Noname manuscript No.
(will be inserted by the editor)

Side Channels as Building Blocks

Markus Kasper · Amir Moradi · Georg T. Becker · Oliver Mischke · Tim Güneysu · Christof Paar · Wayne Burleson

Received: date / Accepted: date

Abstract Since the introduction of the first side-channel analyses in academia about 15 years ago, several physical attacks have been presented that exploit side-channel leakages to break implementations of cryptographic algorithms. This article deals with the same physical property of electronic devices, but focuses on the art of tailoring it for constructive uses. More precisely, two scenarios, i.e., *hardware Trojans* and *IP watermarking*, are illustrated in which the designer of an electronic circuit can add functionality by considering side channels as part of the available design space. Both applications use the same concept, i.e., deliberately leaking a secret through a side channel while keeping the introduced side channel hidden from adversaries and attackers.

This article provides a broad overview of the existing works for both applications and should serve as a comprehensible introduction to the underlying field of research. This includes many subtle details that have not been discussed in literature yet, including existing shortcomings and possible improvements to the existing works. The solutions summarized in this article provide general guidelines for theorists and practitioners to use side channels constructively to achieve designs that are robust against detection and removal. Furthermore we present an entirely new design of a Trojan side-channel. This architecture demonstrates the potential of a Trojan side-channel that is neatly tailored to the targeted implementation. The new design removes all non-invasive starting points a third party could use to analyze or get access to the secret-channel.

M. Kasper, A. Moradi, O. Mischke, T. Güneysu and C. Paar
Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany
E-mail: {mkasper, moradi, mischke, gueneysu, cpaar}@crypto.rub.de

G.T. Becker, C. Paar and W. Burleson
Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA
E-mail: {becker, burleson}@ecs.umass.edu

Keywords Side-channel analysis · hardware Trojan · IP watermarking · AES · masking

1 Introduction

In the early days of research on cryptology the main focus was on developing cryptographic primitives which can provide a secure encryption mechanism. Today there are also primitives that provide not only confidentiality, but also other security services, such as integrity, authenticity or even non-repudiation. On the other hand, not only the primitives became more and more powerful over the years, but also the methods for analysis and attacks developed and the computational power available to perform the attacks improved. Nevertheless, there exists a set of proposed cryptographic routines today, which are able to withstand all currently known cryptanalytic attacks and are thus believed to be secure for all practical purposes. Considering that these algorithms have been tested by experts from all over the world for many years, it is very unlikely (but still not impossible) that a mathematical attack will be developed that severely decreases the security margin of these primitives. In these mentioned early days security engineers only needed to carefully choose well-established security primitives to gain security for their systems. In the late 1990s this situation changed again when side-channel attacks on embedded devices became popular. Side-channel attacks aim to extract a secret stored inside an embedded device. For example, this can be the secret key of a symmetric encryption scheme or challenge response protocol or a private key used in digital signatures. Since the first publications on side-channel analysis (SCA) attacks by Kocher et al. at CRYPTO 1996 and 1999 [19,20], SCA attacks have been extensively studied in scientific literature.

A good list of references to proposed attacks is provided by the CHES workshop series [1] and the DPA Book [23].

The detailed documentation of these methods and their low-cost nature allows even non-professionals to attack embedded devices within reasonable time. This makes side-channel analyses a very dangerous threat for many security related applications: The sensitive devices can be physically accessed by their owners and the low efforts of the attacks motivates to probe the implemented security. Smartcards for payment purposes, car keys and remote control entry systems are typical targets for side-channel attacks [13,25,27,29].

To prevent successful side-channel analyses many commercial products are implemented with side-channel countermeasures to increase the resistance of the device to physical attacks. Unfortunately, implementing countermeasures that provide a sufficient side-channel resistance to render recent attack methods infeasible is a very challenging task with many pitfalls. Many modern security applications thus rely on the commercial availability of trusted side-channel resistant implementations of cryptographic routines.

In this contribution we give an overview of applications, where methods from side-channel analysis are used to actively tailor physical side channels to provide some special functionality. This changes the overall view from the destructive to the constructive perspective, i.e., employing side channels as part of the available design space. For the scope of this article we will only focus on the passive and non-invasive side-channel analysis attacks, as defined in [23] and do not consider any active physical attacks such as differential fault attacks [8].

This work provides two contributions. The first one is a complete introductory overview of the existing literature on Trojan side-channels extended by many new insights, e.g., on shortcomings of the existing works and appropriate solutions. The other contribution are practical results from a new Trojan side-channel design adapted to a side-channel protected implementation of a cryptographic primitive.

The article is organized as follows. Section 2 provides a short overview of side-channel analysis attacks. These well-known methods from side-channel analysis provide the required background to develop and implement an artificial side-channel. Section 3 summarizes the fundamental idea and characteristics of Trojan side-channels as introduced in [22], while the following Section 4 explains practical implementation details. Sections 4 and 5 are provided in form of practical and illustrative case studies. Subsections 4.3 and 4.4 recapitulate the basic toolbox used to constructively tailor a side-channel and Subsection 4.5 presents an entirely new design of a side-channel backdoor. This implementation allows to lift many of the shortcomings of the previously published designs and is the first demonstration of a Trojan side-channel tailored to the popular masking countermeasure used for side-channel resistant implementations of cryptographic primitives. Finally Section 5 illustrates how the presented concepts can be adapted for watermarking applications followed by the conclusions in Section 6.

2 Side-Channel Analysis

Side-channel analysis attacks do not target an algorithm itself, but its implementation. Real-world implementations are not perfect black boxes that perform some calculation and output nothing else but the result of the computation. Instead an algorithm implemented in a device will always be accompanied by many physical properties. For example the implemented algorithm requires a finite time to be executed. Furthermore, the device will heat up during the computation, will draw currents at its power pins, and will produce electro-magnetic radiation. The idea of side-channel analysis attacks is to make use of these additional measurable outputs from the implementation (called side channels) to extract information on the secrets used by the implementation.

2.1 Classifying Side-Channel Attacks

If a measured physical quantity of an implementation depends on the secret used inside the device, this implementation is a good candidate to mount a side-channel analysis (SCA) attack on. An important class of SCA are power analysis attacks which can be divided into two main categories: differential power analysis (DPA) attacks [20] and simple power analysis (SPA) attacks [20]. In both attacks a physical quantity is measured while the algorithm is executed inside the device. In an SPA attack (ideally) a single trace of such a measurement is sufficient to deduce the secret. These attacks often even work by visual inspection, and the most popular representatives of SPA attacks are those on implementations of asymmetric primitives used in digital signatures such as RSA and elliptic curve schemes. If they are implemented using schoolbook square-and-multiply style algorithms to perform the exponentiation (or multiplication respectively), the ability to distinguish between square and multiply operations is sufficient to directly determine a secret key based on the sequence of operations performed by the device [24].

2.2 Differential Power Analysis Attacks

Differential power analysis attacks on the other hand are based on statistic analysis of larger sets of measurements. The idea behind DPAs is that the intermediate values of a cipher will have an impact on the side-channel leakage of a device and thus measuring the device while processing different input values will lead to different measurement values. More precisely in a DPA attack the attacker assumes

that there are classes of intermediate values that will lead to different measurement values. In order to perform the attack a large amount – typically from hundreds (microcontrollers) to millions (high-tech FPGAs) – of measurements are acquired using different input values, i.e., usually plaintexts or ciphertexts. The methods applied later on will use these measurements and attack each measurement point individually. Then the attacker selects an intermediate value of the algorithm, which depends on the parts of the secret (called subkey) and the known input values. In the next step the selected intermediate values are calculated for each measurement's input value and for each possible subkey candidate resulting in sets of hypothetical intermediate values for each subkey candidate. Finally, the measured side-channel leakage is "compared" to these sets of hypothetical values by means of a "distinguisher", to decide which key candidate fits best to the measured values. There are three very popular methods to perform this step of comparing the measurement values and the hypotheses:

Difference of Means - DPA The classical DPA [20] is based on the simple observation, that different intermediate values processed inside a device will lead to different leakage behavior. An attacker uses the predicted intermediate value to sort the measurements into two groups. In this step typically a single bit of the intermediate value is used to determine the groups. Then the average of the samples in each group is calculated giving this distinguisher its name "difference of means". To distinguish correct from false subkey candidates the difference between these average values is used. For a sane grouping of measurements, resulting from a correct subkey hypothesis, the average values will in fact differ. False key candidates will, on the other hand, lead to a random sorting, so that the averages of the two sets will be very close to each other.

CPA The correlation power analysis [12] uses the Pearson's correlation coefficient to distinguish the correct hypothesis amongst the others. Here the attacker first maps the intermediate values to hypothetical power consumption values by means of a model of the leakage behavior. Then, the correlation between hypothetical power consumption values and measured values is calculated. For the correct hypothesis and a suitable leakage model this correlation will be higher than the correlation for the false ones.

MIA The mutual information analysis [6, 14] uses information theory to distinguish the correct from false key hypotheses. To employ the metric of mutual information the attacker has to classify the measurement values into different groups based on a leakage function applied to the calculated intermediate values. If this grouping of measurements is sane with respect to the physical leakage behavior of the target device, the different groups have smaller entropies compared to a random distribution of the values. To distinguish the correct subkey the entropies for the grouping of traces for all subkey candidates are compared. The smallest entropy leads to the highest mutual information and will occur for the correct candidate.

When these distinguishers are applied to the individual sample points of all measurements, there will be a significant difference at exactly the points where the considered intermediate value influences the measured side channel. As a result, the attacks will lead to a small difference of means, a low correlation coefficient, and a low mutual information for almost all attacked sample points. Only the combination of samples at the correct time instances and the correct key hypothesis will show a significant deviation from the common case. Thus, power analysis attacks provide both: *i*) the time instance when the considered intermediate value is processed and *ii*) the correct subkey. Using a divide-and-conquer strategy the full secret can be extracted as a set of recovered subkeys.

3 Side Channels as Building Blocks

In security research side channels were usually treated as a parasitic side effect that can significantly weaken the implemented algorithms in embedded devices. Consequently, most of the research on side channels has been spent on the arms race of securing implementations against side-channel attacks and developing more sophisticated attack methods.

In 2009 Lin *et al.* [22] motivated to change this point of view of side channels as parasitic side effects and stimulated research on their constructive use instead. In their work, the side channels of an SCA-protected implementation of the AES cipher have been tailored to design a covered channel which allows to read out the secret key. According to the backdoor nature of the first proposed application, which is a characteristic of Trojan Hardware Circuits, the introduced concept was named *Trojan Side-Channel* (TSC). Nevertheless, the idea is much more general and the authors provided a framework of requirements to implement hidden low-cost communication channels within the side-channel leakage of an embedded device.

In general it is possible to implement a communication channel within the side-channel leakage of a device, that can be read using a single measurement of the side channel and can be analyzed by visual inspection. Although the TSC concept allows implementing this kind of channel, it is focused on much smaller and much more subtle hidden channels. In addition, TSCs are designed to deny access to the transmitted information for everyone who is not in possession of a TSC specific secret. This is closely related to the concept of kleptography introduced by Young and Yung in [32].

The term kleptography describes the art of stealing information securely and subliminally, and was introduced in the context of backdoors in encryption and signature schemes. The kleptography-backdoors permit access to the secret information only to the implementer, even if the full specification of the backdoor is public. TSCs can implement a similar access control, where the desired secret property is realized by keeping a small part of the implementation detail confidential. The case studies presented later in this paper provide design examples that can provide the desired functionality.

3.1 Characteristics of Trojan Side-Channels

In the following the fundamental characteristics of Trojan side-channels are reviewed while the later sections discuss technical details of TSC implementations. According to the inventors [22], the design of a TSC requires to combine three major ingredients: *i*) a predictable *intermediate state*, *ii*) a *combination function*, and *iii*) a *leakage circuit*.

TSCs are in general designed to provide a subtle and hidden channel. In order to realize this, TSCs transmit their information with a very low signal-to-noise ratio (SNR) such that within the side channel the signal cannot be distinguished from Gaussian noise. To extract the signal the statistical tools known from differential power analysis attacks are applied. The core of these tools is a hypothesis test using one of the distinguishers introduced in Section 2.2.

3.2 Establishing Trojan Side-Channels

The applied hypothesis testing mechanism dictates some important requirements to make use of a TSC. First of all the implementer has to be able to acquire a sufficient large set of measurement samples from the channel.

While constant leakages[1] to encode the secret information might work in theory, they will either be so large, that the hidden property of the TSC is violated, or their decoding requires additional profiling of the side-channel behavior of the targeted device. This profiling might be very hard or even impossible in case the implemented TSC is just a very small part of an unknown and complex circuit design. Thus, TSCs are designed such that the acquired measurement samples vary in a way that the side-channel leakage depends on the secret information to be transmitted.

To encode the constant secret information to a varying leakage, the secret and a varying predictable *intermediate value* are combined to an internal signal controlling the leakage output. Thus, when implementing a TSC the first important design decision is to select an intermediate state for this purpose. The choice has to be a value that is known to the implementer of the TSC or is even under his control. The intermediate state used might even come with some uncertainty as long as it can be guessed correctly with reasonable probability. Here the term reasonable is defined by the additional efforts required to decode the channel using partially wrong hypotheses.

Ideal candidates for intermediate values used to encode the TSC information are predictable states of internal registers of the surrounding implementation. These states can be reused without the need for additional registers and thus allow small TSC designs. Nevertheless, artificially introduced states, that are not part of the pristine implementation without TSC, can also be used as well as a combination of existing and newly introduced intermediate values.

Having selected the intermediate values to be used to encode the secret information, the next step is to define a method to combine the secret information and the known intermediate state. This is the task of the *combination function*.

The combination function uses logic gates to map the set of known/predictable intermediate values and the secret as input to one or more output bits. The output bits are then fed into the last important block of a TSC: the *leakage circuit* (LC). The task of the leakage circuit is to generate additional side-channel leakage depending on the output of the combination function.

It is important to note the fact that the leakage circuit is indeed an important part of the design space of TSCs as it allows adjusting the SNR of a TSC to a desired level. A small SNR makes it hard to detect and demodulate a TSC, but on the other hand it also makes the Trojan more resistant to exposure. Using a small SNR to enforce at least n measurements of the side channel to demodulate the TSC requires a third party to also use n measurements in each approach[2] to analyze the TSC, making many analyses computationally infeasible. A good guideline for a suitable LC design is to make the SNR as small as possible and thus the demodulation by the implementer as hard as the application allows. This way the implementer of a TSC can maximize the difficulty of the TSC to be exposed.

For each possible choice of the transmitted secret information an implementer knowing the used intermediate state, the combination function, and the leakage circuit can predict the leakage behavior of the leakage circuit. When decoding the information transmitted by the covered channel, this knowledge is used to perform a hypothesis test on the side-channel to extract the encoded secret information. To allow for an efficient hypothesis test, the combination function should be designed in a way that different secrets re-

[1] Constant leakage here means that the leakage provided by the TSC does not depend on the intermediate state of the device, and does not change as long as the secret key is fixed.

[2] Since the selected intermediate state and the combination function are kept obscure, the third party needs to guess them and examine the existence of a TSC module for each guess.

sult in very different leakage behaviors. In other words, the outputs for each possible set of two different keys should not show any significant correlation. Furthermore, the implementer should keep the area required for the combination function as small as possible to evade detection of the backdoor by mask inspection.

The leakage circuit mapping the output of the combination function to a physical observable can be designed in many different forms. Additional glitches, toggling circular shift registers, charging and discharging internal capacitances, and pseudo-NMOS gates are just some examples to generate the required additional power consumption for the power side channel.

Having introduced the general idea of side channels as building blocks to implement a covered communication channel, we will give an overview of published and new applications of the TSC concepts. These examples have been selected to illustrate the large design space available to TSC designers and to demonstrate the scalability of the approach reaching from very small add-on backdoors on unprotected implementations of ciphers to much more complex TSCs that are adapted to implement a hidden channel in SCA-resistant implementation schemes. The sophisticated TSC implementation presented in this case study is a new design that outperforms all previously published TSC implementations. The set of examples starts with different approaches to Trojan hardware applications and continues by illustrating examples of watermarking applications.

4 Case Studies: Trojan Hardware

The first class of applications for tailored side channels to be presented are hardware Trojans. For the purpose of this paper we focus on Trojans injected in embedded security modules, as these are high-value targets for malicious covert channels and thus the first type of hardware design that is required to be protected from possible TSC threats.

4.1 Introduction to Trojan Hardware

For a long time high-security applications such as banking or government systems had been the moving power to implement side-channel resistant cryptographic routines in hardware. While they still play the most stimulating role for the development of hardened cryptographic solutions, today also many other commercial off-the-shelf products make use of hardware-based security. Especially the trend towards system-on-a-chip (SoC) solutions has facilitated the integration of highly resistant cryptography in commercial silicon devices.

SoC products are composed of specialized function blocks developed and maintained by independent engineering teams. A typical example for system-on-a-chip applications are high-end cell phone processors. The types of required function blocks ranges from digital application processors and security modules to mixed-signal baseband processors and video encoding. The approach of combining multiple functional blocks on a single chip does not only reduce the power consumption of embedded systems, but also increases the reusability of functional designs and can thus save development costs. Today specialized companies offer all kinds of functional blocks for licensing. These blocks are often termed IP blocks or IP cores as most of them are protected by intellectual property rights.

There are two types of IP cores: hard cores and soft cores. Soft cores are available as sources in a hardware description language like VHDL or Verilog or as netlists, which are harder to reverse engineer and might even be encrypted. The essential property of soft IP cores is that they are synthesizable and thus portable to any process technology. In contrast, hard IP cores are delivered as completely routed blocks that often come as black box implementations with a given interface specification. Hard cores are foundry specific and can in most cases not be transferred to different processes or foundries. Analog circuits always come as hard IP blocks. From a security perspective the system-on-a-chip trend leads to many very challenging tasks. Circuits using third party components are still very hard to be verified as even the implementers themselves do not have access to all details of the circuit. This allows malicious distributors of IP cores to introduce hardware Trojans into their functional blocks.

As an example, a fraudulent company might insert a Trojan into an otherwise secure cryptographic IP-block that is being used for payment systems. Due to the fast growing and changing nature of the IT-Industry such a fraudulent company might appear legitimate or might impersonate a legitimate company. If a stealthy hardware Trojan is used, like the ones we propose in this paper, even an independent security evaluation laboratory might declare the Trojan-infested IP-block as secure and legitimate. As a result companies could end up using an Trojan-infected IP block in payment systems, leaving them wide-open for fraud.

Besides fraudulent companies, also a group of engineers could insert an undetected Trojan as an inside-job. Last but not least, government agencies are interested in introducing hidden backdoors into a variety of applications, ranging from communication systems to access control mechanisms. There is no doubt these agencies possess the required funding and influence to embed Trojan hardware in commercial systems.

In recent years the threat of hardware Trojans has gained increasing attention. Especially the US Department of Defense has realized hardware Trojan as a potential threat [3], which resulted in the "‘DARPA TRUST in Integrated Cir-

cuits"' program. But others such as the US senate as well as the semiconductor industry acknowledge the danger of hardware Trojans [21,5,4] and the research interest in this area has increased significantly over the last few years.

In general hardware Trojans are circuits hidden in a bigger design that add some malicious functionality to the device [11,17,18]. A prominent example are kill-switch designs, which destroy the chip or disable parts of its functionality upon occurrence of some trigger condition. Another example are Trojans, which allow to read out secret encryption keys. Conventional proposals for hardware Trojans used input and output pins of the compromised designs to implement their functionality. This changed with the introduction of Trojan side-channels (TSC) [22]. This class of hardware Trojans employs side channels to implement covert communication channels.

4.2 Attack Scenario and Terminology

For the rest of the paper we will use the following terminology. A Trojan scenario consists of mainly two parties. The first party is the implementer of the Trojan. We refer to this person as the *implementer* or the attacker. The other party is called the *evaluator*. The evaluator is the owner of the system design that hosts the Trojan circuit. The task of an evaluator is to test his devices functionality and to protect it from all possible vulnerabilities.

The goal of the attacker is typically to covertly leak out the secret key of a side-channel protected encryption scheme by means of the Trojan. Note that using a Trojan to covertly transmit a secret key from an implementation does not make much sense, if the attacked device is not side-channel protected. In that case one of the well documented SCA methods can be employed to extract the key without any need to modify the hardware design.

The idea of the TSC paper [22] is to use a side channel as a building block to implement a covert communication channel. This channel is then used to transmit a secret encryption key K, which was assumed to be protected in a way that it cannot be recovered without the Trojan side-channel. The role of the TSC is to encode K into physical leakage. This encoding $e(K)$, has to be designed to reserve usage and detection of the side channel only to the implementer. Once the IC is deployed in the field, an evaluator must not be able to detect the TSC. For the implementer of a Trojan this means that the Trojan has to be very small (i.e., few gates), should have a low power footprint, hide in unused or test areas of the chip and should not interfere with the designed functional behavior of the pristine device. Furthermore, the Trojan should implement some *encryption property*[3] that is designed to avoid usage of the Trojan by others that are aware of the existence of the TSC. TSCs that incorporate such an encryption property require special attention during IC security evaluation.

A good encryption property requires an evaluator to overcome an infeasible computational or experimental effort, e.g., 2^{80} calculations or 2^{40} measurements, to access the secret information that is exposed. On the other side the attacker (who designed the TSC) needs to have some advantageous knowledge, allowing him to make use of the TSC within feasible efforts. Due to the hidden nature of the TSC this type of Trojan does not need an activation mechanism. The TSC can always be active without revealing itself.

4.3 The CDMA Trojan Side-Channel

The first example of a tailored side channel to be given is the CDMA Trojan introduced by the TSC inventors in [22]. This design makes use of a hidden communication channel by means of spread-spectrum communications (also known as code-division multiple access (CDMA)). In this method the information of each bit is spread over many clock cycles.

The basics of the CDMA encoding are very similar to conventional stream ciphers. A bit sequence (the code) is used to modulate information bits using an XOR operation. Contrary to the stream cipher concept, CDMA uses many code bits to transfer a single bit of information, i.e., the code bits are changing much faster than the information bits. This strategy spreads the information contained in a single bit along a longer bit (or code) sequence which allows transmission and recovery of information in subliminal channels transmitted with very small SNR ratios. This property of providing a hidden signal below noise level also makes CDMA the method of choice to implement hidden military communication channels.

The demodulation used to decode CDMA channels helps to understand how CDMA can establish channels in this sub-noise domain. The process of decoding using a correlation demodulator is very close to what the community of cryptographers knows as correlation power analysis. The demodulator uses subsequent power measurements within a single power trace and correlates them to the synchronized code sequence. If the correct code has been used, this leads to a positive correlation coefficient for encoded zeros and a negative correlation coefficient for encoded ones. The more power measurements the demodulator analyzes, the more "process gain" (which is the ratio of code sequence length to the bit information length in spread-spectrum encoding) is available to overcome a low demodulation SNR.

[3] This is a property that is considered for the encoding mechanism $e(K)$, and does not deal with the encryption scheme realized by the target device.

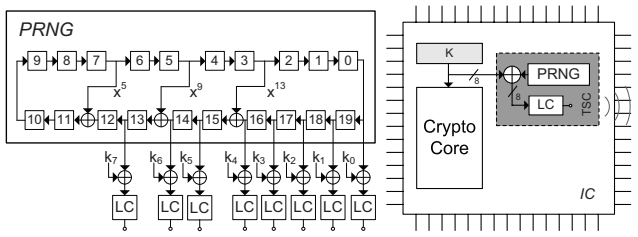

Fig. 1 Block diagram of (right) a CDMA Trojan and (left) an exemplary PRNG (see [22] for more details on the used PRNG)

Note that the CDMA channel can only be demodulated using the correct code sequence and demodulation with different code sequences will not lead to any significant correlation. Therefore, it would also be possible to transfer bits on multiple CDMA channels simultaneously, as each CDMA channel is indistinguishable from noise for all other channels.

The block diagram of an exemplary CDMA TSC is depicted in Fig. 1. The TSC employs the CDMA methods by using a pseudo-random number generator (PRNG) or stream cipher to create a code sequence. This sequence is then used to XOR-modulate the secret information bits. In the TSC the modulated sequence is then forwarded to a leakage circuit (LC) to set up a covert CDMA channel within the power side channel.

In this model, the advantage of the attacker is the knowledge about the exact setup of the code generator (for example the initialization vector and feedback coefficients of an implemented PRNG or the secret key in case of a stream cipher setup) which are required to predict the code sequence. Knowing all details of the code generator gives the attacker an essential advantage over evaluators who cannot distinguish the covert channel from noise.

For decoding this channel, the attacker performs a correlation demodulation[4] on measurement points of subsequent clock cycles as described above. When evaluating side-channel leakage of the compromised device, the leakage due to the TSC will not be detected during attacks tailored to the pristine IC core.

In this TSC the internal state used to encode the secret information is an artificially introduced linear feedback shift register (LFSR) used to generate a code sequence. The combination function is a single XOR gate, that encodes each bit of information at its input to a sequence of bits. In the proof-of-concept ASIC simulation of this method the output of the combination function was attached to a large capacitance. This lead to measurable charging currents at the ground plane of the device under attack.

[4] Note that depending on the LC, it might be necessary to consider a mapping of the used code with respect to a suitable power model of the LC prior to correlation-based demodulation.

This type of Trojan does not need a trigger mechanism. The TSC can be active right from power up as it does not interfere with the correct execution of the core. Knowing the time when the chip was powered up, the attacker can predict the state of the LFSR in a power measurement within a certain margin and align the LFSR sequence and the power traces accordingly.

The demodulation can then be performed similar to a CPA: Given a set of measurement samples t_i of a single power trace, the attacker estimates the Pearsons correlation coefficient as

$$\rho = \frac{\sum_{i=1}^{n}(t_i-\bar{t})\cdot(c_i-\bar{c})}{\sqrt{(\sum_{i=1}^{n}(t_i-\bar{t})^2)(\sum_{i=1}^{n}(c_i-\bar{c})^2)}},$$

where c_i are the bits of the known code sequence (mapped to fit the power model of the leakage circuit), n is the length of the code sequence, and the letters with bar on top denote the mean values calculated as

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

The secret bits are fed bitwise to the TSC. Depending on the attacked cipher this might require some additional circuitry to store a copy of the secret key. In some cases it might be possible to reuse bits available in the pristine cipher implementation and leak those bits to allow an attacker to recover the secret key. In our own proof-of-concept ASIC simulation, we used multiple outputs of the LFSR in parallel to establish multiple parallel CDMA channels each transmitting single bits. Note, that the required number of code bits to transmit a single bit via a CDMA channel depends on the overall signal to noise ratio of the channel. In this context the number n of parallel channels limits the SNR to an upper bound of $1/n$ (see [23] Section 6.3.2).

In our simulation we were able to extract key bits whenever we selected code lengths of more than 1030 bits per transmission. This also determines the number of clock cycles after which the next set of key bits has to be available at the CDMA channel input.

Being related to a stream cipher, the CDMA approach also shares similar properties. In this case the most disturbing fact is the possibility to recover the used code sequence when knowing the leaked secret. An evaluator controlling the secret key bits could simply average many power traces with the leaked secret bit set to logic 1 and repeat this with the power traces with the leaked secret bit set to logic 0. Subtracting those two traces allows to reverse engineer the code sequence and thus to gain full access to the hidden channel.

This shortcoming can be solved by employing an internal state dependent initialization of the code generator. This increases the size of the Trojan, but using a varying known internal state to initialize the generator makes it significantly harder to reverse engineer the details of the generator.

Another problem is the sequence used to encode a single bit. The demodulation is a very robust process and thus allows for incorrect but similar code sequences to still provide a significant correlation. This feature allows an evaluator to use guessed code sequences to analyze the Trojan. Here the use of longer sequences and lower signal to noise ratios can help.

While this approach might have some subtle shortcomings, it still provides enough room to be adapted and extended at the cost of some additional gates to be very powerful. In addition, the approach can also be used in more complex TSC designs as shown later in this article.

4.4 The Input-Modulated Trojan

The second example of a Trojan setup using side channels introduced by the inventors of the TSC in [22] is a Trojan that uses only available internal states of the pristine circuit and can thus be designed to be extremely small. In this example, as shown in Fig. 2, a TSC is attached to an implementation of the AES cipher.

In each round of the cipher the Trojan leaks out a subset of 16 bits of the roundkey. To do so it is attached to two bytes of the registers used to implement the AES keyschedule. Making use of the structure of the AES keyschedule these bytes allow reconstruction of the full secret key. It is also possible to leak just a single byte in each round of the keyschedule, but this would require additional efforts to bruteforce the remaining bits of the full key at the end of the attack.

The combination function of the original TSC publication used 16 bits of the plaintext which were pairwise combined with 16 bits of the registers used during the AES key schedule by means of AND gates. The output of all these AND gates was then fed to a large XOR gate to output a single bit (see [22] for more details about the structure of the underlying keyschedule unit and the combination function). In contrary to the CDMA Trojan, this design of the combination function does not encode the key bits by modulating them with a known sequence, but by using the internal state (here plaintext bits) which changes during multiple encryption runs. Therefore, the attacker needs to measure the transient power consumption of a larger set of encryption runs.

Knowing the used input values (plaintexts in this case) he is able to predict the outcome of the combination function by making an assumption of the leaked secret. For each of the 2^{16} possible secret messages that might be transmitted, the attacker can predict the hypothetical outcome of the combination function for each acquired power trace. Correlating these hypothetical values to the measured power traces will lead to a high correlation for one of the hypotheses (corresponding to the transmitted message) and no correlation for the other hypotheses. An attacker can thus distinguish between transmitted messages.

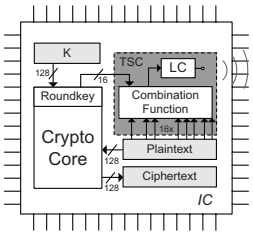

Fig. 2 Block diagram of an Input-Modulated Trojan (an example given in [22])

Because of the specific structure of the key schedule unit considered in [22], within the full power traces the TSC will subsequently transmit the information gained from the different rounds of the keyschedule, so that they can be found by correlating at different positions in time. Note that this Trojan consists of only a (very small) combination function and a leakage circuit. The internal state is a reused value and thus requires only some additional wires to be accessed. This design has been successfully demonstrated in an FPGA proof-of-concept implementation using a LUT realizing the leakage circuit.

A LUT is a look-up-table and corresponds to the smallest available building block inside an FPGA. Besides being used to realize logic functions, FPGA LUTs can also be configured to serve as shift registers. A very efficient way to implement a leakage circuit using a LUT is to fill it with an alternating sequence of ones and zeros and feed back its output to its input. By gating the corresponding clock signal, which is shifting the LUT content, based on the output of the combination function (AND conjunction), an extremely small and yet efficient design can be realized.

This Trojan uses only 15 4-input LUTs for the combination function and 1 LUT for the leakage shift register (16 LUTs in total). Compared to the logic used by the main application, typically taking up thousands of LUTs, the share of the TSC can be considered negligible.

The suggested combination function of the initial TSC paper introduced above comes with a small shortcoming: when all bits of the transmitted secret are zero, then the output of the combination function will be a constant value and thus will not lead to any correlation during demodulation.

While an attacker can distinguish zero-messages by assigning all decodings with absent correlation to the zero-message this behavior is still an undesired property. Fortunately this concept is very flexible and by adding a single additional internal state bit to the XOR gate of the combination function this issue can be solved with small overhead.

What has not yet been discussed is the encryption property denying access to the TSC to others than the implementer. In case of the input-modulated TSC the secret knowledge are the details of the combination function. First of

all, the selection of used plaintext bits as internal state is unknown. Furthermore, the information which plaintext bit is paired to which bit of the secret message is unknown as well as the knowledge of the bits selected to be transmitted within the secret message. If this "keyspace" of choices seems to be too small, the designer can also add an arbitrary amount of different combined plaintext bits as input to the combination function. Selecting input bits that are inverted before entering the combination function makes it even impossible for an evaluator to distinguish the effect of key bits from the effect of inverted bits.

Not knowing all details of the TSC implementation efficiently prevents an evaluator from creating sets of hypotheses to distinguish transmitted messages. Similar to the case of the CDMA TSC, this TSC should be designed to have a small SNR to impede detection of the TSC for an evaluator.

4.5 A TSC Backdoor for Implementations Protected by Hardware Masking

As a third case study on TSCs we introduce a new design based on a CDMA Trojan side-channel to add a hidden backdoor to an implementation protected by a *masking* scheme. Masking is a countermeasure to power analysis attacks which tries to randomize the intermediate values computed by the device by means of a set of random values called masks. The masks are refreshed for every execution of the cryptographic device. The unknown random masks prevent an attacker from predicting intermediate states and therefore deny straightforward DPA attacks.

In contrast to the previously discussed conceptual designs of Trojan side-channels the following example shows how an adversary can adapt a TSC design to construct a backdoor in a real-world SCA-protected implementation. In a reasonable design this backdoor has to be subtle enough that it cannot be detected by means of side-channel analyses performed by evaluation laboratories, but still has to allow the implementer of the backdoor to perform the attack. To achieve this, our backdoor leaks the random mask bits instead of the secret key.

The key recovery process is split in two steps:

1. The bits of the random mask, which are leaked by means of a spread-spectrum TSC, are recovered,
2. Knowing the random mask of many power traces a conventional CPA is performed to recover the secret key.

For the purposes of this case study, the Trojan needs to transmit many bits, i.e., a 128-bit mask for our proof-of-concept implementation. To achieve this, the backdoor implementation employs the combination function introduced in the TSC paper to be used in an input-modulated Trojan. The purpose of this combination function is to compress multiple bits of secret information to single-bit leakages.

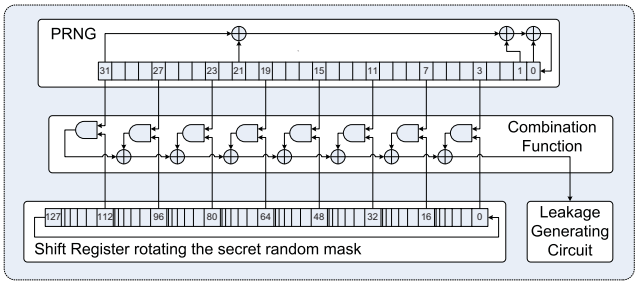

Fig. 3 Design of the implemented backdoor leaking the random mask

The combination function was designed to allow a good discrimination of transmitted values in a differential power analysis and to require only few logic gates.

The mechanism leaking the mask bits is illustrated in Fig. 3. In this Trojan the 128 bits of the mask used to protect the implementation are stored in a circular shift register. Every sixteenth bit of this register is connected to the combination function. In addition to the eight bits from the shift register, another eight state bits from the encoding LFSR, i.e., the code generator, are connected to the combination function. The sequence generated from the later bits can be predicted by the implementer who knows all details of the LFSR, but not by any other party.

The reason for having the circular shift register saving the random mask is twofold:

1. The combination function should be limited in the number of inputs, so (similar to SCA attacks) the adversary is able to apply a divide-and-conquer scheme and keep the search space (for the mask bits) feasible.
2. Instead of using several instances of the combination function for each part of the random mask, the design given in Fig. 3 shares the combination function and the LFSR for each 8-bit part of the random mask. Therefore, every sixteen clock cycles the same eight bits of the random masks are given to the combination function.

This introduced circuit generates a bitstream that is unique for all different masks. As already stated above, each bit of the stream depends on eight bits of the mask and in this TSC every sixteenth bit is encoded using the same eight mask bits. In order to recover the mask, an attacker guesses the value of one of these sixteen eight-bit sets of the mask. Using the known LFSR he then predicts the hypothetical output bit of the TSC for all clock cycles where the attacked set of bits is used. In a final step he can use the corresponding sample points of a long power trace to verify his guess by means of a correlation power analysis attack. The leaking of the complete mask can thus be considered as an interleaved transmission of sixteen 8-bit values by means of CDMA. Note that our proposed method to leak the mask can be extended to transmit much more than 128 bits per power trace.

Having recovered the complete masks for all measured power traces, a first order CPA can be performed. Using the known mask bits an attacker is now able to generate predictions for intermediate states of the cipher that can be used to test key hypotheses with the known methods of side-channel analysis.

The careful reader might question at this point, why this design does not directly leak the secret key. We now show, that this approach provides a significant improvement over all existing works.

As the secret information (i.e., the mask) leaked by the TSC is

1. random and thus not predictable,
2. different for every measurement, and
3. not related to any predictable intermediate state of the protected implementation of the underlying cipher,

the TSC cannot be detected by conventional SCA attacks.

This is a major improvement with respect to the detectability compared to the methods directly leaking the secret key introduced earlier. The introductory designs allowed an evaluator, who knows the secret key, to challenge and thus analyze the TSCs behavior. This left some room for the development of potential detection methods.

In contrast, this implementation is completely non-deterministic and can only be analyzed by the implementer. To illustrate the feasibility to detect the backdoor we compare the effort of the implementer to recover 8-bit of the transmitted mask to the effort of an evaluator to find the backdoor.

Knowing all details of the implemented TSC the implementer only has to predict and correlate the sequences generated for 2^8 possible values of a mask byte. An evaluator, in comparison, has to guess the length of the LFSR, its feedback function, the used initialization vector, the amount and positions of the taps connected to the LFSR, the used combination function, the order of inputs to the combination function, and many more details, which are all only known to the implementer of the backdoor. Again inverted input taps can be used to further improve the used combination function. We thus claim that the design space of this class of backdoor implementations is so big, that it cannot be explored by an evaluator by guessing within feasible time.

Masking Scheme

In order to test our new TSC concept, we implemented a low power serialized masked AES design which is depicted in Fig. 4. The main building blocks are a single masked S-box operation and a masked MixColumns operation.

Both the key schedule and the SubBytes transformation utilize an 8-bit data path to the S-box, while the MixColumns and AddRoundKey transformation make use of a 32-bit wide data path. At the beginning of each round the S-box is initially used by the key schedule to compute the first 32-bit of

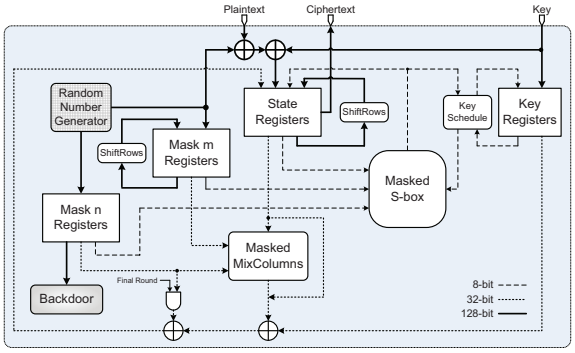

Fig. 4 Block diagram of the target architecture

the next round key. The other bytes of the round key are then computed on the fly while processing the SubBytes transformation on the internal state. After every four clock cycles of SubBytes, the MixColumns and AddRoundKey transformations are applied to the newly computed 32-bit state. This sequence is repeated four times to complete the full AES round function.

While masking of MixColumns is trivial because of its linear properties, the secure masking of the non-linear S-box is quite challenging. In our design we implemented the masking scheme presented in [10][5]. It uses the so-called "tower-field" approach [30] to perform the inversion in the $GF(2^2)$ sub-subfield where inversion is equivalent to squaring and therefore linear. By additively masking the data path and strictly keeping the required order of masked summations, it achieves "perfect masking" by the definition of [9].

It is important to note that this scheme requires two independent 8-bit masks m and n to randomize the input and output of each S-box computation of one round. Therefore, our design requires two random 128-bit masks to successfully protect all 16 bytes of the internal state during the SubBytes and MixColumns transformations. These masks are internally generated by a PRNG and change after each full AES computation.

Since every S-box output is masked by parts of n, knowledge of this mask would enable an attacker to perform a couple of CPA attacks, e.g., using a Hamming Distance (HD) model predicting consecutive S-box outputs being overwritten in the state registers.

Experimental Results

To implement the masked architecture and practically evaluate our proposed scheme we have used a *Side-channel Attack Standard Evaluation Board* (SASEBO) [2] platform that is equipped with an xc2vp7 Virtex-II Pro FPGA. The chip was clocked at a frequency of 3MHz, and the power

[5] Its HDL specification was obtained from the official website of the corresponding author.

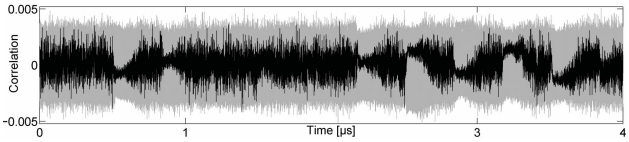

Fig. 5 Result of a CPA attack using a HD model on 8 bits of the state register using 1 000 000 measurements

consumption traces have been acquired by measuring the voltage drop over an 8Ω resistor in the V_{DD} (1.6V) path. All experiments were performed using a LeCroy WP715Zi 1.5GHz oscilloscope and a differential probe.

As a first step we implemented the masked AES-128 scheme on the reconfigurable device and then measured the power consumption of the target implementation at a sampling rate of 5GS/s to collect 1 000 000 traces. All measurements used the same fixed key and uniformly distributed random plaintexts. In order to confirm the resistance of the target implementation against first-order CPA attacks, we evaluated multiple attacks including attacks using the Hamming weight (HW) of the S-box input and output bytes, attacks using the Hamming distance of a part of the state register, and attacks using the zero value model on the S-box input. As expected, none of the performed attacks allowed to extract information on the secret key. To illustrate the achieved resistance against first-order CPA attacks, Fig. 5 shows the result of an attack predicting the HD of 8 bits of the state register.

In the next experiment, we moved another step towards the proposed backdoor scheme by implementing the previously mentioned PRNG. Attached to the PRNG a leakage generating circuit (LC) has been realized by configuring a single LUT as a circular 16-bit shift register which is initialized with $(\mathrm{AAAA})_{\mathrm{h}}$. The output of the PRNG was used to control the clocking of the leakage circuit, i.e., toggling of all 16 bits of the shift register in a clock cycle. At the start of every encryption process the PRNG loads a specific hardwired initialization vector (IV) to achieve a deterministic output sequence that is fixed for each encryption run. Note that in this step we have implemented only the PRNG and the LC without the protected AES and the combination function.

To verify the feasibility of detection of this simple sequence generator, we had to measure a long trace similarly to the experiments used to illustrate the concepts of side-channel watermarking presented at COSADE 2010 [7]. In this experiment we therefore decreased the sampling rate to 250MS/s to be able to cover many clock cycles, e.g., 500 000, for a single power trace. In order to reduce the amount of data for further analysis we compressed the acquired power traces by storing only the average power consumption of each clock cycle of a trace. Afterwards, we simulated the PRNG with respect to the known IV and computed the correlation coefficient between the aligned power trace and the PRNG output sequence.

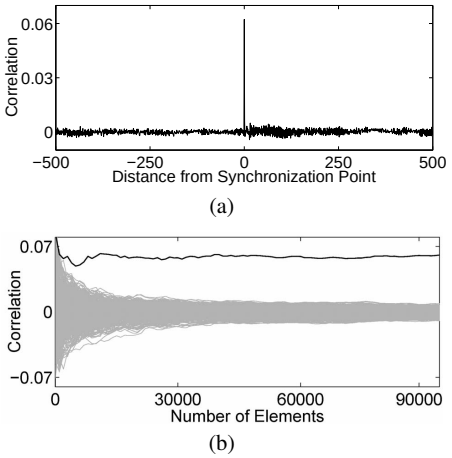

Fig. 6 Correlation between PRNG simulation and measurements aligned on neighboring positions (a) using 500 000 elements , (b) with respect to the length of the analyzed sequence

Since the exact alignment of the power trace and the simulated sequence might be quite challenging, sliding the PRNG output vector over the power trace vector might be necessary. However, the attacker knows the details of the PRNG and the used initial state and hence should be able to predict the state of the PRNG for a given clock cycle with only a small error margin (i.e. below a few hundred or thousand states). In this sliding method the correlation coefficient is computed for every possible alignment position until a prominently deviating correlation coefficient can be found for a specific alignment.

Fig. 6(a) shows the result of such an alignment process using a window of 1000 elements around the correct alignment position for a sequence length of 500 000 clock cycles. Further, Fig. 6(b) illustrates the effect of the used sequence length on the detectability of the PRNG sequence by means of the correlation coefficient.

Having confirmed the feasibility of sequence detection and the required sequence lengths, we added the backdoor circuit into the protected AES implementation. Before giving numbers on the increase in hardware consumption caused by the backdoor implementation we should note that for our chosen case study we have used only one instance of a very compact masked S-box in our low-area implementation. This said, adding the backdoor circuit led to an increase of 162 slice flip-flops, 167 4-input LUTs, and in sum 88 occupied slices.

According to the combination function used by the backdoor (see Fig. 3), the clocking of the leakage generating circuit is in each clock cycle controlled by a combination of 8 bits of the mask and 8 bits of the internal state of the PRNG. Remember that every 16th clock cycle the same 8 bits of

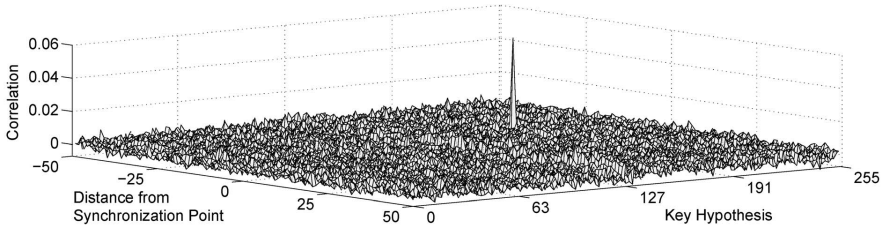

Fig. 7 Result of a mask recovery process using a single trace of the backdoor implementation

the mask serve as the input to the combination function, although in a rotated fashion. Therefore, all power consumption values in the trace having a distance that is a multiple of 16 clock cycles solely depend on the same 8 bits of the mask and the predictable PRNG state. Thus, the compressed traces can be downsampled by extracting every 16th bit resulting in 16 different traces.

Similarly to the divide-and-conquer approach of CPA attacks, each of these downsampled traces can then be used separately to recover the subsets of 8 bits of the mask, i.e., in sum all 128 bits can be recovered. In order to recover the encoded mask the 256 possible output sequences of the leakage circuit corresponding to the 255 possible 8-bit values of the mask (i.e., excluding the zero value) are simulated. Then the method used to detect CDMA sequences described earlier can be used as a distinguisher for the correct mask bits, i.e., the hypothetical sequences are correlated to the corresponding aligned and downsampled power trace. The correct hypothesis can then be identified by a prominent value of the correlation coefficient.

Again, if unknown, the required alignment can be achieved at the cost of some additional computational efforts by employing the sliding approach. As explained before, we are able to cover 500 000 clock cycles in each measured power trace. Thus, every downsampled power trace has a length of around 30 000 elements. According to Fig. 6(b) this number of elements is sufficient to distinguish the correct hypothesis of a sequence amongst the others in our experimental setup. As an example, Fig. 7 shows the result of this process using downsampled power traces with a length of 30 000 elements.

Based on the used combination function, and similarly to the presented results in [22], when all 8 bits of the mask are zero, the output sequence is constant and thus leads to no significant correlation for any of the hypothetical sequences. Therefore, the lack of a significant correlation with any of the predicted sequences serves as an indicator to detect the 8-bit zero mask value case.

Repeating the described mask recovery process on all 16 downsampled power traces allows recovery of all 128 mask bits using a single power trace. Note that since the mask bits change for every execution of the encryption, each power trace must be processed independently and cannot be averaged to facilitate recovery of the used mask bits. It means that for every encryption process a long power trace has to be measured and 16 independent mask recovery processes have to be performed. Fortunately, once the offset for correct alignment, i.e., the distance from the synchronization point in Fig. 7, is revealed, it can be used for all mask recovery processes on all power traces.

As explained before, the masked S-box of the protected AES implementation makes use of two 8-bit masks to randomize both its input byte and its output byte. In our modified implementation the 128-bit mask used to protect the 16 output bytes of the S-box in the first round of the AES, is stored in the mask shift register of the backdoor scheme. As a consequence recovering the mask bits using the described methods reveals all masks of the first rounds S-box outputs.

Also rembember that in the targeted architecture the (masked) S-box outputs are processed byte-wise for every computation of the S-box. Therefore, a possible CPA attack would be to predict the HD of the subsequent S-box outputs which are overwritten in the state register. Since two subsequent S-box outputs depend on two key bytes, for the first CPA attack two key bytes have to be guessed, i.e., 2^{16} hypotheses. However, once the first key bytes are recovered, the CPA attacks targeting the remaining bytes of the key can make use of the already known key bytes to reduce the key search space to 2^8 per attack.

In order to evaluate the attack we measured 7 000 long traces (at a sampling rate of 250MS/s) and recovered the mask bits for each one. Then, we used the same traces to perform the aforementioned CPA attack using the HD model. The result of the attack which recovers two key bytes is presented in Fig. 8(a). Since the state register is shifted in a byte-wise fashion, the correct hypothesis can appear multiple times in the attack results. Furthermore, Fig. 8(b) shows the highest correlation values for all hypotheses with respect to the number of used power traces.

It should be noted that the attack is even robust to erroneous recoveries of mask bits due to noisy measurement setups, an insufficient amount of clock cycles covered by the downsampled traces, or low sampling rates. Failed recoveries of mask bits affect the CPA attack similar to additional noise, and thus only lead to an increase of the required number of measurements, but will not prevent a successful key extraction.

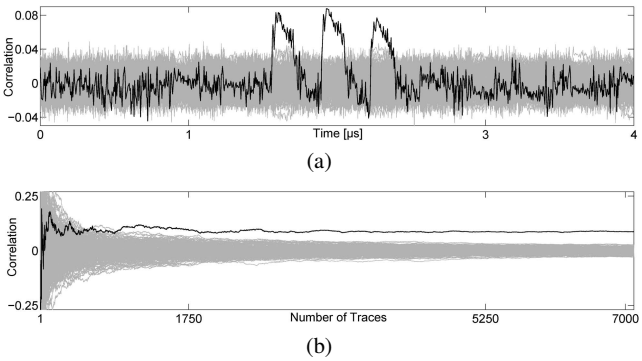

Fig. 8 CPA attack result on the masked implementation using the recovered masks (a) using 7000 traces , (b) with respect to the number of used power traces

4.6 Summary

In the last sections we gave an extensive review of the previous works on Trojan side-channels and combined them to a more sophisticated concept for backdoor implementations that are tailored to protected cryptographic schemes employing masking countermeasures. The given examples illustrate the versatile and subtle nature of Trojan side-channels and demonstrate the serious thread they can pose, once deployed in the field. Our extensions to overcome the shortcomings of the initial TSC proposal, such as using inverted input taps or the combination of CDMA and input modulated TSC concepts, lead to improved TSC designs.

In contrast to conventional architectures of Trojan hardware, the TSC approach allows much more subtle designs, which are very hard to reveal without applying sophisticated mask inspection tools. We have demonstrated sophisticated TSCs, which are very resistant to detection by both functional testing or side-channel evaluation. The flexibility of the very general approach also allows to tailor Trojans to the targeted designs by exploring the given environment and reusing parts of the existing logic where possible.

5 Case Studies: Watermarking

5.1 Motivation for Hardware Watermarking

So far we have discussed scenarios in which side channels are used in a malicious way to leak out secret information. However, the concepts of Trojan side-channels can also be used for constructive applications. In this chapter we demonstrate how Trojan side-channels can be used to build a hardware watermarking mechanism to detect IP theft.

As mentioned in Section 4.1, in most cases ICs are not designed from scratch, but parts of the design are reused or bought from other companies. Given the huge design costs of ICs, these so called IP cores are a very worthwhile target for IP theft and product piracy.

A promising solution proposed to counter the IP theft threat is the concept of hardware watermarking [15], [26], [28], [31]. Hardware watermarks do not make the IP theft itself more difficult, but are used to efficiently detect the IP theft. They can, furthermore, be used to provide proof-of-ownership.

The idea behind hardware watermarks is to hide information in the IP core that can be used to reliably identify the original owner of the IP core. A good hardware watermark should have the following properties:

- It should be easy for the owner of the IP core to detect the watermark.
- It should be impossible/very difficult for an attacker to remove the watermark from an IP core.

In the watermarking scenario the attacker is the person who tries to steal and illegally use the IP core. The goal of the attacker is to make the detection of the watermark impossible so that his fraud stays undetected. [7] introduced a hardware watermarking mechanism based on Trojan side-channels that is very robust to many attack scenarios.

The main idea is to hide the watermark in the power consumption of an electronic device. In order to verify the watermark the IP owner collects measurements of the power consumption of the device and then performs a side-channel analysis. If a protected IP core is used in the tested device, the analysis will reveal the existence of the included watermark.

This approach is similar to the one proposed in [16], where the temperature side channel was used to transmit a watermark instead of the power side channel. Due to the fact that the heat side channel is used in this work, the leakage generator is very big and slow. This has the disadvantage that the overall power consumption of the device increases significantly, which is undesired in embedded applications. Furthermore, the watermark is not very robust as it is easier to detect the watermark signal due to the strong leakage generator. These characteristics makes it possible to add a jamming (or inverse) signal to make the detection of the watermark impossible.

The idea to use the power consumption to transmit a watermark has been proposed in [33], but the involved detection mechanism was not very robust. In this approach the watermark signal is transmitted above the noise level and is very vulnerable to noise. Simply transmitting several watermarks can render the detection impossible. Hence an attacker could again easily create jamming signals to impede or even deny the watermark detection.

In conventional hardware watermarking scenarios the verifier needs to have access to the embedded code, the chip layout or the content of the memory of a device to be able

to verify the watermark. However, in real world applications a verifier needs to employ expensive and time consuming reverse-engineering techniques to get access to this information, which makes the watermark detection both expensive and difficult.

With the recently introduced side-channel watermark technique these efforts are not necessary anymore to perform a verification of a watermark. Only power measurements are needed, which can be acquired with relatively cheap equipment and moderate efforts. This makes the side-channel watermark a superior approach to conventional watermarking techniques and thus a promising candidate to be adapted in many real-world applications.

5.2 Side-channel Based Watermark Design

The side-channel hardware Trojans introduced in Section 4 are designed in a way that the owner of the Trojan can use them to leak out secret information, while staying hidden and invisible for everybody else. These properties are exactly what is needed for hardware watermarks: The owner of the watermark should be able to detect the watermark while they should stay invisible for everybody else. Furthermore it should be impossible for an attacker to remove the watermark. The main difference between side-channel watermarks and side-channel Trojans is the information that is leaked out: In case of the hardware Trojans this information is a secret key while in the watermark case we want to leak out a unique ID.

To adopt the idea of Trojan side-channels to design hardware watermarks one therefore only has to adapt the choice of the leaked secret information. The two different original methods to implement Trojan side-channels, CDMA Trojan and input-modulated Trojan, can both be employed to design side-channel hardware watermarks.

The unique ID that is leaked out should have the property that its ownership can be unambiguously assigned to a single entity. For example, this can be achieved by means of a digital signature or a hash value of the IP owner's name. If a secure hash function is used, it is not possible for an attacker to find a string different to the IP owner's name, that will result in the same ID. Hence, if the watermark ID is detected in a device, the verifier can proof not only the existence of the IP core but also his legal ownership of the IP. Thus, this method provides all proof-of-ownership features required to serve as hardware watermark.

5.2.1 CDMA Side-Channel Watermarks

In Section 4.3 a PRNG is used to encode a bit sequence similarly as it is done in a CDMA system to covertly transmit a signal below the noise level of the power consumption. We applied this technique to covertly transmit a secret key and extended the idea in Section 4.5 to leak the random masks employed in a protected AES implementation.

For our watermark purpose we make use of the same concept, the design of the PRNG, and the leakage circuit. The unique ID of the watermark is XOR-modulated with the output of the PRNG and then transmitted using the leakage circuit. In order to detect the watermark the owner of the watermark can use his knowledge of the PRNG design, the unique ID and the leakage circuit to perform a side-channel analysis just as the attacker did in the Trojan scenario. If the device under test has the watermark embedded, this side-channel analysis will be successful, and the owner of the IP can be sure that his IP core is embedded in the analyzed device.

We have successfully implemented and demonstrated this approach on an FPGA. The results matched the results from the CDMA hardware Trojan which can be found in Fig. 6(a).

5.2.2 Input-modulated Side-Channel Watermark

The other way to implement a side-channel hardware watermark is adapted from the concept of the input-modulated hardware Trojan. An input-modulated watermark consists of a combination function, a leakage circuit, the unique watermark ID and some predictable internal state. The combination function and leakage circuit can be the same as the one from the input modulated Trojan from Section 4.4. Each byte of the unique ID is subsequently fed to the combination function together with an internal state that is predictable by the verifier. The internal state could, for example, be the register values of some registers that directly depend on the input to the IP core. The output of the combination function is then leaked out using the leakage circuit.

With the knowledge of the used combination function and internal state a verifier can check the watermark using a CPA in the same way the key was detected in the input-modulated Trojan. An attacker who does not know the combination function and the used input bits will not be able to detect this type of watermark as the signal is hidden well below the noise floor of the power consumption.

5.3 Robustness of Side-Channel Watermarks

Hardware watermarks are used to detect IP theft. Naturally, an attacker who tries to illegally use an IP core wants to stay undetected. Hence, he will try to remove the watermark or, to be more precise, make the detection of the watermark impossible. In the scientific literature the *robustness* of a watermark is used to describe how difficult it is for an attacker to render the watermark detection impossible. In this section we provide a short discussion of the robustness of the side-channel hardware watermarks.

We discuss two different approaches how an attacker can try to avoid the detection of the watermark:

- Increasing the noise level to deny a successful side-channel analysis
- Locating and modifying parts of the watermark circuitry to disable it

The used detection mechanism, a correlation power analysis (CPA), is very robust to noise. In side-channel attacks small differences in the power consumption can be exploited to mount a CPA. The hardware watermark benefits from this noise resistance as well. The attacker could try to add additional circuitry to decrease the signal-to-noise ratio (SNR) of the watermark. However, even a very small SNR allows for a successful side-channel analysis given enough measurements. To render the detection of the watermark impossible, a huge amounts of noise needs to be added. However, this would also significantly increase the power consumption of the chip and might furthermore result in a big area overhead to implement the noise sources. The level of resistance of side-channel analysis towards noise can be illustrated by the fact that even after many years of research there still does not exist a single side-channel resistant logic style that can defeat all kinds of DPA attacks.

The amount of leakage of the watermark is controlled by the selected leakage circuit. Hence, the amount of leakage of the watermark is part of the design space of the watermark owner and can be chosen so that the watermark can be detected with a reasonable amount of measurements, even in high-noise environments. In practice rendering the detection of the watermark infeasible by decreasing the SNR with additional circuitry can not be achieved.

Another approach the attacker might try is to modify the watermark circuitry itself. For example, if the attacker can change the combination function or the PRNG of the watermark, the owner of the watermark could not detect the watermark anymore. However, to do this the attacker needs to first locate the watermark circuitry and then he needs to be able to change this circuitry without causing any impact on the functionality of the protected IP design.

An attacker would first need to apply some reverse-engineering to locate the watermark circuitry, which is a very difficult and time consuming task in practice that also requires expensive equipment. If the attacker can use some automated search tools to look for suspicious parts that could be a watermark the attacker might be able to reduce the effort needed to reverse-engineer the watermark. For example an attacker might search the design for parts that can hold a watermark ID, such as fuse boxes. Therefore the owner of the watermark needs to make sure that their design is well hidden in between the surrounding logic. The difficulty to achieve this strongly depends on the size and type of design that is being watermarked. If the goal of the watermark is only to quickly detect the IP theft rather than proofing the theft in court, the watermark ID could be omitted, making the reverse engineering much harder.

Hardware watermarks are in general quite small and lightweight and only consist of a few gates. Thus, although reverse-engineering attacks are not impossible, it is safe to assume that the required reverse-engineering is a significant obstacle for the party that tries to steal the IP core. Often the costs of reverse-engineering are close to the costs of designing the system or the royalties required to legally use the IP. This makes IP core theft of watermarked designs uneconomical in most applications. Furthermore, the watermark is not visible to an attacker trying to steal an IP protected core. Hence, the attacker cannot even be sure whether or not a watermark is present and whether or not he has to circumvent any protection.

A great advantage of the side-channel watermark approach is its scalability. Multiple watermarks can be embedded within a single chip or even within a single design. Previously proposed watermarking schemes were not able to provide a menas that allows to independently watermark individual components of a bigger design.

Now IP core owners can tag each sold IP block with a watermark and can verify it without the requirement of having access to input values to the core to feed a specific trigger pattern. Furthermore, multiple watermarks within a single chip do not deny each others verification anymore. This is a major improvement to conventional approaches of hardware watermarks.

6 Conclusions

In this article we have given a comprehensive overview on how to use side channels constructively. Instead of just treating the side channels as parasitic side effect, they are considered as part of the available design space. In order to make use of them we discussed two possible applications: hardware Trojans and watermarks.

In the case of a Trojan side-channel as malicious circuitry, the designer of the hardware Trojan intends to bypass the protection against side-channel leakage of a chip-internal secret. His aim is to be able to extract this usually security sensitive secret while at the same time denying access to the implemented backdoor to everyone else.

We have revised two Trojan side-channel approaches, namely the CDMA and the Input-Modulated Trojan and suggested improvements to avoid the existing shortcomings of these first concepts. These approaches allow very small and subtle solutions, which do not require any external I/O communication to leak the secrets. By combining and improving these ideas we designed a sophisticated kind of Trojan side-channel, which is extremely hard to detect without mask

inspection and provides a backdoor to completely break a first-order side-channel protected AES implementation.

Finally, we have extended the Trojan side-channel principle to make use of specifically tailored side channels as watermarks. In this concept similar techniques as in the hardware Trojan case are employed not to leak out a secret key, but a unique identifier instead.

We presented a design, which is independent of the access to the device I/O pins and, at the same time, allows to stack multiple watermarks in a single device. Thus, this concept allows multiple watermarks of independent parties within a single chip. To the best of our knowledge this has not been achieved by any other watermarking scheme so far.

References

1. Cryptographic Hardware and Embedded Systems. `http://www.chesworkshop.org`.
2. Side-channel Attack Standard Evaluation Board (SASEBO). Further information are available via `http://www.rcis.aist.go.jp/special/SASEBO/index-en.html`.
3. Report of the defense science board task force on high performance microchip supply. Defense Science Board, US DoD, Februar 2005.
4. Innovation at risk: Intellectual property challenges and opportunities. white paper, Semiconductor Equipment and Materials International, June 2008.
5. S. Adee. The hunt for the kill switch. *IEEE Spectrum*, 45(5):34–39, 2008.
6. L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon. Mutual Information Analysis: a Comprehensive Study. *Journal of Cryptology*, 24(2):269–291, 2011.
7. G. T. Becker, M. Kasper, A. Moradi, and C. Paar. Side-channel based Watermarks for Integrated Circuits. In *HOST 2010*, pages 30–35. IEEE Computer Society, 2010.
8. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO 1997*, volume 1294 of *LNCS*, pages 513–525. Springer, 1997.
9. J. Blömer, J. Guajardo, and V. Krummel. Provably Secure Masking of AES. In *SAC 2004*, volume 3357 of *LNCS*, pages 69–83. Springer, 2004.
10. D. Canright and L. Batina. A Very Compact "Perfectly Masked" S-Box for AES. In *ACNS 2008*, volume 5037 of *LNCS*, pages 446–459. Springer, 2008. the corrected version is available at Cryptology ePrint Archive, Report 2009/011 `http://eprint.iacr.org/`.
11. Z. Chen, X. Guo, R. Nagesh, A. Reddy, M. Gora, and A. Maiti. Hardware Trojan Designs on BASYS FPGA Board. In *Embedded System Challenge Contest in Cyber Security Awareness Week - CSAW 2008*, 2008.
12. J.-S. Coron, P. C. Kocher, and D. Naccache. Statistics and Secret Leakage. In *FC 2000*, volume 1962 of *LNCS*, pages 157–173. Springer, 2000.
13. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In *CRYPTO*, volume 5157 of *LNCS*, pages 203–220. Springer, 2008.
14. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In *CHES 2008*, volume 5154 of *LNCS*, pages 426–442. Springer, 2008.
15. A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. Watermarking Techniques for Intellectual Property Protection. In *DAC 1998*, pages 776–781. ACM, 1998.
16. T. Kean, D. McLaren, and C. Marsh. Verifying the Authenticity of Chip Designs with the DesignTag System. In *HOST 2008*, pages 59–64. IEEE Computer Society, 2008.
17. F. Kiamilev and R. Hoover. Demonstration of Hardware Trojans, 2008.
18. S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and Implementing Malicious Hardware. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats - LEET 2008*, pages 1–8. USENIX Association, 2008.
19. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
20. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
21. J. Lieberman. National security aspects of the global migration of the u.s. semiconductor industry. white paper, Airland Subcommittee, US Senate Armed Services Committee, June 2003. `http://lieberman.senate.gov/documents/whitepapers/semiconductor.pdf`.
22. L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *CHES 2009*, volume 5747 of *LNCS*, pages 382–395. Springer, 2009.
23. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
24. T. Messerges, E. Dabbish, and R. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In *CHES 1999*, number 1717 in LNCS, page 144. Springer, 1999.
25. A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks - Extracting Keys from Xilinx Virtex-II FPGAs. In *the 18th ACM Conference on Computer and Communications Security - CCS 2011*. ACM, 2011. to appear. A draft version is available in Cryptology ePrint Archive, Report 2011/390. `http://eprint.iacr.org/`.
26. N. Narayan, R. D. Newbould, J. D. Carothers, J. J. Rodriguez, and W. T. Holman. IP Protection for VLSI Designs Via Watermarking of Routes. In *ASIC/SOC 2001*, pages 406–410. IEEE, 2001.
27. K. Nohl, D. Evans, Starbug, and H. Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, pages 185–194. USENIX Association, 2008.
28. A. L. Oliveira. Techniques for the Creation of Digital Watermarks in Sequentialcircuit Designs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 20(9):1101–1117, 2001.
29. D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *CHES 2011*, LNCS. Springer, 2011. to appear.
30. C. Paar. *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, Institute for Experimental Mathematics, University of Essen, Germany, 1994.
31. I. Torunoglu and E. Charbon. Watermarking-Based Copyright Protection of Sequential Functions. *IEEE Journal of Solid-State Circuits*, 35(3):434–440, 2000.
32. A. Young and M. Yung. The Dark Side of "Black-Box" Cryptography, or: Should We Trust Capstone? In *CRYPTO 1996*, volume 1109 of *LNCS*, pages 89–103. Springer, 1996.
33. D. Ziener and J. Teich. Power Signature Watermarking of IP Cores for FPGAs. *Signal Processing Systems*, 51(1):123–136, 2008.