International Journal of Organizational and Collective Intelligence
Volume 8 • Issue 2 • April-June 2018

Are Timing-Based Side-Channel Attacks Feasible in Shared, Modern Computing Hardware?

Reza Montasari, School of Computing and Digital Technology, Birmingham City University, Birmingham, UK
Amin Hosseinian-Far, Department of Business Systems and Operations, The University of Northampton, Northampton, UK
Richard Hill, Department of Computer Science, University of Huddersfield, Huddersfield, UK
Farshad Montaseri, Independent Researcher, Iran
Mak Sharma, School of Computing and Digital Technology, Birmingham City University, Birmingham, UK
Shahid Shabbir, School of Computing and Digital Technology, Birmingham City University, Birmingham, UK

ABSTRACT

This article describes how there exist various vulnerabilities in computing hardware that adversaries can exploit to mount attacks against the users of such hardware. Microarchitectural attacks, the result of these vulnerabilities, take advantage of microarchitectural performance of processor implementations, revealing hidden computing process. Leveraging microarchitectural resources, adversaries can potentially launch timing-based side-channel attacks in order to leak information via timing. In view of these security threats against computing hardware, the authors analyse current attacks that take advantage of microarchitectural elements in shared computing hardware. This analysis focuses only on timing-based side-channel attacks against the components of modern PC platforms - with references being made also to other platforms when relevant - as opposed to any other variations of side-channel attacks which have a broad application range. To this end, the authors analyse timing attacks performed against processor and cache components, again with references to other components when appropriate.

KEYWORDS

Attack Taxonomy, Hardware Vulnerabilities, Microarchitectural Attacks, Processor, Side-Channel Attacks

1. INTRODUCTION

Side-Channel Attacks, hereafter referred to as SCAs, pose serious security and privacy 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 (Hu, 1992, Page, 2002) and practical (Bernstein, 2005; Osvik et al., 2006) – is carried out through the exploitation

DOI: 10.4018/IJOCI.2018040103

of inadvertent information leakage from computing hardware (Gruss, 2017; Spreitzer et al., 2016) or via 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). For instance, through a SCA, an attacker will be able to exfiltrate secret keys used in cryptographic implementations, or gain information about it by probing the runtime. As an example, in 128-bit AES implementations that utilises four 1KB precomputed SBox tables such as OpenSSL (OpenSSL, 2016; Gullasch et al., 2011; Neve and Seifert, 2006), the probing of the ciphertext can result in the extraction of the complete secret key (Zhang et al., 2014; Agrawal and Mishra, 2012; Neve and Seifert, 2006). Various systems have inherent side-channel vulnerabilities that can be exploited by the attackers to launch devastating SCAs. For instance, an adversary can simplsy carry out a differential power analysis (Kocher 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 (Zhang and Lee, 2014).

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, 2006). 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 et al., 2007; Wang and Lee, 2006). 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).

SCAs have increasingly advanced from attacks on computing devices to attacks on general-purpose computing platforms (Spreitzer et al., 2016) and cloud computing infrastructures (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). Such advancement coupled with evolution in computational power of modern processors (as a result of the sharing of processor units) have provided researchers with new research opportunities in the new field of Microarchitectural analysis to continue to devise new and more sophisticated SCAs (such as that of exploiting the sharing features of processors) as well as Covert Channel Attacks, hereon abbreviated to CCAs. For instance, studies since as far back as 1973 have demonstrated that microprocessors and caches are susceptible to both SCAs and CCAs (Gruss et al., 2017; Pessl et al., 2016; Liu et al., 2015; Hund et al., 2013; Kim et al., 2012; Neve et al., 2006; Osvik et al., 2006; Bernstein, 2005; Percival, 2005; Tsunoo et al., 2003; Kelsey et al., 2000; Kocher, 1996; Hu, 1992; Wray, 1992; Lampson, 1973) by illustrating that cache architecture brings about inconsistency in the runtime because of dissimilar memory accesses. Such vulnerability poses threats to hardware because cache accesses are reliant, for instance, on the inputs of plaintext and the key (Crane et al., 2015; Tromer et al., 2010; Neve and Seifert, 2006).

Furthermore, as well as modern computing processors and caches, SCAs can also be carried out against public key cryptography schemes (Zhang et al., 2014; Bellare et al., 2013), AES (Irazoqui et al., 2015, a; Irazoqui et al., 2014), ECDSA (Benger et al., 2014), TLS messages (Irazoqui et al., 2015, b), items in a shopping cart (Zhang et al., 2014) or even the key strokes typed in a keyboard (Gruss et al., 2015). In addition, SCAs can also be mounted against PaaS clouds (Liu et al., 2015; Zafirt, 2015; Zhang et al., 2014), across processors (Irazoqui et al., 2016; Crane et al., 2015; Hund et al., 2013) and in smartphones (Lipp et al., 2016; Song et al., 2016; Chen et al., 2014). Having examined the state-of the-art reveals that on one hand processor manufacturers incorporate new enhanced security features, but on the other hand security researchers compromise these features just to demonstrate the way in which secret information can be emitted (Grus et al., 2016; Hunger et al., 2015; Wang et

al., 2014; Saltaformaggio et al., 2013; Aciicmez and Seifert, 2007; Wang and Lee, 2006). In light of such increasing attention to both SCAs and CCAs, researchers have proceeded to suggest various hardware and software countermeasures to deal with these attacks (Martin et al., 2012; Kong et al., 2009; Wang and Lee, 2007; Page, 2005).

1.1. Key Contributions and the Methodology Followed

In light of the above discussion, this study thoroughly reviews and analyses both sides of the competition, i.e. attacks and countermeasures, and makes the following contributions that include:

1. A systematic survey of existing literature within the context;
2. Analysing the field of Microarchitectural Analysis;
3. Describing the main characteristics of Microarchitectural elements that enable the exposure of side channels;
4. Identifying and examining existing Microarchitectural SCAs;
5. Summarising the main features of hardware that bring about side channels, in order to enable the research communities and software developers to gain better insight into the way that applications can be pro-actively designed to prevent SCA vulnerabilities.

Therefore, as already stated, this study examines SCAs with the main focus being on Timing-Based Side-Channel Attacks, hereon referred to as TBSCAs, and identify common features between them. The study then proceeds to analyse existing countermeasures and propose new ones. From our analysis, we deduce insight in relation to the current state of knowledge observed in the literature, establish means of attacks, predict possible future modus operandi of attacks, and propose effective future directions for the development of appropriate defence mechanisms. Although we present theoretical work of clear relevance, we primarily focus our attention on practical and established attacks and defence mechanisms. We believe that the understanding obtained from this study enables researchers to establish directions for future research and also to address such attacks on a larger scale.

1.2. Scope of the Survey

Based on our survey of the literature, various Microarchitectural SCAs have been identified, which organise into a taxonomy of 13 general sub-categories, including: Acoustic Cryptanalysis Attack, Branch-Prediction Attack, Cold Boot Attack, Cache Attack, Differential Fault Analysis Attack, DMA Attack, Electromagnetic Attack, Fault-Attacks, Lucky-Thirteen Attack, Pass the Hash Attack, Power-Analysis Attack, Tempest Attack, and Timing Attack. The emphasis of this study is only on Timing-Based Side-Channel Attacks, as opposed to any other 12 variants, with brief reference to other variations only when appropriate. Furthermore, the emphasis of our study on TBSCAs is only on those Timing Attacks that are capable of compromising 'the components of a PC platform' (such as a hard drive or a modern processor that can consist of processor cores and any functional units inside a multi-core multi-threaded processor) and 'entities in a network'. Again, references are made also to TBSCAs in other platforms such as mobile devices or cloud infrastructure only when appropriate. In addition, this study does not explore covert channels even though they are referred to when necessary or appropriate. Any other topics related to side channels are beyond the scope of this paper. Existing countermeasures against TBSCAs within PC platforms and entities in networks are then analysed, and new strategies are proposed.

1.3. Outline of the Paper

The remainder of the paper is structured as follows: Section 2 provides a background for Microarchitectural Analysis. In Section 3, Timing-Based Side-Channel Attack are presented and examined in detail, while in Section 4, Side-Channel Attacks against RSA implementations are

analysed. Finally, Section 5 concludes the study by providing a detailed discussion about trends in attacks, challenges to offset them and the future research direction in this research field. Two main contributions of this paper are the scope of the discussion, since few works of similar scope currently exist, and the provision of an agenda for the direction of future research.

2. BACKGROUND TO MICROARCHITECTURAL SIDE-CHANNEL ATTACKS

In 1996, Kocher (1996) demonstrated that attackers could potentially deduce RSA keys, named after RSA's creators "Rivest-Shamir-Adleman" (Rivest et al., 1978), and also other decipher cryptosystems by carefully calculating the amount of time needed to conduct private key operations. Kocher was able to define SCAs as a method that enables adversaries to extract secret information utilised in a computing process from unintended impacts that the computing process has on its environment (Gruss, 2017; Crane et al., 2015; Genkin et al., 2014). His seminal work can be considered as a foundation for a whole new domain of research into Side-Channels (Gruss, 2017). Kocher carried out an attack which researchers now define as Timing-Based Side-Channel Attacks, attacks that take advantage of differences in runtime of a computing process. Kocher illustrated that a Side-Channel Attack against a weak system would not be computationally difficult and often necessitate only ciphertext. His study revealed that attackers could make relatively precise timing calculations that would result in breaking systems such as "cryptographic tokens, network-based cryptosystems", and other applications (Kocher, 1996). Other seminal works in the field of SCAs include those by Mangard et al. (2008); Quisquater and Samyde (2001), Chari et al. (1999) as well as another study by Kocher himself in Kocher et al. (1999).

In the years that have followed, researchers have been able to illustrate SCAs based on changes in different computing settings (Gruss et al., 2017; Spreitzer and Plos, 2013), including: AES (Zhang et al., 2016, b; Spreitzer and Plos, 2013; Gullasch et al., 2011; Osvik et al., 2006), differential power analysis (Kocher et al., 2017; 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, 2017; 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, 2017; 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).

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 (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 analyse 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) 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. TIMING-BASED SIDE-CHANNEL ATTACKS

In this section, following giving a brief overview of timing channels and TBSCAs, we analyse various TBSCAs and demonstrate that hardware vulnerabilities resulting from various factors such as optimisations on a microarchitectural layer can be exploited by the attackers to launch devastating TBSCAs and as a result compromise the system security.

3.1. Overview

Over the past few years, information leakage via covert channels and side channels have been a grave concern for security researchers (Wu et al., 2015; Luo et al., 2011; Chen et al., 2010). This issue has recently been exacerbated by certain features of modern processor architecture (Liu et al., 2015; Yarom and Falkner, 2014; Hund et al., 2013; Wang and Lee, 2006) such as simultaneous multithreading (Zhang and Reiter, 2013; Domnitser et al., 2012; Tromer et al., 2010), speculative memory accesses (Doychev et al., 2015; Guan et al., 2015; Yarom and Falkner, 2014; Harnik et al., 2010) and shared caches (Gruss et al., 2016; Crane et al., 2015; Zhang and Lee, 2014; Zhang et al., 2012) that can accelerate both covert channels and side channels.

Timing channels that represent both covert and side channels are the focus of this study as stated previously. A timing channel is a communication channel that can transfer information to a recipient and decoder by controlling the timing performance of an object (Biswas et al., 2017; Rowland, 1997) such as inter-packet delays of a packet stream (Wu et al., 2015; Liu et al., 2010; Sultana et al., 2013) or the reordering packets in a packet stream (Biswas et al., 2017; Sultana et al., 2013; Lu et al., 2010; Tsai et al., 2010). Such channels can be considered as a type of computer security attack that enables an adversary to develop an ability to transfer information objects between processes that are not intended to be allowed to communicate by the computer security policy.

The term "timing channel" was coined in 1973 by Lampson (Lampson, 1973) as channels that "are not intended for information transfer at all, such as the service program's effect on system load" to differentiate it from benign channels that are exposed to access controls by computer security. Girling (1987) was first to investigate the usage of delays between packets transferred over computer networks for covert communication. This seminal study became the foundation for many other studies (Wendzel et al., 2015; Mazurczyk et al., 2014; Geddes et al., 2013; Luo et al., 2008; Zander et al., 2007; Partan et al., 2007; Elson et al., 2002; Ahsan, 2002) to identify and analyse timing channels. There exist three different widely-recognised types of Timing Channels, including: Covert Communications (Chen and Venkataramani, 2014; Gianvecchio and Wang, 2011; Liu et al., 2009), Timing-Based Side-Channels (Liu et al., 2015; Meyer et al., 2014; Hund et al., 2013; Stefan et al., 2013), and Network Flow Watermarking (Biswas et al., 2017, Bates et al., 2012; Zander et al., 2007).

In this study, we only focus on providing a detailed analysis of TBSCAs that we classify into two general categories. These include: (1) Timing-Based Side-Channel in a network, where an active entity within a network system communicates with other objects in the network, and in-system Timing-Based Side-Channel in a PC platform, where entities communicate with each other within the PC platform.

A TBSCA represents a type of SCA that exploits differences in the runtime of an algorithm (Brumley and Tuveri, 2011; Aciiçmez et al., 2005; Kocher, 1996). This denotes that by taking advantage of such differences, an adversary can potentially compromise a cryptosystem through the observation of the time required to run cryptographic algorithms (Pornin, 2017; Schneier,

2005; Kocher, 1996). Unlike cryptanalysis which is focused on the mathematics (Song et al., 2013; Otmani et al., 2010), for instance, in differential and linear cryptanalysis (Bogdanov and Rijmen, 2014; Mouha et al., 2011), a TBSCA is based on implementation and applies additional information collected from attacking such implementations (Snow et al., 2013; Hund et al., 2013; Sarwar et al., 2013; Kocher et al., 2011). Therefore, a TBSCA can also exploit the data-dependent performance features of the execution of an algorithm (Pornin, 2017; Chen et al., 2013; Kocher, 1996) as opposed to the mathematical components of the algorithm itself.

Furthermore, contrary to Cache-Based Side-Channel Attacks (CBSCAs), which exploit operational aspects of a system (Such as general-purpose systems) (Ge et al., 2016; Crane et al., 2015; Wang and Lee, 2007), TBSCAs are carried out via timing variation, even when the execution performance of the system is entirely known, and even when there is formal proof of the lack of Cache Channels (Ge et al., 2016; Murray et al., 2013; Schaefer et al., 1977). The time that each logical execution takes in a computer system to run varies according to the input (Patterson and Hennessy, 2017; Kirsch and Sokolova, 2012). With the exact analysis of the time for each execution, an adversary will be able to work backwards to the input (Vétillard and Ferrari, 2010; Kocher, 1996). Calculating the time that a computer system takes to address specific quarries can cause an emission of information from the system (Seibert et al., 2014; Weiß et al., 2012; Tromer et al., 2010; Hopper et al., 2010). The degree to which this information can assist an adversary will be based on certain factors such as crypto system implementation, the algorithms utilised, the CPU running the system, various execution details, timing attack remedies, the precision of the timing measurements, etc.

In addition, TBSCAs can exploit the effects of variations in encryption time caused by conditional branches that occur during encryption processing (Lee et al., 2016; Lawson, 2009; Aciiçmez et al., 2007; Zhou and Feng, 2005; Tsunoo et al., 2003; Kocher, 1996). CPU cache (between the CPU and main memory) misses are also capable of creating such variations (Braun et al., 2015; Bonneau and Mironov, 2006). If the CPU accesses data that does not reside in the cache, a delay will be triggered because the target data must be loaded from main memory into the cache (Irazoqui et al., 2016; Gruss et al., 2016; Tsunoo et al., 2003). Therefore, the measurement of such delay can allow adversaries to establish the occurrence and frequency of cache misses. Last, but not least, Timing Attacks can also be facilitated through shared memory controllers (Ge et al., 2016; Pessl et al., 2016; Wang et al., 2014), where the computing hardware emits portions of its internal state such as confidential information via variations in performance and timing (Shafiee et al., 2015; Wang et al., 2014).

3.2. Prime+Probe Attack

In this attack, the adversary fills a cache set with his own lines, then waits for a specific period and proceeds to establish whether the lines are still cached (Gruss et al., 2017; Crane et al., 2015; Irazoqui et al., 2015, a; Tromer et al., 2010). The adversary will then be able to determine whether the victim accessed the designated cache set in the meantime (Gruss et al., 2017; Inci et al., 2016; Irazoqui et al., 2015, c; Apecechea et al., 2014). This requires that the attacker examines certain cache sets to establish the presence of a cache miss. To do so, he will need to time the accesses to the cache set after the victim executes (Inci et al., 2016; Crane et al., 2015). He assigns a group of cacheline-sized, cacheline-aligned memory blocks in order for such memory blocks to fill a collection of targeted cache sets (Zhang et al., 2016, a; Yarom and Falkner, 2014; Percival, 2005). Having done this, the attacker will then constantly carry out two attack phases including ‘prime phase’ and ‘probe phase’(Gruss et al., 2017; Crane et al., 2015; Tromer et al., 2010). Within the ‘prime phase’, the attacker examines each memory chunk to eject all the victim’s data in such cache arrays. They will then wait for a delay time prior to carrying out the ‘probe phase’, where each memory chunk is examined in the group again and the time of memory accesses is determined (Gruss et al., 2017; Crane et al., 2015; Liu et al., 2015; Varadarajan et al., 2014). Longer access times represent one or more cache misses. This means that this cache array has been accessed by the victim between the prime and probe phases. The attacker will repeat these two phases many times in order to acquire traces that might overlap

with the victim's performance of cryptographic operations (Zhang et al., 2016, a; Liu et al., 2015; Wei et al., 2012; Zhang et al., 2012, a; Ristenpart et al., 2009). As a result, the adversary will be able to establish which cache lines were replaced by the victim and infer more details concerning which addresses the victim accessed. To determine the cache lines that were replaced, the attacker will need to measure the speed of each cache access.

It should be noted that the Prime+Probe Attacks can target both the L1 through the use of cache sets and the Addressing Scheme (Oren et al., 2015; Yarom and Benger, 2014; Apecechea et al., 2014; Yarom and Falkner, 2014; Zhang et al., 2012; Tromer et al., 2010; Aciiçmez, 2007; Osvik et al., 2006; Percival, 2005), as well as the Last Level Cache (LLC) (Irazoqui et al., 2015, a; Liu et al., 2015; Oren et al., 2015; Benger et al., 2014; Ristenpart et al., 2009). To perform the attack on L1, both the attacker and the victim will need to have access to the same physical CPU core concurrently. In contrast, to carry out the Prime+Probe Attack against L3, which is a more advanced version, both the adversary and the victim must be using the same CPU but not "necessarily" the CPU core (Inci et al., 2016; Irazoqui et al., 2014; Yarom and Falkner, 2014; Benger et al., 2014). Many of the advanced Prime+Probe Attacks do not require to be dependent upon De-Duplication (Liu et al., 2015, Irazoqui et al., 2015, a) or core sharing, resulting in them becoming broadly relevant (Inci et al., 2016).

Adversaries can also attack AES in OpenSSL 0.9.8 with Prime+Probe on the L1 data-cache (D-cache) (Liu et al., 2016; Kong et al., 2013; Wei et al., 2012; Brumley, 2011; Tromer et al., 2010; Osvik et al., 2006) and L1 instruction-cache (I-cache) contention (Irazoqui et al., 2015, a; Liu et al., 2015; Aciiçmez, 2007) to establish an end-user's control row (Ge et al., 2016; Jia et al., 2014). This will enable the attacker to differentiate squares and multiples in OpenSSL 0.9.8d RSA (Allan et al., 2016; Liu et al., 2016).

Moreover, an adversary can launch a Prime+Probe Attack against elliptic-curve cryptography (ECC) in OpenSSL 0.9.8k using channel measurements (Garcia and Brumley, 2016; Allan et al., 2016; Ge et al., 2016; Yarom and Falkner, 2014; Yarom and Benger, 2014; Brumley and Hakala, 2009). CPU caches are a potent source of information leakage. Therefore, Prime+Probe Attacks can be mounted against them through manual identification of susceptibilities such as data accesses or instruction execution (Benger et al., 2014; Chen et al., 2013; Gullasch et al., 2011; Brumley and Hakala, 2009; Bonneau and Mironov, 2006; Osvik et al., 2006). In addition, Prime+Probe Attack can also be carried out within cloud computing environments. The cross-VM leakage exists in public clouds and is often a practical attack vector for stealing sensitive data (Inci et al., 2016; Green, 2013; Zhang et al., 2012, a). In the cross-VM context, the adversary and victim have two distinct VMs running as co-tenants on the same server. Thus, the adversary will be able to acquire co-tenancy of a malign VM on the same server as a target (Varadarajan et al., 2014; Xiao and Xiao, 2013; Zhang et al., 2012, a; Ristenpart et al., 2009). For instance, using a Prime+Probe technique, a cross-VM attack can be carried out to obtain ElGamal secret keys from the victim (Irazoqui et al., 2016; Osvik et al., 2006).

3.3. Time Slicing Attack

By performing a Time Slicing Attack (TSA), an adversary will be able to extract kernel and user-level ASLR offset on the branch target buffer (BTB) (Ge et al., 2016; Evtyushkin et al., 2016; Hund et al., 2013; Aciiçmez et al., 2007, Hu, 1992). An Address Space Layout Randomisation (ASLR), first designed and coined by Linux PaX project (PaX, 2001), is a security technique used to prevent exploitation of memory vulnerability in operating systems that guard against buffer-overflow attacks. To provide such security, ASLR functions by randomising the location in which system executables are loaded into memory (Symantec, 2017; Davi et al., 2015; Shacham et al., 2004) and the offset of key program segments in virtual memory (Crane et al., 2015; Backes and Nürnberger, 2014; Bhatkar et al., 2003). In theory, this should render it difficult for the attacker to deduce addresses of certain code objects (Gruss et al., 2016; Evtyushkin et al., 2016). However, as stated above, attackers can undermine the ASLR by mounting a TSA. For instance, a local attacker with restricted privileges can exploit the limitations of kernel space ASLR to launch a TSA against the memory management

system in order to infer information about the privileged address space layout (Hund et al., 2013; Bhatkar et al., 2003). Furthermore, through a TSA combined with another variant of TBSCA (See sub-section 3.7), namely Flush+Reload (Yarom and Falkner, 2014), the adversary will be able to exploit the Translation Lookaside Buffer (TLB) (Wang et al., 2017; Ge et al., 2016; Grunwald and Ghiasi, 2002) contention to overcome ASLR (Jang et al., 2016; Ge et al., 2016; Liu et al., 2015). This can result in the detection of both kernel-level and user-level virtual address space layout on ASLR-enabled Linux platforms since such platforms employ only portion of the virtual address bit as hash tags (Evtyushkin et al., 2016; Wang et al., 2015; Larsen et al., 2014). In addition, invalid mappings that are not loaded into the TLB can also be exploited by the attacker, resulting in a void address which will activate another table walk (Ge et al., 2016; Evtyushkin et al., 2016; Hund et al., 2013).

3.4. Remote Timing-Based Side-Channel Attacks

A Remote Timing-Based Side Channel Attack (RTBSCA), which is carried out within a network setting, enables an attacker to exploit weaknesses of a cryptographic design remotely (Hunger et al., 2015; Liu et al., 2015; Aciiçmez et al. 2007; Brumley and Boneh, 2005). A RTBSCA often remains undetected for a long time before its presence can be detected and the emitted private information can be decoded (Biswas et al., 2017; Lawson, 2009). Usually, the observer will not be able to enhance such properties since the private information is emitted by a defective operation (Biswas et al., 2017; Hunger et al., 2015). Achieving higher Timing-Based Side-Channel capacity is difficult since many constant observations are needed to decrease the error probability to enhance efficiency (Liu et al., 2015; Wu et al, 2015; Kocher, 1996). Acquiring high bandwidth necessitates optimising synchronization (Wu et al., 2015; Liu et al., 2015; Karlof and Wagner, 2003; Katabi, 2003). This denotes that matching clocks in the sender and receiver in order for them to correspond on the time duration for each bit (Hunger et al., 2015; Rhee et al., 2009). Synchronisation allows the sender and receiver to employ basic binary signalling without requiring self-clocking codes and yet obtain low bit error rates (Maurice et al., 2017; Hunger et al., 2015; Welzl, 2012; Welzl, 2005).

Often, there are vulnerabilities associated with SSH that an attacker can exploit to exfiltrate passwords remotely during an SSH session (Biswas et al., 2017; Balduzzi et al., 2012; Song et al., 2001) and private keys from an OpenSSL-based web server (Liu et al., 2015; Brumley and Boneh, 2005). For instance, these include: (1) the exposure of the original data size by the 8-byte limit of transferred packets (Seibert et al., 2014; Aciiçmez et al., 2010; Song et al., 2001) and (2) the emission of the inter-keystroke timing information since each portion of keystroke information is transmitted to the remote machine while in interaction state (Biswas et al., 2017; Seibert et al., 2014; Aciiçmez et al., 2010; Foo Kune and Kim, 2010; Raymond, 2001). Furthermore, under this attack, the adversary can exfiltrate a considerable amount of information that the victim types by employing sophisticated statistical methods (Chen et al., 2010). The attacker will be able to deduce secret information and values from the inter-keystroke timings by utilising, for instance, Hidden Markov Analysis Model (a technique employed to predict the value of a variable, the future value of which remains independent of its past history) and their key forecast algorithm (Biswas et al., 2017; Seibert et al., 2014; Aciiçmez et al., 2010; Foo Kune and Kim, 2010; Raymond, 2001).

Furthermore, adversaries can also launch a remote TBSCA against OpenSSL by exploiting the inherent susceptibility that exists in OpenSSL (in the Montgomery ladder in the Elliptic Curve Cryptosystem) in order to extract the secret key of a TLS server (Benger et al., 2014; Yarom and Benger, 2014; Brumley and Tuveri, 2011). Network Tomography, an essential part of network measurement, is responsible for performing traffic analysis by observing the network to ensure that all the links in a network are healthy (Mardani and Giannakis, 2016; Chawla et al., 2012; Danezis and Clayton, 2007). This is performed through the use of end-to-end queries that are transmitted by agents residing at vantage points in the network (Gong et al., 2012; Shmatikov and Wang, 2006). Using this same approach, that necessitates direct monitoring of network connections at local vantage points, an attacker will be to perform network analysis and as a result to launch a devastating TBSCA against

the hardware agents in the network (Ge et al., 2016; Meyer et al., 2014; Brumley and Boneh, 2005). In unusual cases in which the attacker is 'extremely savvy', he can also employ the same convention to launch a remote TBSCA by exploiting a scheduler (which is a side channel) between himself and the victim (Varadarajan et al., 2014; Wang et al., 2014; Stefan et al., 2013; Gullasch et al., 2011). Our point is substantiated by a study (Gong and Kiyavash, 2013) in which researchers were able to demonstrate that the attacker could establish the entire pattern of the victim by using "Shannon equivocation as a privacy metric" (Gong and Kiyavash, 2013). This attack is made possible if the scheduler is based on the policy of a first-come, first-served basis (Wang et al., 2014; Kadloor et al., 2010; Gupta, 2007). Again, using the same approach, the savvy attacker can launch a remote TBSCA by taking advantage of the Timing Side Channels inside a "home digital subscriber line (DSL) router" (Gong and Kiyavash, 2013; Kurose and Ross, 2010). Performing this attack, the adversary will be able to acquire the victim's secret data such as passwords and voice over IP (VoIP) conversations (Biswas et al., 2017; Lee et al., 2015; Mohaban et al., 2007).

3.5. Access-Driven Cache-Timing Attack

An Access-Driven Cache-Timing Attack (ADCTA) is another variation of SCAs that takes advantage of the emission of the memory locations that the victim process accesses (Kim et al., 2012; Gullasch et al., 2011). This attack involves probing the cache's timings as a source of information emission (Zhang et al., 2012; Neve and Seifert, 2006). In the ADCTA, the cache performance is inspected with a fine granularity (Crane et al., 2015) as opposed to assessing the overall runtime. Using an ADCTA, an adversary will be able to determine whether a cache line has been ejected or not (Crane et al., 2015; Irazoqui et al., 2015; Aciiçmez and Koç, 2006). Furthermore, a savvy attacker can also potentially perform an ADCTA on the Advanced Encryption Standard (AES) block cipher (Irazoqui et al., 2015; Neve and Seifert, 2006; Bonneau and Mironov, 2006) by utilising compressed tables (Zhang et al., 2012; Gullasch et al., 2011) to extract the complete secrete key in real time (for instance, for AES-128) (Crane et al., 2015). To do so, he will need only a restricted number of observed encryptions (without requiring any information concerning plaintext) to be able to exfiltrate the entire key due to, for instance, round analysis from the ciphertext.

ADCTAs are also made possible in cloud computing environments due to the vulnerability in certain hardware components such as the scheduler of the Xen hypervisor (Yarom and Falkner, 2014; Barham et al., 2003). For instance, the attacker can utilise a malicious virtual machine that will enable him to extract detailed, precise information from a victim VM that is running in parallel on the same computer (Yarom and Falkner, 2014; Zhang et al., 2012; Kim et al., 2012). As an example, to perform an ADCTA on a symmetric multiprocessing system (Braun et al., 2015; Winder, 2012), the adversary will require to deal with various challenges such as core migration (Winder, 2012; Bertozzi et al., 2006), multiple sources of channel noise (Braun et al., 2015; Winder, 2012) and also the problems with pre-empting the victim with adequate frequency to acquire detailed information from it (Zhang et al., 2012; Winder, 2012; Bertozzi et al., 2006). However, the attacker can bypass such challenges by utilising, for instance, libgcrypt cryptographic library to exfiltrate an ElGamal (ElGamal, 1985) decryption key of a GnuPG decryption (Yarom and Falkner, 2014), which is running in another guest, from the victim. The ADCTA can also be performed against certain OpenSSL (e.g. 0.9.8n) implementation of AES on Linux systems (Crane et al., 2015; Liu et al., 2015; Brumley and Boneh, 2005). In addition, it can also be used to mount a denial of service (DoS) attack on the task scheduler of Linux systems that allows the attackers to monitor all memory accesses of a victim process (Zhang et al., 2012; Gullasch et al., 2011).

Likewise, adversaries might be able to perform the ADCTA on a time-shared core to take advantage of a shared LLC. In such a case, they will need to exploit the cupid instructions or leverage fence instructions to be able to synchronise the instruction stream (Yarom and Falkner, 2014; Gullasch et al., 2011). Similarly, a successful ADCTA can break the isolation feature of system virtualisation (Yarom and Falkner, 2014; Kim et al., 2012; Zhang et al., 2012). In this situation, by employing and

adapting Bernstein's attack's link (Bernstein, 2005) in CTA (Weiß et al., 2012), the attacker will be able to derive secret information from an isolated execution area. An ADCTA can also be performed to exploit the hardware-assisted multi-threading (Osvik et al., 2006; Percival, 2005) or single-threading (Neve and Seifert, 2006) ability of certain microprocessors so as to execute a spy process quasi in parallel to a cryptography process.

3.6. The Flush+Reload Technique

The Flush+Reload Attack (Yarom and Falkner, 2014), a variation of Prime +Probe Attacks (Irazoqui et al., 2015; Tromer et al., 2010), is based on the sharing of pages between the malicious and victim processes (Gruss et al., 2015; Liu et al., 2015; Yarom and Falkner, 2014; Osvik et al., 2006). By performing this attack, an adversary will be able to eject a particular memory line from the entire cache hierarchy through shared pages (Irazoqui et al., 2015; Yarom and Falkner, 2014). The Flush+Reload Attack has been adapted from Gullasch et al.'s (2011) technique for usage in both virtual and non-virtual settings (Gruss et al., 2015; Irazoqui et al., 2015; Zhang et al., 2014; Yarom and Falkner, 2014). Therefore, it can be performed in both environments, i.e. virtualisation and non-virtualisation. For instance, in cloud and virtual environments, by conducting the Flush+Reload Attack, the adversary will be able to exfiltrate GnuPG (a popular cryptography package that is utilised as the cryptography module of many open-source projects) private keys across several processor cores and virtual machines (Yarom and Falkner, 2014). Due to its generic nature, Flush+Reload Attack can be performed for other malicious purposes too. For instance, an attacker can launch a Flush+Reload Attack to gather statistical data on network traffic by observing network handling code or monitoring keyboard drivers to derive keystroke timing information.

Flush+Reload Attack consists of three stages (Zhang et al., 2014; Yarom and Falkner, 2014), consisting of Flush, Flush+Reload Interval and Reload. Stage one, Flush, involves flushing the observed memory line from the cache hierarchy including the shared last-level cache utilising clflush instruction (Gruss et al., 2015; Zhang et al., 2014; Yarom and Falkner, 2014). In stage two, Flush+Reload Interval, the attacker waits for a "prespecified interval" to enable the victim to access the memory line, while the last-level cache is employed by the victim running on the CPU core. Stage three, Reload, the attacker involves the attacker reloading the memory line, calculating the time to load it. A faster reload will indicate the existence of certain chunks in the last-level cache and the fact that they were run by the victim during the Flush+Reload interval. In contrast, a slower reload signifies the contrary (Zhang et al., 2014; Yarom and Falkner, 2014).

3.7. Asynchronous Attack

An Asynchronous Attack (AA) (Tromer et al., 2010) is a very complex attack that is difficult to detect. As such, it is likely to be carried out only by very advanced attackers. Despite the fact that an AA is similar to a Cache-Based Attack (CBA) in that both are performed against RSA for processors with simultaneous multithreading (Percival, 2005; Osvik et al., 2006), their cryptanalysis is very different because "algorithms and time scales involved in RSA vs. AES" executions are very different (Tromer et al., 2010). Under an AA, the adversary runs its own program on the same process as the encryption application (Tromer et al., 2010, Percival, 2005). This is achieved without any "explicit interaction" including "inter-process communication" (Crane et al., 2015; Gullasch et al., 2011; Osvik et al., 2006). The only insight that the attacker is required to have concerns the plaintexts or ciphertexts (as opposed to their exact values) (Aciiçmez et al., 2007; Osvik et al., 2006; Percival, 2005). If the cipher runs on a simultaneous multi-threading (SMT) machine (Crane et al., 2015, Tromer et al., 2010), and the attacker is able to execute a dummy process concurrently with the cipher process (Gullasch et al., 2011), he will then be able to clear the BTB through the executions of the dummy process and causes a BTB miss during the operation of the target branch (Ge et al., 2016; Evtyushkin et al., 2016; Hund et al., 2013; Aciiçmez et al., 2007' Hu, 1992). In this case, "the BPU automatically predicts the branch not to be taken if it misses the target address in the BTB". Thus, there will be a

misprediction whenever the result of the target branch is "taken". The attacker will be able to simulate the exponentiations partition of the sample according to the outcome of the branch (Aciiçmez et al., 2007; Grunwald and Ghiasi, 2002).

3.8. Evict+Time Attack

Evict+Time Attacks can be defined as a generic Cache-Timing Attack technique, by means of which the adversary activates multiple victim computations and calculates the victim's runtime (Gruss, 2017; Irazoqui et al., 2014; Osvik et al., 2006). In order to do so, he evicts the cache set the victim's runtime in order to calculate the effect of a particular cache set, and then and force the encryption programme to fetch key values from the main program (Crane et al., 2015; Tromer et al., 2010). In cases in which there is a timing difference when ejecting the cache set, the adversary will be able to deduce that the cache set was utilised by the victim computing process (Yarom and Falkner, 2014; Weiß et al., 2012; Aciiçmez et al., 2007, Bernstein, 2005). To carry out an Evict+Time attack, the adversary will need to be able to calculate the precise starting and end time of a victim computing process. Evict+Time Attacks have been extensively covered in the literature. For instance, Osvik et al. (2006) examined Evict+Time Attacks in an attack on OpenSSL AES. Lipp et al. (2016) and Spreitzer and Plos (2013) illustrated that Evict+Time Attacks on OpenSSL AES are also relevant to mobile ARM-based devices. Similarly, Hund et al. (2013) showed that Evict+Time Attacks can be applied to compromise Kernel Address Space-Layout Randomisation (KASLR). Therefore, we do not aim to delve into this attack any further.

3.9. Timing Attacks Against Floating-Point Instructions

An attacker can also launch a TBSCA against the floating-point instructions of modern x86 processors (Andrysco et al., 2015; Coppens et al., 2009). The "running time of floating point addition and multiplication instructions can fluctuate by two orders of magnitude" (Andrysco et al., 2015; Hachez and Quisquater, 2000; Walter, 1999) depending on their operands (Coppens et al., 2009). Multiplying or dividing with subnormal values results in slowdown on, for instance, all tested Intel and AMD processors (Ge et al., 2016; Andrysco et al., 2015), irrespective of employing single instruction multiple data (SIMD) or x87 instructions (Intel®, 2016). Furthermore, by exploiting the aforementioned effects, an attacker will be able to employ the subnormal floating-point numbers to launch a timing attack, for instance, on a scalable vector graphics (SVG) filter, that reads arbitrary pixels from any victim web page though the Firefox browser as demonstrated by (Andrysco et al., 2015). Similarly, in relation to this type of exploit, two other researchers (Stone, 2013; Kotcher et al., 2013) have implemented a new method for cross-origin pixel stealing in the browser, which is a timing side-channel within CSS Scalable Vector Graphics (SVG) transforms. Such transforms can be employed through CSS to any element of a webpage, for instance iframes. Once the content of cross-origin is contained in an iframe, the containing page is then able to employ SVG transformation filters to that iframe (Andrysco et al., 2015). As a result, an attacker will be able to extract any pixel value from a website that he does not own by selecting certain SVG filters and determining the page render times.

3.10. Bernstein's Attack

Bernstein's Attack (Bernstein, 2005) is another variant of TBSCA that is carried out remotely on an AES T-table implementation in which the attacker can recover the AES key from "known-plaintext timings of a network server" on a different computer. This attack is the resultant of the fault in AES design and not to a specific library used by the server (Bernstein, 2005). Through this attack, Bernstein (2005) demonstrated that attacks as such were not restricted just to the Pentium III but instead could be performed against an "AMD Athlon, an Intel Pentium III, an Intel Pentium M, an IBM PowerPC RS64 IV, and a Sun UltraSPARC III". By performing the Bernstein's Attack, the adversary can potentially compromise T-table lookups in a system, that represent "pre-processed S-box computations" based on AES design (Gruss, 2017; Daemen and Rijmen, 2013). Through this attack,

the whole AES algorithm can be utilised as a fast sequence of T-table lookups that will be accessed based on an established implementation (Gruss, 2017; Bernstein, 2005). Such accesses can then be cached, and the timing difference can be monitored (Bernstein, 2005). The attacker will subsequently be able to determine which T-table entry was accessed by monitoring the timing difference. Many researchers (Spreitzer and Gérard, 2014; Weiß et al., 2014; Spreitzer and Plos, 2013; Neve et al., 2006; Bonneau and Mironov, 2006) have reproduced and assessed Bernstein's Attack, that is presented in Bernstein's (2005) seminal study.

3.11. Branch Prediction Attack

A Branch Prediction Attack (BPA) combined with cache performance can be a potential source of control-dependent and data-dependent timing (Coppens et al., 2009; Chen et al., 2003). As already stated, in modern processors, many resources are shared between different threads being run in the system (Ge et al., 2016; Coppens et al., 2009; Aciiçmez et al., 2007). As a result, there will be conflict between those resources (Braun et al., 2015; Fedorova et al., 2010; Bonneau and Mironov, 2006). This leads to "inter-thread timing dependencies" (Ge et al., 2016), where the operation of one thread affects the timing performance of other threads (Martin et al., 2012). Therefore, an attacker will be able to monitor other threads competing for other resources to deduce information on condition that he does not have direct access to the timing of a thread which has come under attack (Coppens et al., 2009; Osvik et al., 2006; Bonneau and Mironov, 2006). To be able to launch a direct timing attack, the adversary needs to know the Branch Prediction Unit (BPU) architecture and also the implementation details of the encryption, as these two elements establish the prediction of the target branch (Aciiçmez et al., 2007). Although this information is not easily available to the attacker, he can still carry out the examination stage speculating each possible state one at a time. The DTA can be used on any system on condition that branch prediction algorithm is applied on it. To compromise a cipher carrying out a BPA, the adversary requires having a result which must rely on the secret/private key of the cipher (Aciiçmez et al., 2007). Furthermore, BPAs that are based on hardware performance can also be performed to elicit RSA keys from "exponentiations" executed in other processes (Gruss, 2017, Bhattacharya and Mukhopadhyay, 2015; Rebeiro et al., 2015).

3.12. Brief Overview of Timing-Based Attacks in Other Platforms

Although the focus of this study has been only on TBSCAs against PC platforms, nevertheless, we consider it worthwhile to provide a generic description of the way in which such attacks can be also carried out against other platforms such as mobile device. TBSCAs against mobile devices take advantage of both physical and software properties. For instance, a malign application can leverage the "accelerometer sensor" so as to launch an attack against the victim input. This is feasible because of the integral input technique that depends on touchscreens (Spreitzer et al., 2016; Aviv et al., 2012; Cai and Chen, 2011). Therefore, to perform a successful TBSCA against a mobile device, the adversary needs either to have a physical access to the device or remotely spread an application that appears to be benign (such as a game app) through an existing App store (Spreitzer et al., 2017; Spreitzer et al., 2016;). For example, through their study, O'Flynn (2016) illustrated that by shorting the "power supply of an off-the-shelf Android smartphone", the attacker would be able to present a fault that can result in an invalid a fault loop count (Spreitzer et al., 2017). Attackers can also exploit the logical property of software provided by the API of the mobile device OS or even the OS itself (Spreitzer et al., 2016; Michalevsky et al., 2015; Zhou et al., 2013) to be able to carry out TBSCAs against such devices. This suggests that smartphones expand the extent of TBSCAs (Acar et al., 2016; O'Flynn, 2016; Spreitzer et al., 2016).

On the contrary, TBSCAs mounted against cloud computing hardware does not require the adversary to be in possession of the physical hardware (This, however, does not apply in cases where the cloud service provider, himself, is the adversary) since the attacker can potentially run a malicious application remotely (Spreitzer et al., 2016). For instance, to do so, he will require to be able to exploit

the Microarchitectural performance or software characteristics in order to be able to deduce private information from co-located processes (Ge et al., 2016; Spreitzer et al., 2016; Gruss et al., 2015; Yarom and Falkner, 2014; Tromer et al., 2010). Furthermore, in certain circumstances, TBSCAs can also be mounted through websites without the adversary relying on the victim to install an application. Furthermore, adversaries can also launch Side-Channel Attacks against mobile devices by taking advantage of the logical property of software that is offered by the API of the mobile device OS or even the OS itself (Spreitzer et al., 2016; Michalevsky et al., 2015; Zhou et al., 2013).

There exist other ways of launching TBSCAs in various platforms, the analysis of which is beyond the scope of this paper due to the page constraints.

4. SIDE-CHANNEL ATTACKS AGAINST RSA

Although many studies have been conducted on the topic of RSA, there are very few works that explore its vulnerabilities to TBSCAs. As a result, RSA's vulnerabilities to TBSCAs are not still fully known by the research community. Therefore, we have assigned this section exclusively to the topic of RSA's vulnerabilities to TBSCAs in an attempt to provide a more in-depth analysis of the principles underlying its susceptibilities to TBSCAs.

4.1. Overview of RSA Algorithm

RSA algorithm named after its creators, Rivest-Shamir-Adleman, (Rivest, 1978) is a public key encryption algorithm that is widely utilised to secure sensitive data transmission, especially when transmitted over an insecure network. RSA can be embedded in SSL (Secure Sockets Layer) to provide security and privacy over the Internet. In cryptography field, an asymmetric key algorithm employs a pair of different cryptographic keys to perform encryption and decryption. Both keys are mathematically connected, denoting that a message encrypted by the algorithm using one key can be decrypted by the same algorithm such as RSA. The RSA algorithm includes four parts: key generation, key distribution, encryption and decryption. The underlying fundamental behind RSA is the notion that it is applied to discover three big positive integers 'e', 'd' and 'n' in such a way that with modular exponentiation for all integer m (with $0 \leq m < n$ and that even knowing e and n or even m), it can be very difficult to identify d:

$$\left(m^{e}\right)^{d} \equiv m\left(mod\, n\right)$$

Furthermore, the RSA consists of both a public key and the private key. The public key allows the sender to carry out the encryption whereas the private key remains secret by the receiver and allows him to conduct the decryption. In more technical details, the public key, which contains the "modulus" n and the public "exponent" e, can be known by all parties and is applied to encrypt sensitive data. The idea behind the public key is that any data encoded with the public key can only be decoded in a sensible amount of time by employing the private key. Integer e and integer n, that are created by two main numbers p and q, represent the public key, and integer d represents the private key. However, integer d can also be utilised while decoding the data. This denotes that integer d can be regarded as a constituent of the private key too. In contrast, integer m represents the message. In contrast with the public key, the private key comprises the "modulus" n and the private "exponent" d, that must be reserved secret. The p, q, and n must also remain secret since they can be employed by the attacker to deduce d. There are two types of asymmetric encryption algorithms, both of which are built upon the Diffie-Hellman key agreement algorithm. Similarly, there are two unique types of symmetric key ciphers including block ciphers (fixed size) and stream ciphers (continuous stream). Although asymmetric encryption is much stronger and more secure than symmetric encryption, as

of yet, there does not exist an asymmetric key algorithm confirmed to be secure enough against a sophisticated mathematical attack. This is due to certain weaknesses in the asymmetric key algorithms that make them vulnerable to Timing Attacks. For instance, a savvy attacker can carefully measure the precise amount of time that it takes hardware to encrypt plaintext in order to facilitate the search for possible decrypting keys. Therefore, the usage of asymmetric key algorithms does not always guarantee security.

The potency of the RSA lies with the fact that it will be hard to factor large numbers. Almost, the most recognised factoring techniques are still considered to be slow as factoring large numbers can take many months and in some cases years. Since so many researchers have been attempting to factor large numbers in an efficient manner without any success, we can assume that at this point in time RSA is secure against factoring attacks for a standard n of 1024 bit in length (Wong, 2005). Securing the RSA against factoring attacks even more is possible by doubling the key length to 2048 bits or even more. Notwithstanding its potency against factoring technique attacks, RSA can be vulnerable to Timing Attacks. An adversary with an advanced knowledge of algorithms and programming skills could potentially exfiltrate RSA secret keys in a stealthy manner without directly breaching the RSA. Such an attack has come to be known as Timing Attack, which we have already covered in detail, that enables an adversary to monitor the execution time of a cryptographic algorithm and as a result infer the secret keys and values in the execution.

4.2. Timing Attacks Against RSA Algorithm

There are various types of covert and overt SCAs that can be mounted against RSA (Inci et al., 2016; Yarom and Falkner, 2014; Wang and Lee, 2006; Percival 2005), including Timing Attacks (the focus of this study) (Brumley and Boneh, 2005; Kocher, 1996), Trace-Driven Instruction Cache (Cai-Sen et al., 2011; Aciiçmez, 2007) Differential Power Analysis (Bauer et al., 2013; Kocher et al., 1999), Electro Magnetic Emanations (Gandolfi et al., 2001; Genkin et al., 2015) and acoustic channels (Genkin et al., 2014; Hutter and Schmidt, 2013). Through a TBSCA against RSA, an attacker will be able to extract RSA private keys by utilising the Instruction Cache and the Branch Prediction Unit as covert channels (Inci et al., 2016; Aciiçmez, 2007; Aciiçmez et al., 2007). For instance, EL-Gamal (Zhang et al., 2012) private keys can be extracted from co-located VMs by taking advantage of emission in upper level caches. To launch a SCA against EL-Gamal, the attacker will require to use the Prime+Probe Attack in the LLC (Zhang et al., 2012; Liu et al., 2015). Likewise, the adversary can also employ the Branch Prediction Performance Counters to extract RSA keys (Inci et al., 2016; Bhattacharya and Mukhopadhyay, 2015). Furthermore, advanced attackers will also be able to extract RSA private keys from co-located VMs (Yarom and Falkner, 2014; Wang and Lee, 2006) by mounting the Flush+Reload Attack while memory De-Duplication is active (Yarom and Falkner, 2014).

A TBSCA against RSA can be carried out if the adversary knows the victim's hardware in adequate detail and if he is mathematically advanced enough to calculate the decryption time for multiple known ciphertexts. In such a case, the attacker will then be able to infer the decryption key, i.e. d, immediately (Kocher, 1999). Such an attack can also be performed against the RSA scheme (Brumley and Boneh, 2005) to extract RSA factorisation over a network connection from a web server enabled with Secure Sockets Layer (SSL). Another way of launching a TBSCA against RSA is through the exploitation of information that has been leaked by the Chinese Remainder Theorems Optimisation (a technique that is often utilised by many RSA implementations).

As already discussed, the Evict+Time Attack (Aciiçmez et al., 2007) against RSA is performed to evict entries from the BTB in a selective manner by running branches at relevant addresses, and then monitor the impact on the runtime of an RSA encryption in OpenSSL. To do so, the attacker will need to calculate the time that is needed to carry out the initial ejection to be able to deduce whether OpenSSL had beforehand run the branch or not. Although the Simple Branch Prediction Analysis (SBPA) (Aciiçmez et al., 2007) uses the same approach as that of the aforementioned approach, it is more powerful as it can exfiltrate most key bits in a single RSA execution. The SBPA is based

on the "fine- grained sharing" that is involved in SMT (Ge et al., 2016). Furthermore, attackers can identify and take advantage of the contention between hardware threads on the multiplier unit (Ge et al., 2016; Aciicmez and Seifert's, 2007; Wang and Lee, 2006). Simultaneous Multi-Threaded (SMT) processors (Tullsen et al., 1995), a method for optimising the overall efficiency of CPUs with hardware multithreading, execute multiple independent processes in parallel in order to make a more effective use of the resources offered by processor designs. The parallel threads share a cache of functional units (FUs) dynamically assigned to each process (Wang and Lee, 2006). Such a sharing creates interference between two difference processes as they will need to compete for FUs. This results in a 'covert channel', the extent of which surpasses those of other covert channels. Similarly, the conflicts between hardware threads on the multiplier unit can create a 'side channel' that enables a malign thread to differentiate multiplications from "squaring in OpenSSL's RSA architecture" (Ge et al., 2016; Aciicmez and Seifert's, 2007; Wang and Lee, 2006). These aforementioned attacks can enable an adversary to calculate the latency created by contending threats that are compelled to remain until they can access the multiplier unit.

The scatter-gather implementation employed in the modular exponentiation routine in OpenSSL is also prone to the CacheBleed attack, in which cache-bank conflicts on the Sandy Bridge microarchitecture can be exploited by the attackers (Fog, 2017; Intel®, 2016; Yarom et al., 2017). CacheBleed Attack, which has successfully been tested on an Intel Xeon E5-2430 processor (Yarom et al., 2017), allows the attacker to detect the cache pool that maintains each given multiplier utilised through the "exponentiation in the OpenSSL constant time RSA design" (Acıiçmez et al., 2007; Brickell et al., 2006;). As a result, it enables the attacker to exfiltrate the entire private key after he has monitored 16,000 decryptions for 4096-bit RSA (Yarom et al., 2017, Ge et al., 2016).

Furthermore, a thread that is running on a design of a SMT processor is also vulnerable to denial of service through a malign thread. This results in a significant reduction in the speed of the original thread. Therefore, an adversary can utilise Performance Counter Hardware to create this type of slowdown by intentionally abusing the shared resources and design decisions that are essential for high speed implementation (Grunwald and Ghiasi, 2002). Consequently, since a given thread can deny other threads (in resource sharing) of their resources through the usage of a multithreading processor, one thread can have an impact on the performance of another thread. Applying exceptional conditions on behalf of one thread can also create a significant performance degradation for another SMT thread (Ge et al., 2016; Grunwald and Ghiasi, 2002). Moreover, in certain processors (such as the Intel Pentium 4), self-adaptive code flushes the trace cache causing a significant reduction in performance (e.g. in a DoS attack). Although control techniques facilitated by resource sharing are capable of enhancing essential processor speed-paths, they can be taken advantage of by a single action by a malicious thread that can create many sets of delays.

Another type of Timing Attack against RSA is "Square vs. Multiplication" that is relevant to parallel multi-threading CPU designs. This attack cannot be carried out on CPU implementations without the aid of SMT hardware. Unlike other aforementioned attacks against RSA that often focus on the notion of a shared resource, Square vs. Multiplication attack is based on the concept that Intel's hyper-threading technology shares the ALU's large floating-point multiplier among its two hardware threads (Aciicmez and Seifert, 2007). An attacker can also exploit the timing performance of the Montgomery multiplications during the initialisation of the table enabling himself to extract one of the main factors of timing RSA moduli by adding to the number of multiplications (Biswas et al., 2017; Aciiçmez et al., 2005; Brumley and Tuveri, 2011; Brumley and Boneh, 2005). In addition, by performing a TBSCA, the entire RSA key in cloud settings can be recovered (Inci et al., 2016; Irazoqui et al., 2014; Yarom and Falkner, 2014) by the attackers, outside the closed-box VM, who can potentially exploit a vulnerable Virtual Machine Monitor (VMM) system that is running on top of a SMT processor (Inci et al., 2016; Irazoqui et al., 2014). A TBSCA on a hyper-threading processor also allows an attacker through a user processor to deduce the RSA key of another processor which is carrying out RSA encryption (Wang and Lee, 2006; Percival, 2005). There will be no need for

any type of special tools to facilitate this attack, and the attack also does not even need to rely on software defects to extract RSA. Only a series of memory accesses are required to be executed by the spy process, followed by its observation of the timing while the victim process is being executed on the same processor (Osvik et al., 2006; Wang and Lee, 2006; Bernstein, 2005; Percival, 2005).

5. CONCLUSION

In this study, we identified and analysed some of the existing known Timing-Based Side-Channel Attacks (TBSCAs), and demonstrated their devastating impacts on shared, modern computing hardware. We thoroughly reviewed relevant literature within the context and we discussed various attack vectors that attackers can adopt to mount such attacks against components of modern PC platforms. Through this systematic literature review and analysis, one can deduce that all Microarchitectural Timing 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. Although this study mainly focused on the review and analysis of Timing Attack vectors, in a follow-up paper as a future work, we are providing the existing countermeasures against such attacks and propose new strategies to deal with these attacks. There are already comprehensive research works (Yang et al., 2018; Sohal et al., 2018; Kuo et al., 2018) covering the response to such attacks; we will discuss these as a future work of this piece of research.

REFERENCES

Acar, Y., Backes, M., Bugiel, S., Fahl, S., McDaniel, P., & Smith, M. (2016). Sok: Lessons Learned from Android Security Research for Appified Software Platforms. In *IEEE Symposium on Security and Privacy (SP)* (pp. 433-451). doi:10.1109/SP.2016.33

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

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

Acıiçmez, O., & Koç, Ç. K. (2006). Trace-Driven Cache Attacks on AES (Short Paper). In P. Ning, S. Qing, & N. Li (Eds.), *Information and Communications Security. ICICS 2006*, LNCS (Vol. 4307). Berlin: Springer.

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

Aciiçmez, O., Schindler, W., & Koç, Ç. K. (2005). Improving Brumley and Boneh timing Attack on Unprotected SSL Implementations. In *Proceedings of the 12th ACM conference on Computer and Communications Security* (pp. 139-146). doi:10.1145/1102120.1102140

Aciicmez, O., & Seifert, J. P. (2007). Cheap Hardware Parallelism Implies Cheap Security. In *IEEE Workshop on Fault Diagnosis and Tolerance in Cryptography* (pp. 80-91). doi:10.1109/FDTC.2007.16

Agrawal, M. and Mishra. (2012). A Comparative Survey on Symmetric Key Encryption Techniques. *International Journal on Computer Science and Engineering*, *4*(5), 877.

Ahsan, K. (2002). *Covert Channel Analysis and Data Hiding in TCP/IP*. Canada: University of Toronto.

Allan, T., Brumley, B. B., Falkner, K., Van de Pol, J., & Yarom, Y. (2016). Amplifying Side Channels through Performance Degradation. In *Proceedings of the 32nd ACM Annual Conference on Computer Security Applications* (pp. 422-435).

Andrysco, M., Kohlbrenner, D., Mowery, K., Jhala, R., Lerner, S., & Shacham, H. (2015). On Subnormal Floating Point and Abnormal Timing. In *IEEE Symposium on Security and Privacy (*pp. 623-639). doi:10.1109/SP.2015.44

Apecechea, G. I., Inci, M. S., Eisenbarth, T., & Sunar, B. (2014). Fine Grain Cross-VM Attacks on Xen and VMware Are Possible! IACR Cryptology.

Aviv, A. J., Sapp, B., Blaze, M., & Smith, J. M. (2012). Practicality of Accelerometer Side Channels on Smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference* (pp. 41-50). doi:10.1145/2420950.2420957

Backes, M., & Nürnberger, S. (2014). Oxymoron: Making Fine-Grained Memory Randomization Practical by Allowing Code Sharing. In *USENIX Security Symposium* (pp. 433-447).

Balduzzi, M., Zaddach, J., Balzarotti, D., Kirda, E., & Loureiro, S. (2012). A Security Analysis of Amazon's Elastic Compute Cloud Service. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (pp. 1427-1434). doi:10.1145/2245276.2232005

Barenghi, A., Pelosi, G., & Teglia, Y. (2010). Improving First Order Differential Power Attacks through Digital Signal Processing. In *Proceedings of the 3rd ACM International Conference on Security of Information and Networks* (pp. 124-133). doi:10.1145/1854099.1854126

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., & Warfield, A. et al. (2003). Xen And the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles* (pp. 164-177).

Bates, A., Mood, B., Pletcher, J., Pruse, H., Valafar, M., & Butler, K. (2012). Detecting Co-Residency with Active Traffic Analysis Techniques. In *Proceedings of the ACM Workshop on Cloud computing security workshop*. doi:10.1145/2381913.2381915

Bauer, A., Jaulmes, E., Prouff, E., & Wild, J. (2013). Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations. doi:10.1007/978-3-642-36095-4_1

Bellare, M., Keelveedhi, S., & Ristenpart, T. (2013). Message-Locked Encryption and Secure Deduplication. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 296-312). Springer.

Benger, N., Van de Pol, J., Smart, N. P., & Yarom, Y. (2014). Ooh Aah... Just a Little Bit": A Small Amount of Side Channel Can Go a Long Way. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 75-92). Springer.

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

Bertozzi, S., Acquaviva, A., Bertozzi, D., & Poggiali, A. (2006). Supporting Task Migration in Multi-Processor Systems-On-Chip: A Feasibility Study. In *Proceedings of the Conference on Design, Automation and Test in Europe* (pp. 15-20). doi:10.1109/DATE.2006.243952

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

Bhattacharya, S., & Mukhopadhyay, D. (2015). Who Watches the Watchmen?: Utilizing Performance Monitors for Compromising Keys of RSA on Intel Platforms. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 248-266). Springer. doi:10.1007/978-3-662-48324-4_13

Biswas, A. K., Ghosal, D., & Nagaraja, S. (2017). A Survey of Timing Channels and Countermeasures. [CSUR]. *ACM Computing Surveys*, *50*(1), 6. doi:10.1145/3023872

Bogdanov, A., & Rijmen, V. (2014). Linear Hulls with Correlation Zero and Linear Cryptanalysis of Block Ciphers. *Designs, Codes and Cryptography*, *70*(3), 369–383. doi:10.1007/s10623-012-9697-z

Bonneau, J., & Mironov, I. (2006). Cache-Collision Timing Attacks against AES. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 201-215). Springer.

Braun, B. A., Jana, S., & Boneh, D. (2015). Robust and Efficient Elimination of Cache and Timing Side Channels. arXiv preprint arXiv:1506.00189.

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

Brumley, B., & Tuveri, N. (2011). Remote Timing Attacks Are Still Practical. In *European Symposium on Research in Computer Security* (pp. 355-371).

Brumley, B. B., & Hakala, R. M. (2009). Cache-Timing Template Attacks. In *15th International Conference on the Theory and Application of Cryptology and Information Security* (pp. 667-684).

Brumley, D., & Boneh, D. (2005). Remote Timing Attacks Are Practical. *Computer Networks*, *48*(5), 701–716. doi:10.1016/j.comnet.2005.01.010

Cai, L., & Chen, H. (2011). TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. *HotSec*, *11*, 9–9.

Cai-Sen, C., Tao, W., Xiao-Cen, C., & Ping, Z. (2011). An Improved Trace Driven Instruction Cache Timing Attack on RSA. IACR Cryptology.

Callan, R., Zajic, A., & Prvulovic, M. (2014). A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events. In *47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 242-254). doi:10.1109/MICRO.2014.39

Carmon, E., Seifert, J. P., & Wool, A. (2017). Photonic Side Channel Attacks Against RSA. IACR Cryptology. doi:10.1109/HST.2017.7951801

Chari, S., Jutla, C. S., Rao, J. R., & Rohatgi, P. (1999). Towards Sound Approaches to Counteract Power-Analysis Attacks. In M. Wiener (Ed.), *CRYPTO, LNCS* (Vol. *1666,* pp. 398–412). Heidelberg: Springer. doi:10.1007/3-540-48405-1_26

Chawla, S., Zheng, Y., & Hu, J. (2012). Inferring the Root Cause in Road Traffic Anomalies. In *12th IEEE International Conference on Data Mining (ICDM)* (pp. 141-150). doi:10.1109/ICDM.2012.104

Chen, C., Wang, T., Kou, Y., Chen, X., & Li, X. (2013). Improvement of Trace-Driven I-Cache Timing Attack on the RSA Algorithm. *Journal of Systems and Software*, *86*(1), 100–107. doi:10.1016/j.jss.2012.07.020

Chen, J., & Venkataramani, G. (2014). Cc-Hunter: Uncovering Covert Timing Channels on Shared Processor Hardware. In *47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 216-228). doi:10.1109/MICRO.2014.42

Chen, L., Dropsho, S., & Albonesi, D. H. (2003). Dynamic Data Dependence Tracking And Its Application To Branch Prediction. In *Proceedings of The 9th IEEE International Symposium on High-Performance Computer Architecture* (pp. 65-76).

Chen, Q. A., Qian, Z., & 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, S., Wang, R., Wang, X., & Zhang, K. (2010). Side-Channel Leaks in Web Applications: A Reality Today, A Challenge Tomorrow. In *IEEE Symposium on Security and Privacy (SP)* (pp. 191-206). doi:10.1109/SP.2010.20

Clements, A. (2006). *Principles of Computer Hardware* (4th ed.). Oxford University Press.

Coppens, B., Verbauwhede, I., De Bosschere, K., & De Sutter, B. (2009). Practical Mitigations for Timing-Based Side-Channel Attacks on Modern X86 Processors. In *30th IEEE Symposium on Security and Privacy* (pp. 45-60). doi:10.1109/SP.2009.19

Crane, S., Homescu, A., Brunthaler, S., Larsen, P., & Franz, M. (2015). *Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity*. doi:10.14722/ndss.2015.23264

Daemen, J., & Rijmen, V. (2013). *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media.

Danezis, G., & Clayton, R. (2007). *Introducing Traffic Analysis*. Boca Raton, FL: Auerbach Publications.

Davi, L., Liebchen, C., Sadeghi, A. R., Snow, K. Z., & Monrose, F. (2015). Isomeron: Code Randomization Resilient to (Just-In-Time) Return-Oriented Programming.

Domnitser, L., Jaleel, A., Loew, J., Abu-Ghazaleh, N., & Ponomarev, D. (2012). Non-Monopolizable Caches: Low-Complexity Mitigation of Cache Side Channel Attacks. *ACM Transactions on Architecture and Code Optimization*, *8*(4), 35. doi:10.1145/2086696.2086714

Doychev, G., Köpf, B., Mauborgne, L., & Reineke, J. (2015). CacheAudit: A Tool for the Static Analysis of Cache Side Channels. *ACM Transactions on Information and System Security*, *18*(1), 4. doi:10.1145/2756550

ElGamal, T. (1985). A Public Key Cryptosystem and A Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, *31*(4), 469–472. doi:10.1109/TIT.1985.1057074

Elson, J., Girod, L. & Estrin, D. (2002). Fine-Grained Network Time Synchronization Using Reference Broadcasts. *ACM SIGOPS Operating Systems Review*, *36*, 147-163.

Evtyushkin, D., Ponomarev, D., & Abu-Ghazaleh, N. (2016). Jump over ASLR: Attacking Branch Predictors to Bypass ASLR. In *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. doi:10.1109/MICRO.2016.7783743

Faruque, A., Abdullah, M., Chhetri, S. R., Canedo, A., & Wan, J. (2016). Acoustic Side-Channel Attacks on Additive Manufacturing Systems. In *Proceedings of the 7th IEEE International Conference on Cyber-Physical Systems*. doi:10.1109/ICCPS.2016.7479068

Fedorova, A., Blagodurov, S., & Zhuravlev, S. (2010). Managing Contention for Shared Resources On Multicore Processors. *Communications of the ACM*, *53*(2), 49–57. doi:10.1145/1646353.1646371

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

Foo Kune, D., & Kim, Y. (2010, October). Timing Attacks on Pin Input Devices. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (pp. 678-680).

Gandolfi, K., Mourtel, C., & Olivier, F. (2001). Electromagnetic Analysis: Concrete results. In Cryptographic Hardware and Embedded Systems (pp. 251-261). Springer Berlin/Heidelberg.

Garcia, C.P. & Brumley, B.B. (2016). Constant-Time Callees with Variable-Time Callers. IACR Cryptology ePrint Archive, Report 2016/1195.

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

Geddes, J., Schuchard, M., & Hopper, N. (2013). Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention. In *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security* (pp. 361-372). doi:10.1145/2508859.2516742

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

Genkin, D., Shamir, A., & Tromer, E. (2014). RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In *International Cryptology Conference* (pp. 444-461). Springer. doi:10.1007/978-3-662-44371-2_25

Gianvecchio, S., & Wang, H. (2011). An Entropy-Based Approach to Detecting Covert Timing Channels. *IEEE Transactions on Dependable and Secure Computing*, *8*(6), 785–797. doi:10.1109/TDSC.2010.46

Girling, C. G. (1987). Covert Channels in LAN's. *IEEE Transactions on Software Engineering*, *13*(2), 292–296. doi:10.1109/TSE.1987.233153

Gong, X., Borisov, N., Kiyavash, N., & Schear, N. (2012). Website Detection Using Remote Traffic Analysis. In Privacy Enhancing Technologies (pp. 58-78). doi:10.1007/978-3-642-31680-7_4

Gong, X., & Kiyavash, N. (2013). Timing Side Channels for Traffic Analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8697-8701).

Green, M. (2013). The Threat in the Cloud. *IEEE Security and Privacy*, *11*(1), 86–89. doi:10.1109/MSP.2013.20

Grunwald, D., & Ghiasi, S. (2002). Microarchitectural Denial of Service: Insuring Microarchitectural Fairness. In *Proceedings of the 35th Annual IEEE/ACM International Symposium Microarchitecture* (pp. 409-418). doi:10.1109/MICRO.2002.1176268

Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., . . . Yarom, Y. (2017). Another Flip in the Wall of Rowhammer Defenses. arXiv:1710.00551

Gruss, D., Maurice, C., Wagner, K., & 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., & Mangard, S. (2015). Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In *USENIX Security Symposium* (pp. 897-912).

Guan, L., Lin, J., Luo, B., Jing, J., & Wang, J. (2015). Protecting Private Keys Against Memory Disclosure Attacks Using Hardware Transactional Memory. In *IEEE Symposium on Security and Privacy (*pp. 3-19). doi:10.1109/SP.2015.8

Guilley, S., Hoogvorst, P., & Pacalet, R. (2004). *Differential Power Analysis Model and Some Results*. doi:10.1007/1-4020-8147-2_9

Gullasch, D., Bangerter, E., & Krenn, S. (2011). Cache Games--Bringing Access-Based Cache Attacks on AES to Practice. In *IEEE Symposium on Security and Privacy (*pp. 490-505). doi:10.1109/SP.2011.22

Gupta, D. (2007). Surgical Suites' Operations Management. *Production and Operations Management*, *16*(6), 689–700. doi:10.1111/j.1937-5956.2007.tb00289.x

Hachez, G., & Quisquater, J. J. (2000). Montgomery Exponentiation with No Final Subtractions: Improved Results. In Cryptographic Hardware and Embedded Systems—CHES (pp. 91-100). Springer Berlin/Heidelberg.

Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2010). Side Channels in Cloud Services: Deduplication in Cloud Storage. *IEEE Security and Privacy*, *8*(6), 40–47. doi:10.1109/MSP.2010.187

Hayashi, Y. I., Homma, N., Mizuki, T., Aoki, T., Sone, H., Sauvage, L., & Danger, J. L. (2013). Analysis of Electromagnetic Information Leakage from Cryptographic Devices with Different Physical Structures. *IEEE Transactions on Electromagnetic Compatibility*, *55*(3), 571–580. doi:10.1109/TEMC.2012.2227486

Homma, N., Aoki, T., & Satoh, A. (2010). Electromagnetic Information Leakage for Side-Channel Analysis of Cryptographic Modules. In *IEEE International Symposium on Electromagnetic Compatibility (EMC)* (pp. 97-102). doi:10.1109/ISEMC.2010.5711254

Hopper, N., Vasserman, E. Y., & Chan-Tin, E. (2010). How Much Anonymity Does Network Latency Leak? *ACM Transactions on Information and System Security*, *13*(2), 13. doi:10.1145/1698750.1698753

Hu, W. M. (1992). Reducing Timing Channels with Fuzzy Time. *Journal of Computer Security*, *1*(3-4), 233–254. doi:10.3233/JCS-1992-13-404

Hund, R., Willems, C., & Holz, T. (2013). Practical Timing Side Channel Attacks against Kernel Space ASLR. In *IEEE Symposium on Security and Privacy (*pp. 191-205). doi:10.1109/SP.2013.23

Hunger, C., Kazdagli, M., Rawat, A., Dimakis, A., Vishwanath, S., & Tiwari, M. (2015). Understanding Contention-Based Channels and Using Them for Defense. In *21st IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 639-650). doi:10.1109/HPCA.2015.7056069

Hutter, M., & Schmidt, J. M. (2013). The Temperature Side Channel and Heating Fault Attacks. In *International Conference on Smart Card Research and Advanced Applications* (pp. 219-235). Springer, Cham.

Inci, M. S., Gulmezoglu, B., Irazoqui, G., Eisenbarth, T., & 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. doi:10.1007/978-3-662-53140-2_18

Intel Corporation. (2016). Intel 64 and IA-32 Architectures Optimization Reference Manual.

Irazoqui, G., Eisenbarth, T., & Sunar, B. (2015a). S$A: A Shared Cache Attack That Works Across Cores and Defies VM Sandboxing-and Its Application to AES. In *IEEE Symposium on Security and Privacy (*pp. 591-604). doi:10.1109/SP.2015.42

Irazoqui, G., Eisenbarth, T., & Sunar, B. (2016). Cross Processor Cache Attacks. In *Proceedings of the 11th ACM Conference on Computer and Communications Security* (pp. 353-364).

Irazoqui, G., Inci, M. S., Eisenbarth, T., & Sunar, B. (2014). Wait a Minute! A Fast, Cross-VM Attack on AES. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 299-319). doi:10.1007/978-3-319-11379-1_15

Irazoqui, G., Inci, M. S., Eisenbarth, T., & Sunar, B. (2015b). Lucky 13 Strikes Back. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security* (pp. 85-96).

Irazoqui, G., Inci, M.S., Eisenbarth, T. and Sunar, B. (2015c). Know Thy Neighbor: Crypto Library Detection in Cloud. In *Proceedings on Privacy Enhancing Technologies* (Vol. 1, pp. 25-40).

Jang, Y., Lee, S., & Kim, T. 2016, October. Breaking Kernel Address Space Layout Randomization with Intel tsx. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security* (pp. 380-392). doi:10.1145/2976749.2978321

Jia, W., Shaw, K. A., & Martonosi, M. (2014). MRPB: Memory Request Prioritization for Massively Parallel Processors. In *20th IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 272-283). doi:10.1109/HPCA.2014.6835938

Kadloor, S., Gong, X., Kiyavash, N., Tezcan, T., & Borisov, N. (2010). Low-Cost Side Channel Remote Traffic Analysis Attack in Packet Networks. In *IEEE International Conference on Communications*. doi:10.1109/ICC.2010.5501972

Kambourakis, G., Damopoulos, D., Papamartzivanos, D., & Pavlidakis, E. (2016). Introducing Touchstroke: Keystroke-Based Authentication System for Smartphones. *Security and Communication Networks*, *9*(6), 542–554. doi:10.1002/sec.1061

Karlof, C., & Wagner, D. (2003). Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Ad Hoc Networks*, *1*(2), 293–315. doi:10.1016/S1570-8705(03)00008-8

Katabi, D. (2003). Decoupling Congestion Control and Bandwidth Allocation Policy With Application To High Bandwidth-Delay Product Networks [PhD Thesis]. Massachusetts Institute of Technology.

Kelsey, J., Schneier, B., Wagner, D., & Hall, C. (2000). 'Side Channel Cryptanalysis of Product Ciphers'. *Journal of Computer Security*, *8*(2-3), 141–158. doi:10.3233/JCS-2000-82-304

Kim, T., Peinado, M., & Mainar-Ruiz, G. (2012). STEALTHMEM: System-Level Protection Against Cache-Based Side Channel Attacks in the Cloud. In *USENIX Security Symposium* (pp. 189-204).

Kirsch, C. M., & Sokolova, A. (2012). The Logical Execution Time Paradigm. In Advances in Real-Time Systems (pp. 103-120). Springer Berlin Heidelberg. doi:10.1007/978-3-642-24349-3_5

Kocher, P., Jaffe, J., & Jun, B. (1999). Differential Power Analysis. In Advances in Cryptology—CRYPTO'99 (pp. 789-789). Springer Berlin/Heidelberg. doi:10.1007/3-540-48405-1_25

Kocher, P., Jaffe, J., Jun, B., & Rohatgi, P. (2011). Introduction to Differential Power Analysis. *Journal of Cryptographic Engineering*, *1*(1), 5–27. doi:10.1007/s13389-011-0006-y

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

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. doi:10.1007/3-540-68697-5_9

Kocher, P. C., Rohatgi, P., & Jaffe, J. M. (2017). U.S. Patent No. 9,569,623. Washington, DC: U.S. Patent and Trademark Office.

Kocher, P.C., Rohatgi, P. and Jaffe, J.M. (2017). Secure boot with resistance to differential power analysis and other external monitoring attacks.

Kong, J., Aciiçmez, O., Seifert, J. P., & Zhou, H. (2009). Hardware-Software Integrated Approaches to Defend Against Software Cache-Based Side Channel Attacks. In *15th IEEE International Symposium on High Performance Computer Architecture* (pp. 393-404). doi:10.1109/HPCA.2009.4798277

Kong, J., Aciicmez, O., Seifert, J. P., & Zhou, H. (2013). Architecting against Software Cache-Based Side-Channel Attacks. *IEEE Transactions on Computers*, *62*(7), 1276–1288. doi:10.1109/TC.2012.78

Kotcher, R., Pei, Y., Jumde, P., & Jackson, C. (2013). Cross-Origin Pixel Stealing: Timing Attacks Using CSS Filters. In *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security* (pp. 1055-1062). doi:10.1145/2508859.2516712

Krämer, J., Nedospasov, D., Schlösser, A., & Seifert, J. P. (2013). Differential Photonic Emission Analysis. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer.

Kuo, C., Chi, P., Chang, V., & Lei, C. (in press). SFaaS: Keeping an eye on IoT fusion environment with security fusion as a service. *Future Generation Computer Systems*. doi:10.1016/j.future.2017.12.069

Kurose, J. F., & Ross, K. W. (2010). *Computer Networking: A Top-Down Approach*. Addison-Wesley.

Lampson, B. W. (1973). A Note on The Confinement Problem. *Communications of the ACM*, *16*(10), 613–615. doi:10.1145/362375.362389

Lange, M., Liebergeld, S., Lackorzynski, A., Warg, A., & Peter, M. (2011). L4Android: A Generic Operating System Framework for Secure Smartphones. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices* (pp. 39-50). doi:10.1145/2046614.2046623

Larsen, P., Homescu, A., Brunthaler, S., & Franz, M. (2014). SoK: Automated Software Diversity. In *IEEE Symposium on Security and Privacy* (pp. 276-291).

Lawson, N. (2009). Side-Channel Attacks on Cryptographic Software. *IEEE Security and Privacy*, *7*(6), 65–68. doi:10.1109/MSP.2009.165

Lee, J., Cho, K., Lee, C., & Kim, S. (2015). VoIP-Aware Network Attack Detection Based on Statistics and Behavior of SIP Traffic. *Peer-to-Peer Networking and Applications*, *8*(5), 872–880. doi:10.1007/s12083-014-0289-8

Lee, S., Shih, M. W., Gera, P., Kim, T., Kim, H., & Peinado, M. (2016). Inferring Fine-Grained Control Flow Inside SGX Enclaves with Branch Shadowing. arXiv:1611.06952

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

Liu, F., Ge, Q., Yarom, Y., Mckeen, F., Rozas, C., Heiser, G., & Lee, R. B. (2016). Catalyst: Defeating last-level cache side channel attacks in cloud computing. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 406-418). doi:10.1109/HPCA.2016.7446082

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

Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A. R., Schulz, S., & Katzenbeisser, S. (2009). Hide and Seek in Time-Robust Covert Timing Channels. In *Proceedings 14th European Symposium on Research in Computer Security*, Saint-Malo, France (pp. 120-135). doi:10.1007/978-3-642-04444-1_8

Liu, Y., Ghosal, D., Armknecht, F., Sadeghi, A. R., Schulz, S., & Katzenbeisser, S. (2010). Robust and Undetectable Steganographic Timing Channels for iid Traffic. *Information Hiding*, *6387*, 193–207. doi:10.1007/978-3-642-16435-4_15

Longo, J., De Mulder, E., Page, D., & Tunstall, M. (2015). SoC it to EM: Electromagnetic Side-Channel Attacks on a Complex System-on-Chip. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 620-640). Springer. doi:10.1007/978-3-662-48324-4_31

Luo, X., Chan, E. W., & Chang, R. K. (2008, June). TCP Covert Timing Channels: *Design and Detection. In IEEE International Conference on Dependable Systems and Networks with FTCS and DCC* (pp. 420-429).

Luo, X., Zhou, P., Chan, E. W., Lee, W., Chang, R. K., & Perdisci, R. (2011). HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. NDSS, 11.

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

Mardani, M., & Giannakis, G. B. (2016). Estimating Traffic and Anomaly Maps Via Network Tomography. *Biological Cybernetics*, *24*(3), 1533–1547.

Martin, R., Demme, J., & Sethumadhavan, S. (2012). Timewarp: Rethinking Timekeeping and Performance Monitoring Mechanisms to Mitigate Side-Channel Attacks. *ACM SIGARCH Computer Architecture News*, *40*(3), 118–129. doi:10.1145/2366231.2337173

Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Boano, C. A., & Römer, K. et al. (2017). *Hello from The Other Side: SSH over Robust Cache Covert Channels in the Cloud*. San Diego, CA, US: NDSS.

Mazurczyk, W., Szaga, P., & Szczypiorski, K. (2014). Using Transcoding for Hidden Communication in IP Telephony. *Multimedia Tools and Applications*, *70*(3), 2139–2165. doi:10.1007/s11042-012-1224-8

Meyer, C., Somorovsky, J., Weiss, E., Schwenk, J., Schinzel, S., & Tews, E. (2014). Revisiting SSL/TLS Implementations: *New Bleichenbacher Side Channels and Attacks. In USENIX Security Symposium* (pp. 733-748).

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

Mohaban, S., Parnafes, I., & Kahane, O. (2007). U.S. Patent No. 7,209,473. Washington, DC: U.S. Patent and Trademark Office.

Moradi, A., Barenghi, A., Kasper, T., & 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). doi:10.1145/2046707.2046722

Mouha, N., Wang, Q., Gu, D., & Preneel, B. (2011). Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In *International Conference on Information Security and Cryptology* (pp. 57-76).

Murray, T., Matichuk, D., Brassil, M., Gammie, P., Bourke, T., Seefried, S., & Klein, G. et al. (2013). seL4: from General Purpose to a Proof of Information Flow Enforcement. In *IEEE Symposium on Security and Privacy (*pp. 415-429). doi:10.1109/SP.2013.35

Neve, M., & Seifert, J. P. (2006). Advances on Access-Driven Cache Attacks on AES. In Selected Areas in Cryptography (pp. 147-162).

Neve, M., Seifert, J. P., & Wang, Z. (2006). A Refined Look at Bernstein's AES Side-Channel Analysis. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security* (pp. 369-369). doi:10.1145/1128817.1128887

O'Flynn, C. (2016). Fault Injection Using Crowbars on Embedded Systems. IACR Cryptology.

OpenSSL. (2016). OpenSSL Cryptography and SSL/TLS Toolkit. Retrieved 12th October 2017 from https://www.openssl.org/

Oren, Y., Kemerlis, V. P., Sethumadhavan, S., & Keromytis, A. D. (2015). The Spy in the Sandbox: Practical Cache Attacks in JavaScript and Their Implications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1406-1418). doi:10.1145/2810103.2813708

Osvik, D. A., Shamir, A., & Tromer, E. (2006). Cache Attacks and Countermeasures: The Case of AES. In *The RSA Conference Cryptographers' Track*. doi:10.1007/11605805_1

Otmani, A., Tillich, J. P., & Dallot, L. (2010). Cryptanalysis of Two McEliece cryptosystems Based on Quasi-Cyclic Codes. *Mathematics in Computer Science*, *3*(2), 129–140. doi:10.1007/s11786-009-0015-8

Owusu, E., Han, J., Das, S., Perrig, A., & Zhang, J. (2012). ACCessory: Password Inference Using Accelerometers on Smartphones. In *Proceedings of the 12th ACM Workshop on Mobile Computing Systems & Applications*. doi:10.1145/2162081.2162095

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

Page, D. (2005). Partitioned Cache Architecture as a Side-Channel Defence Mechanism. IACR Cryptology.

Partan, J., Kurose, J., & Levine, B. N. (2007). A Survey of Practical Issues in Underwater Networks. *Mobile Computing and Communications Review*, *11*(4), 23–33. doi:10.1145/1347364.1347372

Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann.

PaX. (2001). Address Space Layout Randomisation (ASLR). Retrieved 31st December 2017 from https://pax.grsecurity.net/docs/aslr.txt

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

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

PorninT. (2017). Why Constant-Time Crypto? Retrieved 17th December 2017 from https://www.bearssl.org/constanttime.html

Quisquater, J. J., & Samyde, D. (2001). *Electromagnetic Analysis (ema)* (pp. 200–210). Measures and Counter-Measures for Smart Cards. Smart Card Programming and Security.

Raymond, J. F. (2001). Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In Designing Privacy Enhancing Technologies (pp. 10-29). Springer Berlin/Heidelberg.

Rebeiro, C., Mukhopadhyay, D., & Bhattacharya, S. (2015). *'Branch Prediction Attacks'. Timing Channels in Cryptography* (pp. 125–137). Springer International Publishing.

Rhee, I. K., Lee, J., Kim, J., Serpedin, E., & Wu, Y. C. (2009). Clock Synchronization in Wireless Sensor Networks: An Overview. *Sensors*, *9*(1), 56–85. doi:10.3390/s90100056 PMID:22389588

Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009). Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security* (pp. 199-212). doi:10.1145/1653662.1653687

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126. doi:10.1145/359340.359342

Rowland, C. H. (1997). Covert channels in the TCP/IP protocol suite. *First Monday*, *2*(5). doi:10.5210/fm.v2i5.528

Saltaformaggio, B., Xu, D., & Zhang, X. (2013). *Busmonitor: A Hypervisor-Based Solution for Memory Bus Covert Channels.* In Proceedings of EuroSec.

Sarwar, G., Mehani, O., Boreli, R., & 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).

Schaefer, M., Gold, B., Linde, R., & Scheid, J. (1977). Program Confinement in KVM/370. In *Proceedings of the Annual ACM Conference* (pp. 404-410).

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

SchneierB. (2005). AES Timing Attack. Retrieved 23rd September 2017 from https://www.schneier.com/blog/archives/2005/05/aes_timing_atta_1.html

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

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

Seibert, J., Okhravi, H., & Söderström, E. (2014). Information Leaks Without Memory Disclosures: Remote Side Channel Attacks on Diversified Code. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security* (pp. 54-65). doi:10.1145/2660267.2660309

Shafiee, A., Gundu, A., Shevgoor, M., Balasubramonian, R., & Tiwari, M. (2015). Avoiding Information Leakage in the Memory Controller with Fixed Service Policies. In *Proceedings of the 48th ACM International Symposium on Microarchitecture* (pp. 89-101). doi:10.1145/2830772.2830795

Shmatikov, V., & Wang, M. H. (2006). *Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses* (pp. 18–33). Computer Security–ESORICS.

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

Snow, K. Z., Monrose, F., Davi, L., Dmitrienko, A., Liebchen, C., & Sadeghi, A. R. (2013). Just-In-Time Code Reuse: On the Effectiveness of Fine-Grained Address Space Layout Randomization. In *IEEE Symposium on Security and Privacy (*pp. 574-588). doi:10.1109/SP.2013.45

Sohal, A. S., Sandhu, R., Sood, S. K., & Chang, V. (2018). A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. *Computers & Security*, *74*, 340–354. doi:10.1016/j.cose.2017.08.016

Song, C., Lin, F., Ba, Z., Ren, K., Zhou, C., & 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). doi:10.1145/2976749.2978300

Song, D. X., Wagner, D., & Tian, X. (2001). Timing Analysis of Keystrokes and Timing Attacks on SSH. In *Proceedings of the 10th USENIX Security*.

Song, J., Lee, K., & Lee, H. (2013). Biclique Cryptanalysis on Lightweight Block Cipher: HIGHT and Piccolo. *International Journal of Computer Mathematics*, *90*(12), 2564–2580. doi:10.1080/00207160.2013.767445

Spreitzer, R., & Gérard, B. (2014). Towards More Practical Time-Driven Cache Attacks. In *IFIP International Workshop on Information Security Theory and Practice* (pp. 24-39). Springer.

Spreitzer, R., Moonsamy, V., Korak, T., & Mangard, S. (2016). SoK: Systematic Classification of Side-Channel Attacks on Mobile Devices. arXiv:1611.03748

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

Spreitzer, R., & Plos, T. (2013). Cache-Access Pattern Attack on Disaligned AES T-Tables. In *International Workshop on Constructive Side-Channel Analysis and Secure Design* (pp. 200-214). Springer. doi:10.1007/978-3-642-40026-1_13

Stefan, D., Buiras, P., Yang, E. Z., Levy, A., Terei, D., Russo, A., & Mazières, D. (2013). Eliminating Cache-Based Timing Attacks with Instruction-Based Scheduling. In *European Symposium on Research in Computer Security* (pp. 718-735). Springer. doi:10.1007/978-3-642-40203-6_40

Steffan, J. G., Colohan, C. B., Zhai, A., & Mowry, T. C. (2000). A Scalable Approach to Thread-Level Speculation. *ACM SIGARCH Computer Architecture News*, *28*(2). doi:10.1145/342001.339650

Stone, P. 2013. Pixel Perfect Timing Attacks with HTML5. Context Information Security (White Paper).

Sultana, S., Shehab, M., & Bertino, E. (2013). Secure Provenance Transmission for Streaming Data. *IEEE Transactions on Knowledge and Data Engineering*, *25*(8), 1890–1903. doi:10.1109/TKDE.2012.31

Symantec. (2017). Types of Memory Exploit Mitigation Techniques. Retrieved 31st December 2017 from https://support.symantec.com/en_US/article.HOWTO127179.html

Tromer, E., Osvik, D. A., & Shamir, A. (2010). Efficient Cache Attacks on AES, and Countermeasures. *Journal of Cryptology*, *23*(1), 37–71. doi:10.1007/s00145-009-9049-y

Tsai, J. Y., & Yew, P. C. 1996, October. The Superthreaded Architecture: Thread Pipelining with Run-Time Data Dependence Checking and Control Speculation. In *Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques* (pp. 35-46). doi:10.1109/PACT.1996.552553

Tsai, M. F., Chilamkurti, N., Park, J. H., & Shieh, C. K. (2010). Multi-Path Transmission Control Scheme Combining Bandwidth Aggregation and Packet Scheduling For Real-Time Streaming In Multi-Path Environment. *IET Communications*, *4*(8), 937–945. doi:10.1049/iet-com.2009.0661

Tsunoo, Y., Saito, T., Suzaki, T., Shigeri, M., & Miyauchi, H. (2003). Cryptanalysis of DES Implemented on Computers with Cache. In C. D. Walter, Ç. K. Koç, & C. Paar (Eds.), *Cryptographic Hardware and Embedded Systems - CHES '03*, LNCS (Vol. 2779). Berlin: Springer. doi:10.1007/978-3-540-45238-6_6

Varadarajan, V., Ristenpart, T., & Swift, M. M. (2014). Scheduler-Based Defenses against Cross-VM Side-Channels. In *USENIX Security Symposium* (pp. 687-702).

Vétillard, E., & Ferrari, A. (2010). Combined Attacks and Countermeasures. In *International Conference on Smart Card Research and Advanced Applications* (pp. 133-147). Springer.

Walter, C. D. (1999). Montgomery's Multiplication Technique: How to Make It Smaller and Faster. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 80-93). Springer. doi:10.1007/3-540-48059-5_9

Wang, W., Chen, G., Pan, X., Zhang, Y., Wang, X., Bindschaedler, V., . . . Gunter, C. A. (2017). Leaky cauldron on the dark land: understanding memory side-channel hazards in SGX. arXiv:1705.07289

Wang, X., Chen, Y., Wang, Z., Qi, Y., & Zhou, Y. (2015). SecPod: a Framework for Virtualization-based Security Systems. In *USENIX Annual Technical Conference* (pp. 347-360).

Wang, Y., Ferraiuolo, A., & Suh, G. E. (2014). Timing Channel Protection for A Shared Memory Controller. In *20th IEEE International Symposium on High Performance Computer Architecture (HPCA)* (pp. 225-236). doi:10.1109/HPCA.2014.6835934

Wang, Z., & Lee, R. B. (2007). New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks. In *Proceedings of the 34th Annual ACM International Symposium on Computer Architecture* (pp. 494-505). doi:10.1145/1250662.1250723

Wang and Lee. (2006). Covert and Side Channels due to Processor Architecture. In *22nd IEEE Annual Conference on Computer Security Applications* (pp. 473-482).

Wei, M., Heinz, B., & Stumpf, F. (2012). *A Cache Timing Attack on AES in Virtualization Environments. In* Financial Cryptography and Data Security (pp. 314–328).

Weiß, M., Heinz, B., & Stumpf, F. (2012). *A Cache Timing Attack on AES in Virtualization Environments*. Financial Cryptography and Data Security (pp. 314–328).

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

Welzl, M. (2005). *Network Congestion Control: Managing Internet Traffic*. John Wiley & Sons. doi:10.1002/047002531X

Welzl, M. (2012). *Scalable Performance Signalling and Congestion Avoidance*. Springer Science & Business Media.

Wendzel, S., Zander, S., Fechner, B., & Herdin, C. (2015). Pattern-Based Survey and Categorization of Network Covert Channel Techniques. *ACM Computing Surveys*, *47*(3), 50. doi:10.1145/2684195

Winder, D. (2012). Side channel attacks could threaten cloud security in a big way. Best to be prepared. Retrieved 2nd January 2018 from http://www.cloudpro.co.uk/cloud-essentials/cloud-security/5010/cryptography-attack-side-channel-cloud-threat-all-nerd-and-no-k

Wong, W. H. (2005). Timing Attacks on RSA: Revealing Your Secrets through the Fourth Dimension. *Crossroads*, *11*(3), 5–5. doi:10.1145/1144396.1144401

Wray, J. C. (1992). An Analysis of Covert Timing Channels. *Journal of Computer Security*, *1*(3-4), 219–232. doi:10.3233/JCS-1992-13-403

Wu, J., Cheng, B., Yuen, C., Shang, Y., & Chen, J. (2015). Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing*, *14*(4), 688–701. doi:10.1109/TMC.2014.2334592

Wu, J., Cheng, B., Yuen, C., Shang, Y., & Chen, J. (2015). Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing*, *14*(4), 688–701. doi:10.1109/TMC.2014.2334592

Xiao, Y., Zhang, X., Zhang, Y., & 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., & Xiao, Y. (2013). Security and Privacy in Cloud Computing. *IEEE Communications Surveys and Tutorials*, *15*(2), 843–859. doi:10.1109/SURV.2012.060912.00182

Xu, Z., Bai, K., & Zhu, S. (2012). Taplogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors. In *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (pp. 113-124). doi:10.1145/2185448.2185465

Yang, Y., Zheng, X., Guo, W., Liu, X., & Chang, V. (in press). Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system. *Information Sciences*. doi:10.1016/j.ins.2018.02.005

Yarom, Y., & Benger, N. (2014). Recovering OpenSSL ECDSA Nonces Using the FLUSH+ RELOAD Cache Side-channel Attack. IACR Cryptology.

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

Yarom, Y., Genkin, D., & Heninger, N. (2017). CacheBleed: A Timing Attack on OpenSSL Constant-Time RSA. *Journal of Cryptographic Engineering*, *7*(2), 99–112. doi:10.1007/s13389-017-0152-y

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

Zander, S., Armitage, G., & Branch, P. (2007). A Survey of Covert Channels and Countermeasures In Computer Network Protocols. *IEEE Communications Surveys and Tutorials*, *9*(3), 44–57. doi:10.1109/COMST.2007.4317620

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

Zhang, T., & Lee, R. B. (2014). *Secure Cache Modeling for Measuring Side-Channel Leakage (Technical Report*). Princeton University.

Zhang, T., Zhang, Y., & 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. doi:10.1007/978-3-319-45719-2_6

Zhang, Y., Juels, A., Reiter, M. K., & Ristenpart, T. (2012). Cross-VM Side Channels and Their Use to Extract Private Keys. In *Proceedings of the ACM Conference on Computer and Communications Security* (pp. 305-316). ACM. doi:10.1145/2382196.2382230

Zhang, Y., Juels, A., Reiter, M. K., & Ristenpart, T. (2014). Cross-Tenant Side-Channel Attacks in PaaS Clouds. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security* (pp. 990-1003).

Zhang, Y., & Reiter, M. K. (2013). Düppel: Retrofitting Commodity Operating Systems to Mitigate Cache Side Channels in the Cloud. In *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security* (pp. 827-838). doi:10.1145/2508859.2516741

Zhou, X., Demetriou, S., He, D., Naveed, M., Pan, X., Wang, X., & Nahrstedt, K. et al. (2013). Identity, Location, Disease and More: Inferring Your Secrets from Android Public Resources. In *Proceedings of The ACM SIGSAC Conference on Computer & Communications Security* (pp. 1017-1028). doi:10.1145/2508859.2516661

Zhou, Y., & Feng, D. (2005). Side-channel attacks: ten years after its publication and the impacts on cryptographic module security testing. IACR Cryptology.

Reza Montasari is a lecturer at the School of Computing and Digital Technology, Faculty of Computing, Engineering and the Built Environment, Birmingham City University. He holds a BSc (Hons) in Multimedia Computing, an MSc and PhD both in Digital Forensics. He is also a member of IET (MIET) and is currently working towards becoming a CEng. Reza has published numerous research papers and serves as a programme committee member for various reputable international journals and conferences.

Amin Hosseinian-Far holds the position of Senior Lecturer in Business Systems & Operations at the University of Northampton, United Kingdom. In his previous teaching experience, Amin was a Staff Tutor at the Open University, UK, a Senior Lecturer and Course Leader at Leeds Beckett University. He has held lecturing and research positions at the University of East London, and at a number of private HE institutions and strategy research firms. Dr. Hosseinian-Far has also worked as Deputy Director of Studies at a large private higher education institute in London. He received his BSc (Hons) in Business Information Systems from the University of East London, an MSc degree in Satellite Communications and Space Systems from the University of Sussex, a Postgraduate Certificate in Research and a PhD degree titled 'A Systemic Approach to an Enhanced Model for Sustainability' which he acquired from the University of East London. Dr. Hosseinian-Far holds Membership of the Institution of Engineering and Technology (IET), Senior Fellowship of the Higher Education Academy (HEA), and Fellowship of the Royal Society of Arts (RSA).

Richard Hill is Head of the Department of Computer Science and Director of the Centre for Industrial Analytics at the University of Huddersfield, UK. Professor Hill has published widely in the areas of Big Data, predictive analytics, the Internet of Things, edge analytics and Industry 4.0, and has specific interests in digital manufacturing.