
```

Listing 1.3: TEE-BI uses a callback function to handle contigous regions in memory.

5.5 Data Extraction

For the use case of extracting cryptographic key material of ongoing TLS-encrypted connections, we implement a callback function that extracts the Master Secret and the Client Random [7] of a target process that uses the `boringssl` library. We assume that the data structure layouts used by `boringssl` are known or can at least be probed before scanning for the cryptographic key material. The two values underlying secure connections for the `boringssl` library are referenced by the Secure Sockets Layer (SSL)-context data structure. By finding this structure, we can follow the appropriate offsets and memory references to extract the desired information. In particular, we find this data structure by searching for a fingerprint consisting of certain lengths values used for key sizes and cipher ids indicating different ciphers. Additionally, the alignment of the members is considered during the search. Once the structure is identified, we can systematically traverse the memory and retrieve the target information.

5.6 Prototype Limitations

Our TEE-BI prototype is focused on providing a proof of concept for the viability of TEE-based introspection and extraction of cryptographic key material. Therefore, we initiate the extraction right after the process using the `boringssl` library established a secure connection. In a production state, TEE-BI would implement a polling-based mechanism for the extractions step. Since cryptographic keys for TLS connections only temporarily reside in memory, the time frame to extract the keys is limited. TEE-BI cannot rule out the possibility that a key is already wiped from memory, but it can increase the chance of searching at the right point in time by scanning the memory with a higher frequency. Furthermore, a secure timer is needed to trigger and guarantee the execution of the extraction routine without the RichOS being able to interfere with this process.

6 Evaluation

In this section, we evaluate TEE-BI concerning its effectiveness, performance, and compatibility, as well as its compliance with legal requirements.

6.1 Target Application

For testing TEE-BI, we use a client application, which opens an SSL-encrypted connection and transmits data to a remote server. Furthermore, we show the general approach and explain the callback functions used for obtaining the target data. Additionally, we demonstrate how to extract cryptographic key material from the target process. For this purpose, we implement a client and an echo server, which both utilize Transport Layer Security (TLS) encrypted sockets.

To ensure secure communication with the server, the target client uses BoringSSL. After opening a connection, it initiates an SSL-handshake. Next, the client sends data to the server, which the server directly echoes back to the client.

We run the client application as our target in the REE. TEE-BI is able to retrieve the Master Secret and the Client Random from the client application.

6.2 Effectiveness

To show the effectiveness of TEE-BI, we use the target application. The target application running in the REE connects to the server hosted on a remote machine. As mentioned before, the connection between the client and the server is encrypted using SSL (TLS 1.2). We capture the encrypted traffic onto the server machine with `tcpdump` and open it using Wireshark. At this point, it is not yet possible to decrypt the traffic. However, by using TEE-BI, we can now extract the Master Secret and the Client Random that were used for the secure connection. Listing 1.4 depicts the format of the extracted secrets, consisting of three strings on a single line. The first string is the label `CLIENT_RANDOM`, the second string is the 64-digit Client Random encoded as a hex-string, and the third string is the 96-digit Master Secret also encoded in hex.

```
1 CLIENT_RANDOM \
2 f282e8e9873964504ce6f94c8be4d15d\
3         5db4b508de20a61499bad70f0c8ff34c \
4 fc309fdd46cf4f5a6e5754b3c91c6f70\
5         0848991270e4624d771b038c7757136f\
6         a64e33554b713b6bcf751042d29deaf3
```

Listing 1.4: Master Secret and Client Random in correct format for Wireshark.

With the above format in Listing 1.4, Wireshark is able to read the secrets and decrypt the traffic. Since we were able to decrypt the SSL-traffic successfully, TEE-BI is effective in extracting cryptographic key material from a target process running in the REE.

6.3 Performance

We measure the performance of TEE-BI by considering the overall duration for the task list traversal and the extraction of cryptographic secrets. For comparison, we also count the number of pages mapped, and the number of performed page table walks. In order to get reliable results, we execute the task list traversal and the extraction of secrets 1000 times each and take the arithmetic mean of our measurements.

Since TEE-BI needs to map pages for reading their content programmatically, we also need to measure this impact. During the introspection, there were 192 processes running, and 207 pages needed to be traversed for the extraction of the encryption secrets. Table 1 shows the performance measurements. During task list traversal, the virtual address mapping has been changed 2935 times, which includes pages that have been mapped more than one time. In comparison, the virtual address mapping for the extraction was updated 7008 times. Table 1 also shows that for the extraction 48 713 μs of the overall 50 491 μs are necessary for the mapping procedure. Also for the task traversal, the mapping of the pages into the virtual address space of the TEE takes most of the overall duration. Looking at the duration for the page table walk during the extraction of secrets, we can see that most of the time is spent on mapping pages as well. In average, mapping one page takes $48\,713\mu s/7008 = 6{,}95\mu s$. Compared to the overall time for page table walks we can conclude that the page mapping has a strong impact on the performance of TEE-BI.

	page table walks	pages mapped	overall duration
Task List	-	2 935 pages	
Traversal	0 μs	20 598 μs	21 132 μs
Secrets	25 walks	7 008 pages	
Extraction	441 μs	48 713 μs	50 491 μs

Table 1: Performance measurements of TEE-BI on the HiKey development board.

In summary, we can observe that the adjustment of the memory mapping in the TEE has a strong impact on TEE-BI's performance. TEE-BI does not implement sophisticated caching mechanisms yet, which leaves significant room for improvements.

6.4 Compatibility

Since ARM processors hold a large market share, it is important to evaluate the adaptations necessary for porting TEE-BI to other ARM-based systems.

First, we have a look at mobile consumer devices such as smartphones and tablets. The majority of these devices use Android and processors with the ARMv8-A ISA. Therefore, these devices have an MMU and most likely support for ARM TrustZone. These are the same prerequisites we assume for TEE-BI. Given these similarities, using TEE-BI on a mobile consumer device should be straightforward.

Depending on the kernel configuration and the used memory management model, it might be necessary to adjust some parameters, for instance, for the page table walk. These parameters affect the number of paging levels or the granule size of the pages in use [12]. For instance, the kernel on our HiKey development board uses four-level page tables. We designed TEE-BI in a configurable way to enable its portability and, thus, can adjust the number of paging levels and the page size.

Besides mobile consumer devices, ARM processors are used in various embedded devices. Assuming an ARM TrustZone is present on such a device, there is the possibility to implement TEE-based introspection. It might be more complex to adjust the prototype to such a device depending on its memory model. If an MMU is present, the procedure is likely similar to our procedure. In case an MMU is not present, the processor is often used as a microcontroller (ARMv8-M). These processors use a Memory Protection Unit (MPU) instead of an MMU. In this case, the introspection needs to be adjusted to its specific use case. This might be more complex as well, but still possible.

We can summarize that compatibility with architectures commonly used for mobile consumer devices seems given. Also, in the embedded area, our approach could be utilized but requires more adjustments.

6.5 Legal Requirements

As argued by Fröwis et al. [24], digital evidence in criminal proceedings under the rule of law must meet the following requirements: Lawfulness of data processing, authenticity and integrity of data, reliability of the applied methods, qualification of the investigators, verifiability of the evidence and the conclusions, chain of evidence. As we now

argue, the method we present in this paper has advantages over traditional methods of remote forensic telecommunication surveillance, especially with regard to the following criteria, which increase the value of evidence for legal proceedings: Authenticity and integrity, reliability and verifiability.

Authenticity and integrity. Authenticity is given if the data presented as evidence actually originate from the source indicated by the investigating authorities. Integrity can be affirmed if the data has not been altered after their collection [24, p. 5]. The advantage of TEE-BI's method is that after the extraction of the key, the telecommunication data can be obtained from the telecommunication providers in the traditional way as a 1:1 copy following a classical wiretap. The telecommunication provider then guarantees that the data actually resulted from a specific communication event and went "over the network". With TEE-BI, it is, therefore, harder to argue that law enforcement authorities have planted the communication data on purpose on the monitored system. Furthermore, the methods of IT forensics, which have been tried and tested for years, can be used to avoid changes to these data sets [16].

Reliability. The reliability of the investigative method is ensured if it is a precise and scientifically validated method for the respective data processing and if it is used correctly in the individual case [24, p. 5]. The advantage of TEEs, such as TrustZone, lies in the fact that they guarantee that only particular versions of the software get to be executed, namely those that were digitally signed by some trusted party (usually the manufacturer). This means that the remote forensic software must be *signed* before it is able to run. This makes it much harder for law enforcement to deploy arbitrary functionality.

In addition, the installation of the remote forensic software is achieved through methods that the manufacturer explicitly supports. This prevents collateral damage from more "offensive" installation scenarios (e.g., manual or via exploit) that dominate the mobile phone forensics market [20,54] and have problems of their own. TrustZone even provides a technique for *remote attestation*, i.e., the possibility to construct a cryptographic security proof that a particular *version* of software was running on the system at any given point in time. Realizing this would definitely increase the trust in the reliability of the extraction and analysis of the evidence because it not only can be proven that a software package was running at a certain point in time, but also its exact version.

Verifiability. The results of the evaluation of data sets are verifiable if they meet the criteria of repeatability and reproducibility [38,24]. Traditional remote forensic telecommunication surveillance faces the problem here that, by extracting the telecommunication data before or after the sending process, there is no possibility to repeat this process or to show that the data was actually sent later/before. There is also the danger of overapproximating the data that is sent over the network, circumstances which are illegal in Germany [23].

Unfortunately, with TEE-BI, we also cannot repeat the acquisition process at a later time since cryptographic keys are usually eliminated from memory when a connection closes. However, this form of repeatability is also not present in classical forms of source telecommunication surveillance. The fact that the original data comes from the

telecommunication provider makes it harder for law enforcement to plant new evidence into a file with network traffic. What, however, can be done is to extract repeatedly the unencrypted communication data from the encrypted network communication that was seized through wiretapping. This replication step clearly has evidential value in criminal cases.

A threat to reproducibility is the introspection element because there is always the chance that data that already resides in memory might be misunderstood or wrongly interpreted by the software. Cryptographic techniques (like authenticated encryption), however, will ensure that decryption does not produce any false positives, i.e., messages that are "valid" decryptions of the plaintext without being actual messages from the producer. Even though scanning for cryptographic keys is heuristic in nature, the decryption backend can always try more than one decryption key. Therefore, if data can be successfully extracted, then the chosen key is correct, and the extracted data was, in fact, the data that was previously encrypted.

7 Related Work

TEE-BI introspects the translation tables of a lower-privileged execution context from the TEE. This idea of introspection originated from the usage of hypervisors for Virtual Machine Introspection (VMI). VMI describes the access and evaluation of a running system and its resources from outside of the system [25]. The concept of VMI primarily applies introspection to main memory, but, more generically, could also include non-volatile background storage and other hardware. VMI is primarily used for secure monitoring of running Operating Systems (OS's) [34]. An approach using the hypervisor extensions of ARM chipsets, instead of AMD's or Intel's technologies, was proposed by Lengyel *et al.* [36], who evaluated VMI on ARM with the Xen hypervisor. In this context, LibVMI is a promising project to harmonize and simplify the required introspection Application Programming Interface (API) [41].

Other research involving TEE-based translation table introspection was first shown by Azab *et al.* [14]. Their system is executed within the TEE and ensures the integrity of the running rich OS. In contrast to Azab *et al.*'s work, TEE-BI has its focus on the exfiltration of data from userland processes, instead of monitoring the kernel. Furthermore, Guerra *et al.* [32] introduce the ITZ-library, which generalizes the introspection of NW memory from the TEE. For their research, they use the Genode OS Framework [3]. Since the ITZ-library is based on Genode, and OP-TEE does not support the Genode Framework, we do not use the ITZ-library but implement the memory access based on the interface provided by OP-TEE.

Moreover, TrustDump [50] is a forensic tool for reliable and trustworthy memory acquisition on smartphones. This tool uses TZ for the generation of physical memory dumps. Instead of generating a memory dump, TEE-BI evaluates the physical memory while the target operating system is still running.

Next, Stüttgen *et al.* [49] evaluate the possibility of injecting a minimal acquisition module into a running Linux Kernel Module (LKM). This approach reduces the impact on the target system, but still changes the codebase of the targeted OS significantly.

TEE-BI's acquisition is carried out from the isolated context of a TEE and, therefore, naturally has a minimal impact on the integrity of the introspection target.

Further, DroidKex [52] targets Android apps and extracts data from applications' address spaces. DroidKex is a rich OS module and intercepts calls to the SSL library used by targeted applications. TEE-BI's goal is similar to DroidKex's, but we integrate the extraction of the SSL secrets into the TEE, instead of modifying the REE software stack. With our setup, the monitoring runs stealthily alongside the REE and can barely be detected. Effective hiding techniques are a major challenge for other monitoring tools [46,51].

For the live analysis of Android applications, Thing *et al.* [53] developed a tool for the extraction of process memory. The live analysis runs as system service in the Android environment and is able to gather data from user processes such as deleted messages from Instant Messengers (IMs). Thing *et al.*'s research was later enhanced by Grover [31] to no longer require elevated privileges. Using a userland application, they provide an open-source Android monitoring tool for enterprise purposes. All these approaches require software within the REE, which is the main difference to our approach.

8 Conclusions

In this paper, we propose a novel approach for remote forensic telecommunication surveillance which addresses two major downsides of the current practices: Our design increases the evidential value of remote forensic investigations and provides a technique for lawful surveillance of encrypted communication. To achieve these goals, our approach embeds the remote forensic software into the trusted computing base and utilizes characteristics of the ARM TZ, such as the root of trust in secure hardware. Moreover, in our approach, we exfiltrate only a minimal amount of information from the consumer device that is necessary for the surveillance of encrypted communication: the cryptographic key. We argue that this approach is least intrusive to the target device and therefore is the only method, which satisfies legal principles in Europe (Proportionality) and the USA (Balancing Test).

Furthermore, we implement and evaluate TEE-BI, a prototype for our proposed approach. TEE-BI is based on ARMv8-A with ARM TZ technology and, therefore, uses the same foundation as millions of mobile devices around the globe. To demonstrate its practicability, we implemented TEE-BI on real hardware and used an up-to-date Android-based setup for our evaluation. We study TEE-BI's effectiveness, performance, and compatibility as well as evaluate it regarding legal requirements. For the latter, we argue that TEE-BI by design complies with authenticity and integrity, reliability, and verifiability.

Acknowledgements We would like to thank Maria Maras for her constructive feedback. This research was supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant number 393541319/GRK2475/1-2019) and the German Federal Ministry of Education and Research (BMBF) as part of the Software Campus project (Förderkennzeichen: 01/S17045).

References

1. Android Open Source Project: Using reference boards, https://source.android.com/setup/build/devices, Last accessed 15 Sep 2020
2. BoringSSL, https://boringssl.googlesource.com/boringssl/, Last accessed 15 Sep 2020
3. Genode Operating System Framework, https://genode.org/, Last accessed 15 Sep 2020
4. German Federal Constitutional Court BVerfGE 120, p. 274, p. 318-319; Handyside v. The United Kingdom (1976) 24 ECHR (Ser. A) 23 at para. 49;
5. Lorraine v. Markel American Insurance Company, 2007, United States District Court for the District of Maryland, 241 F.R.D. 534 (D. Md. 2007)
6. Open Portable Trusted Execution Environment, https://www.op-tee.org, Last accessed 15 Sep 2020
7. RFC5246 – The Transport Layer Security (TLS) Protocol Version 1.2, https://tools.ietf.org/html/rfc5246, Last accessed 15 Sep 2020
8. Skinner v. Ry. Labor Executives' Ass'n, 489 U.S. 602, 629 n.9 (1989)
9. R. v. Oakes. 1 SCR 103 (1986)
10. Security enclave processor for a system on a chip (2012), https://patents.google.com/patent/US8832465B2/en, Last accessed 15 Sep 2020
11. Abeyratne, R.: More structure, more deference: Proportionality in Hong Kong. Proportionality in Asia (edited by Po Jen Yap) (2019)
12. ARM: Arm® architecture reference manual Armv8, for Armv8-A architecture profile documentation, https://developer.arm.com/docs/ddi0487/latest, Last accessed 15 Sep 2020
13. ARM: ARM security technology: Building a secure system using trustzone technology (2008), https://static.docs.arm.com/genc009492/c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf, Last accessed 15 Sep 2020
14. Azab, A.M., Ning, P., Shah, J., Chen, Q., Bhutkar, R., Ganesh, G., Ma, J., Shen, W.: Hypervision across worlds: Real-time kernel protection from the arm trustzone secure world. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. p. 90–102. CCS '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2660267.2660350
15. Barak, A.: Proportionality - Constitutional rights and their limitations. Cambridge University Press pp. 181–208, concerning the legal sources see pp. 211–241 (2012)
16. Casey, E.: Digital Evidence and Computer Crime - Forensic Science, Computers and the Internet, 3rd Edition. Academic Press (2011), http://www.elsevierdirect.com/product.jsp?isbn=9780123742681, Last accessed 15 Sep 2020
17. Cohen-Eliya, M., Porat, I.: Proportionality and constitutional culture. Cambridge University Press (2013)
18. Cupa, B.: Trojan horse resurrected - on the legality of the use of government spyware. In: Webster, C., William, R. (eds.) Living in surveillance societies: The state of surveillance: Proceedings of LiSS conference 3. pp. 419–428. CreateSpace Independent Publishing Platform (2013)
19. Dent, A.W.: Secure boot and image authentication (2019), https://www.qualcomm.com/media/documents/files/secure-boot-and-image-authentication-technical-overview-v2-0.pdf, Last accessed 15 Sep 2020
20. Dewald, A., Freiling, F.C., Schreck, T., Spreitzenbarth, M., Stüttgen, J., Vömel, S., Willems, C.: Analyse und vergleich von BckR2D2-I und II. In: Suri, N., Waidner, M. (eds.) Sicherheit

2012: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 6. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 7.-9. März 2012 in Darmstadt. LNI, vol. P-195, pp. 47–58. GI (2012), https://dl.gi.de/20.500.12116/18287, Last accessed 15 Sep 2020
21. European Telecommunications Standards Institute: Handover interface for the lawful interception of telecommunications traffic (1999), https://www.etsi.org/deliver/etsi_ts/101600_101699/101671/03.11.01_60/ts_101671v031101p.pdf, Last accessed 15 Sep 2020
22. European Union, Directorate-General for Internal Policies: Legal frameworks for hacking by law enforcement: Identification, evaluation and comparison of practices (2017), http://www.europarl.europa.eu/supporting-analyses, Last accessed 15 Sep 2020
23. Freiling, F., Safferling, C., Rückert, C.: Quellen-TKÜ und Online-Durchsuchung als neue Maßnahmen für die Strafverfolgung: Rechtliche und technische Herausforderungen. Juristische Rundschau pp. 9–22 (2018)
24. Fröwis, M., Gottschalk, T., Haslhofer, B., Rückert, C., Pesch, P.: Safeguarding the evidential value of forensic cryptocurrency investigations. Forensic Science International: Digital Investigation . https://doi.org/10.1016/j.fsidi.2019.200902
25. Garfinkel, T., Rosenblum, M., et al.: A virtual machine introspection based architecture for intrusion detection. In: Ndss. vol. 3, pp. 191–206. Citeseer (2003)
26. GlobalPlatform: TEE sockets API specification v1.0.1 (2017), https://globalplatform.org/specs-library/tee-sockets-api-specification-v1-0-1/, Last accessed 15 Sep 2020
27. Goodison, S.E., Davis, R.C., Jackson, B.A.: Digital evidence and the us criminal justice system: Identifying technology and other needs to more effectively acquire and utilize digital evidence. pp. 9–10. RAND Corporation (2015), www.jstor.org/stable/10.7249/j.ctt15sk8v3, Last accessed 15 Sep 2020
28. Google: Protected confirmation (2020), https://source.android.com/security/protected-confirmation, Last accessed 15 Sep 2020
29. Google Widevine: Widevine – leading content protection for media (2019), https://www.widevine.com/, Last accessed 15 Sep 2020
30. Grabenwarter, C.: ECHR - Commentary (2014), Art. 8 para. 3, 4, 28
31. Grover, J.: Android forensics: Automated data collection and reporting from a mobile device. Digital Investigation **10**, S12–S20 (2013)
32. Guerra, M., Taubmann, B., Reiser, H.P., Yalew, S., Correia, M.: Introspection for ARM TrustZone with the ITZ library. In: 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS). pp. 123–134. IEEE (2018)
33. Hayton, R.: The benefits of trusted user interface (TUI) (2019), https://www.trustonic.com/news/blog/benefits-trusted-user-interface/, Last accessed 15 Sep 2020
34. Hebbal, Y., Laniepce, S., Menaud, J.M.: Virtual machine introspection: Techniques and applications. In: 2015 10th International Conference on Availability, Reliability and Security. pp. 676–685. IEEE (2015)
35. Huawei Technologies: Emui 8.0 security technical white paper (2017), https://consumer-img.huawei.com/content/dam/huawei-cbg-site/en/mkt/legal/privacy-policy/EMUI8.0SecurityTechnologyWhitePaper.pdf, Last accessed 15 Sep 2020
36. Lengyel, T.K., Kittel, T., Eckert, C.: Virtual machine introspection with Xen on ARM. In: Workshop on Security in highly connected IT systems (SHCIS) (2015)
37. Lumme, M., Eloranta, J., Jokinen, H.: Interception system and method (1999), https://patents.google.com/patent/US20020049913, Last accessed 15 Sep 2020

38. Maras, M.H.: Computer Forensics. Jones & Bartlett Learning, second edn. (2015)
39. Marquis-Boire, M., Marczak, B., Guarnieri, C., Scott-Railton, J.: For their eyes only. the commercialization of digital spying. Citizen Lab Report (Sep 2013), https://citizenlab.org/storage/finfisher/final/fortheireyesonly.pdf, Last accessed 15 Sep 2020
40. Neumann, C., Kaye, D., Jackson, G., Reyna, V., Ranadive, A.: Presenting quantitative and qualitative information on forensic science evidence in the courtroom. Chance **29**, 37–43 (2016)
41. Payne, B.D.: Simplifying virtual machine introspection using LibVMI. Sandia report pp. 43–44 (2012)
42. Qualcomm: Qualcomm mobile security (2018), https://www.qualcomm.com/solutions/mobile-computing/features/security, Last accessed 15 Sep 2020
43. Rueckert, C.: Cryptocurrencies and fundamental rights. Journal of Cybersecurity p. 6 (2019)
44. Samsung: Trustonic for KNOX (2015), https://news.samsung.com/global/samsung-and-trustonic-launch-trustonic-for-knox-delivering-a-whole-new-level-of-trust-enhanced-experiences-on-samsung-mobile-devices, Last accessed 15 Sep 2020
45. Samsung: Samsung TEEGRIS (2020), https://developer.samsung.com/teegris/overview.html, Last accessed 15 Sep 2020
46. Samuel, J., Mathewson, N., Cappos, J., Dingledine, R.: Survivable key compromise in software update systems. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. pp. 61–72 (2010). https://doi.org/10.1145/1866307.1866315
47. Sieber, U., von zur Mühlen, N. (eds.): Access to Telecommunication Data in Criminal Justice. A Comparative Analysis of European Legal Orders. Duncker & Humblot, Berlin (2016)
48. Strossen, N.: The fourth amendment in the balance: Accurately setting the scales through the least intrusive alternative analysis. 63 NYUL Rev. **1173** (1988)
49. Stüttgen, J., Cohen, M.: Robust Linux memory acquisition with minimal target impact. Digital Investigation **11**, S112–S119 (2014)
50. Sun, H., Sun, K., Wang, Y., Jing, J.: Reliable and trustworthy memory acquisition on smartphones. IEEE Transactions on Information Forensics and Security **10**(12), 2547–2561 (2015)
51. Sylve, J., Case, A., Marziale, L., Richard, G.G.: Acquisition and analysis of volatile memory from android devices. Digital Investigation **8**(3-4), 175–184 (2012)
52. Taubmann, B., Alabduljaleel, O., Reiser, H.P.: DroidKex: Fast extraction of ephemeral TLS keys from the memory of Android apps. Digital Investigation **26**, S67–S76 (2018)
53. Thing, V.L., Ng, K.Y., Chang, E.C.: Live memory forensics of mobile phones. digital investigation **7**, S74–S82 (2010)
54. WikiLeaks: Spyfiles 4, https://wikileaks.org/spyfiles4/, Last accessed 15 Sep 2020
55. Winter, L.B.: Remote computer searches under spanish law: The proportionality principle and the protection of privacy. Zeitschrift für die gesamte Strafrechtswissenschaft **129**(1), 205–231 (2017)