Chapter

Security at the Edge

Charles J. Gillan and George Karakonstantis

Abstract

The Internet has become an essential part of daily life for almost everyone in society having grown far beyond its roots in the 1970s as the ARPANET, a network that was principally the domain of scientists and engineers. The popularity of the HTTP, developed at CERN in the late 1980s led to the widespread use of the term 'the web' as a generic name for the Internet for many years, at least in the public domain. Of course, the Internet is much more than just web browsing and, in recent years, the term cyberspace has become the most popular term to describe interactions over the Internet. Yet, an unambiguous definition of the term is difficult to formulate . Financial institutions underpinning the economy and the operation of national critical infrastructures, such as monitoring and control of the electricity supply, are now dependent on the Internet. A consequence of this is that cyberattacks become more costly for the victims and perversely more attractive to the criminals who carry them out . The advent of the Internet of Things (IoT) and edge computing as a new paradigm creates the potential for enhanced productivity but at the same time opens up new opportunities for cyberattacks while still being exposed to existing attach vectors such as the well-known denial of service attack (DDoS), which can take place in many forms . In this chapter, we described the challenges in building an edge system that is secure against cyberattack. We begin by briefly reviewing the architecture of communications over the Internet and later consider the new challenges that follow from operating the hardware with values of voltage, frequency and current that enable more energy efficiency.

Keywords: security, energy efficiency, performance, cloud, edge computing

1. Introduction

The Internet has become an essential part of daily life for almost everyone in society having grown far beyond its roots in the 1970s as the ARPANET, a network that was principally the domain of scientists and engineers. The popularity of the HTTP, developed at CERN in the late 1980s, led to the widespread use of the term 'the web' as a generic name for the Internet for many years, at least in the public domain. Of course, the Internet is much more than just web browsing and, in recent years, the term cyberspace has become the most popular term to describe interactions over the Internet. Yet, an unambiguous definition of the term is difficult to formulate [1].

Financial institutions underpinning the economy and the operation of national critical infrastructures, such as monitoring and control of the electricity supply, are now dependent on the Internet. A consequence of this is that cyberattacks become more costly for the victims and perversely more attractive to the criminals who carry them out [2]. The advent of the Internet of Things (IoT) and edge computing

as a new paradigm creates the potential for enhanced productivity but at the same time opens up new opportunities for cyberattacks while still being exposed to existing attack vectors such as the well-known denial of service attack (DDoS), which can take place in many forms [3].

In this chapter, we described the challenges in building an edge system that is secure against cyberattack. We begin by briefly reviewing the architecture of communications over the Internet and later consider the new challenges that follow from operating the hardware with values of voltage, frequency and current that enable more energy efficiency.

2. The structure of the internet: security from data Centre to the edge

There is a proverb in the English language that says that a chain is only as strong as its weakest link. This applies directly as a basic principle of cybersecurity. Edge computing still requires communications to a central data centre, at least some of the time. It follows that it is necessary to consider carefully the WAN and LAN technologies used. **Figure 1** illustrates the networking technologies used and shows the position of edge computing within the wider fog computing environment, which we describe in a later part of this section. The section begins by discussing each networking technology separately and in doing so refers briefly to the history of the development of data networking technologies in general and to the development of the Internet in particular.

2.1 WAN technologies—circuit-based communications

The core transmission technology of the global telephone system developed over several decades from using electromechanical switches and frequency division multiplexing to use digital signals and time division multiplexing by the 1980s. Signals from different sources were multiplexed together in a hierarchy of data rates (2.048 Mbps, 8.448 Mbs, 34.368 Mbps, etc.) for transport across the core network

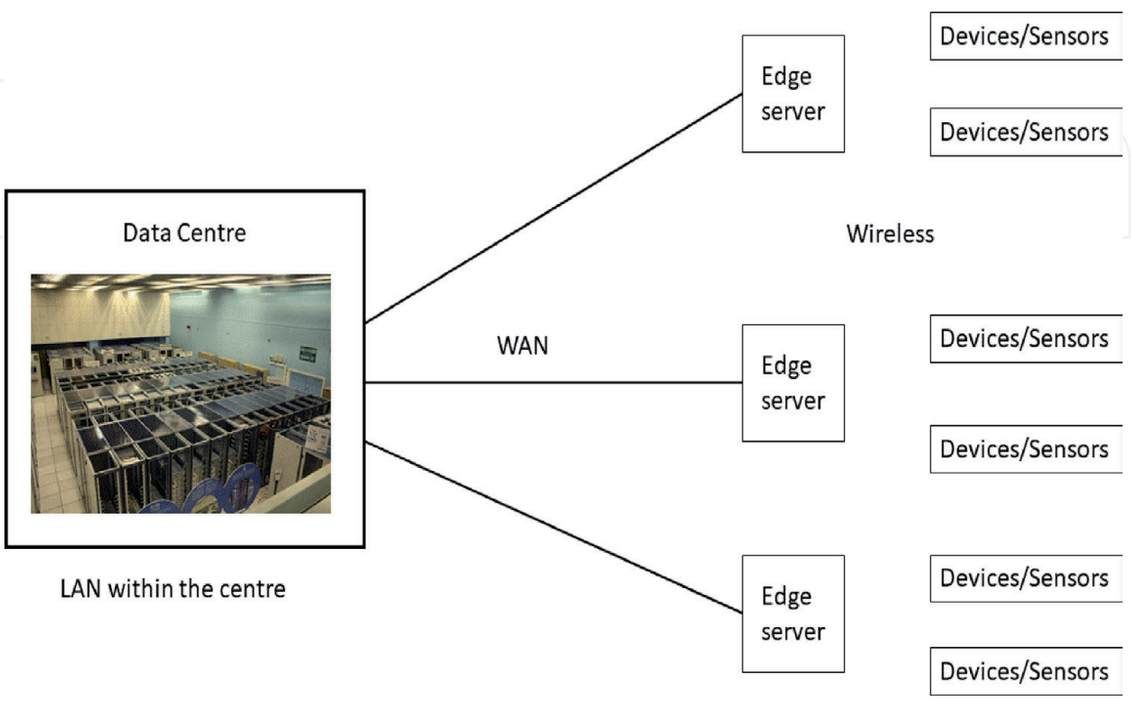

Figure 1.
Illustration of the hierarchy of devices creating the fog computing environment.

before being demultiplexed for transmission to individual receivers. The concept of a unique end-to-end circuit from sender remained clearly identifiable.

The initial plesiochronous digital hierarchy (PDH) handled the fact that lower bit rate sources were not time synchronised (each source had its own clock) by adopting the technique of bit stuffing in order to ensure that the higher rate channels were time synchronised. Thus, equipment inserted extra bits, as needed, at the transmitter and then the receiver removed these bits.

As fibre optic became widely used in the telecommunications industry, PDH was replaced by a different, more scalable, multiplexing technology known as the synchronous digital hierarchy (SDH) in which the equipment across the network is synchronised. SDH works on copper lines and on radio signals as well as fibre optic cables. The ITU-T [4] develops standards for SDH globally. The United States developed the technology under the name Synchronous Optical Network (SONET) around the same time as the ITU-T. In SDH, an aggregate signal composed of virtual containers (VCs) of fixed size is transmitted at a fixed frequency between two pieces of SDH equipment. Each tributary signal arriving at the sender from a source is assigned to one of the VCs with a pointer indicating where the signal is located within the container. Thus by allowing the pointer to vary, the tributary signals are adapted to the synchronised clock of the transmitter and receiver.

While a transmission from source to receiver will pass though many different VCs as it transits the SDH network, essentially using a different VC on each point to point link, the concept of an identifiable circuit remains intact in SDH. This means that distinct users and applications are clearly separated despite the fact that they are carried over the same fibre, wire or radio link. Even if one captures the complete SDH aggregate signal, without knowledge of the mapping of users and applications to the VCs in the signal, it is essentially impossible to extract the targeted data stream.

2.2 Packet communications

The circuit concept in the telephone system described in the above section builds on the idea of reserving bandwidth between the transmitter and receiver although as we have mentioned this confers a certain level of security by separating the signal from others on the same physical medium.

An alternative approach that is available when the transmission is in digital form is to break it into parts and then to transmit these parts in sequence across the digital network. We can define a packet to have three parts: a header, a payload and optionally a trailer. Each part of the digital data is placed uniquely into one packet and the header defines the information that allows the packet to be transmitted across the digital network. This type of transmission, known as packet switching, is the primary basis for data communications in computer networks, whether local or wide area. The definition of the fields in the header (and trailer, if present) plus the functionality associated with each field defines a protocol. The development of early networks, such as the ARPANET discovered that it was useful to encapsulate protocols within other protocols leading to the concept of a layered stack. This was eventually formalised in the definition of a seven-layer abstract model known as the Open Systems Interconnection (OSI) model [5].

As the Internet was adopted globally in the 1990s, intense efforts were applied to use the existing global SDH network, as the wide area networking technology (WAN), to carry the packet protocols that underpin the physical layers of the Internet. Packet over SONET (POS) was developed, defined in RFC 2615 [6] initially, as a way of transmitting packet-based data protocols using point to point protocol (PPP) on each point to point link in an SDH/SONET network.

POS includes the option to apply scrambling to the transmission thereby adding an extra layer of security.

2.3 From cloud to edge to fog computing

The global adoption of the Internet enabled cloud computing paradigm. Large data centres, using virtualisation technology, can offer end users scalable compute resource on a pay per use basis. This approach is well suited to traditional enterprise computing freeing businesses from capital expenditure on computing systems transferring the cost to operational expenditure and off-loading risk to cloud service providers.

Newer applications, such as the Internet of Things (IoT), involve data collection at end user devices, equipment that is often mobile and therefore linked by wireless to edge nodes. The complete system is geographically diverse, with Smart Cities being one of the best illustrations of this being. The opportunity to redistribute computation across the hierarchy from user device, through edge, and back to the data centre when needed is now called fog computing. A hypothetical IoT service with a target end-to-end latency of 200 ms can easily expect, for a roundtrip to the cloud, to spend half of its budget in the network. This leaves a very tight time budget for execution of the actual processing to at the data centre. Fog has the potential to eliminate most, if not all, of the communication latency and, therefore, can permit the option of running the edge systems at lower frequency and voltage; for example, operating at 50% of the peak frequency with 30% less voltage translates to running with 50% less energy and 75% less power. Edge servers can also benefit from virtualisation, running multiple virtual machines to separate functionality. Furthermore research suggests that compute accelerators, in particular GPUs, may be enabled at the edge though virtualisation [7].

Figure 2 shows an analysis of the operation of an edge server, operating in extended margins, presented by the Horizion 2020 project Uniserver (http://www.uniserver2020.eu). UniServer created a cross layer approach from the hardware levels up to the system software layers. The following system enhancements were identified:

i. **at the circuit, micro-architecture and architecture layer** by automatically revealing the possible operating points (i.e. voltage, frequency) of each hardware component no worse than the worst-case operating points used, thus helping to boost performance or energy efficiency at levels closer to the Pareto front maximising the returns from technology scaling;

ii. **at the firmware layer** with low-level handlers by monitoring and controlling the operating status of the underlying hardware components and updating a 'HealthLog', as well as performing periodical benchmarking of the hardware and reporting the findings in a 'StressLog'. The logs with the collected information are communicated to the software stack (hypervisor) in a generic way, allowing easy adoption and exploitation of the observed margins;

iii. **at the software layer** by enabling an easy programmability, ensuring high dependability and full utilisation of the margins observed in the underlying hardware. State-of-the-art software packages for virtualisation (i.e. KVM) and resource management (i.e. OpenStack) will be ported on the micro-server further strengthening its advantages with minimum intrusion and easy adoption.

DOI: http://dx.doi.org/10.5772/intechopen.92788

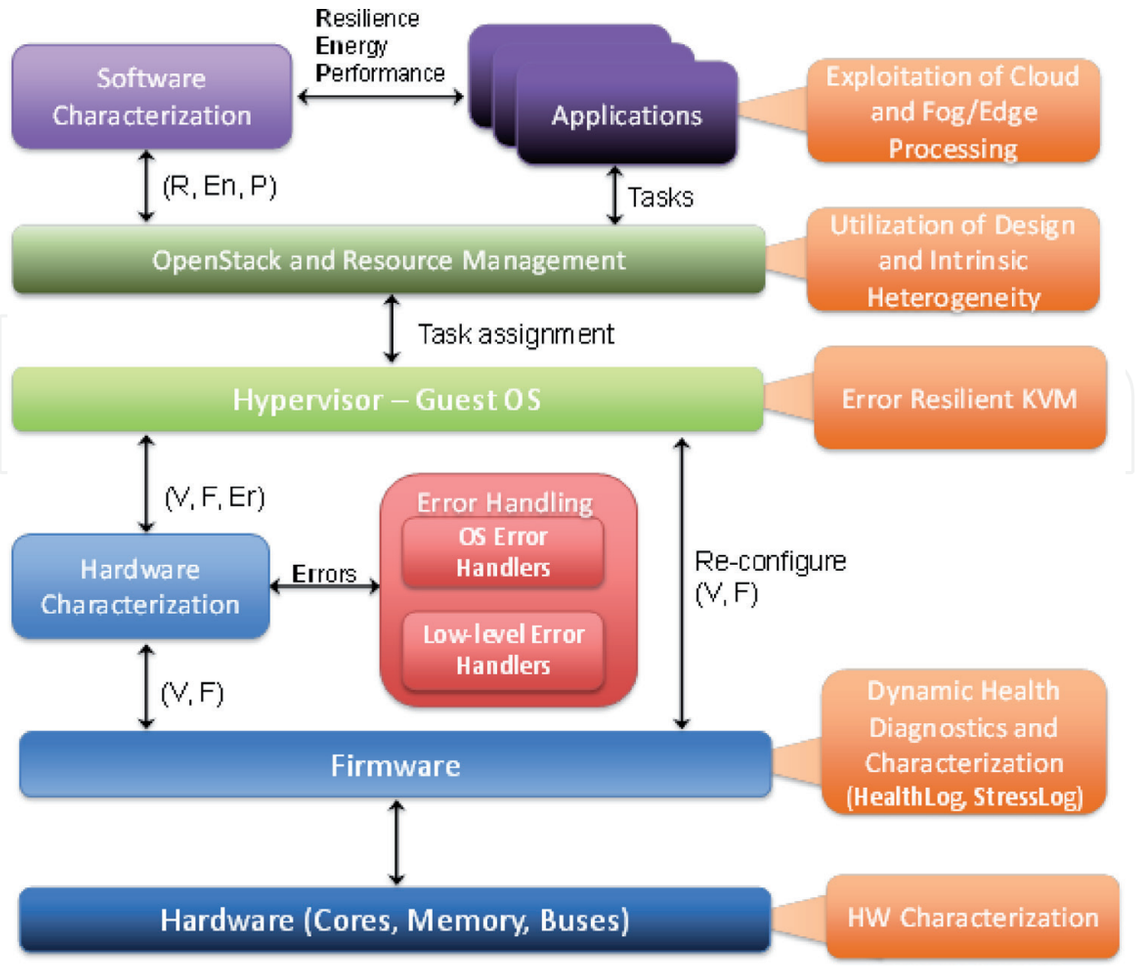

Figure 2.
Perspective from the UniServer project on the enhancement of the edge server.

In this chapter, we focus on the security challenges at the edge components and as we have outlined the WANs that link these edge nodes back to the data centres.

3. Cyber security at the edge

The edge computing paradigm moves significant amounts of computation from the data centre closer to the source of the data, reducing but not eliminating the need for packet communications. It follows that there are larger number of smaller servers deployed at the edge and therefore energy efficiency of the server operation becomes a significant factor. Edge servers have fewer CPUs and less DRAM and limited power budgets when compared to rack mounted servers in data centres. One driver for this is often the fact that physical form factor of the edge server is significantly limited compared to rack space in the data centre.

Manufacturers of server components define operational limits for parameters such as voltage, frequency and current. Routine adherence to these limits in the production of commercial servers reflects, in part, the need to account for the expected performance degradation of transistors and potential functionality failures due to the increased transistor variability in nanometre technologies. In general, the values adopted are quite pessimistic. DRAM manufacturers, in an effort to limit the potential faults, adopt a high operating supply voltage and refresh rate according to the assumed rare worst-case conditions [7]. This leads to the observation that DRAM alone can account for up to 40% power usage. Researchers have however investigated the operation of these electrical components in regions

of voltage and current beyond the conservative limits [8] and report 8.6% system energy savings on average for non-virtualised and 8.4% for virtualised workloads while ensuring the seamless server operation even under extreme temperatures.

Relaxation of voltage, timing and refresh-rate limitations may put at risk the correct functionality of the CPUs and DRAMs due to the potential failures that may occur at lower voltages and dynamically changing operating/environmental conditions (e.g. temperature). Such timing and memory failures may disrupt the operation of the server and/or directly impact the expected Quality of Service (QoS), which can be quantified in terms of throughput and quality-of-results (e.g. in terms of Bit-Error-Rate). As a consequence, such failures will affect service level agreements (SLA) in terms of availability, latency, accuracy and throughput as agreed at the higher level between the service user and the service provider. A further consequence of operating in these extended margins is that new security vulnerabilities may arise in addition to the cyber threats that already exist.

In contrast to a centralised cloud data centre, edge deployments will be constituted from many small clusters or individual installations, where elevated levels of physical security are not economically viable. Physical security of the micro-server may consist primarily of a light-weight enclosure and, from a security perspective, it should be assumed that a determined attacker will be able to gain full access to the system. This creates a larger threat surface, which now incorporates physical attacks, posing threats to the micro-server and the wider network it connects to. Deployments at the edge should be made under the assumption that networks are operating over untrustworthy links, with the use of encrypted tunnelling through VPNs, malware detection, firewalls, intrusion detection/prevention systems and DNSSEC all considerations for an endpoint security policy.

Threats posed by attackers gaining physical access to a system require consideration from both hardware and software security disciplines. Applications developers should employ secure coding practises, particularly when operating on any sensitive information. Care should also be taken to minimise, or, if possible, to avoid the storage of secret information in physical memory. The use of software, or ideally hardware-based, hard disk encryption technologies can offer protections, even when the disk is removed from a system.

Side-channel attacks can potentially be used to reveal sensitive information. In the UniServer system, sensitive extended margin information could be targeted to create denial of service attacks or cause system instability. The variation of voltage and frequency margins, core features of the UniServer solution, may also influence the relative amount of side-channel leakages. Side-channel resilient countermeasures, employing masking and hiding strategies, should be employed to help counteract such threats.

The differing deployment architectures of full stack and bare metal are considered. In the full stack deployment, representing a micro-server data centre, the UniServer software is running under the host OS, abstracted from other guest applications under separate virtual machines. However, in the bare metal deployment, the UniServer software runs along-side other system applications. It is in this deployment architecture where the UniServer system is most exposed to interference by other applications. The UniServer log files are identified as high value assets that need to be protected from tampering, since it could potentially lead to system instability or denial of service attacks. It is therefore a recommendation that the log and policy files are stored in an encrypted format, to avoid reading and manipulation by others. Additionally, consideration should be given as to whether the files should be digitally signed, to provide assurance that they come from a trusted source. These recommendations would naturally have overheads in terms of real-time operation, so their implementation would need to be considered carefully in

terms of system performance. The use of encryption, and possibly digital signing, will likely be candidate to form a security solution.

3.1 General attack vectors

In this section we consider the threats posed to both traditional networked server infrastructure and to the class of physical attacks, discussing the threats and countermeasures used to mitigate against them.

The primary aims of information security are to ensure the confidentiality, integrity and availability of a system [9]. There is generally no single solution to a security problem, since threats and vulnerabilities originate from many sources; rather the aim is to provide a series-layered security response, delivering defence in depth. An overall security response should be considered in the wider sense, consisting of measures that span the range of administrative, logical/technical and physical solutions.

3.1.1 Security of the operating system

The operating system (OS) is the fundamental software layer upon which the rest of the system software is built. In the common four-ring model, shown in **Figure 3**, the operating system is separated into two distinct regions of Kernel space, incorporating kernel memory, components and drivers from rings 0 to 2, and user space in ring 3, where end user applications may be run.

For most commercial operating systems, control of user access is organised under discretionary access control (DAC), providing privileges at the individual user account level. However, unlike a system under mandatory access control (MAC), where applications run in isolated memory with strong separation, typical OS's are running in a multi-tasking environment where resources are shared and are potentially accessible between applications [10]. Security is, therefore, ultimately left up to the system administrator to ensure that appropriate measures are in place and that the system is configured appropriately. Some general recommendations for operating system security, which apply to both cloud and edge deployments, are summarised below [11].

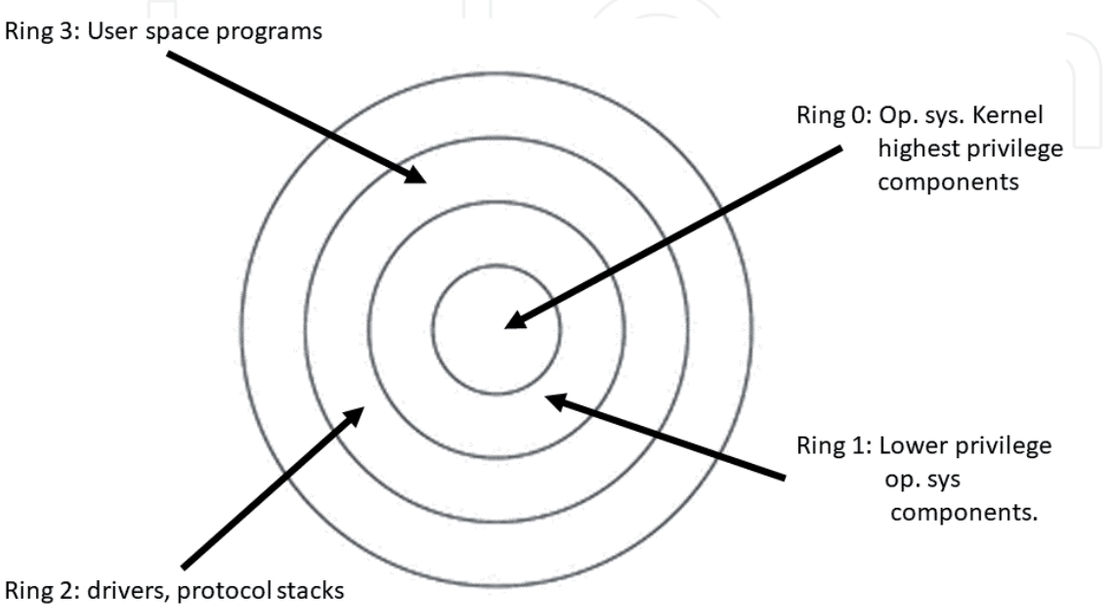

Figure 3.
Layers of protection in the operating system.

3.1.2 System integrity

- Build production systems from a known and repeatable process to ensure system integrity.
- Check systems periodically against snapshots of the original system.
- Use available third-party auditing software to check system integrity.
- Backup system resources on a regular basis.

3.1.3 User accounts

- Limit the number of user accounts.
- Ensure that only a few trusted users have administrative access.
- Assign the minimum required access permissions for the account that runs an application.

3.1.4 Password policies

- Require the use of secure passwords, that is, passwords of sufficient length, using a mix of letters, numbers and symbols. Do not re-use passwords and avoid the use of any personal information or dictionary words.
- Use automated tools to try and crack any weak passwords and require their update by users.
- On a UNIX operating system, activate the shadow password file.
- Use two-factor authentication.

3.1.5 File system

- Deny access by default.
- Provide minimal access rights where necessary, for example, read only.

3.1.6 Network services

- Provide the minimum number of required services.
- Reduce the level of access permissions for network services users.
- Ensure that user accounts that have access to the Web server do not have access to shell functions.
- For UNIX/Linux, ensure that unused services do not exist in the rc files, rc0-rc6, in the /etc. directory.
- Ensure that unused services are not running, and that they do not start automatically on MS Windows.

- Reduce the number of trusted ports specified in the /etc./services file.
- Protect your system against NetBIOS threats associated with ports 137, 138 and 139.
- Use wrapper services, such as iptables.
- Avoid using services that have a GUI, since such services introduce many known vulnerabilities.

3.1.7 System patches

- Run the latest, vendor-recommended patches for the operating system.
- Schedule regular maintenance of security patches.

3.1.8 Operating system minimisation

- Remove non-essential applications to reduce possible system vulnerabilities.
- Restrict local services to those required for operation.
- Implement protection for buffer overflow.

3.1.9 Logging and monitoring

- Log security-related events, including successful and failed logons, logoffs and changes to user permissions.
- Monitor system log files.
- Use a time server to correlate time for forensics.
- Secure the system log files by restricting access permissions to them.
- Secure the logging configuration file.
- Consider the use of a remote server for storage of logging information.
- Enable logging of access requests on web servers.

3.1.10 Hyperjacking

Hypervisor technology enables the deployment of numerous virtual machines (VMs) on the one system, indeed it is a key concept in shared cloud infrastructure. However, the deployment of multiple systems adds complexity and consequently the possibility for new exploits. The term virtualisation escape, or VMEscape, refers to the process by which an attacker can escape the confines of the virtual environment and is then able to exploit the host OS. Virtualised systems should therefore still be deployed under the supervision of firewalls, while guests with differing security levels, such as DMZ and internal, should not be combined on the same host.

It has been reported that malware rootkits have also been developed that act as hypervisors, installing themselves below operating systems, in a process referred to as hyperjacking. Since this software operates ostensibly outside the scope of the operating system, it can evade malware scans and also spy on the system, gathering information such as logging of passwords. In 2009, researchers from Microsoft and North Carolina State University revealed Hooksafe [12], a hypervisor class anti-rootkit, aiming to demonstrate the provision of generic protection against kernel-mode rootkits.

3.1.11 Network attacks

Access via network ports forms the basis of most remote attacks on cloud-based infrastructure. The ports of machines around the world are continually being probed to see if any ports have been left open or unsecured. It is therefore a basic preventative measure to close any unused ports and restrict access and secure those essential ports that are required to remain open. Improperly implemented TCP/IP stacks are vulnerable to various attacks such as buffer overflows, SYN flood attacks, denial of service attacks such as Smurf, ping and Fraggle and fragment attacks such as Teardrop to name but a few. These attacks can be largely mitigated by applying the appropriate configuration to disable services and apply the relevant patches.

Under the assumption that edge-deployed servers are more exposed, there are numerous means by which the traditional networking security elements of firewalls, proxies, virus scanners can be circumvented, creating a means by which other nodes of the network may be exposed. In 2014, the Gameover Zeus (GOZ) botnet was responsible for the global distribution of the CryptoLocker ransomware, which encrypted the victim's hard drive and required payment to receive the decryption key.

Since network connections could be exposed, the communications channel of an edge device should be considered untrustworthy, since attacks such as eavesdropping on network traffic, man-in-the-middle, modification or replay attacks are all possible. It is recommended that an encrypted VPN tunnel should be used between the edge server and other elements of the network to mitigate against such attacks.

DNS hijacking exploits the vulnerability in the way local or caching DNS servers obtain information from root servers regarding the identity of the authoritative servers for a domain. It is possible for an attacker to send falsified replies, and thus control the domain resolution, forwarding the user to the attacker's server [13]. The most effective countermeasure against DNS hijacking is to upgrade DNS to Domain Name System Security Extensions (DNSSEC).

When considering the above attacks, it is evident that edge deployments should incorporate their own endpoint security, consisting of elements such as inbound/outbound firewalls, malware scanning and intrusion detection/prevention systems as necessary security countermeasures.

3.2 Physical attacks and countermeasures for edge deployments

We now turn our attention to the situation in which a determined attacker has been able to bypass the limited protections of an enclosure and has gained direct physical access to the system, providing an enhanced ability to tamper with the system. There are many such physical attacks referenced in the literature; here we aim to give an overview of attacks, providing examples for the most relevant and practical attacks, along with examples of suggested countermeasures to those attacks.

3.2.1 Memory attacks

High-performance, processor-based, systems will generally include the following types of memory: L1/L2/L3 cache, DRAM, Flash Firmware and Hard-Disk Drives. Each of these is a potential threat vector for an attacker.

3.2.2 Timing attacks

Timing attacks exploit the differences in time required to perform specific operations. For example, the time required to calculate division and multiplication instructions, or the time necessary to fetch data when a cache hit, or cache miss, is experienced. Similarly, the difference in timings when conditional branching is used, or when optimisations are used by a programmer to skip unnecessary operations, may improve application performance but at the same time can reveal sensitive information about underlying code and values being processed. A classic example was shown by Kocher in [14] where the timings for modular multiply operations in exponentiation operations, and modulo reductions of the Chinese Remainder Theorem (CRT) optimisation in RSA, could lead to the discovery of the entire encryption key on a PC.

An example of a remote network-based attack is that of Bernstein in [15], demonstrating a timing attack on OpenSSL AES, on a UNIX x86 server. The server was profiled using a known key to determine the timing characteristics for the input plaintext values. During the attack, plaintexts were sent to the server, with their timing profiles compared to the profiled reference. The information leakage was reported to be due to the non-constant timing of table lookups.

Cache-timing attacks were first proposed by Page in [16] and demonstrated by Tsunoo et al. in [17], where DES was broken with a > 90% success rate. In [18], Tromer et al. showed that the full AES key could be extracted using DM-CRYPT disk encryption on Linux with only 800 accesses to an encrypted file. The attack took 65 ms of measurement time and 3 seconds to analyse. The OpenSSL library was also attacked in as little as 13 ms, with 300 encryptions.

Countermeasures to timing attacks generally aim to perform operations in constant time. However, this is not a straight-forward task since compilers can often provide optimisations that affect timing behaviour. In addition, cache hits and variances in instruction timings are generally outside the control of the software designer. A clock-skipping countermeasure was initially proposed by Kocher in [19], which inserted random delays to try and break up characteristic timing patterns, but this was later shown to be equivalent to adding noise to the power waveforms and could be overcome by analysis with a larger number of traces.

In [18], Tromer et al. considered various countermeasures against cache attacks. They suggested:

1. Avoid the use of memory accesses by replacing lookups with equivalent logical operations. This is a possibility for algorithms such as AES. However, there will be a performance trade-off.

2. Use of a bit-slicing approach.

3. Use of a cache no-fill mode, where memory is accessed from the cache during a hit and serviced from memory when there is a cache miss.

4. Dynamic table storage, where the contents of the table lookup are cycled around in memory during encryption operations to de-correlate it.

Guidance for coding standards for cryptographic implementations in software can be found in [20]. For example, in the context of timing attacks, it is recommended:

1. Do not compare secret values on a byte-by-byte basis.

2. Avoid branching predicated on secret data.

3. Avoid the use of lookup tables indexed by secret data.

4. Avoid loops that are bounded by a secret value.

The software developer can also make use of libraries, written with security in mind, such as NaCl [21] and some processors also include custom instruction sets dedicated to cryptography, such as the Intel AES-NI instructions referenced in [22] and the ARM cryptography extensions discussed in ARMv8 [23].

3.2.3 DRAM attacks

Buffer Overflow is a well-known attack that can enable execution of malicious code. Strategies to counteract this attack include the use of improved input validation and bounds checking at the programmer level, or at the system level through approaches such as the randomisation of memory layout or the structuring of buffer memory to incorporate memory spaces, sometimes termed 'canaries', that actively monitor to detect when unauthorised overflows occur.

The purposeful use of errors, exceptions and crashes can also be used to initiate memory dumping, where the entire contents of system memory are exported to enable readout of sensitive values stored in memory. It is recommended that sensitive values should not be stored in memory in the clear, rather they should be stored in encrypted form, or represented as hashed values and compared against re-computed hashes when required.

With direct physical access to a system, such as with an exposed and isolated edge server, an attacker can potentially remove DIMM memory modules from the system board. As described in [24], the use of cooling sprays can enable a DIMM memory module to retain memory, without error, for several minutes. The memory can then be plugged into another system and sensitive information read out. This attack has been shown to make on-the-fly software-based disk encryption systems such as BitLocker, FileVault and TrueCrypt vulnerable. One countermeasure approach would be to avoid the use of pre-computed tables of information for encryption routines, which would typically be stored in DRAM, although this will have performance penalties associated with it since the values will need to be computed on-demand each time.

RowHammer is a more recent memory attack that exploits a weakness identified in commodity DRAMs, where repeated row activations can cause bits to flip in adjacent rows. A recent attack [25] used generic memory functions such as libc, memset and memcpy for attack primitives, making the attack more accessible.

3.2.4 Re-flashing attacks

Re-flash attacks target the replacement of existing system firmware with that of compromised firmware images. This can enable attackers to circumvent protections that would otherwise be in place. Due to the low-level nature of firmware access and

control, such attacks can have a powerful effect on a system. Countermeasures may include incorporating password access for flashing operations.

3.2.5 Hard disk drive attacks

Hard drives will generally host the main operating system and the application software that loads on the system, but also potentially swap page information, which may hold sensitive information temporarily stored from primary DRAM memory. Hard disks, and particularly hot-swappable server-class drives, can be removed from a system at ease, and then connected to another system by plugging in a power and data cable. The disks can then be mounted as secondary drives to be copied, interrogated, or have additional malware or software installed. All of this is outside the scope of any protection from intrusion prevention systems of the original host. It is therefore advisable to consider the deployment of disk encryption technologies, such as software-based encryption, or preferably, hardware-based total disk encryption.

3.2.6 Side-channel attacks

We now consider a class of physical attacks termed as side-channel attacks. These attacks target the leakage of information from a system and are primarily concerned with the discovery of the secret information such as encryption keys that underpins modern cryptographic processing. The same approach can be targeted at modelled leakages of any other high-value information that is processed in a system.

3.2.7 Power analysis attacks

Power analysis is a powerful technique used to obtain side-channel information from a system. The power analysis attack can be categorised into two types: simple power analysis and differential power analysis.

In simple power analysis, the individual power waveform acquisitions are observed to see if information can be gleaned from them. In the attack of [14], it was observed that a single power consumption trace could reveal the entire encryption key by simply interpreting the pattern of the power trace, since modular multiply operations in exponentiation operations took varying times depending on whether the portion of the encryption key was a '1' or a '0'.

In differential power analysis (DPA), a series of power consumption measurements are recorded while the device is processing the target information, typically a secret encryption key, and is then compared against a set of hypothesised power models to determine a portion of the key. The analysis is repeated for the remainder of the key portions until the complete encryption key is recovered, enabling the attacker to decrypt any data, previously encrypted with the same key. Power consumption is typically modelled by estimating the number of '1's in a register via a Hamming weight or Hamming distance power model. Several differing methods of statistically comparing the modelled versus measured power consumptions are commonly used, such as difference of means, distance of means and Pearson's correlation coefficient [26].

Power analysis attacks are device specific and it can take from several hundred, to several million, traces to break an implementation with a DPA attack; this is dependent on the signal/noise (S/N) ratio and whether any countermeasures are present. Research has been carried out on a multitude of low-frequency embedded

systems, where the approach has proved very successful. The attack works best when a clean voltage signal is available, preferably from the processor core of the device, where S/N is typically optimal; however, attacks can also be mounted by measuring the global power supply of a device through the voltage drop across a small resistor placed between supply and ground. There are fewer published works that address attacks on full-scale server boards, due to the additional complexities introduced by higher frequencies of operation, lack of access to processor core voltage and the additional noise generated by numerous system hardware elements.

Countermeasures against power analysis attacks aim to break the statistical link between the power consumption and the sensitive intermediate data values. For defence against simple power analysis, countermeasures primarily focus on disturbing the power waveform to disrupt the observable pattern, and so remove the discernible information. This can be accomplished by increasing background noise signals, introducing random insertions or delays, or by removing conditional branching and employing constant time algorithms.

Protecting a device from DPA is a much more challenging task, since this attack uses advanced statistical techniques to extract information from many traces. Countermeasures can be classed into two broad categories, namely whether they aim to hide or mask the data [27]. Hiding approaches do not attempt to change the intermediate values that are processed, rather they try to change the power waveform by applying some randomisation or by making it constant. Randomising approaches were mentioned above for simple power analysis measures and could also include approaches such as shuffling or skipping of instruction clocks. To make the power consumption constant, approaches have been proposed such as the use of dual-rail pre-charge (DRP) logic styles, which uses two wires that are complementary for each signal. Other logic styles, such as Sense Amplifier Balanced Logic (SABL), were proposed by Tiri et al. in [28] to provide resistance against DPA. However, these approaches require custom ASIC design with careful layout considerations and have still been shown to be vulnerable to DPA attacks.

The masking countermeasure aims to change the sensitive intermediate values by applying and then removing a temporary mask operation, a simple example being an XOR with a random value. This then breaks the link between what the power model expects and what is processed inside the device. The disadvantage of masking is that it can require the application and removal of multiple masks, for example switching between Boolean and multiplicative masks. This has a processing overhead and can be complicated to design and implement.

3.2.7.1 Electro-magnetic attacks

Electro-magnetic (EM) attacks [29] are a variation of power analysis attacks. They differ in the method of acquisition, which uses an electric or magnetic field probe to convert EM radiation into voltage signals that are proportional to the power consumption. The probing is generally classed as being either near-field or far-field. Near-field probing is considered to be the short-range distance that is typically less than one-wavelength from the source. At this distance, the field strength is proportional to $1/r^3$ in strength, therefore placing the probe as close as possible to the source will maximise signal strength. A more invasive attack can be to remove the chip package surface and enable a fine point-tip probe to be placed very close to the exposed integrated circuit (IC); however, this requires more time and generally a laboratory environment. A less invasive approach is to rest a simple loop antenna or EM probe tip against the surface of the IC, and to use active amplification to improve signal strength for appropriate quantisation scaling during acquisition.

Far-field EM attacks work at multiple wavelength distances and typically use a high-frequency directional antenna to receive signals. The waveforms being captured here have escaped the confines of the near field and are propagating over free space [30]. This form of attack would likely only be possible for exposed, non-shielded enclosures.

An EM acquisition can have advantages over that of traditional power analysis attacks. Firstly, it can have a lower invasiveness. In comparison to a power analysis attack, where a resistor may need to be soldered into place, the EM probe can often be placed in close proximity, without any evidence of tampering. Secondly, there is the possibility to improve the localisation of the probe, that is, to position it directly around the circuitry processing the sensitive information. This can help reduce the contributions of the EM fields generated from other elements of the overall power consumption. This can improve the S/N ratio, making it easier to visually identify leakages on an oscilloscope and improves the statistical analysis.

The countermeasures of hiding and masking, discussed above, also provide general protection against both EM analysis. However, for non-invasive attacks with an EM probe, physical shielding countermeasures can offer some further resistance. In [31], Yamaguchi *et al.* applied thin magnetic film to shield an integrated circuit device and reported a 6 dB reduction in magnetic field signal strength.

3.2.7.2 Profiling attacks

Profiling, or template, attacks [32, 33] use a reference device to build a characteristic power model of a device for various test inputs. The power model can then be compared against the power consumption measurements of an identical device to reveal what data have been processed internally. The template attack can potentially reveal the secret key with as little as one power trace; however, to obtain a power model with high fidelity may require the acquisition and pre-processing of many power traces, which may be a time-consuming exercise. Masking or the randomisation of execution order could be used as potential countermeasures.

3.2.7.3 Machine learning attacks

Machine learning is an emerging approach to side-channel attacks. Although numerous algorithms can potentially be used, the specific feature selection and data set size have the major influence on the success of the attack. Examples of approaches are supervised learning, support vector machines, random forest, neural networks and unsupervised learning. To date, most research has focussed on support vector machines [34–36], random forest [37] and neural networks [38]. Countermeasures to machine learning may include higher order masking approaches and the use of poisoned data.

3.2.8 Fault attacks

Fault attacks aim to induce erroneous behaviour in devices by inserting transient faults that propagate through the system and reveal secret information as a consequence. The transient nature of the targeted faults means that an attack can be attempted repeatedly, and the attack developed. This approach means that no permanent damage is caused to the device and therefore it is less likely that any evidence remains that an attack has taken place. In [39, 40] it was shown that faults could be induced in smart card devices by varying the system supply voltage, clock speed and ambient temperatures. Since these same characteristics are altered in

UniServer, it is an area of active investigation in the project, for example in terms of generation of memory and system errors.

Fault attacks in the literature have targeted both public and private key algorithms. Consider, for example, the attack on the Chinese reminder theorem (CRT) computation in RSA of [41] and the targeting of AES in [42, 43]. The attack of [43] demonstrating that inducing two faults in the 9th round of AES key scheduling was enough to break the encryption system. For active attacks, the most common approach is that of fault injections, as detailed in [44].

Countermeasures to fault injections include established techniques in communications engineering, such as the use of error codes and parity checking, along with newer proposals such as concurrent error detection (CED) which suppress the operation of a circuit when error states are detected. The aim of CED is to halt the propagation of the error to the output, where the attacker can analyse whether the fault attack was successful or not. Additional proposals for countermeasures include the duplication of circuitry, or repeated computation, to provide comparators. With duplication of hardware the cost penalty is high, while with repeated computation the execution time may increase significantly. Other, more efficient, schemes have been proposed, such as suggested in [45], requiring only one parity bit for each internal state of AES. The approach detects all odd errors, and in many cases the even errors, and may be a promising approach for implementation in both the hardware and software contexts.

Proposals have also been made to secure the CRT computations of RSA. In [46], the arguments of the CRT were calculated using an approach termed efficient redundancy, where values are verified before their use in the RSA algorithm. This approach, which adds little timing overhead, improves upon previous approaches requiring full redundancy.

3.2.9 Out-of-order execution attacks

At the time of writing, two new side-channel attacks [47], targeting the out-of-order execution of instructions on processors, were announced. Meltdown exploits the scenario where a speculatively executed instruction, although aborted, permits the bypassing of memory protections and thus the ability to read Kernel memory from user space. The attack is deemed to affect Intel processors primarily. In the short-term, a patch based on the KAISER countermeasure of [48] has been released. This countermeasure re-maps the memory space in software. A more permanent solution will likely require architectural changes at the hardware level to control the order of permission checks for access to memory and improvements to memory segmentation.

The Spectre attack exploits the use of speculative branch predictions to store information to cache memory that can then be targeted with side-channel techniques such as flush+reload or evict+reload cache attacks. The attack is considered more universal than Meltdown, and has already been shown to affect Intel, AMD and ARM processors. Countermeasures against Spectre also appear difficult to implement. Simply disabling speculative execution would result in an unacceptable performance loss, while inserting temporary blocking instructions is also seen as a challenging task. Potential updates to processor microcode may be possible as a form of software patch, but likely to impact performance considerably.

4. Conclusions

The move from cloud deployment model to the edge has implications for security. In contrast to a cloud data centre, housed within a large building complex with a significant level of security, the edge deployment will constitute a large number of

small clusters or individual installations, where high levels of physical security are not economically viable. In many situations, physical security of the micro-server may consist primarily of a light-weight enclosure, designed to protect the system from environmental factors and vandalism or casual tampering efforts. For the determined attacker, this may not prove to be an effective barrier and it should be assumed that a realistic worst-case scenario is that an attacker will be able to gain full access to the system. This then creates a larger threat surface, now incorporating physical attacks that can be used to compromise the individual micro-server, and potentially, the wider network.

Deployment at the edge still requires the implementation of traditional server and network security practises, such as those outlined in this chapter. In addition, deployment at the edge should assume that networks are operating over untrustworthy links and therefore the use of encrypted tunnelling through VPNs, and the use of malware detection, firewalls, intrusion detection/prevention systems and DNSSEC should all be considered as forming the basis of an endpoint security policy.

The use of virtualisation is a core element of cloud and resource sharing technologies; however, it also opens the possibility for attacks exploiting VMEscape. Accommodating guests with differing security levels, such as DMZ and internal, on the same host, should be avoided.

Edge deployment should consider the further threats posed from an attacker gaining partial, or full, physical access to a system. This requires input not only from a hardware security standpoint, but also from software perspectives. Applications developers should employ secure coding practises, particularly when operating on any sensitive information, as highlighted in the discussions of memory attacks in Section 2.2.1. Care should also be taken to minimise or, if possible, to avoid the storage of secret information in physical memory, since attacks such as buffer overflows and removal of frozen DRAM modules have been shown as effective means to extract information stored in the clear. User passwords, for example, should be stored as hashed values and passwords requested on demand for comparison or verification. The use of software, or ideally hardware-based, hard disk encryption technologies can offer protections, even when the disk is removed from a system.

Side-channel attacks can potentially be used to reveal sensitive information such as the extended margin information stored in the log and policy files. Indeed, the variation of voltage and frequency margins, core features of the UniServer solution, may also influence the relative amount of side-channel leakages. A countermeasure to this threat is the deployment of encryption using side-channel resilient countermeasures, such as masking, to break the statistical link between power measurements and hypothetical power models.

In the full stack deployment, representing a micro-server data centre, the UniServer software is running under the host OS, abstracted from guest applications operating under VMs. However, in the bare metal deployment, the UniServer software runs along-side other system applications. It is in this deployment architecture where the UniServer system is most exposed to interference by other applications, which can potentially view and access each other's files or resources. The UniServer log files were identified as high value assets that need to be protected from tampering, since it could potentially lead to system instability or denial of service attacks. It is therefore a recommendation that the log and policy files are stored in an encrypted format, to avoid reading and manipulation by others. Additionally, consideration should be given as to whether the files should be digitally signed, to provide assurance that they come from a trusted source. These recommendations would naturally have overheads in terms of real-time operation, so their implementation would need to be considered carefully in terms of system performance. The use of encryption, and possibly digital signing, will likely be candidate to form a security solution.

Author details

Charles J. Gillan and George Karakonstantis*
The School of Electrical and Electronic Engineering and Computer Science (EEECS), Queen's University Belfast, Northern Ireland

*Address all correspondence to: g.karakonstantis@qub.ac.uk

References

[1] Stojmenovic I, Wen S. The fog computing paradigm: Scenarios and security issues. In: 2014 Federated Conference on Computer Science and Information Systems (FedCSIS). Warsaw, Poland: IEEE; 2014. pp. 1-8

[2] Alrawais A, Alhothaily A, Hu C, Cheng X. Fog computing for the internet of things: Security and privacy issues. IEEE Internet Computing. 2017;**21**(2):34-42

[3] Mukherjee M, Matam R, Shu L, Maglaras L, Ferrag MA, Choudhury N, et al. Security and privacy in fog computing: Challenges. IEEE Access. 2017;**5**:19293-19304

[4] Cabric M. Corporate Security Management. Challenges, Risks, and Strategies. 1st edition. London: Butterworth-Heinemann; 2015:242. ISBN: 9780128029343

[5] S Institute. Operating System Security and Secure Operating Systems [Online]. Available from: https://www.giac.org/paper/gsec/2776/operating-system-security-secure-operating-systems/104723 [Accessed: December 2017]

[6] IBM. Business Intelligence Architecture and Deployment Guide—Securing the Operating System [Online]. Available from: https://www.ibm.com/support/knowledgecenter/en/SSEP7J_10.2.1/com.ibm.swg.ba.cognos.crn_arch.10.2.1.doc/c_securing_the_operating_system.html [Accessed: December 2017]

[7] Wang Z, Jiang X, Cui W, Ning P. Countering kernel rootkits with lightweight hook protection. In: Proceedings of the 16th ACM Conference on Computer and Communications Security. 2009. pp. 545-554

[8] Friedl S. An Illustrated Guide to the Kaminsky DNS Vulnerability [Online]. Available from: http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html [Accessed: December 2017]

[9] Kocher P. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Advances in Cryptology (CRYPTO '96). Lecture Notes in Computer Science. Vol. 1109. Heidelberg, Berlin: Springer; 1996. pp. 104-113

[10] Aly H, ElGayyar M. Attacking AES using Bernstein's attack on modern processors. In: Youssef A, Nitaj A, Hassanien AE, editors. Progress in Cryptology – AFRICACRYPT 2013. Vol. 7918. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2013. pp. 127-139

[11] Page D. Theoretical use of cache memory as a cryptanalytic side-channel. IACR Cryptology ePrint Archive. 2002:1-23

[12] Tsunoo Y, Saito T, Suzaki T, Shigeri M, Miyauci H. Cryptanalysis of DES implemented on computers with cache. Cryptographic Hardware and Embedded Systems-CHES. 2003;**2003**

[13] Tromer E, Osvik D, Shamir A. Efficient cache attacks on AES, and countermeasures. Journal of Cryptology. 2010;**23**(1):37-71

[14] Kocher P, Jaffe J, Jun B. Differential power analysis method and apparatus. U.S. Patent 7587044; 2009

[15] Cryptocoding.net. Cryptographic Coding Standards. Available from: https://cryptocoding.net/index.php/Cryptography_Coding_Standard; 2013

[16] NaCl: Networking and Cryptography library. Available from: https://nacl.cr.yp.to/

[17] Tsunoo Y, Saito T, Suzaki T, Shigeri M, Miyauchi H. Cryptanalysis

of DES implemented on computers with cache. In: Walter CD, Koç ÇK, Paar C, editors. Cryptographic Hardware and Embedded Systems - CHES 2003. Vol. 2779. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2003. pp. 62-76

[18] Tromer E, Osvik DA, Shamir A. Efficient cache attacks on AES, and countermeasures. Journal of Cryptology. 2010;**23**:37-71. DOI: 10.1007/s00145-009-9049-y

[19] Kocher P. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In: Advances in Cryptology - CRYPTO'96. Berlin: Springer; 1996. pp. 104-113

[20] Qiao R, Seaborn M. A new approach for rowhammer attacks. In: 2016 IEEE international symposium on Hardware oriented security and trust (HOST). McLean, Virginia, USA: IEEE; 2016

[21] Brier E, Clavier C, Olivier F. Correlation Power Analysis with a Leakage Model in Cryptographic Hardware and Embedded Systems - CHES. In: Joye M, editor. Berlin, Heidelerg: Springer; 2004:16-29. DOI: 10.1007/978-3-540-28632-5_2

[22] Mangard S, Oswald E, Popp T. Power Analysis Attacks: Revealing the Secrets of Smart Cards. New York, USA: Springer-Verlag US; 2007

[23] Tiri K, Akmal M, Verbauwhede I. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: Proceedings of the 28th European Solid-State Circuits Conference (ESSCIRC 2002). University of Bologna; 2002

[24] Quisquater JJ, Samyde D. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Smart Card Programming and Security (E-smart 2001). Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; Vol. 2140. 2001. pp. 200-210

[25] Agrawal D, Archambeault B, Rao JR, Roha P. The EM side-channel(s). In: Cryptographic Hardware and Embedded Systems (CHES). Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; Vol. 2523. 2002. pp. 29-45

[26] Yamaguchi M, Kobayashi S, Sugawa T, Toriduka H, Homma N, Satoh A, et al. Development of an on-chip micro shielded-loop probe to evaluate performance of magnetic film to protect a cryptographic LSI from electromagnetic analysis. In: International Symposium on Electromagnetic Compatibility (EMC). Lauderdale , Florida: IEEE; 2010. pp. 103-108

[27] Fahn P, Pearson P. IPA: A new class of power attacks. In: Cryptographic Hardware and Embedded Systems. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; Vol. 1717. 1999. pp. 173-186

[28] Tiri K, Verbauwhede I. Charge recycling sense amplifier based logic: Securing low power security ICs against differential power analysis. In: Proceedings of the 30th European Solid-State Circuits Conference, ESSCIRC. IEEE; Sept 2004. pp. 179-182

[29] Hospodar G, Gierlichs B, De Mulder E, Verbauwhede I, Vandewalle J. Machine learning in side-channel analysis: A first study. Journal of Cryptographic Engineering. 2011;**1**(4):293-302

[30] Heuser A, Zohner M. Intelligent machine homicide: Breaking cryptographic devices using support vector machines. In: Constructive Side-Channel Analysis and Secure Design—COSADE 2012. Vol. 7275. Series LNCS. Berlin, Heidelberg: Springer; 2012. pp. 249-264

[31] Lerman L, Bontempi G, Markowitch O. Side channel attack: An approach based on machine learning. In: Constructive Side-Channel Analysis and Secure Design—COSADE. 2011

[32] Markowitch O, Medeiros S, Bontempi G, Lerman L. A machine learning approach against a masked AES. Journal of Cryptographic Engineering. 2013;**5**(2):62-75. DOI: 10.1007/s13389-014-0089-3

[33] Gilmore R, Hanley N, O'Neill M. Neural network based attack on a masked implementation of AES. In: IEEE International Symposium on Hardware Orientated Security and Trust. IEEE; 2005

[34] Anderson R, Kuhn M. Tamper resistance—A cautionary note. In: Proceedings of Second USENIX Workshop on Electronic Commerce. Oakland, California: USENIX; 1996. pp. 1-11. Available from: https://www.usenix.org/legacy/publications/library/proceedings/ec96/index.html

[35] Anderson R, Kuhn M. Low cost attacks on tamper resistant devices. In: Proceedings of 5th Security Protocols Workshop. Lecture Notes in Computer Science. Berlin, Heidelbergs: Springer; Vol. 1361. 1997. pp. 125-136

[36] Boneh D, DeMillo R, Lipton R. On the importance of eliminating errors in cryptographic computations. Journal of Cryptology. 2001;**14**(2):101-119

[37] Piret G, Quisquater J-J. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In: Walter CD, Koç CK, Paar C editors. Proceedings of the 5th International Workshop, Cologne, Germany, September 8-10. Vol. 2779. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2003. pp. 77-88. ISBN: 978-3-540-45238-6

[38] Kim CH, Quisquater J-J. New differential fault analysis on AES key schedule: Two faults are enough. In: Grimaud G, Standaert FX, editors. Smart Card Research and Advanced Applications. Lecture Notes in Computer Science. Vol. 5189. Berlin, Heidelberg: Springer; 2008. pp. 48-60. DOI: 10.1007/978-3-540-85893-5_4

[39] Bar-Eli H, Choukri H, Naccache D, Tunstall M, Whelan C. The sorcerer's apprentice guide to fault attacks. Proceedings of the IEEE. 2006;**94**(2):370-382

[40] Bertoni G, Breveglieri L, Koren I, Maistri P, Piuri V. Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. IEEE Transactions on Computers. 2003;**52**(4):492-505

[41] Shamir A. Method and apparatus for protecting public key schemes from timing and fault attacks. U.S. Patent 5991415; 1999

[42] The Real Story of Stuxnet. IEEE Spectrum Online [Online]. 2013. Available from: https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet

[43] USBKiller V3. USBKill [Online]. Available from: https://usbkill.com/ [Accessed: December 2017]

[44] Lipp M, Schwarz M, Gruss D, Prescher T, Haas W, Mangard S, et al. Meltdown and Spectre. Bugs in Modern Computers Leak Passwords and Sensitive Data [Online]. 2018. Available from: https://meltdownattack.com/meltdown.pdf

[45] Gruss D, Lipp M, Schwarz M, Fellner R, Maurice C, Mangard S. KASLR is dead: Long live KASLR. In: International Symposium on Engineering Secure Software and Systems. Austria: University of Graz; 2017

[46] Kocher P, Genkin D, Gruss D, Haas W, Hamburg M, Lipp M. Meltdown and Spectre. Bugs in Modern Computers Leak Passwords and Sensitive Sata [Online]. 2018. Available from: https://spectreattack.com/spectre.pdf

[47] Watson RNM, Woodruff J, Roe M, Moore SW, Neumann PG. Capability Hardware Enhanced RISC Instructions (CHERI): Notes on the Meltdown and Spectre Attacks. University of Cambridge Technical Reports, UCAM-CL-TR-916, Feb 2018. Available from: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-916.pdf

[48] Gruss D, Lipp M, Schwarz M, Fellner R, Maurice C, Mangard S. KASLR is dead: Long live KASLR. In: Bodden E, Payer M, Athanasopoulos E, editors. Engineering Secure Software and Systems. ESSoS 2017. Lecture Notes in Computer Science. Vol. 10379. Cham: Springer; 2017. pp. 161-176