Countermeasures for Timing-Based Side-Channel Attacks against Shared, Modern Computing Hardware

Abstract: There are several vulnerabilities in computing systems hardware that can be exploited by attackers to carry out devastating Microarchitectural Timing-Based Side-Channel Attacks against these systems and as a result compromise the security of the users of such systems. By exploiting Microarchitectural resources, adversaries can potentially launch different variants of Timing Attacks, for instance, to leak sensitive information through timing. In view of these security threats against computing hardware, in a recent study, titled "Are Timing-Based Side-Channel Attacks Feasible in Shared, Modern Computing Hardware?", currently undergoing the review process, we presented and analysed several such attacks. This extended study proceeds to build upon our recent study in question. To this end, we analyse the existing countermeasures against Timing Attacks and propose new strategies in dealing with such attacks.

Keywords: side channels; timing attacks; hardware attacks; channel attacks; digital investigations; countermeasures;

1. Introduction

In recent years, Side-Channel Attacks, hereafter referred to as SCAs, have grown from theoretical attacks (Kelsey et al., 2000; Kocher et al., 1999; Chari et al., 1999; Kocher, 1996; Rivest et al., 1978;) to highly advanced and practical attacks on general-purpose computing platforms (Gruss et al., 2017, a; Gruss et al., 2017, b; Schwarz et al., 2017; Irazoqui et al., 2016; Spreitzer et al., 2016; Pessl et al., 2016; Brumley and Boneh, 2003) and cloud computing infrastructures (Xiao et al., 2016; Liu et al., 2015; Zafirt, 2015; Zhang et al., 2014), and finally to attacks on mobile platforms (Lipp et al., 2016; Song et al., 2016; Chen et al., 2014; Sarwar et al., 2013). This growth has been the result of a consistent evolution of performance optimisation of modern microarchitectures that simultaneously manage multiple hardware resources (Thomas, 2017; Irazoqui, 2017; Borkar and Chien, 2011). SCAs pose serious security threats to modern and shared computing hardware (Ge et al., 2016; Liu et al., 2015; Xiao and Xiao, 2013; Kong, 2009). They are the result of spatial and temporal sharing of processor components between various applications as they run on the processor. A SCA – both theoretical (Chari et al., 1999; Kocher, 1996; Hu, 1992, Page, 2002) and practical (Xiao et al., 2016; Zhang et al., 2014; Yarom and Falkner, 2014; Bernstein, 2005; Osvik et al., 2006) – is carried out, for instance, through the exploitation of inadvertent information leakage from computing hardware (Gruss et al., 2017, a; Gruss et al., 2017, b; Spreitzer et al., 2016) or through the exploitation of Microarchitectural channels in order to deduce secrete keys such as those utilised in symmetric cryptography (Inci et al., 2016; Yarom and Benger, 2014; Zhang et al., 2014).

Various systems have inherent side-channel vulnerabilities that can be exploited

by the attackers to launch devastating SCAs. For instance, an adversary can simply carry out a differential power analysis (Cryptography Research, Inc et al., 2017; Moradi et al., 2011; Kocher et al., 2011; Barenghi et al., 2010; Coppens et al., 2009; Schramm et al., 2004; Guilley et al., 2004; Kocher et al., 2004) or monitor electromagnetic radiation (Longo et al., 2015; Hayashi et al., 2013; Homma et al., 2010), etc., in order to deduce vital data from the victims' systems (Gruss et al., 2017, a; Gruss et al., 2017, b; Zhang and Lee, 2014; Xiao et al., 2016; Irazoqui et al., 2016). Furthermore, processor architecture features including simultaneous multithreading (Tromer et al., 2010; Aciiçmez et al., 2007; Percival, 2005), control speculation and shared caches (Steffan et al., 2000; Tsai and Yew, 1996) can unintentionally accelerate side channels or enable new side channels (Yarom and Falkner, 2014; Wang and Lee, 2007). As a result, attackers can detect and exploit contention between hardware threads on the multiplier unit (Ge et al., 2016; Guan et al., 2015; Chen and Venkataramani, 2014). Such contention can be also exploited to create a side channel (Liu et al., 2016; Hunger et al., 2015; Ristenpart et al., 2009), for instance, to enable a malicious thread to differentiate multiplications from squaring in OpenSSL's RSA implementation (Aciiçmez, and Schindler, 2008; Wang and Lee, 2007). These attacks can determine the latency which result from contentious threats that are made to wait for access to functional units (Ge et al., 2016; Tromer et al., 2010; Ristenpart et al., 2009).

In view of the above security threats posed by SCAs against computing hardware, in a recent study, titled "Are Timing-Based Side-Channel Attacks Feasible in Shared, Modern Computing Hardware?" (Montasari et al., submitted), currently under the review process, we identified and analysed several Timing-Based Side-Channel Attacks, hereon abbreviated to TBSCA, a variant of SCAs (in the paper, SCAs have organised into a taxonomy of 13 distinct sub-categories). This extended study proceeds to build upon the study in question (Montasari et al., submitted) by analysing the existing countermeasures against Timing Attacks and proposing new strategies in mitigating such attacks.

The remainder of the paper is structured as follows: Section 2 provides a background for Microarchitectural Analysis, while Section 3 provides a brief review of the of SCAs proposed in the Literature. Countermeasures to address TBSCAs are then provided in Section 4. Finally, Section 5 concludes the study by providing a detailed discussion about trends in attacks and the future research direction in this research field. Two main contributions of this paper are the scope of the discussion as demonstrated in Table 1, since few works of similar scope currently exist, and the provision of an agenda for the direction of future research. Table 1 maps our work to the existing similar surveys on the subject.

Table 1. The scope of our work compared to the existing survey papers on the subject.

Year	Author/s	Platform	Title
2006	Neve and Seifert	Hardware	Advances on Access-Driven Cache Attacks on AES
2015	Doychev et al.	Hardware	CacheAudit: A Tool for the Static Analysis of Cache Side Channels
2014	Zhang and Lee	Hardware	Secure Cache Modeling for Measuring Side-channel Leakage
2016	Bosman et al.	Hardware	Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector
2014	Yarom and Falkner	Hardware	FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack
2007	Aciiçmez	Hardware	Yet another MicroArchitectural Attack: exploiting I-Cache
2014	Kim et al.	Hardware	Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors
2017	Gruss	Software	Software-based Microarchitectural Attacks
2010	Tromer et al.	Software	Efficient Cache Attacks on AES, and Countermeasures
2016	Aweke et al.	Software	ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks
2013	Andriotis et al.	Smartphone	A pilot study on the security of pattern screen-lock methods and soft side channel attacks
2012	Aviv et al.	Smartphone	Practicality of accelerometer side channels on smartphones
2016	Genkin et al.	Smartphone	ECDSA key extraction from mobile devices via nonintrusive physical side channels
2012	Owusu et al.	Smartphone	ACCessory: password inference using accelerometers on smartphones
2011	Cai et al.	Smartphone	TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion
2014	Irazoqui et al.	Cloud	Wait a minute! A fast, Cross-VM attack on AES
2016	Pessl et al.	Cloud	DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks
2016	Inci et al.	Cloud	Cache Attacks Enable Bulk Key Recovery on the Cloud
2017	Gruss et al., a	Cloud	Strong and Efficient Cache Side-Channel Protection using Hardware Transactional Memory
2018	**Our study (Montasari et al., submitted)**	**Hardware, Cloud, Smartphone**	**Countermeasures for Timing-Based Side-Channel Attacks against Shared, Modern Computing Hardware**

2. Background and Related Work

Since gaining attention in 2003, when Brumley and Boneh (2003) carried out a successful remote timing attack on real applications over a local network (Brumley and Boneh, 2003), research efforts have been made on the security analysis of PC platforms from a side-channel perspective (Gornik et al., 2015; Kong et al., 2008; Aciiçmez, 2007). The research community's realisation that the performance of some microprocessors could create critical side-channel emissions has led to the birth of the Microarchitectural Analysis research field, which is now an emerging area of side-channel cryptanalysis. Microarchitecture represents the method by which a specific instruction set architecture (ISA) is implemented in a particular processor (Flynn, 2007; Murdocca and Heuring, 2007; Clements, 2006), and Microarchitectural Analysis (MA) examines the impacts of common processor components and their performance upon the security of software cryptosystems (Ge et al., 2016; Aciiçmez, 2007; Osvik et al., 2006). Microarchitectural Attacks take advantage of the microarchitectural features of a processor to expose cryptographic keys. The performance of some processor components creates data-dependent variations in terms of runtime and power consumption signatures, during the execution of cryptosystems. Such variations either directly leak out the key value during a single cipher execution or emit information that can be collected during many executions and examined to exploit the system. To launch a SCA, adversaries do not require elevated system privileges as all the malicious activities are performed within their authorised privilege level. Attackers require neither knowledge of the plaintext, nor the ciphertext (Zhang et al., 2015; Zhang and Lee, 2014; Gullasch et al., 2011; Osvik et al., 2006).

With the rapid evolution of MA, in the years that have followed since 2003, researchers have been able to illustrate SCAs based on changes in different computing settings (Gruss et al., 2017, a; Gruss et al., 2017, b; Spreitzer et al., 2017), including: AES

(Zhang et al., 2016, b; Gullasch et al., 2011; Osvik et al., 2006), differential power analysis (Kocher et al., 2011; Barenghi et al., 2010; Guilley et al., 2004; Schramm et al., 2004; Kocher et al., 2004), monitor electromagnetic radiation (Longo et al., 2015; Hayashi et al., 2013; Homma et al., 2010), sound and electromagnetic emission (Faruque et al., 2016; Genkin et al., 2014; Callan et al., 2014; Cai and Chen, 2011), photonic side-channel leakage emission (Carmon et al., 2017; Krämer et al., 2013; Schlösser et al., 2012) and many more. All such attacks necessitate adversaries to be able to have a physical access to the victim device so as to monitor and deduce the secret information (Gruss et al., 2017, a; Gruss et al., 2017, b; Ge et al., 2016; Spreitzer et al., 2016; Liu et al., 2015). However, the latest and more advanced SCAs including Cache-Timing Attacks (Ge et al., 2016; Yarom and Falkner, 2014; and Tromer et al., 2010) and DRAM row buffer attacks (Gruss et al., 2017, a; Gruss et al., 2017, b; Schwarz et al., 2017; Pessl et al., 2016) can be carried out remotely by running malicious software within a cloud setting (Spreitzer et al., 2016; Xiao et al., 2016; Irazoqui et al., 2016).

Also with the emergence of cloud computing phenomenon, the extent of SCAs has also evolved considerably since 2000s (Spreitzer et al., 2016; Kim et al., 2012). Likewise, with the rapid advancements in mobile technology, researchers have been able to demonstrate even more sophisticated SCAs compromising smartphones (Spreitzer et al., 2016; Song et al., 2016; Sarwar et al., 2013; Owusu et al., 2012; Lange et al., 2011). For instance, new attacks (Simon et al., 2016; Aviv et al., 2012; Xu et al., 2012; Cai and Chen, 2011) enable adversaries to deduce keyboard input on touchscreens through "sensor readings from native apps" (Spreitzer et al., 2016; Kambourakis et al., 2016; Aviv et al., 2012). Because typing on various places on the screen creates different vibrations, data from Motion (Cai and Chen, 2011), a SCA on touch screen smartphones with soft keyboards data, can be employed by an attacker to deduce the keys being typed. One of the methods to deduce keystrokes via the Motion, is to utilise a mobile application such as TouchLogger, an Android application that derives "features from device orientation data" (Cai and Chen, 2011). More advanced and new attacks can also enable the attackers to infer a user's geographical location through the power consumption (Grus, 2017; Spreitzer et al., 2016; Mangard et al., 2008) and a victim's identity through the procfs (Spreitzer et al., 2016; Zhou et al., 2013) that is available from the proc filesystem (procfs) (Spreitzer et al., 2016; Michalevsky et al., 2015).

In the following section, we shall provide a brief review of various TBSCAs in relation to components of modern and shared PC platforms with references made also to other platforms (such as cloud computing and smart mobile phones, the analysis of which is beyond the scope of this paper) when relevant. To this end, a particular focus will be placed on TBSCAs with again making references to other variations of SCAs only when appropriate.

3. Review of Side-Channel Attacks Proposed in the Literature

Various researchers (Liu et al., 2015; Zhang et al., 2012; Aciiçmez et al., 2010; Tromer et al., 2010; Aciiçmez and Schindler, 2008; Aciiçmez, 2007; Osvik et al., 2005; Percival, 2005) have utilised Prime+Probe as a method of attack against different processor caches such as the L1 data cache, L1 instruction cache and also the branch prediction cache. For instance, Liu et al. (2015) demonstrated that a new type of Prime+Probe can be exploited for LLC attacks. Using their Prime+Probe attack technique, Liu et al leveraged hardware elements that are beyond the control of the cloud provider but often activated in

the VMM for operation reasons. Kim et al. (2012) proposed an StealthMem OS-based Technique, which is a system-level protection against TBSCAs in virtualised environments. StealthMem is based on software mechanism that places pages of a virtual machine into the cache and stops their ejection by other VMs. In this respect, Barthe et al. (2014) have provided a formalisation of such an approach. Raj et al. (2009) suggested system-level protections for segregating machines in virtualised environments, including "cache-aware CPU core assignment" and "cache-aware memory-management".

Ford (2012) proposed information-flow control which is based on clear timing labels, along with OS support for its implementation. Wang and Suh (2012) demonstrated that on-chip network often dynamically shared between applications that running at the same time on a chip-multiprocessor (CMP) can facilitate timing attacks. In their study, Wang and Suh (2012) showed that these shared resources suggest that applications can impact each other's timing features via interference in shared resources, and revealed that such an interference is an attack vector by means of which a malign application can deduce data-dependent information about other applications. Ristenpart et al. (2009) illustrated that by performing a SCA, an adversary could potentially co-locate and detect co-location in public IaaS clouds. Similarly, Zhang et al. (2011) showed that a tenant could identify co-location in the same core by monitoring the L2 cache. In the same vain, Bates et al (2012) proposed a co-location test based on network traffic analysis. Zhang et al. (2014) revealed that the de-duplication allows co-location detection from co-located VMs in PaaS clouds. Inci et al. (2016) demonstrated that by performing the Prime+Probe attack, an attacker could extract an RSA secret key from a co-located instance in cloud environments. Yarom and Falkner (2014) showed that by performing the Flush+Reload Attack during memory de-duplication, an adversary would be able to extract RSA secret keys across co-located VMs. Likewise, Bhattacharya and Mukhopadhyay (2015) demonstrated that by exploiting the branch prediction performance counters, an attacker could deduce RSA keys.

Irazoqui et al. (2014) performed a cross-VM Flush+Reload Attacks against VMs in VMware by exploiting resource sharing features in virtual environments. As a result, they were able to extract an AES' keys in OpenSSL 1.0.1 running inside the victim VM. Wang et al. (2014) illustrated that shared memory controllers are susceptible to SCAs that take advantage of memory interference as timing channels. Likewise, Evtyushkin et al. (2015) demonstrated that covert channels shared between processor resources can facilitate secret communication between malign processes. Utilising the processor branch prediction unit, Evtyushkin et al. (2015) showed the way in which a trojan and a spy can potentially compromise the branch prediction tables (BPT). Similarly, Hunger et al (2015) demonstrated a covert channel through branch predictor that can transmit a 1 by executing a significant number of branches, each of which is taken with 50% probability that reduces the branch prediction accuracy of the receiver. Although both Evtyushkin et al.'s (2015) and Hunger et al's (2015) studies are similar in that each introduced a new covert channel that can enable adversaries to launch timing attacks, they have differences in the way of their implementations of these covert channels. For instance, in Hunger et al's (2015) covert channel, the malicious activities during the transmission of a 1 generates the branch predictor property conflict, that is identified by the spy utilising performance counters. On the contrary, in Evtyushkin et al.'s (2015) proposed attack, a covert channel does not depend on branch predictor conflict but instead utilises the remaining state of the BPT. Furthermore, Evtyushkin et al.'s covert channel can be created without depending on performance counters by observing the runtime of the spy.

Attackers can also exploit hardware threading to examine a competing thread's L1 cache usage in real time (Ge et al., 2016; Percival, 2005). Simultaneous multithreading (the sharing of the operation resources of a superscalar processor between multiple execution threads) is a feature implemented into Intel Pentium 4 processors. Under this implementation, the sharing of processor resources between threads spreads beyond the operation units. This denotes that the threads also share access to the memory caches. Such shared access to memory caches can facilitate side channels and enable a malign thread with restricted privilege to scan the operation of another thread. In turn, this results in allowing the attackers to steal cryptographic keys (Genkin et al., 2015; Liu et al., 2015; 2015; Percival, 2005). In addition, by exploiting side-channel information based on CPU delay adversaries could potentially mount TBSCAs against the Data Encryption Standard (DES) implemented in some applications. Such cryptanalysis technique applies side-channel information on encryption processing to gather plaintexts for cryptanalysis and infers the information on the extended key from the acquired plaintexts (Tsunoo et al., 2003). Through this attack, the adversary will be able to break the cipher with plaintexts.

Furthermore, Time-Driven Attacks can be performed against AES (Crane et al., 2015; Gullasch et al., 2011; Coppens et al., 2009; Bernstein, 2005, Percival, 2005) in a virtualisation setting (Weiß et al., 2014; Kim et al., 2012; Gullasch et al., 2011). An example of such an attack is that of the "PikeOS Microkernel Virtualization Framework" (Weiß et al., 2014) which was successfully mounted against AES on an actual CPS. KASLR has also been shown to be bypassed by applying the branch-target buffer. In their study, Evtyushkin et al. (2016) were able to locate the place in the kernel in which code had been run based on the mapping from virtual addresses to the branch-target buffer cache lines. Furthermore, a malign operating system can also reverse-engineer the control flow of SGX enclaves via branch-prediction analysis (Lee et al., 2016). In addition, adversaries might be able to mount Timing Attacks against secret-dependent data access patterns on the sliding-window modular exponentiation implementation (Liu et al., 2015; Aciiçmez et al. 2007; Brumley and Boneh, 2005). The scatter-gather technique, which is a commonly-implemented method to stop time-based attacks, can also be exploited through a timing attack called CacheBleed (Yarom et al., 2017). CacheBleed, which is also a Timing-Based Side-Channel Attack, takes advantage of "cache-bank collisions" (Fog, 2017; Intel®, 2016) to generate quantifiable timing differences (Ge et al., 2016).

Instruction Cache (I-cache), a processor component, poses grave security susceptibilities, and as a result, it can be exploited in a TBSCA as a source of information emission. An I-Cache attack is capable of exposing full operation arrangement of a cryptosystem during a single execution. In this regard, Aciiçmez (2007) proposed a software-based I-cache attack on OpenSSL's RSA implementation. To demonstrate that I-cache attacks are capable of exposing the execution flow of cryptosystems such as RSA, that can result in a full breach if the cryptosystem is enforced with key-dependent execution flow. I-cache is capable of breaking into security systems even when robust security mechanisms such as sandboxing and virtualisation have been implemented as I-Cache leverages deep processor functionalities that are below the trust architecture boundary of such security implementations.

Moreover, by performing a Cache Template Attack (CTA) (Gruss et al., 2015) adversaries will be able to profile and take advantage of cache-based information emission of programs automatically. To carry out this attack, attackers do not require to know the specific software versions or system information in advance. The CTA includes two stages

consisting of profiling phase and exploitation phase. In the profiling stage, the attacker establishes dependencies between the processing of private information such as certain key inputs or secret keys of cryptography and particular cache accesses. In the exploitation stage, the adversary exfiltrate the private keys having monitored cache accesses. Besides, the CTA has been shown to be capable of deducing keystrokes as well as detecting particular keys on Linux and Windows user interfaces. By performing this attack, adversaries will be able to conduct automated attacks on the Ttable-based AES design of OpenSSL. In addition, by exploiting the vulnerabilities found in OpenSSL implementations, such as their accesses to different data structures during square and multiplication processes, attackers will be able to deduce the process sequence of RSA by observing the cache activities (Percival, 2005).

Likewise, adversaries can perform SCAs against ciphers that utilise table-lookups in large tables and AES (Daemen and Rijmen, 2013; NIST, 2001) because of the ciphers' susceptibilities to such attacks (Osvik et al., 2006; Bernstein, 2005; Tsunoo et al., 2003). Through a SCA against ciphers, attackers will be able to acquire the secret key by taking advantage of the time that an AES process takes because of the storage of the large table in cache. Last, but not least, an adversary can also leverage hardware-based information emission by calculating the power consumption with an oscilloscope to launch a TBSCA. In this context, the attacker will need to have physical access to the victim device

4. Countermeasures against Side-Channel Attacks

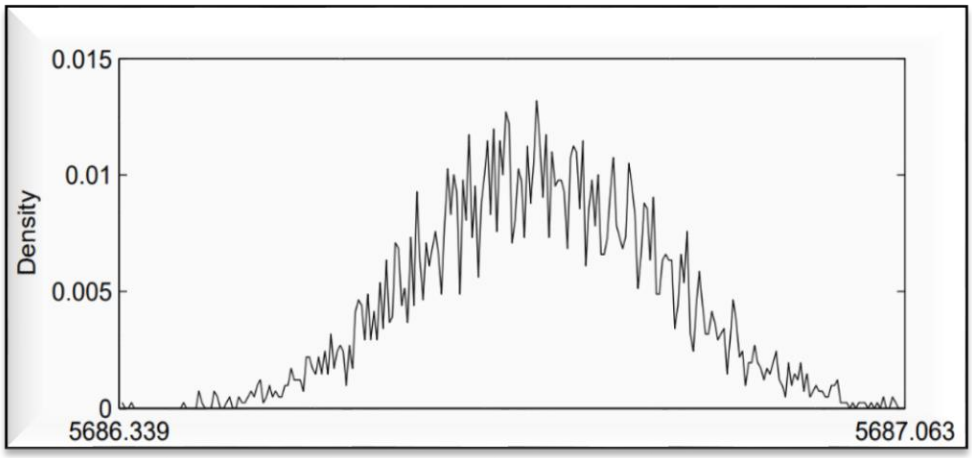
This section provides potential countermeasures that can be adopted to mitigate TBSCAs against components of PC platforms such as a processor, a cache or memory as "The most commonly exploited leakage in the shared resource systems stem from the cache and the memory" (Inci et al., 2016). Despite the focus being on countermeasures for such components, nevertheless we make brief references to countermeasures for other platforms such as cloud environments.

4.1 Noise Injection

One way of mitigating a TBSCA is to inject noise into the adversary's timing measurements in order to render the timing calculation ineffective (Ge et al., 2016; Hu, 1992). Any form of SCAs such as Flush+Reload will require the availability of "High-Resolution Clock" (Yarom and Falkner, 2014). Therefore, reduction in the resolution of clock or injecting noise to the clock measurement (Vattikonda et al., 2011; Hu, 1992) can be applied as a defence mechanism against TBSCAs. However, this countermeasure is restricted in its application because the adversary will be able to utilise other attack vectors to create high resolution clocks (Yarom and Falkner, 2014). Furthermore, Fuzzy Time (Hu, 1992), a specific type of Noise Injection implemented in VAX security Kernel, includes a set of techniques that lowers bandwidths of Timing Channels by injecting too much noise to all clocks that are available to a computing operation such as pre-emptions and interrupt delivery. Compacting, Randomising and Preloading S-Box tables (implemented on most modern processors) (Brickell et al, 2006, a) can also be utilised to inject noise to the cache footprint emitted through AES operations (Ge et al., 2016). This technique helps to safeguard against a timing attack mounted by an adversary who cannot monitor cache access performance more often than the time needed by the cryptographic operation to run an AES session (Brickell et al., 2006, a; Bernstein, 2005). Therefore, the distribution of AES execution times for this implementation of AES, in which execution times are averaged, follows a "Gaussian" distribution over a significant number of random

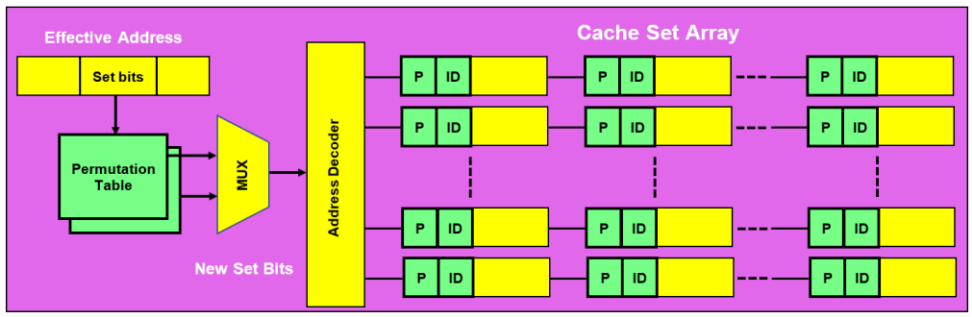
plaintexts. Figure 1., obtained from Brickell et al. (2006, a), represents such a measurement that has been taken based on Bernstein's (2005) study.

Figure 1. Average execution time cycle showing the distribution of execution times for Brickell et al.'s (2006, a) implementation of AES (Brickell et al., 2006, a).



However, such techniques can bring about 100% to 120% performance overhead (Ge et al., 2016). Partition-Locked cache (PLcache) and Random Permutation cache (RPcache) (Wang and Lee, 2007) can also be employed as countermeasures in order to randomise cache indexing scheme and safeguard every attribute in every cache line. The "RPcache" is responsible for achieving the permutation of the memory-to-cache mapping. This is performed by applying indirection in cache indexing. A given process in RPcache contains a Permutation Table, that stores memory-to-cache mappings as demonstrated in Figure 2, adopted and modified from Wang and Lee's (2007) study.

Figure 2. The logical view of Wand and Lee's proposed RPcache implementation (Wang and Lee, 2007).



The number of entries in the table are equal to those in the cache sets, and each given entry includes a different "M-bit number", that signifies the new set. Then, the permutation table is indexed with the M set bits for every specific cache access for every given cache access in order to acquire the new set bits, that are utilised to store the selection of the cache set. Every cache line includes an ID showing the process of its owner (Ge et

al., 2016; Wang and Lee, 2007). Those cache lines that have different IDs will not be able to eject each other. Instead, RPcache chooses a set of cache in a random manner and eject a cache line in the given set. Kong et al. (2009) question Wang and Lee's (2007) proposed Partition-Locked cache (PLcache) and Random-Permutation cache (RPcache) methods arguing that despite the fact that they are effective in mitigating performance overhead and improving the security level, they might still be susceptible to advanced attacks themselves. As a result, they proceeded to propose their own countermeasure that they claimed to introduce various "trade-offs between hardware complexity and performance overhead". This countermeasure, which is claimed to have low overhead, is based on the implementation of a 'specific' RPcache, that is aimed only at secret data including AES tables.

The Xen hypervisor can also be adapted so that noise can be injected into the high-resolution time measurements in VMs. This can be achieved by changing the values that are returned by the rdtsc instruction (Ge et al., 2016; Vattikonda et al., 2011). In addition, a timing channel can also be mitigated by rendering internal time sources imprecise; for instance, the implementation of the x86 rdtsc instruction can be adapted to stop operation till the end of a "a predefined epoch", and then inserting a randomised number "between zero and the size of the epoch", therefore fuzzing the time counter (Ge et al., 2016; Martin et al., 2012). In a cloud setting, bystander VMs can be employed to insert noise on the "cross-VM L2-cache covert channel" that has a configurable workload (Ge et al., 2016). Such a method can enable bystander workloads (on each server) to create high level of error to timing channels and therefore reduce the threat of cross-VM timing channels (Zhang et al., 2015).

It should be noted that many of the existing solutions such as noise injection cannot provide a high level of security, a point also acknowledged by Cock et al. (2014). There are ways to increase the level of security offered by the existing solutions. For instance, creating anti-correlated noise can in theory block the timing channel. However, creating such trade-off is impractical in most situations. The level of actual noise needed grows significantly with the declining channel capacity. Therefore, this will result in significant degradation in system performance and renders it impossible to decrease channel bandwidth by "more than about two orders of magnitude" (Ge et al., 2016; Cock et al., 2014).

4.2 Lattice Scheduling

Arguing that hardware sharing results in timing-channel susceptibilities in hardware components including memory controllers and shared memory, Ferraiuolo et al. (2016) proposed Lattice Priority Scheduling (LPS), which is a memory scheduling algorithm that enhances performance by accurately fulfilling the target system's security requirements. Similarly, Wang et al. (2014) proposed a memory controller implementation that allows secure sharing of main memory between mistrusting parties by eradicating memory timing channels. Wang et al. (2014) determined the sources of interference in a conventional memory controller implementation and further introduced a protection design to remove the interference from security domains. Similarly, Cock et al. (2014) presented a "Lattice Scheduler" for the domain-switched version of seL4. Applying their "mechanisation of the probabilistic program logic pGCL", Cock (2013) proposed a lattice scheduler (a mitigation method) using randomisation as a way of ensuring "starvation-freeness". Cock (2013) illustrated that their lattice scheduler imposed "probabilistic non-

leakage" as well as "probabilistic non-leakage". This scheduler has been implemented in such a way to reduce information flow via a shared cache while ensuring fairness, simplicity and efficiency.

4.3 Countermeasures against the Flush+Reload Attack

To carry out a Flush+Reload Attack, an adversary will need to consider a set of four aspects (Yarom and Falkner, 2014), including: data flow from sensitive data to memory access patterns, memory sharing between the malicious and the victim processes, precise time calculations and the unrestricted use of the clflush instruction. Therefore, blocking any of these four elements will prevent this type of attack from occurrence. The X86 design does not provide permission checks for utilising the clflush instruction. The Flush+Reload Attack takes advantage of the fact that there are no limitations in relation to the usage of the clflush instruction. The absence of such usage limitations can be considered as a security laxity caused by Intel architecture of the X86 design. Therefore, one countermeasure would be to restrict the power of the clflush instruction, the key aim of which is to apply memory consistency (Intel® Corporation, 2016; Yarom and Falkner, 2014).

Software diversification (Forrest, 1997) can also be employed as a countermeasure to block memory page sharing and, therefore, to reduce the likelihood of the Flush+Reload Attack (Yarom and Falkner, 2014). Also, in virtual and cloud settings, Static Memory Deduplication (Kil et al., 2006; Bhatkar et al., 2003; Forrest et al., 1997) can be employed to generate duplicates of programs in each given virtual machine. Since such duplicates are not present beyond the boundary of the given VM, the programme pages will not be copied, and sharing will be blocked. Diversifying the programme at execution time (Curtsinger and Berger, 2013) will block sharing of the programme text even if the adversary has been able to access the binary file (Yarom and Falkner, 2014). Another countermeasure to mitigate the occurrence of a Flush+Reload Attack is to turn off memory deduplication (Yarom and Falkner, 2014). Similarly, preventing page sharing can block the occurrence of this attack.

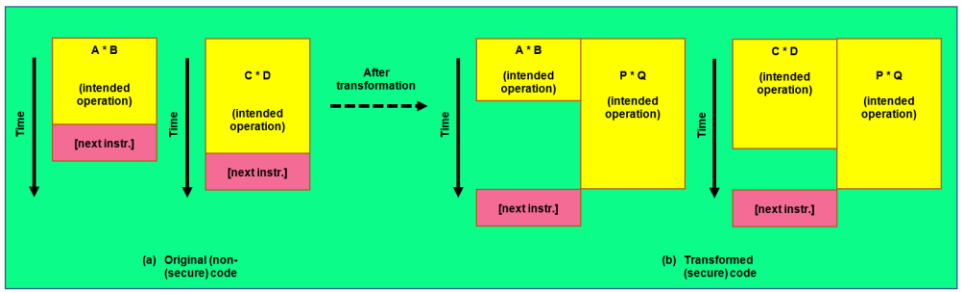
4.4 Constant Time Techniques

One method of safeguarding cryptographic code is to implement the code in such a way that its performance will not be reliant upon data. This denotes, for instance, that the order of cache accesses will not rely on the key or plaintext. This technique is common to address the runtime and also to be used in local contention-based channels (Ge et al., 2016; Bernstein, 2005). In this regard, Brickell (2011) recommends not using secret-dependent memory access "at coarser than cache line granularity". Similarly, others (Bernstein et al., 2013; Osvik et al., 2006) have argued that processors such as Intel processor can emit low-address-bit information (this is the offset in a cache line). This argument is further backed up by Yarom et al. (2017), who were able to demonstrate that OpenSSL implementation was susceptible to the CacheBleed attack. Coppens et al. (2009) revealed numerous leaks such as instructions with "data-dependent runtimes, register dependencies and data dependencies through memory". However, no implementation is so far understood to have been compromised by side-channel attacks via such leaks. Other researchers (Valgrind, 2017 and Langley, 2010) have introduced examination applications to direct the implementation of constant time code. These tools can monitor the flow of secret information to inform whether such information is being utilised in branches or as a memory index (Ge et al., 2016).

Köpf et al. (2012) introduced a method for automatically extracting "upper bounds" on the quantity of information concerning the input that an attacker can derive from a program by monitoring the CPU's cache performance. This method, that facilitates the measuring procedure, is built on top of the "AbsInt TimingExplorer", which is claimed to be one of the most advanced engines for static cache analysis. Doychev et al. (2015) built upon Köpf et al's (2012) work and proposed CacheAudit, which is a multipurpose framework for the automatic examination of cache side channels that facilitates more effective abstractions and a higher accuracy. The contribution of Köpf et al's (2012) work consists of new abstractions to compute accurate "over-approximations" of the potential side-channel monitoring for attackers. Such approximations result in producing "upper bounds" on the quantity of information which is exposed.

Andrysco et al. (2015) proposed a benchmark that counts the timing mutability of floating point operations. As part of their study, Andrysco et al. (2015) also developed a "fixed-point, constant-time math library titled 'libfixedtimefixedpoint'" to mitigate the timing channel on floating-point operations. Rane et al. (2016) introduced a countermeasure that blocks side channels introduced by floating-point computations while maintaining the accuracy of non-secure operations. This countermeasure takes advantage of microarchitectural elements of the SIMD lanes in x86 SSE and SSE2 instruction sets architecture to facilitate fixed time floating-point executions. The main perception underlying Rane et al.'s Escort's secure executions as shown in Figure 3, acquired and modified from Rane et al.'s (2016) study, is that the latent period of SIMD instructions are decided by the lowest execution between the SMID lanes.

Figure 3. The main insight underlying Rane et al.'s Escort's secure executions (Rane et al., 2016).

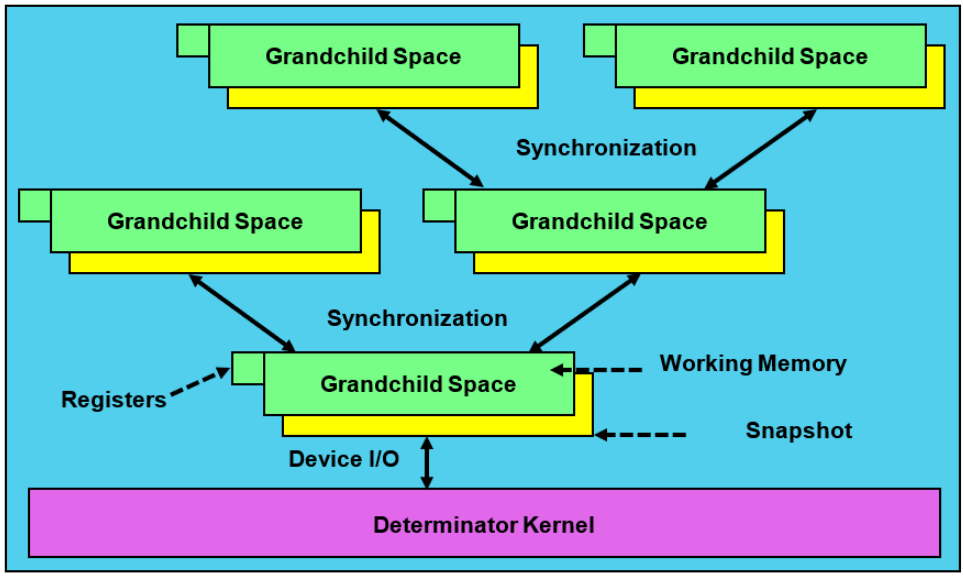


Thus, this Escort compiler can facilitate the execution of each instruction together with a dummy instruction, the operand of which will result in generating the longest latent period. Rane et al.'s (2016) assessment reveals a 0 to 1.5% timing difference of floating-point executions with various types of inputs on the testing machine. Ge et al. (2016) argues that the key shortcoming of such a method is that a constant-time operation on one hardware platform is likely not to perform on a different hardware platform. To attest their point, Ge et al. (2016) refer to Cock et al.'s (2014) study, in which Cock et al. illustrated that the constant-time fix for preventing Lucky 13 attack (Al Fardan and Paterson, 2013), a type of a remote SCA, in OpenSSL 1.0.1e still comprises a side channel on the ARM AM3358 platform.

4.5 Virtual Time

The virtual time method attempts to eradicate access to real time, presenting only virtual clocks, the progress of which is entirely deterministic, and independent of the operations of susceptible elements. Deterministic operation can present various advantages, for instance, in relation to debugging, fault tolerance, and security (Ge et al., 2016; Aviram et al., 2012). However, existing techniques of performing parallel programs tend to bring about high costs and permit misbehaved software applications to outdo repeatability (Aviram et al., 2012). Therefore, to deal with such issues, one can employ a parallel programming paradigm in combinations with "Determinator" (Aviram et al., 2012) that should be implemented in a way to transform write/write conflicts into identified contentions conflicts as shown in Figure 4, obtained and modified from Aviram et al.'s (2012) Study.

Figure 4. The kernel's hierarchy of spaces with each having private register and virtual memory state (Aviram et al., 2012).



Ford (2012) built upon the Aviram et al.'s (2012) model by presenting queues to enable carefully I/O with the outside world without introducing real-time clocks. Similarly, Wu et al. (2015) built upon Ford et al.'s (2015) model by introducing a prototype "hypervisor-enforced timing mitigation" to regulate timing channels in cloud environments. This method blocks "reference clocks" that are internal to the cloud by enforcing a deterministic view of time on guest code, and employs "timing mitigators to pace I/O and rate-limit potential information emission to external observers.

4.6 Countermeasures against Timing Attacks on RSA

One method to mitigate Timing Attacks against RSA is to cause the decoding process to take a continuous amount of time for every ciphertext (the outcome of encryption carried out on plaintext employing an algorithm). It should be noted that this method can considerably affect performance. Therefore, to avoid the reduction in

performance, cryptographic blinding technique (CBT) can instead be employed in the RSA implementations. The CBT takes advantage of RSA's multiplicative property. This denotes that rather than computing cd (mod n), one should select a secret random value r and compute (rec)d (mod n). The outcome of such computation process when utilising Euler's Theorem will be rcd (mod n). Thus, the impact of r can be eliminated through multiplication of its reverse. A new value of r must then be selected for each ciphertext. After the CBT has been applied successfully, the decoding time will no longer be associated with the value of the input ciphertext. As a result, the TBSCA will end in failure. CBT, itself, might be prone to Timing Attacks by 'very advanced' attackers. However, at this stage, we do not have any data that can substantiate such an assumption. However, as a future work, we are aiming to carry out a set of experimental Timing Attacks against the CBT to determine whether we can substantiate this assumption by attempting to break the CBT (i.e. if we can exfiltrate the private key) either fully or partially.

4.7 Network On-Chip and Quashi Partitioning

By temporally-partitioning arbitrators, network capacity can be dynamically assigned to domain in order to mitigate TBSCAs by contending on "on-chip interconnects". To mitigate timing channel attacks enabled by on-chip network, Wang and Suh (2012) proposed an implementation that employs "priority-based arbitration" and a "static limit mechanism" to facilitate one-way information-emission protection. Wang and Suh's (2012) results revealed that the protection design eradicated a timing channel from high-security to low-security domains while performance overheads were kept to minimum for traffic patterns. Also, decreasing the conflict and efficacy of TBSCAs also requires stopping an adversary from dominating resources. Zhou et al. (2016) proposed an approach to prevent TBSCAs that exploit LLCs (the last level caches) shared between cores to emit information between security domains. As part of their study, Zhou et al. (2016) implemented the CacheBar, a memory management subsystem, to prevent TBSCAs within cloud environments. The CacheBar dynamically ejects memory contents from the cache so that protection domains reside only in a restricted quantity in each cache set. According to Ge et al. (2016), CacheBar is fundamentally a software design of the solution that Domnitser et al. (2012) proposed.

4.8 Performance Counters

Counter-based detection along with machine learning methods can also be utilised in PC platforms to detect TBSCAs and restrict the level of emitted information (Chiappetta, 2016). Furthermore, performance counters can also be applied within a cloud environment to identify cross-VM SCAs. In this regard, CloudRadar (Zhang et al., 2016, a) implementation can be used to identify SCAs in multi-tenant cloud systems. This implementation functions by taking advantage of signature-based detection to determine the cryptography process performed by the safeguarded virtual machine (VM). It also utilises "anomaly-based detection" methods to observe the co-located VMs so that abnormal cache behaviours (that are common in SCAs) can be detected. CloudRadar appears to be an effective implementation in mitigating SCAs within a cloud setting. It uses "metamorphic attack" code which makes the implementation difficult to bypass by the attackers. A similar implementation setting titled "Cloak" (Gruss et al., 2017, a), which is also designed for application in a cloud setting, can deal with SCAs in multi-tenant settings. This implementation leverages "hardware transactional memory" to block the attackers from monitoring cache misses of secret code and data. Cloak has been reported to be capable of blocking data emission (which is caused through side channels) from "enclaves"

dealing with one of the shortcomings of SGX (Gruss et al., 2017, a). Figure 5. demonstrates the code for decision tree categorisation of Gruss et al.'s (2017, a) Cloak.

Figure 4. The kernel's hierarchy of spaces with each having private register and virtual memory state (Aviram et al., 2012).

```
using Nodes = nelem_t*;
using Queries = Matrix<float>;
using LeafIds = uint16_t*;

using Nodes = ReadArray<nelem_t, NCS_R>;
using Queries = ReadMatrix<float, NCS_R>;
using LeafIds = WriteArray<uint16_t, NCS_W>;

void _tsx_protected_lookup_leafids(
Nodes& nodes, Queries& queries, LeafIds&
    leafids) {
  nodes.preload();
  queries.preload();

  for (size_t q=0; q < queries.entries();
     q++) {
    if (!(q % 8)) leafids.preload();
    size_t idx = 0, left, right;
    for(;;) {
      auto &node = nodes[idx];
      left = node.left;
      right = node.right_or_leafid;
      if (left == node) {
        leafids[q] = right;
        break;
      }
      if (queries.item(q, node.fdim) <=
    node.fthresh)
        idx = left;
      else
        idx = right;
    }
  }
}
```

4.9 Defence Methods Suggested in the Literature

Kocher (1996) stated that attackers could potentially identify fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems by calculating time needed to perform private key executions. As a result, he proposed various techniques for mitigating the attacks against RSA and presented Diffie-Hellman. Brickell et al. (2006, b) proposed several countermeasures against SCAs. They state that permuting the AES lookup tables can mitigate Access-Driven Attacks and suggest the uses of smaller lookup tables such as original AES S-box, during the first and last rounds of AES computations (Aciiçmez, 2009). Aciiçmez et al. (2007) examined the ability of the RSA implementation of OpenSSL-0.9.7. To this end, they focused on Branch Prediction Analysis and identified multiple several flaws that needed to be addressed. Acıiçmez et al. (2007) informed the OpenSSL organisation of such vulnerabilities and suggested the removal of certain conditional branches that could impact the strength of RSA implementations. As a result, OpenSSL organisation considered the proposal and made the suggested modification to their RSA implementations. In the same vain, Agosta et al. (2007) proposed a defence mechanism for susceptibilities found in branch prediction. In this regard, they proposed the implementation of conditional branches via indirect branching.

Stating that SCAs could compromise a system's confidentiality by exploiting information leakage from the victim system, Zhang and Lee (2014) suggested an Interference Matrix to assess a system's susceptibilities to SCAs. Inam et al. (2014) proposed an implementation of the Multi-Resource Server (MRS) which allows predictable performance of real-time applications on multi-core platforms. The MRS presents temporal abstraction between operations executed on the same core and operations executed on different cores. The latter could, without MRS, interfere with each other due to contention on a shared memory bus. Inam et al. (2014) illustrated that that the MRS can be applied "to encapsulate legacy systems" and to provide adequate resources to achieve their goal. Doychev et al. (2015) suggested a framework titled "CacheAudit" for the automatic, static examination of side channels. Hunger et al. (2015) presented a method for detecting savvy attackers who attempted to hide while executing covert channel eavesdropping attacks. Chiappetta et al. (2016) proposed a method for real time detection of SCAs by employing hardware performance counters. Arguing that SCAs would pose security threats in multi-tenant environments such as modern cloud data centers, Gruss et al. (2017, a) proposed Clock to mitigate malicious monitoring of cache misses on secret code and data.

In addition to the above suggested defence mechanisms, Selective Partitioning and the Random Permutation Cache (RPCache) can also be used as counter-measures to prevent certain Cache-Based Side-Channel Attacks with minimal hardware costs and insignificant performance effect (Wang and Lee, 2007). Covert Channels that are based on Intel Quick Path Interconnect (QPI) lock mechanism have also been identified (Gruss et al., 2017, a; Gruss et al., 2017, b; Wu et al., 2012). One method of mitigating TBSCAs is to utilise per-domain queuing structure and static allocation of time slots in the scheduling algorithm (Evtyushkin et al., 2015; Wang et al., 2014). A framework titled 'CC-Hunter" has also been proposed to identify the presence of covert channels by dynamically tracking conflict patterns over employment of shared processor hardware (Evtyushkin et al., 2015; Chen, J. and Venkataramani, 2014). Since the CCHunter is built around identifying conflict, it is not relevant to identifying the covert channels via branch predictors as such channels are not generated based upon conflict. Gate-Level Information Flow Tracking (GLIFT), a mechanism that constructs the system from the ground up, can also be employed to identify TBSCAs (Oberg et al., 2014; Tiwari et al., 2009). Evtyushkin et al. (2015) state that while GLIFT can be effective, it necessitates substantial "rearchitecting and redesign of the entire system".

One of the countermeasures against Evict+Time Attacks is to adopt Complex Addressing Functions and Replacement Policies of modern processors that can render ejection harder and as a result Evict+Time Attacks more difficult. There exist other countermeasures focusing on identifying side-channel emissions, for instance through static source code analysis (Doychev et al., 2015; Köpf et al., 2012) or by conducting "anomaly detection" employing CPU performance counters (Gruss et al., 2017, a; Gruss et al., 2017, b; Chiappetta et al., 2015). Gruss et al. (2016) delved into the latter method and implemented an attack technique titled "Flush+Flush", a variation of Flush+Reload, that could bypass it. The Flush+Flush attack depends only on the runtime of the flush instruction and data that is cached. This technique does not make "any memory accesses" as opposed to other cache attacks. As a result, it does not generate cache misses. Consequently, the Flush+Flush attacks are very quiet. This denotes that the "spy process" cannot be noticed based on cache hits and misses.

Last, but not least, the usage of caches in the memory system can result in considerable reduction of efficient memory access time even though it is one of the most essential characteristics for enhancing performance in modern processors. Nevertheless, the different access time features because of cache hits and misses create information emission that an attacker can exploit. Therefore, to mitigate this threat, one can simply deactivate cache. However, this approach can significantly degrade the system performance which becomes another major issue on its own right.

5. Discussion

In this extended study, we provided an overview of various Timing Attack vectors, followed by a detailed examination of countermeasures that can be employed to mitigate such attacks. Through the findings of the study, we can deduce that despite the many advancements in computer security, SCAs continue to evolve and wreck a havoc on shared, modern computing hardware. Our study shows that security researchers are simply playing a catch-up game with criminals. This is demonstrated by the various sophisticated TBSCAs that have been illustrated in the literature that can pose various security threats against the users of such systems. Everytime a new side channel susceptibility is discovered, a new mitigation research path also begins. As a result, security researchers have proposed several countermeasures and continue to suggest new ones. However, despite being useful, these proposed countermeasures are often limited to academic suggestions and are either ineffective or unable to address practical TBSCAs fully. Furthermore, almost all of the existing solutions incur significant overheads or degradation in computing performance, which, in turn, weakens the optimisation of modern computing hardware such as processors. The level of such overheads and degradation in hardware performance, often caused by the methods, themselves, has not been adequately reported in the research papers. Thus, it is difficult to assess the effectiveness of such proposed countermeasures. It is also reasonable to state that an interaction between Microarchitectural side channels on one hand and inclination to optimise hardware performance via Microarchitectural improvements on the other hand have given rise to the absence of reliable defence mechanisms when producing hardware components.

Adding to the above aforementioned issues, it is clear that the existing countermeasures for TBSCAs are not proactive but reactive in that they are there to mitigate single, specific attacks that are known to the research community. Nor are they generic to be able to address future, unknown attacks that take advantage of other variations of SCAs. Moreover, there is very little information provided in the existing studies with regards to false positives and negatives of the proposed defence techniques. Likewise, although the effectiveness of some of these defence mechanisms can be evaluated against various criteria, such assessment has been limited to qualitative analysis, a point also acknowledged by Zhang and Lee (2014).

6. Research Directions

In light of the discussion presented in the previous section, as potential directions for future research, we, therefore, recommend the followings:

1. The focus of many studies has been on side channels, for instance via branch prediction units. Thus, it is imperative that future studies take into account mitigation methods that can address the potentials for both side channels and covert channels, for instance, via shared branch prediction units and also other

shared properties.

2. Generic countermeasures will need to be devised that can be independent of both application and architecture to mitigate the majority of known and unknown TBSCAs. Such countermeasures must make it extremely difficult for an adversary to be able to differentiate between various types of side channels. We believe that it is by adopting such an approach that a SCA can be proactively foiled in advance of the attack being carried out.

3. In relation to the point 2, proactive countermeasures must not necessitate a significant modification of hardware and must not certainly result in degradation of system performance or an increase in overheads.

4. To provide accurate measurements in relation to the efficacy of the proposed countermeasures, quantitative methods will need to be developed to be able to determine the true potential values of such countermeasures.

5. Aciiçmez et al. (2007) report several attack scenarios aimed at RSA that take advantage of the CPU's Branch Prediction Unit. Therefore, developing effective countermeasures against these attacks can be considered a feasible direction for future research. For instance, studies need to be performed to determine whether these attacks can be successfully carried against symmetric-key algorithms that utilise the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. In cases where an algorithm is proved to be susceptible, changes will need to be applied to it.

6. All Microarchitectural attacks, irrespective of their type, can exploit security systems regardless of advanced partitioning methods (e.g. memory protection), sandboxing or even virtualisation. Hence, it is vital to identify every conceivable Microarchitectural susceptibility in order to comprehend the potential of Microarchitectural analysis and design to implement more secure systems. This can be achieved by employing appropriate software countermeasures and making specific hardware changes in future architectures.

7. The topic of SCAs against mobile devices is a new area of research that still lacks adequate studies. Thus, more works need to be carried out in the domain of side channels in mobile devices. From the discussion presented in this study, it can be deduced that the existence of specific multi-threading architectures necessitates a profound comprehension of the interaction between the underpinning hardware and software so that one is able to evaluate the true ramifications of the discussed security more effectively.

8. Last, but not least, as future work, we intend to identify more side channels resulting from the inherent vulnerabilities that exist in PC hardware components such as processors and caches, and find solutions to this increasing security threat. We believe that this study will inspire new studies in the implementations of secure PC hardware that do not undermine performance, cost and energy consumption.

References

Aciiçmez, O., Brumley, B.B. and Grabher, P. (2010). New Results on Instruction Cache Attacks. Proceedings of 12th International Workshop on Cryptographic Hardware and Embedded Systems, pp. 110-124. Santa Barbara, USA.

Aciiçmez, O., 2009. Microarchitectural Attacks and Countermeasures. In Cryptographic Engineering, pp. 475-504. Springer, Boston, MA.

Aciiçmez, O. and Koç, Ç, K. (2009). Microarchitectural Attacks and Countermeasures. In Cryptographic Engineering, pp. 475-504. Springer, Boston, US.

Aciiçmez, O. and Schindler, W. (2008). A Vulnerability in RSA Implementations Due to Instruction Cache Analysis and Its Demonstration on OpenSSL. Proceedings of the Cryptopgraphers' Track at the RSA Conference on Topics in Cryptology, pp. 256-273.

Aciiçmez, O. (2007). 'Yet Another Microarchitectural Attack: Exploiting I-Cache'. Proceedings of the ACM Workshop on Computer Security Architecture, pp. 11-18.

Aciiçmez, O., Koç, Ç.K. and Seifert, J.P. (2007). Predicting Secret Keys Via Branch Prediction. Proceedings of the 7th Cryptographers' Track at the RSA Conference on Topics in Cryptology, pp. 225-242.

Agosta, G., Breveglieri, L., Pelosi, G. and Koren, I. (2007). Countermeasures against Branch Target Buffer Attacks. IEEE Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 75-79.

Al Fardan, N.J. and Paterson, K.G. (2013). Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. IEEE Symposium on Security and Privacy (SP), 2013, pp. 526-540. IEEE.

Andriotis, P., Tryfonas, T., Oikonomou, G. and Yildiz, C. (2013). A Pilot Study on The Security of Pattern Screen-Lock Methods and Soft Side Channel Attacks. Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 1-6.

Andrysco, M., Kohlbrenner, D., Mowery, K., Jhala, R., Lerner, S. and Shacham, H. (2015). On Subnormal Floating Point and Abnormal Timing. IEEE Symposium on Security and Privacy (SP), pp. 623-639.

Aviram, A., Weng, S.C., Hu, S. and Ford, B. (2012). Efficient System-Enforced Deterministic Parallelism. Communications of the ACM, 55(5), pp.111-119.

Aviv, A.J., Sapp, B., Blaze, M. and Smith, J.M. (2012). Practicality of Accelerometer Side Channels on Smartphones. Proceedings of the 28th Annual Computer Security Applications Conference, pp. 41-50.

Aweke, Z.B., Yitbarek, S.F., Qiao, R., Das, R., Hicks, M., Oren, Y. and Austin, T. (2016). ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks. ACM SIGPLAN Notices, 51(4), pp.743-755.

Barenghi, A., Pelosi, G. and Teglia, Y. (2010). 'Improving First Order Differential Power Attacks through Digital Signal Processing'. Proceedings of the 3rd ACM International Conference on Security of Information and Networks, pp. 124-133.

Barthe, G., Betarte, G., Campo, J., Luna, C. and Pichardie, D. (2014). System-Level Non-Interference for Constant-Time Cryptography. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 1267-1279.

Bates, A., Mood, B., Pletcher, J., Pruse, H., Valafar, M. and Butler, K. (2012). Detecting Co-Residency with Active Traffic Analysis Techniques. Proceedings of the ACM Workshop on Cloud Computing Security Workshop, pp. 1-12.

Bernstein, D.J., Chou, T. and Schwabe, P. (2013). McBits: Fast Constant-Time Code-Based Cryptography. In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 250-272. Springer, Berlin, Heidelberg.

Bernstein, D. (2005). 'Cache-timing attacks on AES'. Available at: https://cr.yp.to/antiforgery/cachetiming-20050414.pdf (Accessed: 23rd May 2017).

Bhatkar, S., DuVarney, D.C. and Sekar, R. (2003). Address Obfuscation: An Efficient Approach to Combat a Broad Range of Memory Error Exploits. In USENIX Security Symposium, 12(2), pp. 291-301.

Bhattacharya, S. and Mukhopadhyay, D. (2015). Who Watches the Watchmen?: Utilizing Performance Monitors for Compromising Keys of RSA on Intel Platforms. International Workshop on Cryptographic Hardware and Embedded Systems, pp. 248-266. Springer, Berlin, Heidelberg.

Borkar, S. and Chien, A. A. (2011). 'The Future of Microprocessors.' Communications of the ACM, 54(5), pp.67-77.

Bosman, E., Razavi, K., Bos, H. and Giuffrida, C. (2016). Dedup Est Machina: Memory Deduplication as An Advanced Exploitation Vector. IEEE Symposium on Security and Privacy (SP), pp. 987-1004.

Brickell, E. (2011). Technologies to Improve Pl tf S it Platform Security.

Brickell, E., Graunke, G., Neve, M. and Seifert, J.P. (2006, a). Software Mitigations to Hedge AES against Cache-Based Software Side Channel Vulnerabilities. IACR Cryptology ePrint Archive, p.52.

Brickell, E., Graunke, G. and Seifert, J.P. (2006, b), February. Mitigating cache/timing based side-channels in AES and RSA software implementations. In RSA Conference 2006 session DEV-203.

Brumley, D. and Boneh, D. (2005). Remote Timing Attacks Are Practical. Computer Networks, 48(5), pp.701-716.

Brumley, D., and Boneh, D. (2003). 'Remote Timing Attacks are Practical'. Proceedings of the 12th USENIX Security Symposium, pp. 1-14.

Cai, L. and Chen, H. (2011). TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. HotSec, 11, pp.9-9.

Callan, R., Zajic, A. and Prvulovic, M. (2014). A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events. 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 242-254.

Carmon, E., Seifert, J.P. and Wool, A. (2017). Photonic Side Channel Attacks Against RSA. IACR Cryptology ePrint Archive, p.108.

Chari, S., Jutla, C.S., Rao, J.R. and Rohatgi, P. (1999). 'Towards Sound Approaches to Counteract Power-Analysis Attacks'. In: Wiener, M. (ed.) CRYPTO. LNCS, 1666, pp. 398–412. Springer, Heidelberg.

Chen, Q.A., Qian, Z. and Mao, Z.M. (2014). 'Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks'. In USENIX Security Symposium, pp. 1037-1052.

Chen, J. and Venkataramani, G. (2014). 'Cc-Hunter: Uncovering Covert Timing Channels on Shared Processor Hardware'. 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 216-228.

Clements, A. (2006). Principles of Computer Hardware. Fourth ed., Oxford University Press.

Chiappetta, M., Savas, E. and Yilmaz, C. (2016). 'Real Time Detection of Cache-Based Side-Channel Attacks Using Hardware Performance Counters'. Applied Soft Computing, 49, pp.1162-1174.

Cock, D., Ge, Q., Murray, T. and Heiser, G. (2014). The Last Mile: An Empirical Study of Some Timing Channels on seL4. In ACM Conference on Computer and Communications Security, pp. 570-581.

Cock, D. (2013). Practical Probability: Applying pGCL to Lattice Scheduling. In International Conference on Interactive Theorem Proving, pp. 311-327. Springer, Berlin, Heidelberg.

Coppens, B., Verbauwhede, I., De Bosschere, K. and De Sutter, B. (2009). Practical Mitigations for Timing-Based Side-Channel Attacks on Modern X86 Processors. 30th IEEE Symposium on Security and Privacy, pp. 45-60.

Crane, S., Homescu, A., Brunthaler, S., Larsen, P. and Franz, M. (2015). 'Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity'. NDSS, pp. 8-11.

Cryptography Research, Inc., Kocher, P.C., Rohatgi, P. and Jaffe, J.M. (2017). Secure Boot with Resistance to Differential Power Analysis and Other External Monitoring Attacks. U.S. US 9569623 B2.

Curtsinger, C. and Berger, E.D. (2013). Stabilizer: Statistically Sound Performance Evaluation. In ACM SIGARCH Computer Architecture News, 41 (1), pp. 219-228.

Daemen, J. and Rijmen, V. (2013). The Design of Rijndael: AES-the Advanced Encryption Standard. Springer Science & Business Media.

Domnitser, L., Jaleel, A., Loew, J., Abu-Ghazaleh, N. and Ponomarev, D. (2012). Non-Monopolizable Caches: Low-Complexity Mitigation of Cache Side Channel Attacks. ACM Transactions on Architecture and Code Optimization (TACO), 8(4), p.35.

Doychev, G., Köpf, B., Mauborgne, L. and Reineke, J., 2015. 'CacheAudit: A Tool for the Static Analysis of Cache Side Channels'. ACM Transactions on Information and System Security (TISSEC), 18(1), p.4.

Evtyushkin, D., Ponomarev, D. and Abu-Ghazaleh, N. (2016). 'Jump over ASLR: Attacking Branch Predictors to Bypass ASLR'. 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1-13.

Evtyushkin, D., Ponomarev, D. and Abu-Ghazaleh, N. (2015). Covert Channels through Branch Predictors: A Feasibility Study. Proceedings of the 4th Workshop on Hardware and Architectural Support for Security and Privacy, p. 5.

Faruque, A., Abdullah, M., Chhetri, S.R., Canedo, A. and Wan, J. (2016). Acoustic Side-Channel Attacks on Additive Manufacturing Systems. Proceedings of the 7th IEEE International Conference on Cyber-Physical Systems, p.19.

Flynn, M.J. (2007). Computer Architecture: Pipelined and Parallel Processor Design. Jones and Bartlett Learning.

Fog, A., 2017. The microarchitecture of Intel, AMD and VIA CPUs/An optimization guide for assembly programmers and compiler makers.

Ford, B. (2012). Plugging Side-Channel Leaks with Timing Information Flow Control. HotCloud, pp. 1-5.

Forrest, S., Somayaji, A. and Ackley, D.H. (1997). Building Diverse Computer Systems. The 6th Workshop on Hot Topics in Operating Systems, pp. 67-72.

Ge, Q., Yarom, Y., Cock, D. and Heiser, G. (2016). A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware. Journal of Cryptographic Engineering, pp.1-27.

Genkin, D., Pachmanov, L., Pipman, I., Tromer, E. and Yarom, Y. (2016). ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels. Proceedings of ACM Conference on Computer and Communications Security, pp. 1626-1638.

Genkin, D., Pachmanov, L., Pipman, I. and Tromer, E. (2015). Stealing Keys from PCs by Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. Cryptology ePrint Archive, Report 2015/170.

Genkin, D., Shamir, A. and Tromer, E. (2014). 'RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis'. In International Cryptology Conference, pp. 444-461. Springer, Berlin, Germany.

Gornik, A., Moradi, A., Oehm, J. and Paar, C. (2015). A Hardware-Based Countermeasure to Reduce Side-Channel Leakage: Design, Implementation, and Evaluation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34(8), pp.1308-1319.

Gruss, D., Schuster, F., Ohrimenko, O., Haller, I., Lettner, J. and Costa, M. (2017, a). Strong and Efficient Cache Side-Channel Protection Using Hardware Transactional Memory. In Proceedings of the 26th USENIX Security Symposium. Vancouver, Canada.

Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., Schoechl, W. and Yarom, Y. (2017, b). 'Another Flip in the Wall of Rowhammer Defenses'. arXiv preprint arXiv:1710.00551, pp. 1-17.

Gruss, D., 2017. Software-based Microarchitectural Attacks. PhD Thesis, Graz University of Technology.

Gruss, D., Maurice, C., Wagner, K. and Mangard, S. (2016). Flush+ Flush: A Fast and Stealthy Cache Attack. In Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 279-299. Springer International Publishing.

Gruss, D., Spreitzer, R. and Mangard, S. (2015). 'Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches'. USENIX Security Symposium, pp.897-912.

Guan, L., Lin, J., Luo, B., Jing, J. and Wang, J. (2015). Protecting Private Keys Against Memory Disclosure Attacks Using Hardware Transactional Memory. IEEE Symposium on Security and Privacy (SP), pp. 3-19.

Guilley, S., Hoogvorst, P. and Pacalet, R. (2004). Differential Power Analysis Model and Some Results. Smart Card Research and Advanced Applications Vi, pp.127-142.

Gullasch, D., Bangerter, E. and Krenn, S. (2011). 'Cache Games--Bringing Access-Based Cache Attacks on AES to Practice'. IEEE Symposium on Security and Privacy (SP), pp. 490-505.

Hayashi, Y.I., Homma, N., Mizuki, T., Aoki, T., Sone, H., Sauvage, L. and Danger, J.L. (2013). Analysis of Electromagnetic Information Leakage from Cryptographic Devices with Different Physical Structures. IEEE Transactions on Electromagnetic Compatibility, 55(3), pp.571-580.

Homma, N., Aoki, T. and Satoh, A. (2010). Electromagnetic Information Leakage for Side-Channel Analysis of Cryptographic Modules. IEEE International Symposium on Electromagnetic Compatibility (EMC), pp. 97-102.

Hu, W.M., 1992. Reducing Timing Channels with Fuzzy Time. Journal of Computer Security, 1(3-4), pp.233-254.

Hunger, C., Kazdagli, M., Rawat, A., Dimakis, A., Vishwanath, S. and Tiwari, M. (2015). Understanding Contention-Based Channels and Using Them for Defense. 21st IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 639-650.

Inam, R., Mahmud, N., Behnam, M., Nolte, T. and Sjödin, M. (2014). The Multi-Resource Server for Predictable Execution on Multi-Core Platforms. 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 1-12.

Inci, M.S., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T. and Sunar, B. (2016). 'Cache Attacks Enable Bulk Key Recovery on the Cloud'. In International Conference on Cryptographic Hardware and Embedded Systems, pp. 368-388. Springer, Berlin, Germany.

Intel® Corporation. (2016). 'Intel® 64 and IA-32 Architectures Optimization Reference Manual'. Intel Order Number, 64.

Irazoqui, G. (2017). Cross-core Microarchitectural Side Channel Attacks and Countermeasures. Doctoral dissertation, Worcester Polytechnic Institute.

Irazoqui, G., Eisenbarth, T. and Sunar, B. (2016). 'Cross Processor Cache Attacks'. Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 353-364.

Irazoqui, G., Inci, M.S., Eisenbarth, T. and Sunar, B. (2014). 'Wait a Minute! A Fast, Cross-VM Attack on AES'. International Workshop on Recent Advances in Intrusion Detection, pp. 299-319.

Kambourakis, G., Damopoulos, D., Papamartzivanos, D. and Pavlidakis, E. (2016). Introducing Touchstroke: Keystroke-Based Authentication System for Smartphones. Security and Communication Networks, 9(6), pp. 542-554.

Kelsey, J., Schneier, B., Wagner, D. and Hall, C. (2000). 'Side Channel Cryptanalysis of Product Ciphers'. Journal of Computer Security, 8(2-3), pp.141-158.

Kil, C., Jun, J., Bookholt, C., Xu, J. and Ning, P. (2006). Address Space Layout Permutation (ASLP): Towards Fine-Grained Randomization of Commodity Software. 22nd IEEE Annual Conference in Computer Security Applications, pp. 339-348.

Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J.H., Lee, D., Wilkerson, C., Lai, K. and Mutlu, O. (2014). Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In ACM SIGARCH Computer Architecture News, 42(3), pp. 361-372.

Kim, T., Peinado, M. and Mainar-Ruiz, G. (2012). 'STEALTHMEM: System-Level Protection Against Cache-Based Side Channel Attacks in the Cloud'. USENIX Security Symposium, pp. 189-204.

Kocher, P., Jaffe, J., Jun, B. and Rohatgi, P. (2011). 'Introduction to Differential Power Analysis'. Journal of Cryptographic Engineering, 1(1), pp. 5-27.

Kocher, P., Lee, R., McGraw, G., Raghunathan, A. and Moderator-Ravi, S. (2004). Security as a New Dimension in Embedded System Design. Proceedings of the 41st Annual Design Automation Conference, pp. 753-760.

Kocher, P., Jaffe, J. and Jun, B. (1999). 'Differential Power Analysis'. In Advances in Cryptology—CRYPTO'99 pp. 789-789. Springer Berlin/Heidelberg.

Kocher, P.C. (1996). 'Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems'. In Annual International Cryptology Conference, pp. 104-113. Springer, Berlin, Germany.

Kong, J., Aciiçmez, O., Seifert, J.P. and Zhou, H. (2009). Hardware-Software Integrated Approaches to Defend Against Software Cache-Based Side Channel Attacks. 15th IEEE International Symposium on High Performance Computer Architecture, pp. 393-404.

Kong, J., Aciicmez, O., Seifert, J.P. and Zhou, H. (2008). 'Deconstructing New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks'. Proceedings of the 2nd ACM workshop on Computer Security Architectures, pp. 25-34.

Köpf, B., Mauborgne, L. and Ochoa, M. (2012). Automatic Quantification of Cache Side-Channels. In International Conference on Computer Aided Verification, pp. 564-580. Springer Berlin Heidelberg.

Krämer, J., Nedospasov, D., Schlösser, A. and Seifert, J.P. (2013). Differential Photonic Emission Analysis. International Workshop on Constructive Side-Channel Analysis and Secure Design, pp. 1-16. Springer, Berlin, Heidelberg.

Lange, M., Liebergeld, S., Lackorzynski, A., Warg, A. and Peter, M. (2011). L4Android: A Generic Operating System Framework for Secure Smartphones. Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 39-50.

Langley, A. (2010). 'Checking That Functions Are Constant Time with Valgrind'. Available at: https://github.com/agl/ctgrind (Accessed: 02nd January 2018).

Lipp, M., Gruss, D., Spreitzer, R., Maurice, C. and Mangard, S. (2016). 'ARMageddon: Cache Attacks on Mobile Devices'. USENIX Security Symposium, pp. 549-564.

Liu, F., Ge, Q., Yarom, Y., Mckeen, F., Rozas, C., Heiser, G. and Lee, R.B. (2016). Catalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing'. IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 406-418.

Liu, F., Yarom, Y., Ge, Q., Heiser, G. and Lee, R.B. (2015). 'Last-Level Cache Side-Channel Attacks Are Practical'. IEEE Symposium on Security and Privacy (SP), pp.605-622.

Longo, J., De Mulder, E., Page, D. and Tunstall, M. (2015). SoC it to EM: Electromagnetic Side-Channel Attacks on a Complex System-on-Chip. International Workshop on Cryptographic Hardware and Embedded Systems, pp. 620-640. Springer, Berlin, Germany.

Martin, R., Demme, J. and Sethumadhavan, S. (2012). Timewarp: Rethinking Timekeeping and Performance Monitoring Mechanisms to Mitigate Side-Channel Attacks. ACM SIGARCH Computer Architecture News, 40(3), pp.118-129.

Michalevsky, Y., Schulman, A., Veerapandian, G.A., Boneh, D. and Nakibly, G. (2015). PowerSpy: Location Tracking Using Mobile Device Power Analysis. USENIX Security Symposium, pp. 785-800.

Mangard, S., Oswald, E. and Popp, T., 2008. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer Science & Business Media.

Montasari, R. Hosseinian-Far, A., Hill, R. Montaseri, F. and Dilshad, S (Submitted). Microarchitectural Cache-Based Side-Channel Attacks on Computing Hardware: Methods of Exploits and Countermeasures, Digital Investigation.

Moradi, A., Barenghi, A., Kasper, T. and Paar, C. (2011). On the Vulnerability of FPGA Bitstream Encryption Against Power Analysis Attacks: Extracting Keys from Xilinx Virtex-II Fpgas. In Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 111-124.

Murdocca, M. and Heuring, V. (2007). Computer Architecture and Organization. Hoboken: Wiley.

NIST. (2001). 197: Advanced encryption standard (AES). Federal information processing standards publication, 197(441), p.0311.

Oberg, J., Meiklejohn, S., Sherwood, T. and Kastner, R. (2014). Leveraging gate-Level Properties to Identify Hardware Timing Channels. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(9), pp.1288-1301.

Osvik, D.A., Shamir, A. and Tromer, E. (2006). 'Cache Attacks and Countermeasures: The Case of AES'. The RSA Conference Cryptographers' Track, pp. 1-20.

Owusu, E., Han, J., Das, S., Perrig, A. and Zhang, J. (2012). ACCessory: Password Inference Using Accelerometers on Smartphones. Proceedings of the 12th ACM Workshop on Mobile Computing Systems & Applications, p. 9.

Page, D. (2002). 'Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel'. University of Bristol Technical Report: CSTR-02-003, pp. 1-23.

Percival, C. (2005). 'Cache Missing for Fun and Profit'. Available at: http://www.daemonology.net/papers/htt.pdf (Accessed: 19th June 2017).

Pessl, P., Gruss, D., Maurice, C., Schwarz, M. and Mangard, S. (2016). 'DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks'. Proceedings of the 25th USENIX Security Symposium, pp. 565-581.

Raj, H., Nathuji, R., Singh, A. and England, P. (2009). Resource Management for Isolation Enhanced Cloud Services. Proceedings of the ACM Workshop on Cloud Computing Security, pp. 77-84.

Rane, A., Lin, C. and Tiwari, M. (2016). Secure, Precise, and Fast Floating-Point Operations on x86 Processors. In USENIX Security Symposium, pp. 71-86).

Ristenpart, T., Tromer, E., Shacham, H. and Savage, S. (2009). 'Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds'. Proceedings of the 16th ACM Conference on Computer and Communications Security, pp.199-212.

Rivest, R.L., Shamir, A. and Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21(2), pp.120-126.

Sarwar, G., Mehani, O., Boreli, R. and Kaafar, M.A. (2013). 'On the Effectiveness of Dynamic Taint Analysis for Protecting against Private Information Leaks on Android-based Devices'. In SECRYPT, pp. 461-468.

Schramm, K., Leander, G., Felke, P. and Paar, C. (2004). A Collision-Attack on AES. Workshop on Cryptographic Hardware and Embedded Systems, pp. 163-175.

Schwarz, M., Weiser, S., Gruss, D., Maurice, C. and Mangard, S. (2017). 'Malware Guard Extension: Using SGX to Conceal Cache Attacks'. arXiv preprint arXiv:1702.08719.

Schlösser, A., Nedospasov, D., Krämer, J., Orlic, S. and Seifert, J.P. (2012). Simple Photonic Emission Analysis of AES. Cryptographic Hardware and Embedded Systems–CHES, pp. 41-57.

Simon, L., Xu, W. and Anderson, R. (2016). Don't Interrupt Me While I Type: Inferring Text Entered Through Gesture Typing on Android Keyboards. Proceedings on Privacy Enhancing Technologies, 2016(3), pp.136-154.

Song, C., Lin, F., Ba, Z., Ren, K., Zhou, C. and Xu, W. (2016). 'My Smartphone Knows What You Print: Exploring Smartphone-Based Side-Channel Attacks against 3d Printers'. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 895-907.

Spreitzer, R., Moonsamy, V., Korak, T. and Mangard, S. (2017). Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices. IEEE Communications Surveys & Tutorials, PP (9). p. 1.

Spreitzer, R., Moonsamy, V., Korak, T. and Mangard, S. (2016). 'SoK: Systematic Classification of Side-Channel Attacks on Mobile Devices'. arXiv preprint arXiv:1611.03748.

Steffan, J.G., Colohan, C.B., Zhai, A. and Mowry, T.C. (2000). A Scalable Approach to Thread-Level Speculation. ACM SIGARCH Computer Architecture News, 28(2), pp.1-12.

Thoms, N. (2017). Evolution of the CPU: Too Much to Process?. Available at:
https://www.fasthosts.co.uk/blog/digital/evolution-cpu-too-much-process (Accessed: 2nd November 2017).

Tiwari, M., Wassel, H.M., Mazloom, B., Mysore, S., Chong, F.T. and Sherwood, T. (2009). Complete Information Flow Tracking from The Gates up. In ACM Sigplan Notices, 44(3), pp. 109-120.

Tromer, E., Osvik, D.A. and Shamir, A. (2010). 'Efficient Cache Attacks on AES, and Countermeasures'. Journal of Cryptology, 23(1), pp.37-71.

Tsai, J.Y. and Yew, P.C., 1996, October. The Superthreaded Architecture: Thread Pipelining with Run-Time Data Dependence Checking and Control Speculation. Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques, pp. 35-46.

Tsunoo Y., Saito T., Suzaki T., Shigeri M., Miyauchi H. (2003) Cryptanalysis of DES Implemented on Computers with Cache. In: Walter C.D., Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2003. CHES. Lecture Notes in Computer Science, vol 2779. Springer, Berlin, Heidelberg.

Valgrind (2017). 'Current Release: Valgrind-3.13.0' Available at: http://valgrind.org/ (Accessed: 02/02/2018).

Vattikonda, B.C., Das, S. and Shacham, H. (2011). Eliminating Fine Grained Timers in Xen. Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, pp. 41-46.

Wang, Y., Ferraiuolo, A. and Suh, G.E. (2014). Timing Channel Protection for A Shared Memory Controller. 20th IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 225-236.

Wang, Y. and Suh, G.E. (2012). Efficient Timing Channel Protection for On-Chip Networks. Networks on Chip (NoCS), 6th IEEE/ACM International Symposium on Networks on Chip (NoCS), pp. 142-151.

Wang, Z. and Lee, R.B. (2007). 'New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks'. Proceedings of the 34th Annual ACM International Symposium on Computer Architecture, 35(2), pp. 494-505.

Weiß, M., Weggenmann, B., August, M. and Sigl, G. (2014). On Cache Timing Attacks Considering Multi-Core Aspects in Virtualized Embedded Systems. International Conference on Trusted Systems, pp. 151-167. Springer, Cham.

Wu, J., Cheng, B., Yuen, C., Shang, Y. and Chen, J. (2015). Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks. IEEE Transactions on Mobile Computing, 14(4), pp.688-701.

Wu, Z., Xu, Z. and Wang, H. (2012). Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud. In USENIX Security Symposium, pp. 159-173.

Xiao, Y., Zhang, X., Zhang, Y. and Teodorescu, R. (2016). 'One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation'. In USENIX Security Symposium, pp. 19-35.

Xiao, Z. and Xiao, Y. (2013). 'Security and Privacy in Cloud Computing'. IEEE Communications Surveys and Tutorials, 15(2), pp. 843-859.

Xu, Z., Bai, K. and Zhu, S. (2012). Taplogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors. Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 113-124.

Yarom, Y., Genkin, D. and Heninger, N. (2017). CacheBleed: A Timing Attack on OpenSSL Constant-Time RSA. Journal of Cryptographic Engineering, 7(2), pp. 99-112.

Yarom, Y. and Benger, N. (2014). 'Recovering OpenSSL ECDSA Nonces Using the FLUSH+ RELOAD Cache Side-channel Attack'. IACR Cryptology ePrint Archive, p.1-11.

Yarom, Y. and Falkner, K. (2014). 'FLUSH+ RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack'. The Proceedings of the 23rd USENIX Security Symposium, pp. 719-732.

Zafirt. (2015). 'Is Your "Cloud" Safe from Cross-Tenant Side-Channel Attacks?'. Available at: http://oversitesentry.com/is-your-cloud-safe-from-cross-tenant-side-channel-attacks/ (Accessed: 30th December 2017).

Zhang, L., Ding, A. A., Fei, Y. and Jiang, Z. H. (2016, b). Statistical Analysis for Access-Driven Cache Attacks Against AES. IACR Cryptology ePrint Archive, p. 970.

Zhang, T., Zhang, Y. and Lee, R.B. (2016, a). Cloudradar: A Real-Time Side-Channel Attack Detection System in Clouds. In International Symposium on Research in Attacks, Intrusions, and Defenses, pp. 118-140. Springer International Publishing.

Zhang, R., Su, X., Wang, J., Wang, C., Liu, W. and Lau, R.W. (2015). On Mitigating the Risk of Cross-VM Covert Channels in a Public Cloud. IEEE Transactions on Parallel and Distributed Systems, 26(8), pp.2327-2339.

Zhang, T. and Lee, R.B. (2014). 'Secure Cache Modeling for Measuring Side-Channel Leakage'. Technical Report, Princeton University, pp. 1-27.

Zhang, Y., Juels, A., Reiter, M.K. and Ristenpart, T. (2014). 'Cross-Tenant Side-Channel Attacks in PaaS Clouds'. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 990-1003.

Zhang, Y., Juels, A., Reiter, M.K. and Ristenpart, T. (2012). Cross-VM Side Channels and Their Use to Extract Private Keys. Proceedings of the ACM Conference on Computer and Communications Security, pp. 305-316. ACM.

Zhang, Y., Juels, A., Oprea, A. and Reiter, M.K. (2011). Homealone: Co-Residency Detection in The Cloud Via Side-Channel Analysis. IEEE Symposium on Security and Privacy (SP), pp. 313-328.

Zhou, X., Demetriou, S., He, D., Naveed, M., Pan, X., Wang, X., Gunter, C.A. and Nahrstedt, K. (2013). Identity, Location, Disease and More: Inferring Your Secrets from Android Public Resources. Proceedings of The ACM SIGSAC Conference on Computer & Communications Security, pp. 1017-1028.

Zhou, Z., Reiter, M.K. and Zhang, Y. (2016). A Software Approach to Defeating Side Channels in Last-Level Caches. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 871-882.