TEL AVIV UNIVERSITY

THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING
The Zandman-Slaner Graduate School of Engineering

SECURE HARDWARE – PHYSICAL ATTACKS AND COUNTERMEASURES

By

Yossef Oren

THESIS SUBMITTED TO THE SENATE OF TEL-AVIV UNIVERSITY

in partial fulfillment of the requirements for the degree of

"DOCTOR OF PHILOSOPHY"

May 2013

TEL AVIV UNIVERSITY

THE IBY AND ALADAR FLEISCHMAN FACULTY OF ENGINEERING
The Zandman-Slaner Graduate School of Engineering

SECURE HARDWARE – PHYSICAL ATTACKS AND COUNTERMEASURES

By

Yossef Oren

THESIS SUBMITTED TO THE SENATE OF TEL-AVIV UNIVERSITY

in partial fulfillment of the requirements for the degree of

"DOCTOR OF PHILOSOPHY"

Under The Supervision of Prof. Avishai Wool
May 2013

Secure Hardware – Physical Attacks and Countermeasures

Dedicated to my wife, and to my children.

Contents

List of Figures

List of Tables

Abstract

Any cryptographic functionality, such as encryption or authentication, must be *implemented* in the real world before it can be put to practical use. This implementation typically takes the form of either a *software implementation* for a general-purpose device such as a personal computer, or as a dedicated *secure hardware device*, whose main purpose is to embody the cryptographic functionality. Examples of such secure hardware devices include smart cards, car alarm key fobs and computerized ballots. To evaluate the security of a cryptographic system, researchers look for flaws which allow an attacker to break the security assumptions of the system (for example, allowing an unauthorized party to view or modify a message intended for someone else). *Physical attacks* (also called *implementation attacks)* compromise the system by taking advantage of the physical aspects of the algorithm's implementation. Some physical attacks (such as, for example, power analysis) recover the secret key used by the secure device by analyzing physical effects produced during its use; Others (such as, for example, relay attacks) disable or otherwise limit its secure behaviour by exploiting design or implementation flaws or by changing the underlying assumptions made by the designers of the system.

This research focuses on physical attacks on secure hardware devices and on countermeasures which protect against these attacks. My goals were to investigate vulnerabilities in current secure hardware implementations and to evaluate the effectiveness of current and proposed countermeasures against these vulnerabilities. The two main tracks of my research are *side-channel analysis* (and explicitly power analysis) and *secure RFID.*

In the **side-channel analysis track,** I investigated ways of reducing the data requirements of power analysis attacks. We showed how to mount key recovery attacks on a secure device using an extremely low amount of measurement data. The main novelty of our attack was the use of a pseudo-Boolean optimizer – a powerful general-purpose constraint solving tool – to quickly and efficiently explore the huge space of possible solutions and find the correct one.

In the **secure RFID track**, I researched physical attacks on near-field RFID systems, specifically in relation to the proposed Israeli electronic voting scheme. The electronic characteristics of near-field RFID systems require a very short distance between the tag and the reader (typically less than 10cm) for the tag to operate. This leads to the (mistaken) implicit assumption that whenever a reader can communicate with a tag one can assume that the tag is physically very close to the reader. As described in [KW05], relay attacks challenge this underlying assump-

tion and allow a nearly unlimited distance between tag and reader. We describe how relay attacks and other lower-cost attacks can be used to target the reliability, anonymity and trustworthiness of the proposed electronic voting scheme, and what countermeasures can be considered to protect against these attacks.

The results described in this dissertation show how important it is to consider not only the algorithmic component of security, but also the physical aspects of building and breaking secure systems.

Introduction

Cryptographic theory is a mature and advanced field which offers theoretical solutions to many practical problems such as encrypting secure data, signing messages to ensure their authenticity and even allowing reliable and anonymous voting. Theory in itself, however, is not enough – any cryptographic functionality must be *implemented* in the real world before it can be put to practical use. This implementation typically takes the form of either a *software implementation* for a general-purpose device such as a personal computer, or as a dedicated *secure hardware device*, whose main purpose is to embody the cryptographic functionality. Examples of such secure hardware devices include smart cards, car alarm key fobs and computerized ballots. One special class of secure hardware device which has recently gained interest are *secure RFID tags* – a family of low-cost and low-power ubiquitous computers used for security applications such as access control, anti-counterfeiting and even voting.

To evaluate the security of a cryptographic system, researchers look for flaws which allow an attacker to break the security assumptions of the system (for example, allowing an unauthorized party to view or modify a message intended for someone else). *Cryptanalytic attacks* focus on the theoretical and algorithmic aspects of the system, while *physical attacks* (also called *implementation attacks)* compromise the system by taking advantage of the physical aspects of the algorithm's implementation. Some physical attacks (such as, for example, power analysis) recover the secret key used by the secure device by analyzing physical effects produced during its use; Others (such as, for example, relay attacks) disable or otherwise limit its secure behaviour by exploiting design or implementation flaws or by changing the underlying assumptions made by the designers of the system.

During my Ph.D. studies I was a part of Tel-Aviv University's Cryptography and Network Security Lab, where I worked under the supervision of Prof. Avishai Wool on the topic of *physical attacks and countermeasures in secure hardware devices*. My research was focused on physical attacks on secure hardware devices and on countermeasures which protect against these attacks. My goals were to investigate vulnerabilities in current secure hardware implementations and to evaluate the effectiveness of current and proposed countermeasures against these vulnerabilities. The two main tracks of my research are *side-channel analysis* (and explicitly power analysis) and *secure RFID*.

In the **secure RFID track**, I researched physical attacks on near-field RFID systems, specifically in relation to the proposed Israeli electronic voting scheme. The electronic characteristics of near-field RFID systems require a very short distance

between the tag and the reader (typically less than 10cm) for the tag to operate. This leads to the (mistaken) implicit assumption that whenever a reader can communicate with a tag one can assume that the tag is physically very close to the reader. As described in [KW05], relay attacks challenge this underlying assumption and allow a nearly unlimited distance between tag and reader. In two consecutive works published in IEEE RFID 2010[OW10b] and Smart SysTech 2012[OSW12] we described how relay attacks and other lower-cost attacks can be used to target the reliability, anonymity and trustworthiness of the proposed electronic voting scheme, and what countermeasures can be considered to protect against these attacks.

In the **side-channel analysis track,** I investigated ways of reducing the data requirements of power analysis attacks. Conventional power analysis methods rely on statistical relations between different power traces created by the device under test. The statistical structure of these attacks requires the attacker to obtain many power traces (typically thousands) before the attack succeeds, a quantity which may be impossible in many practical situations. In [OW] and in two works which appeared in the proceedings of CHES 2010[OKPW10] and CHES 2012[ORSW12], we showed how to mount key recovery attacks on a secure device using an extremely low amount of measurement data. The main novelty of our attack was the use of a pseudo-Boolean optimizer – a powerful general-purpose constraint solving tool – to quickly and efficiently explore the huge space of possible solutions and find the correct one. During the course of this research, which was performed in collaboration with researchers from Austria and Belgium, I gained experience in understanding the internals of cryptographic devices, as well as applied experience in signal processing and experience in launching and managing experiments involving multiple large-scale computations.

Document Structure

Part I discusses physical attacks on RFID tags, and possible countermeasures against these attacks. In chapter 1 I introduce the topic of physical attacks. In chapter 2 I describe the proposed Israeli e-Voting scheme. In chapter 3 and chapter 4 I show how the proposed system can be attacked using relay-based and non-relay based attacks. In chapter 5 I present experimental validation of our attacks. In chapter 6 I propose possible countermeasures for the attack, and the part concludes with some discussion in chapter 7.

Part II discusses low-data-complexity side-channel attacks. In chapter 8 I present an overview of the subject. In chapter 9 I describe the Tolerant Algebraic Side-Channel Attack (TASCA) methodology. In chapter 10 and chapter 11 I apply this methodology to attack the Keeloq and AES ciphers. The second part of the dissertation concludes in Part III with some discussion.

Finally, the dissertation ends with a general discussion and conclusions in Part III .

Part I.

Physical Attacks on RFID

1. Overview

In 2007-2008 the Interior Ministry of Israel started a process to design a new voting system, with a goal to transition from a traditional paper ballot system to an e-voting system. During 2007 the scheme was passed through countrywide pilot testing in several municipal elections in Israel [otISU07]. The system began the process of legal ratification (as of 2013 the process has not been completed) [Gov09]. The scheme is officially described in a patent claim recently granted to the Government of Israel by the World International Property Organization [ORM+10, Dol09], as well as by the public tender to contractors implementing the scheme [Gov08] and by the Israeli law governing the election process [Shi09]. Further information was provided by discussions with Yoram Oren, one of the co-developers of the scheme [Ore] (not related to the author).

The novelty of the system is that instead of using paper ballots, the votes in the proposed system are cast on contactless smartcards. To cast their votes, the voters use a computer terminal to write their choice into a contactless smartcard, and then physically deposit this smartcard into a ballot box. By encrypting the ballot as it is cast, the system aims to protect the privacy and authenticity of the votes, while still allowing the votes to be counted manually. For a more detailed description of the scheme see chapter 2. The designers of the Israeli e-voting scheme chose near-field contactless readers instead of traditional smartcards for non-security-related reasons. First and most important is the issue of cost and reliability – since a contactless smartcard reader has no mechanical interface and no moving parts (in contrast to a traditional smart card or magnetic-stripe reader), it can survive many more repeated uses with a reduced opportunity for damage or deliberate vandalism. In addition, as observed in [HMM09], contactless smartcards are easier to use than magnetic stripe cards or traditional smart cards since they work regardless of the way the card is oriented with respect to the reader. Cost saving is also reportedly the reason why the system has absolutely no paper trail – the designers wished to save on the cost of maintaining and supplying paper to thousands of printers on election day. The cards chosen for use in the Israeli scheme are Global Platform Java cards conforming to the ISO/IEC 14443 [ISO01] standard family.

1.1. Contributions

This work reports on a series of potential vulnerabilities of the proposed system. Some of these attacks are of the general category of relay attacks [KW05], which

use a pair of specially-located transceivers to arbitrarily extend the interrogation range of RFID tags beyond their nominal range. Another set of attacks works on a more fundamental level and do not require a full relay system to be built. On the basis of these attacks we argue that the proposed e-voting system was insecure and unusable.

We made a preliminary version of our report available to the Israeli Government in early 2010. On April 8, 2010 the government issued a formal response to our report [oFSU10]. The Interior Ministry noted that our attacks were only theoretical in nature and could not succeed in practice, mostly due to the differences between the common RFID tags attacked by previous works in the field and the high-security cards used in the proposed system. In this work we dismiss this claim by reporting on the actual implementation of two of these attacks: zapping and jamming'. We report on the range at which the attacks are possible in practice – we successfully implemented a jamming attack from more than 2m away, using power that can be supplied by a car battery.

The results reported in this part appeared in [OW10b] and [OSW12].

1.2. Related Work

An excellent survey on the history and state of the art on relay attacks has been written by Hancke et al. [HMM09]. As stated in the survey, the first published discussion of a relay attack is arguably the "chess grandmaster" attack, which Conway writes about in his famous book "On Numbers and Games"[Con76]. In the chess grandmaster attack, the adversary successfully plays simultaneously against two chess grandmasters by relaying their moves – even though he has no knowledge or understanding of chess. In 1987 Desmedt et al. [DGB87] wrote of the "mafia fraud" attack, the first discussion of a full-fledged relay attack used to compromise a security protocol (specifically, the Fiat-Shamir authentication protocol).

In 2005 Kfir and Wool [KW05] first noted that applying a relay attack to a near-field contactless system will cause the security of such a system to "collapse". They also used a detailed physical model of the physical layer to evaluate how the ghost and leech distances can be significantly improved beyond the nominal 10cm at a reasonable cost. They predicted that the leech range can be increased to 40 to 50cm and that the ghost range can be made as large as 50m. In that same year Hancke [Han06, Han05] demonstrated a working relay attack on an NFC system using a high-speed radio link, using standard equipment for the ghost and leech radio interfaces and a low-cost radio transciever for the relay link. This setup provided a relay range of about 50 meters. In 2006 Kirschenbum and Wool [KW06] actually built a functioning leech element of the relay by demonstrating a low-cost RFID skimmer that provides a leech range of 25cm, thus validating the model-based prediction of [KW05]. The article also explained how the leech range can be improved even further while staying within a very low budget. A possible way to protect against a

relay attack would be through the use of Faraday cage shielding, or by employing a hardware based distance-bounding protocol which strongly proves physical proximity (see [DM07]), but even these protocols cannot reliably detect a short-range relay attack.

In 2007 Drimer et al. [DM07] applied a relay attack to a contact-based smart card system implementing the high-security UK EMV payment system [EMV09]. One original aspect of this system was the design of the attacker's ghost and leech devices. Since the tag under attack was contact-based and designed to be used in a point-of-sale setting, the authors took special care to make the ghost look as much as possible like a standard EMV credit card. They achieved this feat by taking a genuine card and connecting a long cable to the gold pads interfacing the smardcard with the reader, with the other end of the cable wired to a hardware device implementing the relay functionality. This resulted in a card that looked authentic when viewed from the top side, but was actually connected to external hardware on the back side (see figure 2 in [DM07]). To make this attack undetectable in the field, the relay device and connecting cable could presumably be hidden up an attacker's sleeve. The leech device (which is supposed to receive communications from the victim's tag) was placed inside the box of a standard Chip & PIN terminal, resulting in a device that looks perfectly genuine to the victim.

1.3. Structure

In chapter 2 I describe the Israeli e-voting scheme and its security features in more detail. In chapter 3 I describe a series of proposed relay-based attacks, while in chapter 4 I describe some non-relay attacks. In chapter 5 I describe experimental validation of several of our proposed attacks. Finally, chapter 6 discusses some countermeasures against our attack, and the part concludes with some discussion in chapter 7.

2. The Israeli e-Voting Scheme

2.1. A Typical Magnetically Coupled RFID System

The proposed voting scheme is based on RFID technology. An RFID environment in general consists of a **reader**, a device connected to an external power source, communicating via a **wireless medium** with a multitude of inexpensive **tags**. While there are several classes of such tags, this discussion specifically deals with **passively powered, magnetically coupled** RFID tags (also referred to as **tokens**, **prox-cards** or **contactless smartcards**). These tags are used in new e-passports [EC06], credit cards [EMV09], public transportation [Mif03] and secure entrance systems.

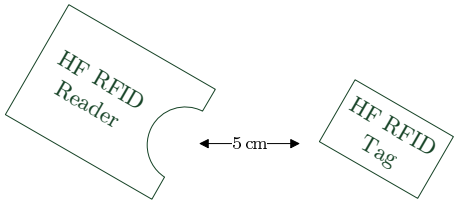


Figure 2.1.: A simple magnetically-coupled RFID channel

The cards chosen for use in the Israeli scheme, which is discussed in this chapter, are Global Platform Java cards conforming to the ISO/IEC 14443 [ISO01] standard family. These tags contain no power source and rely on the reader to provide them with operating power. Due to the physical principles behind magnetic coupling, the reader is only capable of delivering this power to the tag if the tag is near the reader's antenna. Standard ISO/IEC 14443 readers and tags are designed for a 5cm operating range, as illustrated in Fig. 2.1. This severe limit on the usable range of the system is sometimes considered erroneously as a security feature.

To provide for the case of multiple tags sharing the same air space (such as communication with one out of several contactless smartcards which are all located in the same wallet), the ISO/IEC 14443 standard specifies an **anticollision** protocol which allows each tag to be interrogated in turn without corrupting the communications with other tags. An in depth introduction to RFID can be found in [Fin03].

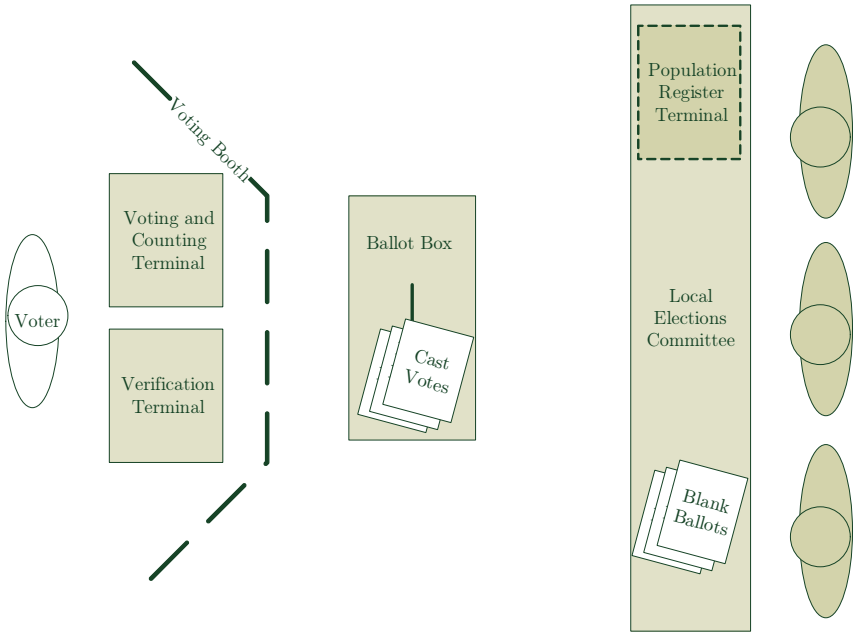


Figure 2.2.: Physical layout of the proposed Israeli e-voting scheme. Illustrated from left to right are the voting booth, the cast ballot box and the local election commitee's desk area.

2.2. Components of the Scheme

The components of a voting station are illustrated in Fig. 2.2. Each voting station consists of the following elements:

- A **voting terminal** (a portable computer with a contactless smartcard reader). The voter uses this terminal to cast his vote. The vote is recorded twice: First, the individual vote is written onto the blank ballot (a contactless RFID smart card). Second, the total vote count is immediately tallied and this total is written to a regular contact-based smart card plugged into the voting terminal.
- A **verification terminal** (another portable computer with a contactless smartcard reader). This terminal is only capable of reading (and not writing) ballots. The voter can optionally place his written ballot on this terminal to make sure his vote was correctly cast.
- A set of **blank ballots**, taking the form of secure contactless smart cards which are cryptographically paired with this specific instance of voting and verification terminals (see sec. 2.4).
- A **voting booth**, formed by folding a cardboard divider, that hides the voting and verification terminal from the elections committee and allows the voter to vote in privacy.

- A **ballot box,** where cast ballots (written contactless smartcards) are physically collected. The ballot boxes are typically made out of cardboard sheets, which are delivered flat and folded into shape on election day.
- The **local elections committee**, which typically consists of three mutually distrustful (and technologically inexperienced) members nominated by the parties participating in the elections. These local commitees in turn report to a central elections committee which oversees the election process. The committee uses a **population register terminal** to verify that each voter accessing the voting station is eligible to vote and has not voted before.

The election body has a very limited control over the physical environment in which the system is deployed – typically the voting station is located in a classroom or another location which is freely accessible to the adversary before election day. Furthermore, it is quite likely that one or more members of the local elections committee are in the service of one of the candidates (since the candidate parties nominate the members of these committees).

2.3. The Proposed Voting Process

The actual voting process is illustrated in Fig. 2.3, and is carried out in the following way:

1. The **voter** approaches the **elections committee**. The committee members verify his eligibility to vote, take his ID card (which is mandatory in Israel) and provide him with a **blank ballot** (contactless smartcard).
2. The voter enters the **voting booth**, where his actions cannot be seen by the committee. He next places his blank ballot on the reader connected to the **voting terminal**. He selects the vote he would like to cast via a touchscreen interface. Once the voter is satisfied with his vote it is written electronically to his ballot. The running total count of votes is also written to a (non-contactless) smart card embedded in the voting machine. The voter is allowed to change his mind and update his vote multiple times, with both the voting terminal's internal smart card and the ballot tracking the latest choice in real time.
3. If the voter wishes to convince himself that the vote he has selected was correctly recorded on his ballot, he may place the ballot on a reader connected to the **verification terminal**, which simply displays his selected vote. As noted in sec. 2.4, the verification terminal is only capable of reading votes cast in this particular voting station. The ballot is supposed to be cryptographically secured and its contents cannot be read back by any other way.
4. The voter, now satisfied with his vote, drops his written contactless smartcard into the ballot box, in plain sight of the voting committee.

5. After witnessing the voting process, the elections committee returns the voter's ID card to the voter.
6. At the end of the day, the local elections committee retrieves the contact-based smart card from the voting terminal and delivers it to the central elections committee, where the cards collected countrywide are tallied to calculate the nationwide preliminary election results. These results are formed by adding together the running totals stored on the smart cards embedded in the voting terminals.
7. After the preliminary results are announced, the local elections committee manually counts all votes found inside the ballot box by passing them one by one through the verification terminal. As specified in [Shi09], the final manual count takes precendence over the preliminary computer-counted results. If the "hand-counted" votes and the computer-counted votes disagree by more than 5 votes, the ballot is "flagged" and must be examined for tampering before being accepted in the total count. If the two tallies differ by more than 30%, all votes in this voting station are immediately ruled invalid. Any subsequent vote recounts, if desired, are performed against the smartcards inside the ballot box.

Note that the security properties of population register terminal are not covered in this report.

2.4. Security Features of the Scheme

The Israeli e-Voting Scheme was designed with a certain emphasis on security. The Global Platform Java cards used by the system conform to Common Criteria EAL 4+ [Com09] and are used in other high-security applications such as e-commerce and access control. The voting and verification terminals are cryptographically paired with the blank ballots used in each specific station, meaning that (at least as designed) a ballot cannot be read from or written to outside its specific voting terminal[1]. This means an attacker cannot steal a voting terminal from one voting station and use it to his advantage in another station. The voting terminals have no online connection either – the identity of the voter is only verified by using the population register terminal used by the voting committee and is not recorded on the ballots.

The redundancy in the vote counting process offers another degree of security, since the voting tallies which are written to the secure smart card inside the voting terminal must match the count of votes in the ballot box. Thus, an attacker would theoretically need to subvert both locations before compromising the election results.

[1] According to the proposed design, even the government's "master key" is incapable of rewriting a ballot. It can only format the contactless smart card to a blank state

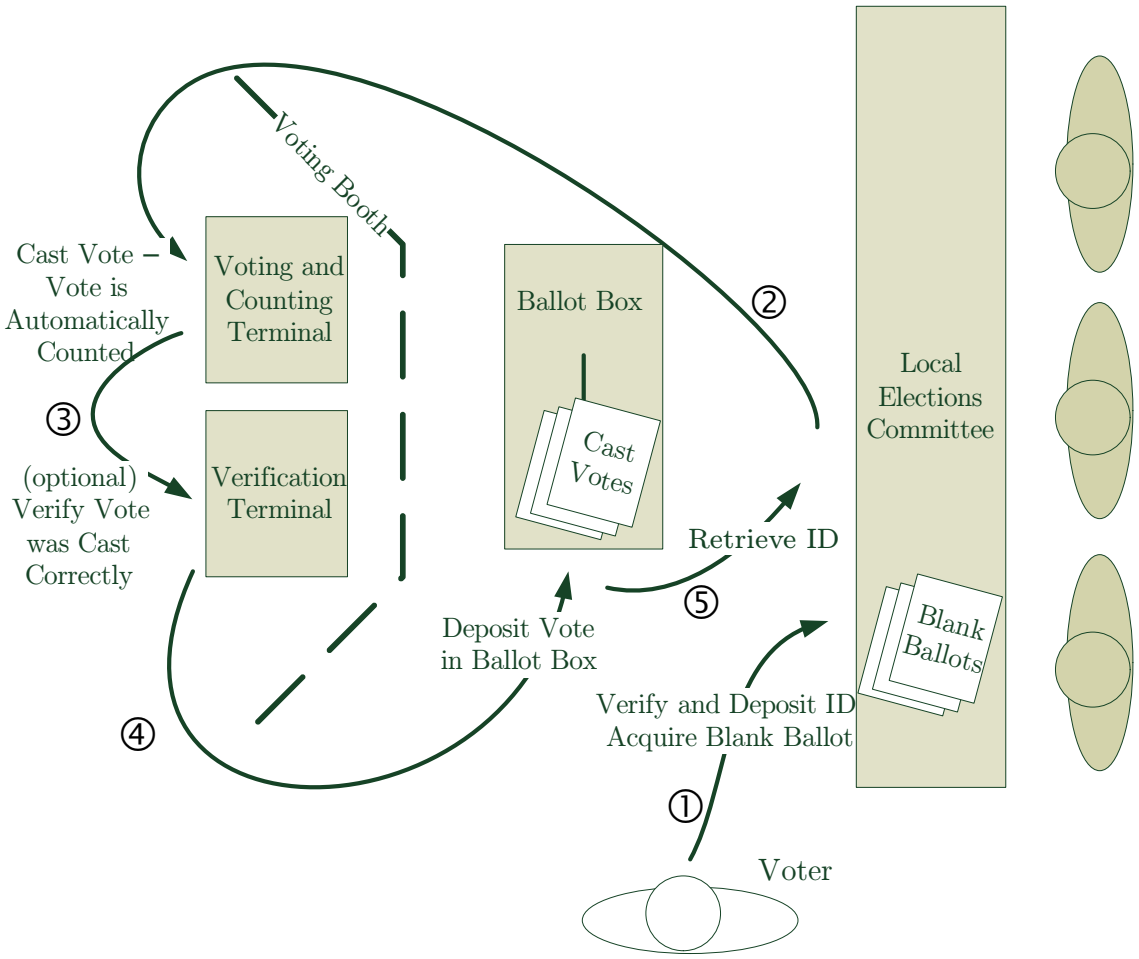


Figure 2.3.: The proposed Israeli e-voting scheme in action. Illustrated from left to right are the voting booth, the cast ballot box and the local election commitee's desk area. The arrows show the path followed by a voter through the three areas of the voting station.

We must note that the specific protocol chosen to form the interface between tag and reader is of no interest to us. Indeed, as shown in [DM07], even the most advanced distance bounding protocols cannot defend against a properly implemented relay over a distance of less than 6 meters.

The designers of the Israeli e-voting scheme chose near-field contactless readers instead of traditional smartcards for non-security-related reasons. First and most important is the issue of cost and reliability – since a contactless smartcard reader has no mechanical interface and no moving parts (in contrast to a traditional smart card or magnetic-stripe reader), it can survive many more repeated uses with a reduced opportunity for damage or deliberate vandalism. In addition, as observed in [HMM09], contactless smartcards are easier to use than magnetic stripe cards or traditional smart cards since they work regardless of the way the card is oriented with

respect to the reader. Cost saving is also reportedly the reason why the system has absolutely no paper trail – the designers wished to save on the cost of maintaining and supplying paper to thousands of printers on election day.

Note that this system has been criticized by many (cf. [RTS, Wei09]), specifically for its lack of a paper trail, but also for its cost and possible discriminatory nature against the technologically challenged such as the poor, uneducated or elderly voters. In this work our goal is to provide a purely technical critique that is based on the suspectibility of the system to physical layer attacks and to relay attacks in particular.

3. Relay Attacks

As mentioned before, the fact that legitimate contactless smartcards have a limited read range should not be considered a security feature. Specifically, the fact that a certain tag is communicating with a certain reader does not imply with any certainty that the tag and reader are in proximity. In our attacks, we use this flaw to create a communications link from the voting and verification terminals inside the voting booth to the RFID-based ballots inside the ballot box that carry votes which were already cast.

While relay attacks have been discussed in other contexts, this is the first time they are considered within a voting system. One specific aspect of the voting process makes relay attacks even more effective in this context. In conventional relay attacks the attacker's tag (or ghost device) is generally used in a legitimate point-of-sale or conditional entry scenario, and as such must look very similar to an authentic tag to avoid suspicion. Fortunately for our attacker, this limitation on the physical form of the ghost does not exist when attacking the Israeli voting system – when performing relay attacks on the election terminals, the attacker is hidden inside the booth and is promised a high degree of privacy by the very nature of the voting process itself. This means that the ghost end of the attacker's relay setup can take an arbitrary form. The attacker may even use his time in the voting booth to replace the entire ensemble of voting and verification terminals with hacked machines running his own code (as the authors of [DM07] did when attacking EMV systems). The reader end (or leech device) can likewise be given an arbitrary shape, if we make the reasonable assumption that the voter can approach the voting commitee carrying a backpack or bag and ask them to mind his bag while he votes, or even consider the possibility that one of the three members of the elections committee is collaborating with the adversary.

3.1. The Theory Behind Relay Attacks

As stated in [Fin03], most contactless smart card systems (and specifically the ISO 14443 [ISO01] family) operate on the physical principle of **magnetic coupling**. The nominal range of such a magnetically coupled channel is very low – typically 5cm or less. This choice of **air interface**, which is used both to provide power to the smartcard and to communicate with it wirelessly, is supposed to impose a very severe limit on the distance between the reader and the tag. This leads to the

(mistaken) implicit assumption that whenever a reader can communicate with a tag one can assume that the tag is physically very close to the reader.

As described in [KW05], relay attacks challenge this underlying assumption and allow a nearly unlimited distance between tag and reader. The attack achieves this ability by placing a **relay**, consisting of custom-designed tag and reader hardware connected by a high-range communication link, between the victim's tag and reader.

As illustrated in Fig. 3.1, the **leech** device (or proxy-reader) presents itself to the **victim tag** as a legitimate reader, while the **ghost** device (or proxy-token), presents itself to the **victim reader** as a legitimate tag. When the victim tag and reader wish to exchange data, their messages are captured by the relay devices and sent across the fast high-range communication link. Thus, any data exchange between the victim's tag and reader can be carried out over the relay channel, even if it is strongly encrypted and authenticated. Such a relay attack on a credit card system, for instance, would allow the attacker to make a purchase and pay using an unsuspecting victim's credit card. The attacker will present the ghost device to the point of sale system, while the leech would be placed near the victim's wallet – perhaps in a totally different location.

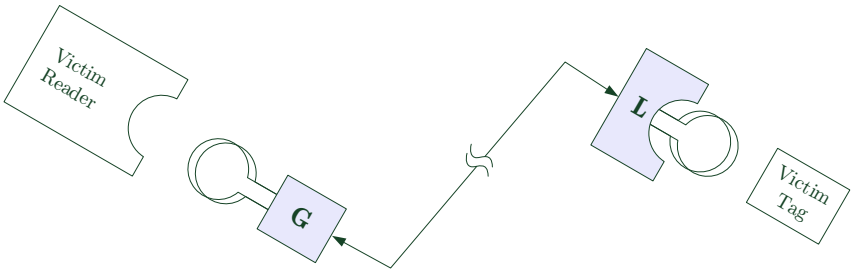


Figure 3.1.: An RFID channel under a relay attack. Device "L" is the leech, while device "G" is the ghost.

Since the attacker' ghost and leech hardware are custom manufactured and do not need to comply with any regulatory standard, the adversary has a large degree of freedom when designing his system. The adversary can specifically design his ghost and leech so that they have an internal power supply, a different form factor or (most importantly) a higher operating range.

Thus, there are three ways in which relay systems increase the distance between the victim tag and victim reader:

1. Increasing the range between the **victim tag** and the **leech (leech range)**
2. Increasing the range of the communications link between the **leech** and the **ghost (relay range)**
3. Increasing the range between the **ghost** and the **victim reader (ghost range)**

We will now outline several relay attack scenarios that are possible against the proposed Israeli voting system. This is not an exhaustive list: there may be other scenarios which we did not consider.

3.2. The Ballot Sniffing Attack

This attack, as illustrated in Fig. 3.2, allows an adversary to learn the complete contents of all votes already cast into the ballot box.

3.2.1. Attack Description

To carry this attack, a relay is set up between the votes inside the ballot box and the verification terminal inside the voting booth. Next, the adversary repeatedly activates the verification terminal, each time with a different ballot. Since the verification terminal is paired with all cast votes in the ballot box, the adversary can now enumerate all votes inside the ballot box and display the votes recorded on them. This process can be carried out reliably and efficiently even if there are many votes already in the ballot box, as long as the "leech" component of the relay uses the RFID anticollision protocol to activate the tags one at a time[1].

3.2.2. Implications

Using this attack, an adversary can determine the partial results of the vote before the voting day is over. If the votes in a certain ballot do not fit the outcome expected by the adversary, he may use this data to decide to disqualify the entire voting terminal using other methods described below. More importantly, if this attack is carried out twice in one day on the same ballot box it will teach the adversary about the votes cast in the interval between the two attacks. Thus, this attack can be used to verify that a voter had voted a certain way, violating voter privacy and allowing coercion to be inroduced into the voting process.

3.3. The Single Dissident Attack

This attack, as illustrated in Fig. 3.3, lets the adversary suppress nearly all undesirable votes in a certain voting station while remaining virtually undetectable.

[1] According to [Ore], even though the ISO 14443 anticollision protocol was officially designed to discriminate between up to 2^{32} tags, it has problems in practice when dealing with more than 3 tags at the same time. If indeed this is the case, this may limit the effectiveness of the ballot sniffing attack.

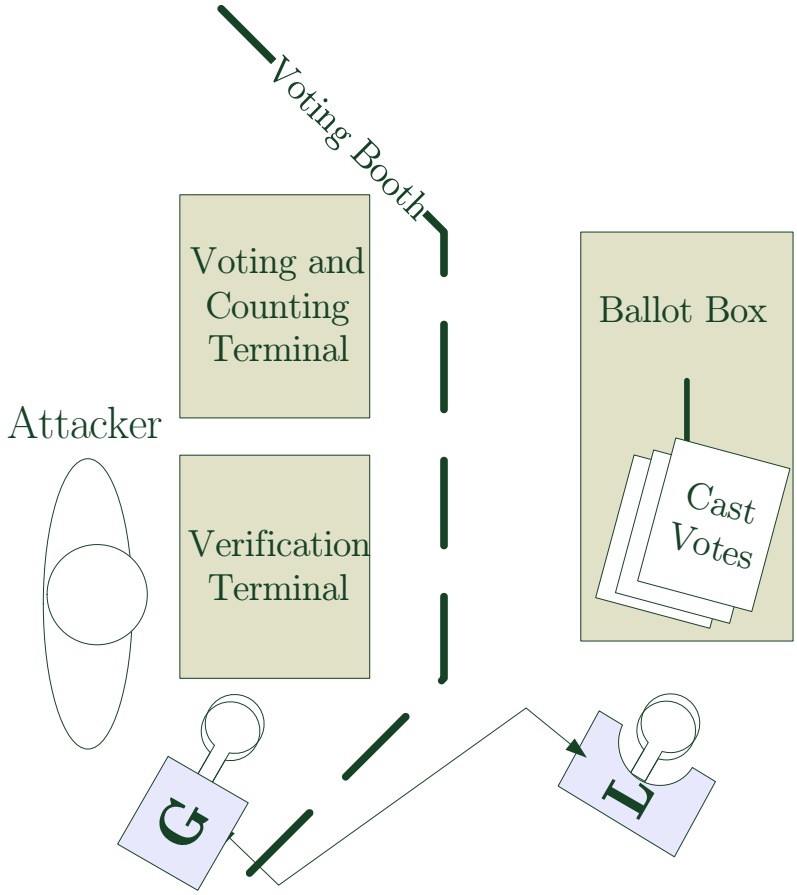


Figure 3.2.: Ballot Sniffing Attack

3.3.1. Attack Description

To carry out this attack, the adversary needs to be present near the voting terminal while another voter is voting (for example, the adversary may volunteer to serve on this specific election committee or simply stand outside). As a prerequisite to this attack, the adversary sets up a relay between the voting and verification terminals and a single previously cast vote located inside the ballot box, hereby called the "single dissident vote". This relay can be turned on and off as desired by the adversary. The adversary then selects a certain group of voters whose voice he wishes to suppress[2]. In general, the relays will operate only when one of these "dissidents" is inside the voting booth. As stated previously, an attacker can easily obtain such fine-grained control over the voting and verification systems if he replaces both terminals inside the voting booth with similar-looking computers running his

[2]The attack builds on the assumption that the the adversary can guess the vote of a some voters based on their external characteristics or some other auxiliary information. This is very reasonable in general, and even more so in Israel given the highly heterogenous nature of the Israeli voting body.

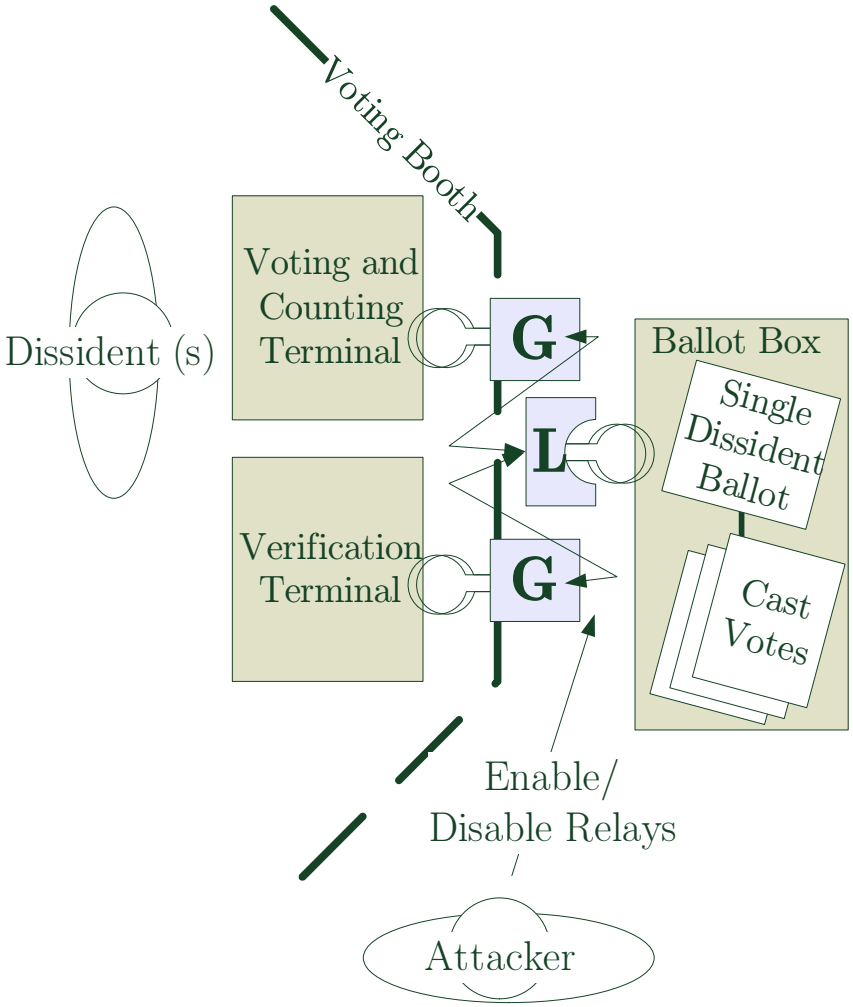


Figure 3.3.: Single Dissident Attack

own code (see [DM07]).

When the relay attack is active, all of the dissenting voter's actions (voting, verifying, etc.) are not performed against the blank ballot he is holding, but rather shunted to the single dissident vote already in the ballot box. Since this single ballot is properly authenticated, both the voting and the verification terminals happily register the vote and display its correct value. Thus, the voter has no idea that the attack is taking place, but his personal ballot always stays blank. After the voter exits the booth, he casts his blank vote into the ballot box and his vote is effectively disqualified and ignored.

3.3.2. Implications

If the adversary manages to correctly guess the disposition of the voter, this attack makes sure the ballot box contains no more than one dissenting vote for each undesirable party. Because of the relay attack all other dissenting voters will register as blank, or invalid, votes, and have no effect on the outcome of the elections. Since the votes inside the ballot box correlate directly with the values written to the smart card inside the voting terminal, the recount at the end of the voting day will find no discrepancy between the two. However, there is only one vote left to represent any amount of dissenting voters.

While this attack seems at first difficult to set up and carry out, it has the advantage of being virtually undetectable. If one of the dissenting voters gets suspicious and tries to find traces of this attack, even manual examination of the per-box vote counts will not reveal the subterfuge in this case, since each voter can be certain only of his personal vote, and at least one dissenting vote has been registered in this ballot. Thus, each dissident will be forced to conclude that his accomplices had a change of heart at the ballot, and that he is indeed the only one to have placed a dissenting vote in this specific ballot box.

If the attacker is somewhat uncertain of his ability to guess how a voter will behave inside the booth, he can use not one but several dissident votes and map the dissenting voters at random into one of this small set whenever one of them enters the booth. Since only a single vote for each party needs to be registered at the voting terminal for the attack to be undetectable, the probability of failure can be significantly reduced by choosing an appropriate size for this group.

Note that according to [Ore] the voting terminal can somehow keep track of which ballots it wrote to, and refuse to connect to previously written tags as soon as it encounters a new tag. We cannot comment on this mechanism since the documentation given in [ORM+10] does not specify exacly what identifier is recorded and how the mechanism should work. Indeed, it seems difficult to provide this functionality without exposing the system to voter privacy violation or to denial of service attacks. Nontheless, such a mechanism may limit the effectiveness of the single dissident attack.

3.4. The Ballot Stuffing Attack

This attack, as illustrated in Fig. 3.3, gives an adversary complete control over previously cast votes, using a relay attack to rewrite them to the candidate of his choice.

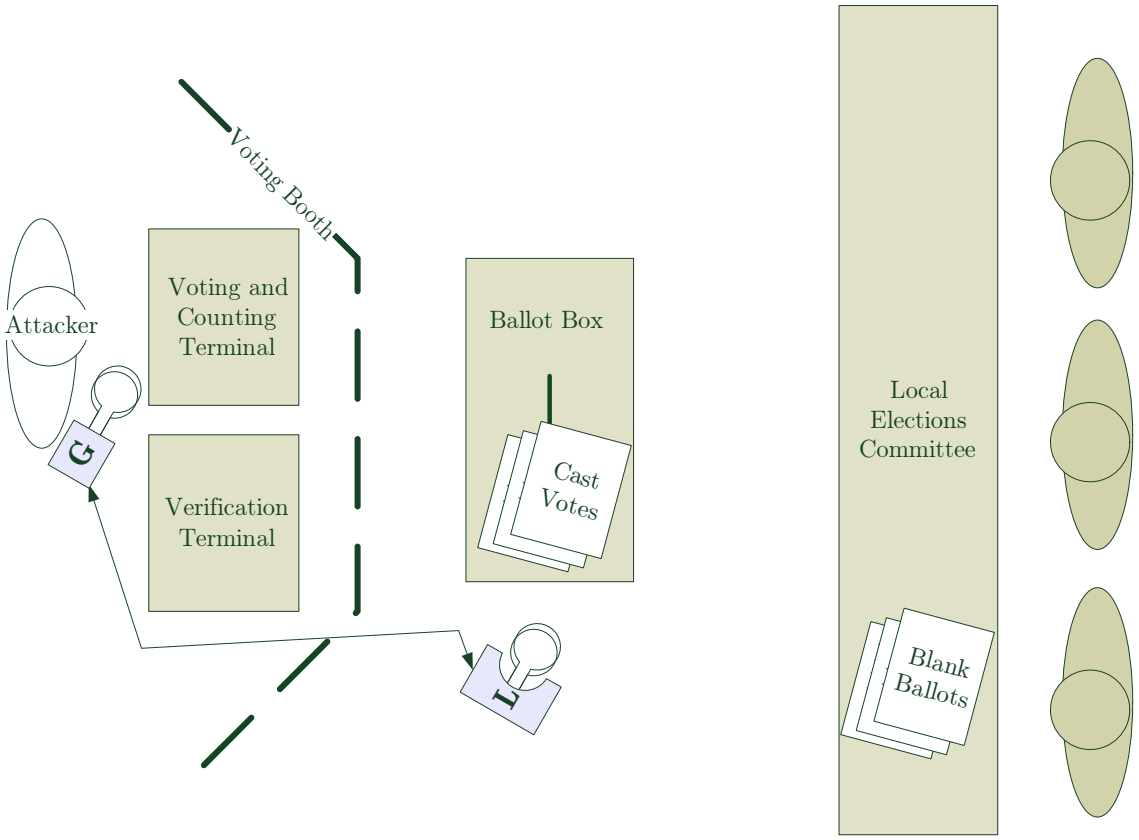


Figure 3.4.: Ballot Stuffing Attack

3.4.1. Attack Description

In this attack the relay is set up between the ballot box and the voting terminal. The adversary then enters the voting booth and repeatedly votes for the candidate of his choice, each time engaging with a different ballot selected from the ballots already inside the ballot box. Since the voting terminal is paired with the previously cast votes, it has no reason to prevent the votes from being modified. Since the voting terminal itself is used to perform the attack, it constantly updates its running vote counts and thus remains in perfect sync with the cast votes instead the ballot box[3]. This means that this attack causes no discrepancy which can be detected during a recount. The RFID anticollision protocol allows this process to be carried out reliably and efficiently even if there are many votes already in the ballot box. The ability to rewrite ballots multiple times is a **confirmed feature** of the system which is stated in the voting law [Shi09].

[3]We assume that the voting terminal handles a voter's vote changes by rewriting the ballot, subtracting one vote from the old choice and adding one vote to the new choice on the internal smartcard, and that there is no "finalizing" step which locks the ballot from subsequent edits. This interpretation was validated by [Ore].

3.4.2. Implications

This attack is the most conspicouous of all attacks presented here, but it is the most powerful, since it allows the entire set of ballots to be rewritten arbitrarily. As noted in the previous subsection, the only sort of fraud that is detectable by mutually distrustful voters would be the case when a party the voter had personally voted for ends up with zero votes in this ballot. If the adversary had previously applied the ballot sniffing attack to read all votes in the ballot, he can avoid this form of detection by registering one vote for each party originally represented in the ballot under attack and giving all other votes to his candidate of choice. The lack of paper trail means that this attack is undetectable and unprovable, even though it has the potential to arouse voter suspicion if used crudely. Note that the countermeasures sketched in the two previous subsections affect this attack as well.

4. Non-Relay Attacks

The relay attacks described in the previous section require some sophistication and a minor budget from the attacker. In addition to these attacks, the RFID technology used in the proposed voting system is also susceptible to simpler hardware-based attacks.

4.1. The Zapper Attack

This relatively low-tech attack, illustrated in Fig. 4.1, can quickly and easily disqualify a certain ballot box, allowing an adversary coarse-grained control over the results of the election.

4.1.1. Attack Description

This attack assumes that the adversary had decided (based on apriori demographic knowledge or on information gained by the ballot sniffing attack described in sec. 3.2) that the votes cast by the entire population of a single ballot box are not to his liking. The adversary can make use of a low-cost "RFID Zapper" device [MM05], consisting of a pulsed power source (most conveniently, the flash circuit from a disposable camera) connected to a properly shaped reader antenna. When the pulsed power source (e.g. flash camera) is activated, a powerful electromagnetic pulse flows through the antenna and "zaps" any RFID tag close enough to couple magnetically with this antenna with a powerful surge of current which overwhelms its input circuits and generally renders it unusable. A live video demonstration of such a device being used to disable an ISO/IEC 14443 tag can be found in [Run05, starting at 19:00]. Once 30% of the tags inside the ballot box are "zapped", this will cause a discrepancy between the count of ballots held by the smartcard located inside the voting terminal and those counted manually by the elections committee, causing the entire ballot to be legally disqualified.

Zapping attacks against a variety of ISO/IEC 14443-compatible HF tags were experimenatally verified by several research groups [Run07, N3l06]. The zapped tags included low-cost inventory tags as well as high-security tags similar to the ones used in the Israeli elections system[1].

[1] Specifically, an admission ticket to the 2006 FIFA world cup.

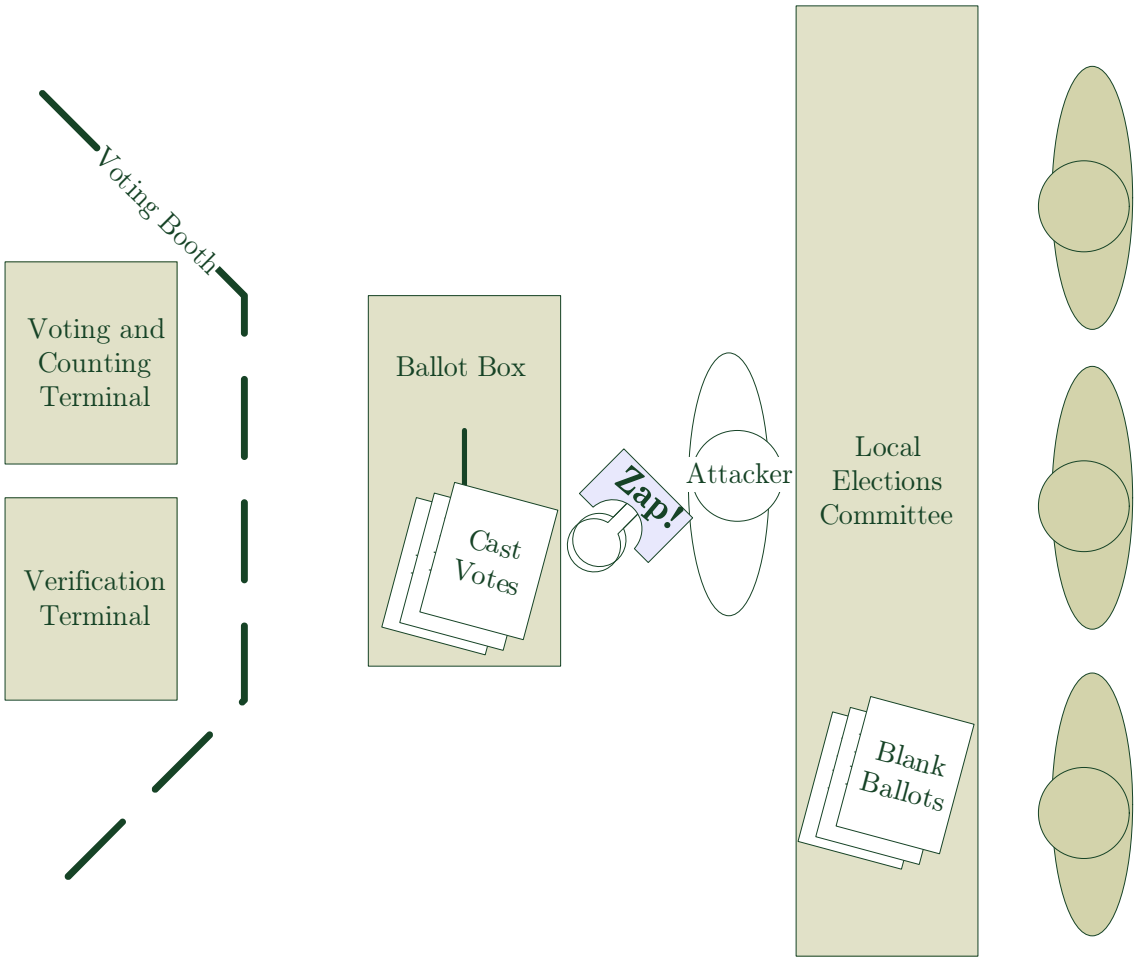


Figure 4.1.: Zapper Attack

4.1.2. Implications

This attack allows anybody with access to a ballot box to quickly and undetectably disqualify its contents and remove it from the election process. The ballot box can be zapped during the election process or later, while the ballots are kept pending the manual recount. In a particularly clever twist on this attack, a small group of crafty adversaries can zap their *own* cards before depositing them in the ballot box, thus immediately disqualifying the entire ballot (of course, the privacy features inherent to the voting process will prevent these individuals from being tracked or caught). The zapping attack is so easy and inexpensive to carry out that it may be even done for reasons of malice or protest, and not just for changing the election results.

4.2. Jamming, Blocking and Selective Denial of Service

The jamming attacks illustrated in Fig. 4.2 can disrupt the operation of the RFID reader at a distance. They can be turned on and off on demand.

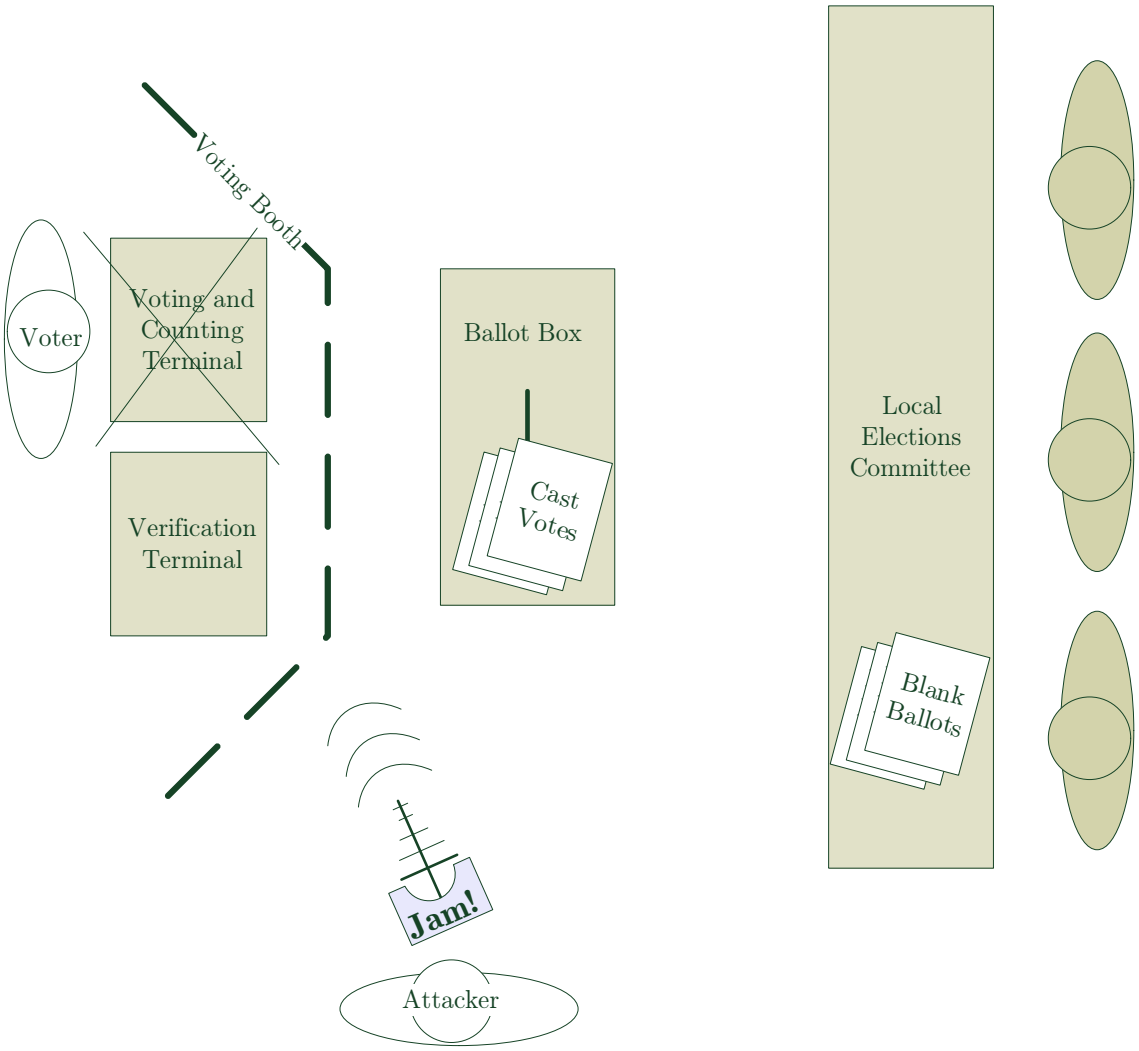


Figure 4.2.: Jamming Attack

4.2.1. Attack Description

While the physical interface used by the tag is based on magnetic coupling, most RFID readers actually use a standard radio decoding technique called single sideband (SSB) demodulation to decode the tag responses. This property allows the ghost component of a relay to work at distances of up to 50 meters, but it also

means that the reader's standard operation can be disrupted by SSB modulated signals sent to it from a distant radio antenna. To carry out this attack the attacker uses a directed antenna to transmit a random SSB-modulated bit pattern toward the reader at the tag's upper side-band frequency of 14.4075 MHz. The reader connected to the voting terminal will not be able to distinguish between this disruptive signal and the tag's response, rendering the reader inoperative and creating an effective denial of service attack. According to [KW05] the operational range of such an attack can be up to 50 meters.

A similar attack can be performed by hiding a "Blocker Tag" [JRS03] somewhere within the voting booth (for example, between the voting terminal and the table on which it is set). This tag is a special hardware device which conforms only partially to the RFID specification – specifically, it ignores the tag anti-collision protocol under certain conditions, disrupting tag and reader communications. The blocker tag presented in [JRS03] prevents RFID readers from communicating with tags whose ID matches certain criteria while allowing all other communications. The design can be modified to support a high-range remote control interface allowing blocking to be turned on and off on demand, creating another hard to detect denial of service system.

4.2.2. Implications

The jamming attack is a **selective denial of service attack**, since it is easy to apply selectively only to a certain subset of voters at the discretion of the attacker. This attack is also unique in its very high operating range. It cannot be prevented unless electromagnetic shielding is applied to the walls, doors and windows of every voting station (and not just the ballot box itself).

4.3. Fault Attacks

As demonstrated in [HSP08], HF RFID tags are highly susceptible to Optical Beam Induced Current (OBIC) attacks, which belong to the general category of of *fault attack*. When subjected to this attack, the tags' writing mechanism becomes completely unreliable. In the case of HF RFID tags, this attack is not at all invasive or expensive, requiring no manipulation of the tag other than shining a high-power LED at the its chip area. Such a LED can be easily hid by the adversary inside the voting terminal. Quoting from [HSP08], the authors show that "RFID tags are exceedingly vulnerable to faults during the writing of data that is stored into the internal memory [...] it is possible to prevent the writing of this data as well as to allow the writing of faulty values. In both cases, tags confirm the operation to be successful."

Such an attack can have a devastating effect on the described voting system, since a failed write which is reported to the voting terminal as successful immediately translates to a discrepancy between the voting station's tally and the final manual count, causing in turn the entire voting station to be disqualified during the manual recount phase.

4.4. Faulty Implementation Attacks

As stated in sec. 2.4, the Israeli voting scheme has certain security properties built in by design, most significantly the cryptographic pairing of blank ballots and voting terminals. This pairing means that a compromised terminal stolen from one voting station cannot be used to attack another station. Throughout this article we assumed that this security feature was properly implemented. A proper implementation would mean that both terminals and blank ballot authenticate each other via public key, they execute a properly randomized key exchange protocol resulting in the generation of a random session key, and all messages sent under this session key are properly randomized, padded, encrypted and MACed to prevent replay or message modification attacks. However, implementation flaws in any part of the cryptographic pairing process immediately make the system even more vulnerable. Here are some examples:

1. If the authentication process consists of a highly secure and authenticated "unlocking" phase followed by a less secured exchange of commands with the "unlocked" tag, the attacker can wait until a tag is legitimately unlocked and then take control of the conversation and send his own commands to the unlocked tag, allowing him to influence the voting results or disqualify an entire voting terminal. This mode of operation is actually common in several low-security RFID protocols, such as the EPC Gen-2 memory write command[C1G05].
2. If encryption is not randomized, then there would only be as many possible encrypted messages as the number of parties in the elections – well under 100 values in Israel.
3. If the messages are not padded to a fixed length and the plaintext includes the full name of the selected party, then the encrypted message length divulges information about the encrypted content.

5. Experimental Validation

In the previous subsections we described several attacks on the proposed system. If the attacker is in possession of a **relay device** [KW05], he can mount a **ballot sniffing attack** (which allows him to learn at any time which votes were already cast into the ballot box), a **single dissident attack** (which can undetectably suppress the votes or any amount of voters), and finally a **ballot stuffing attack** (which gives the adversary complete control over previously cast votes). If the attacker does not use a relay he can mount a **zapping attack** (which can quickly and easily disqualify an entire ballot box), a **jamming attack** (which can disrupt the operation of the voting station at a distance), or a **fault attack** (which can cause the voting station to enter an unpredictable state and thus disqualify it).

In this subsection we report on actual implementations of two of the above attacks: the **zapping attack** and the **jamming attack**.

5.1. System Design

5.1.1. RFID zapping

5.1.1.1. Description

RFID zapping is a well known attack, having previously been demonstrated in several places, including the 25th Chaos Communications Congress [Run05]. As stated in [Run07], the RFID zapper attack is built to attack the RF front-end of RFID tags. To carry out this attack, the adversary sends a short high-power pulse through an antenna placed next to the tag under attack. Because of the coupling between the zapper and tag antenna, this causes a high-power pulse to flow through the tag's antenna. This pulse causes the RF power harvesting system of the tag to be overwhelmed, permanently disabling the tag. This attack is particularly effective against passively-powered tags, since their only power source is the RF power harvester. The overall energy used in the attack is not very large if the high-power pulse is made short enough, allowing this attack to be carried out using inexpensive and portable components – one particularly common configuration is to reuse a disposable film camera, replacing the flash bulb with an appropriate antenna and pressing the camera shutter to activate the attack.

5.1.1.2. Attack setup

Figure 5.1.: The Zapper, shown next to an Israeli e-voting card

The attack setup is illustrated in Fig. 5.1. For our attack we followed the recommendation of [Run07] and purchased a disposable film camera with built-in flash. The total price of the attack was 40 NIS (about 8 Euros) for 3 cameras. We removed the flash bulb and replaced it with a hand-made PCB antenna, with the same size and geometry as the RFID tag under attack. The camera is powered by a single 1.5 battery, which is used to charge a 68 μF electrolytic capacitor to a voltage of approximately 250 V. This battery can supply enough power for dozens of flash activations.

We used this zapper to attack a high-security ISO/IEC 14443 [ISO01] tag provided to us by a contractor of the Israeli Ministry of the Interior. To carry out the attack, we first verified that the tag works properly by placing it on a standard ACR122 NFC reader [Ltd08] connected to a PC. We then placed the tag next to the zapper and activated the zapper once. Next, we placed the zapper tag on the reader to verify that it cannot be read any more. A video demonstration of the attack can be found online [OW10a].

5.1.1.3. Results and Discussion

As our video demonstration shows, the zapper attack was completely capable of disabling the high-security tag in the proposed system. Evidently, the increased ESD protection and other countermeasures which exist in the high-cost EAL 4+ cards used in the system was not sufficient to prevent the zapping attack from being carried out. We note that it is quite simple to build zappers which are even more powerful than the one we constructed, for example by replacing the camera's built-in capacitor with a higher-capacity element.

5.1.2. RFID jamming

5.1.2.1. Description

An adversary who wishes to disturb the normal course of the elections in a certain ballot station can synthesize a jamming signal, thus preventing the RFID reader from communicating with the tag and recording the votes. The signal can be transmitted from outside the room, and can be turned on and off at the adversary demand. This way the attacker can create a denial of service attack at will, depending on the people currently entering to vote.

In order to block the communication between the reader and the tag there is a need to transmit a signal which mimics the load modulation of a tag, thus preventing the reader from receiving the tag's reflected signal. In [JRS03] and [RCT07] the authors implemented RFID blockers by building an active tag emulator which transmitted a fake UID in order to interfere with the anti-collision algorithm of ISO14443. We suggest a more straightforward method of transmitting a powerful signal on the sub-carrier used for the tag load modulation.

An ISO14443 tag transmits its response using load modulation on a sub-carrier of the reader's carrier signal (13.56 MHz). The sub-carrier frequency $\frac{f_c}{16} = 848$ kHz produces side bands at 12.712 MHz and 14.408 MHz. The two sidebands function both as carriers for the tag's data, and are basically the same. According to [FPB11] a typical ISO14443 compliant reader evaluates only the upper side band. Therefore, in order to block the signal from the tag it suffices to transmit a powerful signal on the upper side band (14.408 MHz).

Blocking of the signal can be performed either by transmitting a powerful carrier signal that will interfere with the the legitimate tag's signal at the receiver, or by transmitting a modulated signal similar to ISO14443 load modulation. In the first step of our research we examined the performance of each of these methods.

In [FPB11] Finkenzeller et al, demonstrate an extension of RFID transmission range by using an active load modulation, and a large loop antenna. As mentioned above, in order to block the tag's signal we need to produce a modulated signal on the upper side band, hence, the challenge of jamming is similar to range extension using active load modulation. However in the case of jamming there is no concern about bit errors – which should allow the jamming range to be higher than the communication range.

5.1.2.2. Using a monopole antenna

As part of the attack we investigated the possibility of transmitting the jamming signal using an HF monopole antenna rather than a loop antenna. RFID communication is based on magnetic coupling between two loop antennas. As explained in

[FPB11] an effort to increase the range of an active transmitting signal requires either increasing the current injected to the antenna, or increasing the area of the loop. An alternative approach is to use the field generated by an HF monopole antenna. Monopole antennas are designed for the electric field, or plane wave, transmission rather than magnetic coupling. However, the antenna still produces a magnetic field in the near field region. Moreover, there may be a coupling between the electric field produced by the monopole antenna to the reader's circuit, which will also contribute to the jamming.

There are a few advantages of using a monopole antenna for this attack. First, since it usually looks like a simple pole it is easier to hide. Second, there is a variety of commercial antennas in the radio amateurs market which are designed for the desired frequency range. And third, we hypothesize that the jamming range will be longer, and the power consumption will be reduced in comparison to the loop antenna.

According to [SKS+07] the magnetic field at the near-field region around a monopole antenna (assuming an infinitely thin wire) as derived from Stratton [Str07] is given by:

$$H_\phi(\rho, z) =$$

$$\frac{jI_0}{4\pi \cdot \rho \cdot \sin(kh)}[e^{-jkr_0} - \cos(kh) \cdot e^{-jkr} - \frac{jz}{r} \cdot \sin(kh) \cdot e^{-jkr}]$$

where ρ is the distance from the antenna, k is the wave number given by $k = \frac{2\pi}{\lambda}$, z is th height above ground, h is the length of th antenna, and:

$$r = \sqrt{\rho^2 + h^2}$$

$$r_0 = \sqrt{\rho^2 + (z-h)^2}$$

We compare the predictions for the magnetic field produced by a $\lambda/4$ monopole antenna, with the predictions of the magnetic field produced by a 39 cm loop antenna. According to [FPB11] the magnetic field produced by a loop antenna is given by:

$$H_\rho(\rho, \theta) = \frac{jka^2 I_0 \cos\theta}{2\rho^2}\left(1 + \frac{1}{jk\rho}\right)e^{-jk\rho} \tag{5.1}$$

$$H_\theta(\rho,\theta) = -\frac{(ka)^2 I_0 \sin\theta}{4\rho}\left(1+\frac{1}{jk\rho}-\frac{1}{(k\rho)^2}\right)e^{-jk\rho} \tag{5.2}$$

Fig. 5.2 presents the magnetic field as a function of distance from the antenna for: (a) An ideal monopole antenna ($h=\frac{\lambda}{4}\approx 5\,m$) applied with $I_0 = 1\,A$ measured at a height of $z=\frac{h}{4}\approx 1.3\,m$. (b, c) A magnetic loop antenna with a diameter of 39 cm and $I_0 = 1A$ (both H_r and H_θ) [Str03, §5-3]. We note that for $\rho > 20\,cm$ the field produced by the monopole antenna exceeds the field produced by the loop antenna. Based on the above, we predict that using an HF vertical antenna will result in a better jamming range.

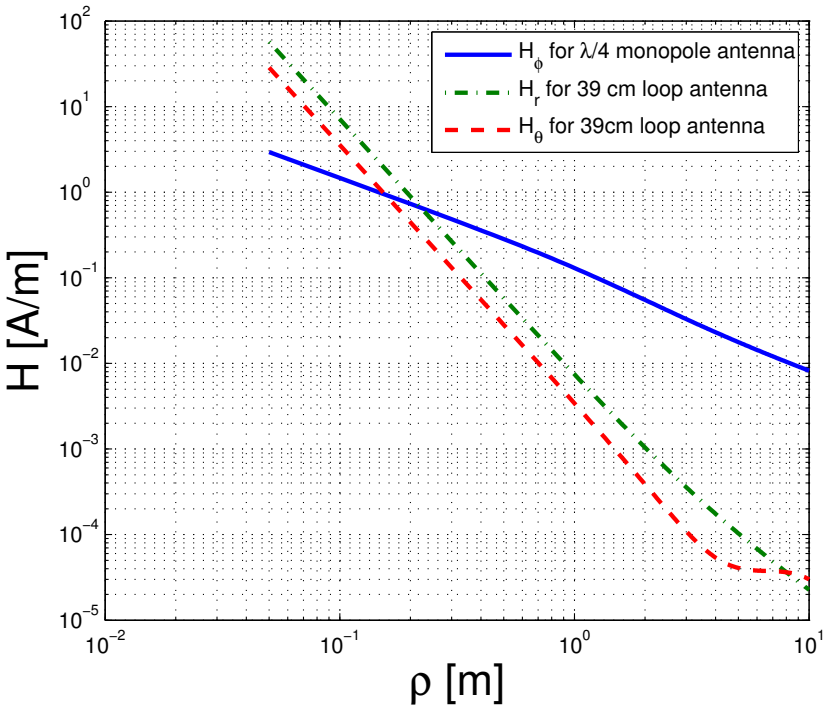


Figure 5.2.: Magnetic field of a $\lambda\backslash 4$ monopole, and a 39 cm loop antenna, for a current of 1 A as a function of the distance from the antenna.

5.2. Experimental Results

5.2.1. Jamming technique

Before checking the jamming distance we wanted to choose the preferred jamming technique. We examined two techniques: (a) Transmitting a continuous wave at the

upper sub-carrier frequency of 14.408 MHz that would interfere with the legitimate load modulation signal at the reader's receiver. (b) Producing a clock signal at 212 kHz, which is the bandwidth of the Manchester coded bit stream the tag transmits (according to ISO14443a [ISO01]), and modulating it using ASK modulation on the upper sub-carrier frequency.

We compared between the two techniques in the lab, measuring the jamming distance achieved using a 39 cm copper tube loop antenna. For the second technique, the 212 kHz clock signal was produced by a pattern generator (Agilent 81110a).

Fig. 5.3 presents the jamming range achieved for each of the techniques for different input powers. One can notice that there is hardly any difference between the performance of the two techniques. Therefore, the attack setup described next uses the first technique only, since it requires less equipment from the attacker.

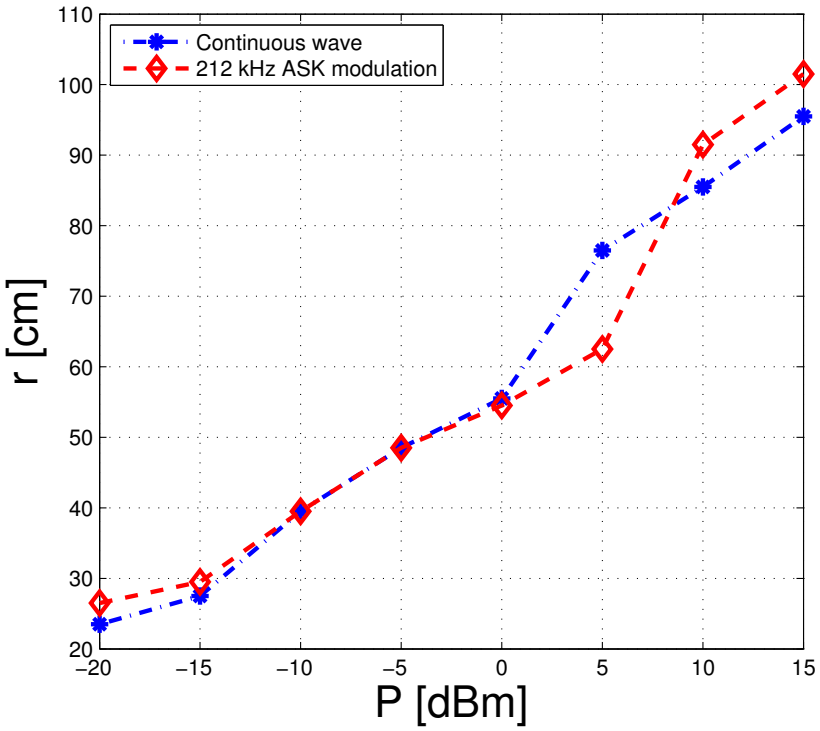


Figure 5.3.: Comparing the two jamming techniques, as a function of transmitted power.

5.2.2. Attack setup

The setup for the jamming attack includes a RF signal source, an amplifier, and an antenna. As mentioned above, the ideal monopole antenna for the desired frequency is over 5m long, and requires a large metal surface for a ground plane. This antenna

is undesirable in the attack setup, which should be mobile, and look innocent to the average eye. Therefore, for our attack setup we used two kinds of mobile HF antenna which are about 1.5m long. In these antennas the lack of height, which results in a capacitive load, is compensated by a large coil.

We examined two kinds of antennas: (a) A radio amateur's antenna. (b) A military broadband helically wound antenna. According to [Str03, §6-37] a helically wound antenna with a height of $\lambda/20$ is similar in performance to a $\lambda/4$ monopole.

Fig. 5.4 illustrates the setup we used for the jamming attack. The setup includes the following equipment:

- RF Signal Generator - to produce the 14.408 MHz sub-carrier signal. We used an Agilent E4438C [Tec08].
- Power Supply - In our experiment we used a lab power supply. The power consumption from the power supply was at most 15W, thus an attacker can also use a car's battery.
- Amplifier - In our experiment we used a Mini-Circuits ZHL-32A[MC09] amplifier.
- Antenna - We examined two mobile HF antenna (both of them about 1.5m long):
  - (a) New-Tronics Hustler: MO-4 (mast) + RM-20-S (resonator), which is designed for the 14–14.35 MHz ham radio band. – estimated cost: $125 [NT08] (See [Str03, §6-29]
  - (b) Broadband vertical helically wound antenna: NVIS-HF1-BC – estimated cost: $1500 (See Fig. 5.5 and [Str03, §6-37])

The jamming signal was produced by the RF generator with an output power of 15 dBm, then amplified by 25 dB using the amplifier, and transmitted through the antenna. Note that for a smaller and more mobile setup the adversary can use a 14.408 oscillator and a pre-amp instead of the RF generator, and he can supply power for all the setup from a car battery instead of the power grid.

5.2.2.1. Coupling effects

During our initial experiments we observed a surprisingly long jamming range of about 10m using the helical antenna. Although we carefully separated the reader from our jamming setup, we later noticed, thanks to an observation by K.Finkenzeller [Fin], that the coax cable connecting the amplifier and the antenna was passing close to the reader. As observed by [TSTMM11], cables, power wires, and even wall framings act as very good antenna relays at HF frequencies. Therefore, the surprisingly long distance was a result of the coupling from the coax cable.

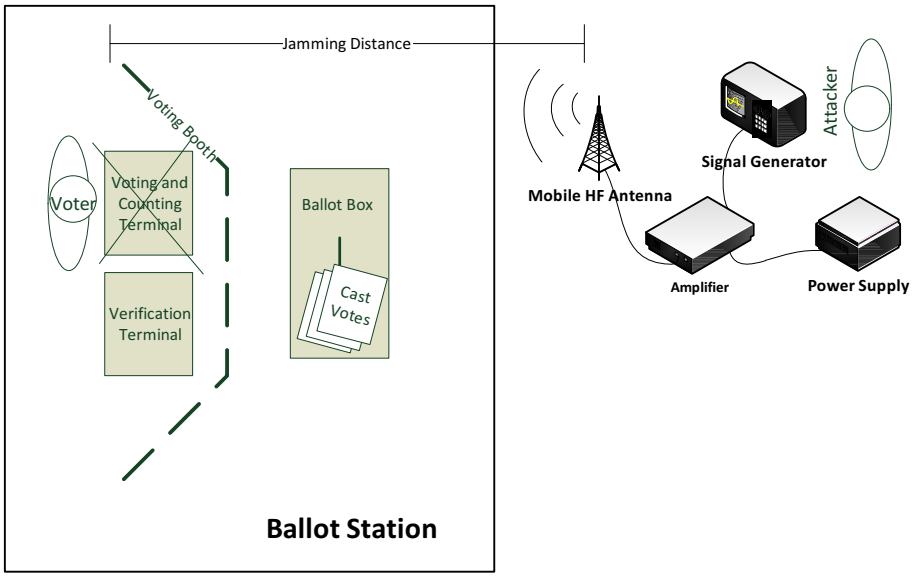


Figure 5.4.: Jamming attack setup

5.3. Discussion

The maximum jamming range was measured for the two mobile HF antenna, and a 39 cm copper tube loop antenna. Jamming was identified using a ISO14443A compliant tag placed next to TI MF S4100 Reader [Ins05]. Using TI's demo software the computer beeps every time a tag is recognized. When placing a tag on top of the reader frequent beeps are heard (about 5-10 beeps per second). We distinguish between two jamming types: full jamming is defined when no beep is heard from the reader for 10 seconds, while partial jamming is defined when 1-2 beeps per second are heard, but still significantly less beeps than with no jamming signal at all.

The maximum jamming ranges for each jamming type, and each antenna are summarized in Tab. 5.1. We notice that using the helically wound antenna we achieve a significant improvement over the loop antenna.

In addition, we wanted to check the effect of the distance between the tag and the reader on the jamming distance. Therfore, we repeated the expirement with the helical antenna, this time with the tag seperated from the reader by 3 cm producing about 20 beeps per minute. In this setup we managed to get an improvement of 30 cm, producing a jamming distance of 2.3 m.

The jamming attack described above can be easily mounted on a car by replacing the RF signal generator with a circuit containing oscillator and a pre-amp. Since in our setup the generator produced a signal with only 15 dBm $\approx$ 30 mW, this circuit can be powered by a battery. Most of the power demands of setup comes from the amplifier which in our experiment consumed a current of about 0.5 A, at a voltage of 24 V. Thus, the power consumed by the entire setup is about 12 W, an amount

which can be supplied from a regular car battery.

Our results indicate that using a mobile HF antenna and some affordable RF equipment that can be easily mounted on a car, one can block the communication of a RFID reader from a distance of few meters. This effectively means that the attacker can place his setup right outside the wall of the ballot station's room and still be able to prevent the voting terminal from working at his command.

The jamming attack is a selective denial of service attack, since it is easy to apply selectively only to a certain subset of voters at the discretion of the attacker. Thus, the attacker can consult any apriori information he has on a voter entering the voter booth (i.e. age, skin color, etc) to decide "on the fly" whether or not to disallow voting for this particular voter. The attack is very difficult to prevent, unless electromagnetic shielding is applied to the walls, doors and windows of every voting station (and not just the ballot box itself) – a very difficult undertaking.

5.4. Future Work

For better results the HF antennas should be placed over a large ground plane. Our experiments were conducted with a 50x30 cm metal plate we had available in the lab as a ground plane.

Furthermore, our amplifier could produce power up to 10 W, using a small HF power amplifier (a variety of these are available in the radio amateurs' market. Increasing the transmission power this way will increase the jamming distance while maintaining the ability to mount the setup on a car, and using the car battery for power supply.

Antenna	Full jamming range [m]	Partial jamming range [m]
39 cm loop	0.95	1.25
Hustler	1.1	1.65
Helical	2	2.3

Table 5.1.: Jamming distance using different antennas

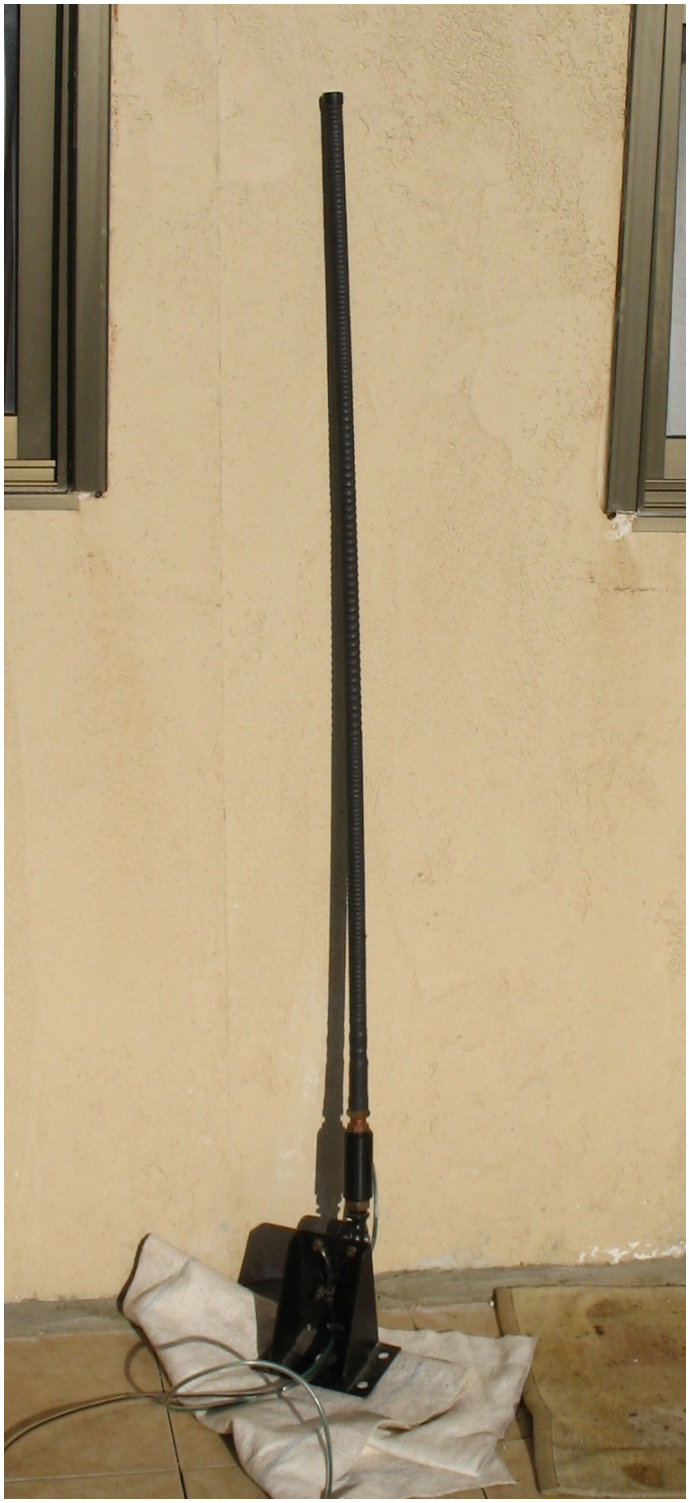


Figure 5.5.: NVIS-HF1-BC antenna. The antenna height is 1.5 m.

6. Countermeasures

This section lists several countermeasures which can be applied to increase the security of the proposed system. Even with all of these countermeasures applied, the system as proposed seems "too broken" to be used in a high-stakes democratic process. However, it would make good sense to apply them if a future variant of the scheme is used for a less important purpose than a democratic election.

6.1. Physical Layer Security

The immediate and most crucial protection mechanism against relay attacks is to shield the voting station and its environment from all forms of electromagnetic radiation. Most importantly, the ballot box should be made into a Faraday cage by constructing it from a sufficiently thick sheet of conducting material such as aluminum. Ferric materials such as iron are probably less suitable since they may not block magnetic coupling between reader and tag. Unfortunately, it was shown that RF shielding against relay attacks is difficult in practice to do properly and cheaply [PHH08], especially for ballot boxes which should be portable, foldable and lightweight. RF shielding is also made difficult by the fact that the ballot box should have a slot for accepting votes[1]. More troubling is the fact that electromagnetic shielding should also be applied to the walls, windows and doors of the voting station itself, to prevent jamming, zapping and selective denial of service attacks from being carried out from a distance. A handy indicator of proper shielding would be the total lack of cellphone reception inside the voting station. When outside the ballot box, the individual ballots (used or not) should be carried inside envelopes made of conducting metal.

The ballot box should also be kept away from any object large enough to hide a leech device. To be on the safe side, no object should be allowed to come within a 1 meter radius of the ballot box. In particular, the ballot box should **not** be on or near the committee table. One open question is how to allow an untrusted voter to approach the ballot box and deposit his vote.

[1] A possible approach would be to protect the slot by a catflap-type arrangement or to line it with fine metal bristles which touch when no card is present.

6.2. Audio Feedback and Cooldown

The voting and verification terminals should be programmed to emit a loud beep each time they read or write a tag. In addition, the terminals should have a "cooldown" period of at least 30 seconds between each use, or at least before switching from one tag to another. Any attempts at wholesale vote sniffing or rewriting will now take a long time and result in many conspicuous beeps, a fact that should arouse the suspicion of the voting commitee. Note that these countermeasures have no effect on the single dissident attack (which registers a single vote per voter), nor do they prevent a single voter from modifying a small but constant amount of votes.

An additional form of audio feedback can be implemented by prepopulating the ballot box with a "canary" tag whose sole functionality would be to beep when interrogated. Since votes inside the ballot box are not typically expected to be interrogated, any attempts to perform a relay attack without first dealing with the "canary" tag will be quickly detected.

6.3. Single-Write Ballots

The voting station should include some sort of irreversible "commitment" or ballot cancellation process (similar to the one performed on paper stamps) which will finalize the vote so the RFID smartcard can be read from but not written to again. In the best case this can be accompanied by a visible mark on the ballot itself. The single dissident attack described in subsection sec. 3.3 will still work in this case if the adversary belongs to the elections committee, since he can keep one un-committed smart card especially for this purpose and perform the relay attacks against this dedicated card. Note that this countermeasure makes the e-voting scheme less cost effective, since it makes the ballots one-use-only and precludes them from being used over multiple elections. According to [Ore], the proposed implementation is supposed to include an (as yet undocumented) mechanism along these lines.

6.4. Strong Probabilistic Encryption

To protect against implementation attacks, the data exchanged between the ballot and the reader should use well-established security best practices. All conversations between the tag and the reader should be authenticated by the private keys of both tag and reader and encrypted by a per-session key. The voting data, which probably consists of a very short payload indicating the selected party, should be padded with random bytes to a constant length and contain a sequence number (to prevent replay attacks on voters who change their minds).

7. Conclusion

In this work we have shown how any party with moderate technical expertise and a fairly small budget can completely compromise the proposed Israeli e-voting system. Our attacks are aided by the fact that the voter's actions in the voting booth are unmonitored, by the design choice of making the tags rewritable, and by the additional fact that the local elections commitee are typically technologically inexperienced.

Despite the threat of relay attacks, contactless RFID smardcards are in use in sensitive applications such as credit cards, e-passports and secure entrance systems, so one may think that they are good enough for e-voting as well. Unfortunately, in these other applications there are always additional security mechanisms: audit trails, credit card statements, insurance, security cameras, human guards, store clerks or border officials. In a voting application, voter privacy **by design** eliminates the possibility for all of these additional safeguards. Unless the attacker is caught "red handed" during the attack, a relay attack on an election may well be a perfect crime.

The designers of the voting system most probably chose contactless technology for cost and reliability reasons. However, this choice led to a devastating consequence in the form of the relay attack. In this work we have shown how any party with moderate technical expertise and a fairly small budget can completely compromise the new Israeli e-voting system. Our attacks are aided by the fact that the voter's actions in the voting booth are unmonitored, by the design choice of making the tags rewritable, and by the additional fact that the local elections commitee are typically technologically inexperienced. In its current form, the proposed system cannot guarantee the privacy of voters, and it cannot promise that cast votes were not manipulated after the fact.

We have offered a few suggestions that, if properly implemented, can mitigate some of the attacks. Nevertheless, viewed together with the substantial non-technical flaws with the system (as discussed in sec. 2.4), we believe the proposed system is strictly inferior to, and completely unacceptable as a replacement for, the plain-paper ballot system currently used in Israel.

Part II.

Side-Channel Attacks

8. Overview

8.1. Introduction to Side Channel Cryptanalysis

8.1.1. Background

Side-channel cryptanalysis has been an active field of research for the last 15 years, having been first described in an academical setting by Kocher et al. in [KJJ99][1]. As formally defined in [MOP07], side-channel attacks are attacks which reveal the secret keys of cryptographic devices by observing their physical properties. The work of [KJJ99] and others has shown that the physical properties of a cryptographic device (such as its temperature, its electromagnetic emanations, and so on) depend on the secret key, and that it is thus possible to extract the key by processing this information leakage.

This chapter deals with a specific type of side-channel leakage called the power side-channel. The power side-channel is obtained by precisely measuring the instantaneous power consumption of the device under test (DUT) as it performs a cryptographic operation, then using this leakage to learn about the internal state of the DUT. As stated in [MOP07], cryptographic hardware typically uses a CMOS fabrication process. CMOS-based registers (or flip-flops) consume more power as they switch between logic states, and less power if they remain in the same state. Thus, the instantaneous power consumption of a DUT is correlated with the number of device registers changing state in a particular time period, or the **Hamming distance** between two consecutive hardware states. If more information about the hardware is available, it is possible to construct more elaborate leakage models which capture additional information about the DUT.

For the simplest devices, that are susceptible to Simple Power Analysis attacks (SPA) [KJJ99], the secret key can be read directly from the shape of a side-channel trace (power consumption, EM radiation, etc.). More commonly, the cryptanalyst needs to use differential (DPA) analysis [KJJ99, MOP07]. DPA techniques typically require multiple traces, often hundreds or more, to overcome the measurement noise via signal processing and statistical estimation techniques. Obtaining all these traces places a significant burden on the attacker, and it is quite interesting to discover ways to extract the secret key data from a *single trace* from devices that are not susceptible to SPA.

[1] It is believed that this form of attack was well known to the signals intelligence community from as early as WWII.

8.1.2. Causes of Errors in Side-Channel Information

The side-channel information emitted by a cryptographic device is an analog high-frequency signal that is measured with a suitable instrument. In case of the power consumption side-channel, the logic cells in the digital circuit draw power from the supply according to their state and activity. This instantaneous power consumption signal is measured with an oscilloscope. The measurement process includes an analog-to-digital conversion of the sampled values. On their way from the logic cell to the oscilloscope's digital output, the power values are influenced by all kinds of physical effects and other signals. These influences are commonly denoted as *noise*. This noise can cause *decoding errors* when trying to estimate the original power consumption of the logic cell.

The overall noise that is present in measured power traces can be divided into electronic noise, quantization noise, and switching noise. Electronic noise is present in every measurement in practice. It includes the noise that occurs in conductors (e.g. thermal noise) and semiconductors (e.g. generation-recombination noise). Furthermore, sources of electronic noise are the conducted and radiated emissions from all components that are part of the control and measurement setup and from external components that operate in the vicinity of the measured cryptographic device. These components include the supply unit that powers the device and the oscilloscope. Another important source of electronic noise is the clock generator that supplies the digital circuit in the cryptographic device with the clock signal. Due to its typical rectangular shape, this signal contains high-frequency components that also influence the measured power values.

The digital oscilloscope contains another source of noise. The analog-to-digital conversion process that it performs introduces small errors in the measured values. The effect of these errors can be modeled as noise in the measured signal, commonly called *quantization noise*. The higher the resolution of the oscilloscope, the lower is the amount of quantization noise.

The third main type of noise is switching noise. Besides the power consumption of the logic cells we are interested in, typically also other cells contribute to the total power consumption value at a specific point in time. The power signals from these other cells are denoted as switching noise. The main parameters of the control and measurement setup that influence the amount of switching noise for a specific point in time are the bandwidth of the power measurement system and the clock frequency. The lower the bandwidth of the measurement path the more the distinct power consumption signals of individual logic cells get blurred together and the amount of switching noise increases. A higher clock frequency can also have such an effect [MOP07].

8.2. Introduction to Algebraic Side-Channel Attacks

As stated in [BBWW07], a constraint solver is a piece of software which tries to find an assignment over a set of variables such that a set of user-defined constraints is satisfied. The first attempt to attack a modern cipher (DES [Nat99]) using constraint solvers was published in 2000 by Massacci et al. in [MM00]. This attack methodology was named Logical Cryptanalysis or Algebraic Cryptanalysis. In Algebraic Cryptanalysis attacks, cryptosystems are represented as systems of equations. A constraint solver is then applied to find the cryptographic key satisfying these equations. Massacci et al.'s work and subsequent followups ([SNC09, MZ06, JJ05, CB07]) demonstrated that modern cryptographic algorithms cannot be directly attacked by constraint solvers.

In [RS09] and subsequent works Renauld et al. applied SAT solvers to the problem of side-channel analysis. Interestingly, while straightforward cryptanalysis was consistently shown to be beyond the capability of solvers, once equations based on side-channel information were introduced into the equation set the situation changed. Renauld et al. showed how a solver-based attack on the modern cipher AES can terminate in less than a minute with the correct key when provided with suitable side-channel information. This new attack methodology was named *Algebraic Side-Channel Analysis* (ASCA).

To carry out an ASCA attack, the attacker recovers a vector of **side-channel leaks** (such as Hamming weights or Hamming distances) from the power trace, then writes an **equation set** mapping these leaks to the evolution of the internal state of the device; finally, a **constraint solver** is used to find the secret key satisfying these equations. In [RS09] and [RSVC09] it was shown that if the side-channel vector is represented perfectly it is possible to recover the key from unprotected AES [Nat01] and PRESENT [BKL+07a] software implementations with very low data complexity (typically one or two power traces).

Renauld et al. suggest in [RSVC09] to perform algebraic side-channel attacks in two separate phases: the first phase is the *estimation phase*, where information is extracted from the power traces using signal processing techniques, while the second is the *key recovery phase*, where this information is processed to return cryptanalytically significant results. In particular, [RSVC09] uses *algebraic methods* for the key recovery phase, converting the problem into a Boolean satisfiability (SAT) problem, then using a SAT solver.

One drawback of the ASCA method is that it requires perfectly accurate side-channel data. Algebraic cryptanalytic attacks are difficult to apply directly to side-channel attacks, since the SAT representation of a cryptosystem and its side-channel measurements is extremely sensitive to noise — indeed [RSVC09] were only able to solve problems with an error rate well under 1%, which is much lower than realistic noise on a single trace. Side-channel analysis using standard solvers was also suggested by [PRR+07].

The sensitivity of the ASCA methodology to noise or decoding errors in the side-channel leak vector severely limits its practicality. In [OKPW10] a new attack methodology called *Tolerant Algebraic Side-Channel Analysis (TASCA)* was presented. This methodology allows the algebraic methods of [RSVC09] to be used for key recovery from a very small amount of side-channel information, even in the presence of reasonable amounts of measurement noise. A modification to the standard ASCA attack which can tolerate some noise was also introduced in [ZWG+11].

8.2.1. Anatomy of a ASCA attack

In an algebraic side-channel attack (ASCA), the attacker is provided with a **device under test** (DUT) which performs a cryptographic operation (e.g. encryption). While performing this operation the device emits a measurable **side-channel leakage** that is expected to be data dependent. A typical example of such a leakage is a power consumption or electromagnetic radiation trace. As a result of the data dependence, a certain amount of **leaks** is modulated into the trace. These leaks are functions of the internal state of the DUT, which can teach the attacker about intermediate computations at various stages of the cryptographic operation. The trace, and its embedded leaks, are subjected to some **noise** due to interference and to the limitations of the measurement setup [OKPW10, S 1.2].

The ASCA methodology uses the following steps to recover the secret key from a power trace:

1. In a first offline phase, the DUT is analyzed in order to identify the position of the leaking operations in the traces, for instance by using classical side-channel attacks like CPA [BCO04] or template attacks [CRR02].
2. Next, in a second offline phase, the DUT is profiled and a decoding process is devised, in order to map between a single power trace and a vector of leaks. A common output of the decoder would be the Hamming weight of the processed data as in [RSVC09], but many other decoders are possible.
3. After the offline phase, the attacker is provided with a small number of power traces (typically, a single trace). The traces are accompanied by **auxiliary information** such as known plaintext and ciphertext. The decoding process is applied to the power trace, and a vector of leaks is recovered. This vector of leaks may contain some errors, e.g. due to the effect of noise.
4. The leak vector, together with a formal description of the algorithm implemented in the DUT, is represented as a system of equations. This equation set also includes any additional auxiliary information.
5. A machine solver such as SCIP [BHPW09] evaluates the equation set and attempts to find a candidate key satisfying it. In the case of an optimizing solver, a goal function is also specified to define the optimality of each candidate solu-

tion. The solver may fail to terminate in a tractable time, or otherwise return a candidate key.

6. Eventually, and optionally, some post-processing can be used, as stated in sec. 8.2.3, in order to brute force the remaining key candidates provided by the solver.

As indicated in the above list, there are several conditions which must all hold true before such an attack succeeds. First, the correct key should not be excluded from the set of solutions to the equation system. This can happen if the traces are too noisy, or if the decoder is not adapted to the attacked device. Next, the solver should not run for an intractable time. This can happen if not enough side-channel information is provided. Finally, the returned key should be the correct key, or at least a key close enough to allow an efficient enumeration.

8.2.2. The TASCA problem space

As discussed in the previous subsection, the TASCA solver is presented with **leak equations** and with **auxiliary information** such as known plaintext and ciphertext, and is tasked with finding the secret key. Let us discuss the behaviour of the solver as the function of the amount of leaks it is provided.

At the most extreme case of limited information, the solver is provided only with equations describing the encryption algorithm, but with no leak equations at all and with no auxiliary information. It is immediately evident that this instance is **trivial to solve** but will return an **incorrect key**, since any key will satisfy this extremely underspecified problem. The same holds if no leaks are known and only the plaintext or the ciphertext (but not both) are provided as auxiliary information.

In contrast, if the plaintext and ciphertext are *both* provided, and no leak equations exist, the problem degrades to that of standard algebraic cryptanalysis – with high probability only a single key exists that will satisfy this plaintext-ciphertext relation, but this cryptanalytic instance is **too difficult to solve**, making it impossible for a solver to find the key in a reasonable time [MM00].

Another extreme case is the case where all side-channel data is presented to the solver, without any measurement error. As shown in [RSVC09], even with no auxiliary information the solver has enough information to **solve the problem successfully and efficiently** and find the correct key in a reasonable time. Obviously, any additional auxiliary information such as known plaintext or ciphertext only increases the probability of a successful outcome.

The amount of leak information provided to the solver is measured both by the number of measurement equations and by the amount of error tolerance built into each equation. As opposed to ASCA problems, for which the leak equations are precisely defined, TASCA (and Set-ASCA) leak equations admit several possible

values for each leak, allowing the solver to tolerate some amount of noise in the measurements.

One method for dealing with measurement noise was introduced in [RSVC09] and more recently investigated in [ZWG+11]. In this approach, which we call *Set-ASCA*, each entry in the recovered leak vector is not represented as an equation allowing a single acceptable value. Instead, the equations accept any value from a *set* of several possible values. The values in the set are each equally acceptable, that is, there is no incentive for the solver to choose one value over another. The resulting equation set is then submitted to a standard SAT solver. As shown in [ZWG+11], this approach is quite satisfactory given that enough leaks and auxiliary information are provided to the solver.

In [OKPW10] a different method for dealing with noise is described, called *Tolerant Algebraic Side-Channel Analysis* (TASCA). The main ingredient in [OKPW10] is the use of an optimizing solver, with a goal function that minimizes the amount of modeled noise. In this work we evaluate the TASCA approach and compare it to the Set-ASCA approach.

8.2.3. Dealing with incorrect answers

The solver does not always succeed in recovering the correct key. There are three possible failure modes:

- The first type of failure occurs when the solver reports that the problem is **unsatisfiable**. The source of this failure is a **decoding failure** – the decoder introduces more noise into the leak equations than the solver is capable of dealing with.
- The second type of failure occurs when the solver **times out** and does not return any answer within the specified time.
- The third type of failure occurs when the solver **returns an incorrect key**. Typically this failure is caused by too few leak equations, or too much error tolerance, which causes the problem to be under-defined.

A specific type of the third failure mode, which we discovered to be quite common, results in the solver returning a **partially correct key**. Due to the specific byte-oriented micro-architecture we address in this work, this mode is usually characterized by a partition of the key bytes into a group of perfectly correct bytes and a group of completely random bytes. Since the error tends to be localised to only a few bytes, the key can be recovered in some cases from the incorrect result using a moderate amount of brute-force searching.

To analyse the brute-force effort required by an attacker, assume that e of the 16 AES key-bytes are incorrect, and that the rest are correct. The attacker must go over all $\binom{16}{e} \approx 2^{4e}$ possible locations for those errored bytes, then try $256^e =$

2^{8e} possible candidate assignments for these positions, resulting in an approximate total effort of $2^{4e} \cdot 2^{8e} = 2^{12e}$ AES operations. Most modern Intel CPUs have a native implementation of AES (AES-NI), which allows a sustained rate of more than 2^{31} AES operations per second on a contemporary system [ADF+10][2]. As illustrated below in Tab. 8.1, an attacker can use a single machine with an AES-NI implementation to probe the close neighborhood of a candidate key and find the correct key within several hours, even if as many as 4 of the 16 bytes returned are incorrect.

Incorrect Key Bytes	0	1	2	3	4	5
AES operations required	1	2^{12}	2^{24}	2^{36}	2^{48}	2^{60}
Estimated time using AES-NI[ADF+10]	<1 sec	<1 sec	<1 sec	25 sec	24 hr	9 yr

Table 8.1.: Estimated time for brute-force searching for incorrect key candidates

8.2.4. Evaluating solver performance

A central component of our attack is a pseudo-Boolean optimizer, described in more detail in [Ach07]. There are several ways of constructing such a solver, for example by extending a SAT solver or by constraining an Integer Programming solver. Regardless of the design choice, a pseudo-Boolean optimizer is generally slower and less efficient than a single-purpose SAT solver.

An interesting alternative to an optimizing solver was introduced in [RSVC09] and more recently investigated in [ZWG+11]. In this approach each entry in the recovered leak vector is not represented as an equation allowing a single acceptable value. Instead, the equations accept any value from a *set* of several possible values. The values in the set are each equally acceptable, that is, there is no incentive for the solver to choose one value over another. The resulting equation set, which is essentially similar to the straight ASCA equation set, is then submitted to a standard SAT solver. As shown in [ZWG+11], this approach is quite satisfactory given that a enough leaks and auxiliary information are provided to the solver. Thus, it is interesting to compare this approach (which we call *Set-ASCA*) to a full optimizer-based TASCA attack.

When evaluating the performance of various solving methodologies, we used the following metrics of performance:

1. The **time** it takes the solver before the key is successfully recovered. This metric gives an advantage to SAT solvers, which are simpler in construction than PB optimizers and are dramatically faster for instances of similar size.

[2]It must be noted that the sustained AES rate described by Intel relates to a sequence of encryptions all using the same secret key. A brute force attack, as described here, uses many different secret keys in rapid progression. This difference may result in a lower practical performance.

2. The **success probability** of the attack to succeed, given a typical power trace. While this is an important practical consideration, we note that from a security standpoint an attack that can break AES for a small but significant fraction of the instances it encounters is arguably just as dangerous as one that can break nearly all of the instances.
3. The **amount of leaks** required – naturally, the attack which requires the least amount of leaks is the most useful. We consider this metric significant, especially considering the large amount of profiling and preprocessing involved in a complete attack and the interplay between the amount of required leaks and the maximum tolerable error rate (see sec. 9.4).
4. The ability of the solver to **tolerate noise** and error. There are several conflicting ways of treating this metric and we will attempt to unify them in this discussion.

8.3. Contributions

One primary limitation of the ASCA method lies in its intolerance to errors: if the correct key is excluded from the system of equations, the attack will fail. This can be somehow mitigated by the robustness vs. information tradeoff, but only up to a certain point, as the loss of too much information makes the computation time intractable. In this work, we first show how an optimizing solver can use probability data to retain the robustness required to be error-tolerant, while losing less information than a SAT solver, at the cost of a larger problem representation. Our key observation is that a SAT representation does not offer a very convenient or efficient method to deal with errors in side-channel information. Instead, we suggest casting the problem in the more expressive language of non-linear *pseudo-Boolean optimization* (PBOPT).

A PBOPT representation offers several properties that are suitable for single-trace side-channel attacks in the presence of errors: (a) A side-channel measurement is typically the Hamming weight w of some hardware feature: this is naturally represented by equating a sum of state bits to the *integer* w – in contrast, representing integers in a SAT instance is quite awkward; (b) It is straight-forward to add variables representing error quantities to the side-channel equations; (c) Unlike a SAT, that is basically a decision ("yes/no") problem, a PBOPT instance includes an objective function, and the solver finds a solution that minimizes this objective.

Luckily, PBOPT offers more than a convenient representation formalism. Research on non-linear pseudo-Boolean equation solvers is a field which displays remarkable activity, and even has a highly-competitive yearly evaluation of solvers [MR09]. Thus, a PBOPT instance representing a single-trace side-channel attack with errors can actually be solved efficiently, leading to our new attack methodology: *Tolerant Algebraic Side-Channel Analysis (TASCA).*

In this work we show how the TASCA or Set-ASCA methodology can be used to recover secret keys from cryptographic devices even if the data available to the attacker is limited both in quantity and in quality. To demonstrate the viability of our TASCA approach, we mounted successful and efficient single-trace attacks, against real, fielded ciphers, even in the presence of realistic error rates of 10%–20%. We show a practical attack on the Keeloq and AES ciphers. We define the *set size k* as a parameter describing the error-tolerance of our attack and provide a theoretical analysis which connects the measurement signal-to-noise ratio and the set size with the success probability. Comparing TASCA and Set-ASCA, we show that optimizing solvers have a distinct advantage over non-optimizing solvers in our scenario. Our results show that using the TASCA method the secret key can be recovered from 60%-70% of AES instances, even when only a single power trace is provided, and even when 20% of the trace signal is corrupted by noise.

To generalize our attack, we describe a novel way of describing measurement equations directly as vectors of aposteriori probability, using the objective function of the optimizing solver. Next, we discuss the generalization of ASCA from the case of Hamming weight leakages to generic (template-based) models. We show that template attacks and ASCA can be naturally integrated, both with standard solvers and optimizers. We additionally provide experimental results allowing to put forward the strengths and weaknesses of the newly proposed probabilistic TASCA and set-ASCA. Overall, the resulting attacks allow strongly reduced data complexity template attacks, when compared to standard divide-and-conquer key recovery attacks.This new cryptanalytic capability may compromise secure systems whose defense against (statistical) side-channel attacks is an aggressive re-keying schedule which results in a small amount of traces per given key.

The results reported in this part appeared in [OW], in [OKPW10] and in [ORSW12].

8.4. Related Work

Algebraic side-channel attacks were introduced by Renauld et al. in [RS09, RSVC09], and first applied to the block ciphers PRESENT [BKL+07b] and AES [Nat01]. These works showed how keys can be recovered from a single measurement trace of these algorithms implemented in an 8-bit microcontroller, provided that the attacker can identify the Hamming weights of several intermediate computations during the encryption process. Already in these papers, it was observed that noise was the main limiting factor for efficient ASCA. To mitigate this issue, a heuristic solution was introduced in [RSVC09], and further elaborated in [ZWG+11]. The main idea was to adapt the leakage model in order to trade some loss of information for more robustness, for example by grouping hard to distinguish Hamming weight values together into sets. We will denote this approach as set-ASCA. Other improvements regarding the error tolerance of ASCA have also been discussed in [MBZ+12]. In parallel, an alternative proposal was introduced at CHES 2010, and denoted as Tolerant ASCA

(TASCA) [OKPW10]. Here, the idea was to include the imprecise Hamming weights in the equation set, and to deal with these imprecisions via the solver. The authors showed how leaking implementations of Keeloq [Daw98] could be attacked in this way, and recently extended their results to the AES case [OW12b].

Other non-algebraic methods have also been suggested for dealing with single-trace power analysis in the presence of noise. A side-channel attack using the Viterbi iterative algorithm [Vit67] for dealing with errors, first presented in the context of elliptic-curve operations in [KW03], is one example.

8.5. Structure

This part is organized as follows: chapter 9 introduces the TASCA method of [OKPW10] and describes a theoretical analysis, connecting the measurement signal-to-noise ratio and the set size in a Set-ASCA or TASCA solver with the success probability. In chapter 10 I describe an attack on the Keeloq cipher. In chapter 11 I describe a series of attacks on the AES cipher and show how the attacker can exploit probabilistic information and extend the attack beyond the well-known Hamming weight leakage model. In chapter 12 the part concludes with some discussion.

9. Tolerant Algebraic Side-Channel Attacks

9.1. Motivation

As stated in [RSVC09], the cryptanalytic problem needs to be transformed into a set of equations before being submitted to the equation solver. This equation set typically consists of a general description of the cryptographic algorithm, together with an assignment of any known inputs to the algorithm. If the equation set represents an algebraic side-channel attack, it will contain additional equations which describe the side-channel emanations of the system in addition to the standard known plaintexts and ciphertexts. Building on the results of [RSVC09], we can assume that an errorless description of the side-channel data will lead to successful key recovery. However, such an equation set is very sensitive to noise: a single errored side-channel measurement will create an equation set that is either unsatisfiable, or is satisfied by the wrong key.

The equations are presented to the solver using the solver's problem description language. The authors of [RSVC09] used a SAT solver which accepts its input in the form of conjugate normal form (CNF) SAT statements. As we shall see, we use the richer and more powerful pseudo-Boolean optimization representation.

9.2. Naïve Methods of Dealing with Errors

Assume that the vector z represents some side-channel information extracted from a certain cryptographic operation (for example Hamming weights or Hamming distances) under a certain key k_c, and that there exists some distance function $d(k, z)$ which indicates how likely a given vector z is to be the result of the operation under a certain key k. As noted in the introduction, the raw side-channel measurement (or *trace*) in itself does not typically have the form of a vector of Hamming weights and must pass some preprocessing before being used. We consider this process, called *estimation*, external to the attack itself.

A typical way of implementing the distance function $d(k, z)$ is to perform a power simulation of the cryptographic operation using a hardware model of the cryptographic device assuming the key k, obtain from this simulation a vector of simulated

side-channel measurements z^k and return the mean-squared error (or L_2 distance) of the two vectors:

$$d(k, z) = \sum_i \left(z_i^k - z_i\right)^2 \tag{9.1}$$

We can assume that the measurement z was created from the "optimal" measurement z_c by the addition of some noise vector:

$$z = z_c + e \tag{9.2}$$

The magnitude of the vector e is defined by the noise model and the performance of the estimator. This analysis assumes a moderately effective estimator and limits the discussion to cases in which the maximum amplitude of e is ± 1 bits in each measurement. An estimator is a *hard estimator* if its outputs are discrete symbols without any confidence information. Under our assumptions the hard estimator will always have a measurement error of either -1, 0 or 1 bits. We can now quantify the errors by considering only P_{err}, the probability that e is nonzero in a given location.

We will now describe several well-known ways of attempting to identify and eliminate noise in decoding problems.

9.2.1. Random Subset Decoding

If P_{err} is very low, we can try to sample a random subset of measurement locations. If by chance none of the measurements are errored, we can attempt to recover k_c from the sample and verify its correctness using trial decryption. Assuming a vector with an i.i.d. probability of hard error of P_{err}, the probability that a set of m indices will contain no errors is $(1 - P_{err})^m$. If we assume, for example, that $P_{err} = 0.01$ and that 128 indices are required for an attack to succeed, the overall probability of success is only 27%. For higher error probabilities this method quickly becomes impractical.

9.2.2. Standard Algebraic Attack with Duplication

The algebraic attack presented in [RSVC09] requires that the measured side-channel information contains no errors. In such a model, a variant of the random subset method can be used: instead of selecting a subset of the data, we can enumerate over all possible locations of errors in the measured data, then create many duplicate instances, each "fixing" the anticipated measurement errors in a certain location and then attempting to carry out an algebraic attack. All duplicate instances are then combined, while we specify to the solver that a single one of the instances needs to be satisfied. Let us assume for example that we have 128 side-channel measurements

and we assume that at most 2 locations out of the 128 contain single-bit errors. In this case we can create $\binom{128}{2}$ duplicate instances, each assuming the errors occurred at a certain pair of locations and "fixing" them. We then specify to the solver that only one out of all of the duplicate instances needs to be satisfied. While most of these duplicates will be unsatisfiable (or result in a wrong recovered key), in one of them the measurement error will indeed be cancelled out by our guess, leading to a successful key recovery. The duplication method is obviously only suitable for a very small amount of errors, since the number of additional instances grows exponentially with the amount of anticipated errors.

9.2.3. Iterative Methods

If the cipher uses the key bits sequentially (bit by bit) in the encryption or decryption process, an iterative Viterbi-like algorithm [Vit67], which is described in detail in [KW03], can be used to recover errors. The Viterbi algorithm's main parameter is its data structure size, which controls the number of key candidates the algorithm maintains during its operation. Letting this size approach $2^{keysize}$, we can treat the iterative algorithm's output as an effective ordering of all key candidates with increasing distance from the measured side-channel information z. The index of the correct key candidate in this ordered list can be an indication of the effectiveness of the iterative approach for solving this specific problem.

The main disadvantage of the iterative method is that it operates in a greedy manner, and cannot return to a key candidate once it has been disqualified. Essentially, this limits the amount of usable side-channel data to a single use of each key bit. In addition, *diffusion* elements (such as the AES MixColumns operation) highly complicate the operation of iterative methods, since many state bits change almost simultaneously and affect every side-channel measurement.

9.3. Handling Errors by Pseudo-Boolean Representation

9.3.1. Side-Channel Analysis as a Pseudo-Boolean Problem

Before we present our approach, let us return to the fundamental problem of side-channel analysis, which can be described as follows:

> Given the algorithmic description of a cryptographic algorithm, the physical power model of the device under attack and the side-channel measurements, output a key assignment for which the expected side-channel information is as close as possible to the measured side-channel information.

When written in the above form, it is clear to see that side-channel analysis is naturally represented as an **optimization problem:**

> Find the **minimal** assignment to an **error vector** such that it is possible for the **cryptographic algorithm**, operating under a certain unknown **key** and in a certain **physical power model,** to produce the **measured side-channel information** affected by this error**.**

We call the class of attacks which performs cryptanalysis using an optimizer instead of a solver **Tolerant Algebraic Side-Channel Analysis (TASCA)**.

9.3.2. An Introduction to Pseudo-Boolean Optimizers

The field of pseudo-Boolean optimization (PBOPT) problems is a special case of integer programming problems [BW05]. Stated informally, a PBOPT instance consists of an *objective (goal) function* and a series of *inequality constraints*, both of which are defined over some set of Boolean variables. A solution to the PBOPT instance must satisfy all inequality constraints while minimizing the objective function. Unlike standard Boolean satisfiability (SAT) problems, a PBOPT problem instance admits multiple solutions, choosing the one solution that minimizes the objective function.

As stated formally in [BHP09], a linear PB problem is an optimization problem over n binary (Boolean) variables $x_1 \cdots x_n$ having the following form:

$$\min c^{\mathrm{T}} x \tag{9.3}$$

$$Ax \geq b \tag{9.4}$$

$$x \in \{0,1\}^n \tag{9.5}$$

where all the coefficients are signed integers: $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^n$. The term $c^{\mathrm{T}} x$ is the *objective function* and the row inequalities in $Ax \geq b$ are the *linear constraints*. The solvers we are interested in can also accept *non-linear constraints* of the form $\sum_{i=1}^{t} d_i \prod_{j=1}^{k} \ell_{i,j} \geq r_i$, where $\ell_{i,j} \in \{x_{i,j}, \bar{x}_{i,j}\}$. Because all coefficients are signed values, equality constraints (of the form $\sum d_i \prod \ell_{i,j} = r_i$) and less-than constraints (of the form $\sum d_i \prod \ell_{i,j} \leq r_i$) can also be reduced to the above form.

Because of their relation to both SAT, linear programming, and integer programming, PB instances can be solved using a variety of approaches. Some solvers attempt to compile the PB instance into a SAT instance and apply a standard SAT solver, possibly multiple times; others map the problem into an integer programming instance; some solvers use a hybrid approach, combining the best features of the two.

The pseudo-Boolean description language is very expressive and allows relatively complex constraints to be described quite efficiently. Notably, each errored side-channel measurement can be efficiently written down as a single equation.

The solver we chose to use is SCIPspx version 1.2.0 [BHP09, BHPW09, Ach07]. SCIPspx won the first prize for non-linear optimizer in the Pseudo-Boolean Evaluation Contest of SAT 2009 [MR09]. SCIPspx solves the optimization problem by using integer programming and constraint programming methods. It performs a branch-and-bound algorithm to decompose the problem into sub-problems, solving a linear relaxation on each sub-problem and finally combining the results. The linear relaxation component of SCIPspx is the standalone LP solver SoPlex [Wun96].

9.3.3. Elements of a TASCA Equation Set

To represent a side-channel attack as a PB optimization instance a TASCA equation set is written, consisting of the following four sections:

1. **A general description of the cryptographic algorithm as a set of equations**: The cryptosystem is described by writing down internal state transformations leading from plaintext to ciphertext. The specification is very hardware-minded, with each state bit/memory element (flip-flop) typically represented as a sequence of variables representing its evolution in time, and each combinational element (gate) finding its way into an equation connecting the variables. For example, the AES state has 16 bytes, each of which changes its value 4 times in each round (other than the first and the last). This means that the state of each subround is represented by 16×8 binary (0-1) variables, for a total of $16 \times 8 \times 41 = 5248$ variables for an entire AES encryption. There will also be variables for every key bit and every subkey bit, and a set of equations representing the subkey expansion.

2. **An assignment of any known inputs to the algorithm:** These can be known plaintext or ciphertext, or even more subtle hints such as the relationship between two consecutive unknown plaintexts.

3. **A specification of the measurement setup:** The actual side-channel measurement is mapped to the internal state according to the structure of the physical hardware device. For example, an 8-bit microcontroller-based implementation will typically leak the Hamming weight of individual state bytes as they are accessed, while a parallelized ASIC will typically leak the Hamming distance between the former and present values of all bits in the device's internal state. Note that both in the case of Hamming distance and in the case of Hamming weight the measurement equation consists of an equality between a sum of Boolean variables on one side and an integer value on the other – $\sum_j state_{i,j} = m_i$, where $state_{i,j}$ is the value of state variable j at time i and m_i is the side-channel measurement at time i. This form of equation is natural to write down using the PB syntax. It should be noted that when attacking the same cipher running on different target architectures, the measurement setup is usually the only section of the equation set which needs to be modified.

4. **A set of potentially errored measurements:** This section matches the measurements described in the previous section to actual outputs of the estimation phase. As stated previously, the main point of the TASCA approach is to allow errors in the estimation. This is done in practice by adding additional *error variables* to the above-mentioned measurement equations. These error variables are used to cancel out errors in the measurements. In our implementation we included two error variables per measurement (one with a plus sign and one with a minus), which allow the true side-channel value to be within ± 1 bits of the measured one. It should be noted that this section is the only part of the equation set which tolerates errors (all other sections are explicitly defined), and that this section only accounts for 1% to 5% of the entire set of equations for the cryptosystems we tested.

In addition to the equation set, the solver is provided with a **objective function** which it is required to minimize. In our case, our objective is to use as few error variables as possible.

9.4. Theoretical Modeling of Error Tolerance

Since there are different ways of encoding error into an equation set, it is difficult to compare the ability of different methods to deal with errors. Therefore, it is useful to find a way to discuss errors in terms of standard metrics such as signal to noise ratio or bit error rate.

As stated in sec. 8.2.1, the side-channel leak is first passed through a decoder, where it is converted to set of leak equations, and next to a solver. The equation representations we discuss here deal with errors by permitting more than one valid assignment to each individual leak (i.e. they follow either the Set-ASCA or the TASCA methodologies). Thus, a set of k values are accepted for each leak. A necessary condition for the solver (Set-ASCA or TASCA) to succeed is that the correct value of each leak is a member of the acceptable set presented to the solver[1].

Let us now assume that we wish to carry out an attack based on Hamming weights of internal calculations performed by an 8-bit micro-controller. In this model each leak x_i is an integer value which takes a value between 0 and 8. The leak is modulated onto the amplitude of the power trace, subjected to some additive noise, and finally recovered by the attacker.

As shown in [MOP07], in the case of an 8-bit microcontroller the amplitude of the power trace is approximately linearly dependent on the Hamming weight. To recover the Hamming weight from the power trace, the attacker typically measures the amplitude of the power trace at one or more points in time, then applies an

[1] This is not a sufficient condition – even if all leaks are recovered correctly the problem may still be under-defined or computationally intractable.

affine transform to these measurements to arrive at $\hat{x}_i$, the estimated value for the leak x_i. This estimated value can thus be seen as the result of the original integral leak x_i and an additive Gaussian noise element $\nu_i \sim \mathcal{N}(0, \sigma)$: $\hat{x}_i = x_i + \nu_i$.

To put this model into standard engineering signal and noise terms, the signal power P_s can be defined as the *variance of the Hamming weights of the inputs*, assuming a uniform distribution of the inputs. The noise power P_n can be defined as the *variance of the noise* ν. This leads to the standard definition of signal-to-noise ratio, which is typically measured in dB:

Let $HW(i)$ denote the Hamming weight of an integer $i \in [0, 255]$

Claim:$SNR \approx -20 \log_{10} \sigma + 3$

Proof:

$$SNR = 10 \log_{10} \frac{P_s}{P_n}$$

$$= 10 \log_{10} \frac{\frac{1}{256} \sum_{i=0}^{255} \left(HW(i) - E\left[HW(i)\right]\right)^2}{\sigma^2}$$

Since $E\left[HW(i)\right] = 4$ it can be verified that the numerator equals 2, giving

$$SNR = 10 \log_{10} 2 - 10 \log_{10} \sigma^2$$

$$\approx -20 \log_{10} \sigma + 3 \,\square$$

Now that $\hat{x}_i$ is defined, it needs to be presented to the solver. Assuming the solver is a TASCA or Set-ASCA solver with a set of size k, the natural approach would be for the decoder to populate the set of valid solutions with the k integer values closest to $\hat{x}$. A necessary condition for the solver to succeed would then be that one of the elements of the set output by the decoder is the original x_i. The elements of this set may all be considered equally desirable (in the case of Set-ASCA), or otherwise ranked according to their distance from $\hat{x}_i$ (in the case of TASCA). For example, if the set size k is 1 (corresponding to ASCA) and the decoder observation was $\hat{x}_i = 2.2$, the set will contain the value 2. In other words, if the decoder outputs the symbol closest to $\hat{x}_i$, and the original symbol x_i was 2, the attack will succeed only if the decoded value $\hat{x}_i$ is between 1.5 and 2.5, or equivalently if the noise ν_i is between -0.5 and 0.5. If k is 2, the equations will allow the values 2 and 3, and the attack will succeed only if $\hat{x} \in (1.5, 3.5)$. In general, if k values are acceptable then the noise ν_i must be in the range $\left[-\frac{k}{2}, \frac{k}{2}\right]$. For a **decoding success** we require this condition to hold for all m leaks in the leak vector simultaneously. Assuming the noise is i.i.d. for all leaks, the probability of such an event is:

$$\Pr\left(\text{Decoding Success}\right)$$

$$
\begin{aligned}
&= P\left(\hat{x}_i \in \left[x_i - \frac{k}{2}, x_i + \frac{k}{2}\right] | x_i\right)^m \\
&= P\left(x_i + \nu_i \in \left[x_i - \frac{k}{2}, x_i + \frac{k}{2}\right] | x_i\right)^m \\
&= P\left(\nu_i \in \left[-\frac{k}{2}, \frac{k}{2}\right] | x_i\right)^m \\
&= P\left(\nu_i \in \left[-\frac{k}{2}, \frac{k}{2}\right]\right)^m \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\frac{k}{2}}^{\frac{k}{2}} e^{-\frac{x^2}{2\sigma^2}} dx\right)^m
\end{aligned}
$$

By fixing the set size k and the number of leaks m and solving for σ, we can apply Proposition 1 and find the maximum Gaussian noise power tolerable by sets of size k if a certain success probability is desired.

If we define the per-leak error rate as the probability that a set of size 1 does not contain the correct value of x_i, we can convert the value of σ to an error rate using the following relation:

$$
P_{error} = P\left(\nu_i \notin \left[-\frac{1}{2}, \frac{1}{2}\right]\right) = 1 - \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{-\frac{x^2}{2\sigma^2}} dx
$$

Tab. 9.1 summarises these results for relevant ranges of parameters. It can be seen from the table that each set size corresponds to a certain maximal error rate and, equivalently, to a certain minimal signal-to-noise ratio. We can see that for a set size $k = 1$ (corresponding to the ASCA method) the maximum error rate is between 0.05% and 0.1% for $m \in [100, 200]$. Such a low error rate is impractical to achieve even with high-end measurement equipment. However, with $k = 3$ it is possible to tolerate an error rate of almost 20%, which is within reason.

Leak Count	m =100 leaks		m =200 leaks	
Set size k	min. SNR [dB]	max. error rate	min. SNR [dB]	max. error rate
1	20.8	0.1%	21.2	0.05%
2	14.8	5.2%	15.2	4.3%
3	11.3	19.5%	11.6	17.7%
4	8.8	33.1%	9.1	31.1%

Table 9.1.: Relation between error rate and set size for normally distributed noise when we require Pr (Decoding Success) = 99%

Fig. 9.1 displays the probability that a decoder using a set of size $k = 3$ will correctly capture all measurements with a fixed error rate. It can be seen that as the number

of equations m grows, the probability of decoding success falls exponentially. We see that an increase in information causes a decrease in the robustness of the attack, making it more sensitive to errors. Thus (and perhaps counter-intuitively) it makes sense to provide the solver with the *least* amount of side-channel information required for a successful key recovery, even if more information is available.

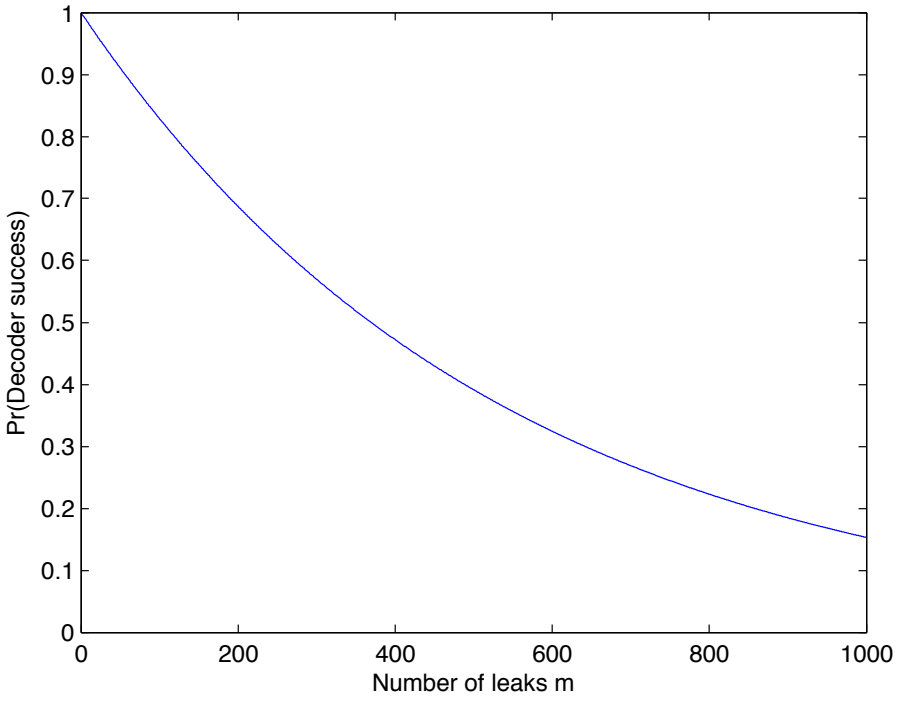


Figure 9.1.: Decoder success rate as a function of leak count m, with a fixed set size $k = 3$ and an error rate of 30% ($\sigma = 0.4824$, SNR=9.3 dB)

9.5. Solver Software and Hardware

The solver used in our experiments was SCIP version 1.2.0 compiled for Windows 64-bit[BHPW09]. When we began running our experiments in late 2009, this solver was listed by [MR09] as the best non-commercial solver available for non-linear optimization problems. The solver was run on on a quad-core Intel Core i7 950 at 3.06 GHz with 8 MB cache, running Windows 7 64-bit Edition. To take advantage of the multiple hyper-threading cores of the server, six instances of the solver were run in parallel. It should be noted that running a single instance at a time will result in noticeably better performance due to less contention on the L2 cache (equation solving is very RAM intensive) and due to Intel's Turbo Boost feature which speeds up one computational core when the others are idle[Int08]. A set of MATLAB scripts was used to create random instances, run the solver and collect the results automatically.

For the set-ASCA experiments we used CryptoMiniSAT 2.9 [Soo]. This solver won several prizes in SAT competitions (SAT Race 2010 [Sin10] and SAT competition 2011 [sat]) and is well adapted to deal with cryptographic problems, as XOR operations (very frequent in cryptographic algorithms) are managed by the solver using specific optimized clauses. The solver was run on on a quad-core Intel Core Intel Xeon X5550 processor, running at 2.67 GHz with 8 MB cache.

10. Attacks on Keeloq

Keeloq is a block cipher which is most commonly used in remote keyless entry (RKE) systems, e.g. for cars. We chose to attack this cipher first since it has a very simple round structure which is relatively easy to represent as equations. Furthermore, a reduced version of Keeloq (using 140 rounds instead of the full 528) was already broken using standard algebraic techniques in [CBW08] without requiring side-channel inputs. In [EKM+08] a physical ASIC implementation of the Keeloq cipher was shown vulnerable to a standard DPA attack, an attack we were also able to reproduce.

Because it only operates on a single bit of the key in each round, Keeloq is very effectively attacked using the iterative approach described in sec. 9.2.3.

10.1. The Keeloq Algorithm

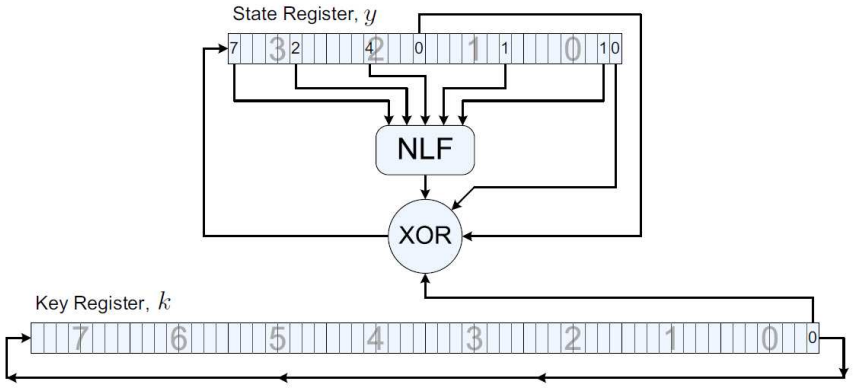


Figure 10.1.: Structure of the Keeloq cipher (taken from [EKM+08])

The Keeloq algorithm [Daw98] is a block cipher designed for efficient hardware implementation. Keeloq has a block size of 32 bits and a key size of 64 bits. As shown in Fig. 10.1 (taken from [EKM+08]), its main components include an internal state register (32 bits) and a non-linear feedback function (NLF). In each round of the cipher, NLF operates on five bits from the cipher's current state. The output from NLF is mixed with some prior state bits and with one of the 64 key bits and finally shifted back into the state register. To perform encryption, the plaintext is

loaded into the state register, the key is loaded into the key register, and the entire system is clocked for 528 rounds. After these 528 rounds the state register contains the ciphertext. To perform decryption the ciphertext is placed in the state register and the system is clocked 528 times in the opposite direction. The progression of the state register is typically modeled as a vector of bits $S_0 \cdots S_{559}$, with $S_0 \cdots S_{31}$ being the plaintext and $S_{528} \cdots S_{559}$ the ciphertext.

10.2. An Equation Set for Keeloq

As stated in sec. 9.3.3, a TASCA equation set consists of four elements - the algorithm, the inputs, the measurement setup and finally the (potentially errored) measurements.

The algorithmic description of a single Keeloq round is a simple set of 2 PB equations:

$$NLF_i = NLF(S_{i+31}, S_{i+26}, S_{i+20}, S_{i+9}, S_{i+1}) \tag{10.1}$$

$$S_{i+32} = NLF_i \oplus S_i \oplus S_{i+16} \oplus K_{i \bmod 64} \tag{10.2}$$

The function NLF is a 5-to-1 non-linear function defined such that $NLF(a, b, c, d, e)$ is bit number $abcde_b$ (binary) of the hexadecimal constant $3A5C742E$, where bit 0 is the least significant bit. It has no efficient linear or algebraic representation, and is represented as a single disjunctive normal form (DNF) equation based on the function's truth table. The XOR function on 4 variables (effectively 5, since we realize the function $x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = 0$) is also represented by a single equation. Each of these equations is repeated 528 times to lead from the plaintext ($S_0 \cdots S_{31}$) to the ciphertext ($S_{528} \cdots S_{559}$).

In its most common mode of operation, Keeloq uses *rolling codes* which mandates that the ciphertext is known to the attacker but not the plaintext. Accordingly, the only known input to our solver was the ciphertext.

If we assume Keeloq is implemented on an ASIC, the power traces tend to be correlated with the *Hamming distance* of the entire 32 bits of the state register between the current round and the previous round (a similar attack can also be mounted if the device leaks the Hamming weight). To put this into equation form, we define the Hamming distance between each two consecutive bits of the state progression and group them in sets of 32. Finally, we add two additional Boolean variables to the measurement sums to allow for errors:

$$hd_i = S_i \oplus S_{i-1} \tag{10.3}$$

$$HD_i = \sum_{j=i}^{i+32} hd_j \tag{10.4}$$

$$\widehat{HD_i} = HD_i + e_i^+ - e_i^-$$

The number of rounds for which we produce side-channel measurement equations (m_{sc}) is a configuration parameter of the system: the following subsection shows how to select a proper value for m_{sc}. A Keeloq key recovery instance with side-channel measurements equations applied to the final $m_{sc} = 90$ rounds of encryption contains a total of 428 equations and has a file size of about 140 KB. A partial listing of a sample PB instance is provided for reference in the Appendix to this dissertation.

10.3. Attack Results

We performed a power simulation of 300 ASIC-based Keeloq decryptions, corrupted the simulated power measurements with different probabilities of error and submitted them to the SCIP PB-solver. For each decryption and tested error probability, we selected $P_{err} \times 528$ rounds and corrupted the side-channel values measured in those rounds (specifically, the device-total Hamming distance) by ± 1. In each experiment the solver was asked to recover the 64-bit key from the errored side-channel outputs produced by the final 90 rounds of encryption.

Fig. 10.2 shows our results. For reference we also show the performance of an iterative attack on the final 64 rounds of the encryption using comparable bit error rates. We emphasize that the attack is on the **full** 528-round cipher, even though it uses only a subset of the measured side-channel data.

It can be seen that the solver was able to find the key even with $P_{err} = 18.8\%$ with an average running time of 3.8 hours per instance, and that the time grows superlinearly with the error probability. The TASCA solver is 10 to 100 times faster than the iterative solver on instances with $P_{err} \geq 11\%$.

We also noted it was important to properly choose the number of side-channel measurements passed to the solver (m_{sc}). When too few rounds were passed, the optimal solution found was not necessarily the correct key. When too many rounds were passed, the computational burden involved slowed down the solver, as shown in Fig. 10.3. In the case of Keeloq a good tradeoff was $m_{sc} = 90$ rounds which provided enough information to find the key in nearly all of the cases. For lower m_{sc}values the solver returned incorrect results for at least 25% of the instances.

As an aside note, the iterative decoder, which struggled with single-trace key recovery, had much better performance when attacking multiple traces. The algebraic solver did not perform as well with additional inputs, since each additional trace significantly increased the size of the equation set.

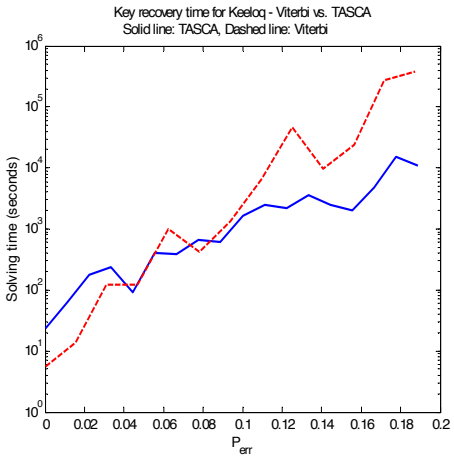


Figure 10.2.: TASCA key recovery from the final 90 rounds of Keeloq

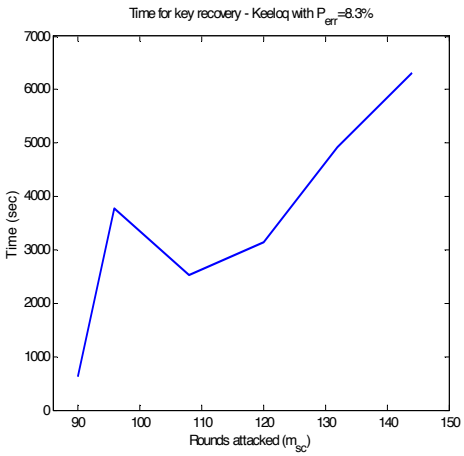


Figure 10.3.: TASCA speed as a function of m_{sc}

11. Attacks on AES

11.1. The AES Algorithm

We chose to model our device under attack as naïve 8-bit microcontroller implementation of AES-128 [DR98], based on the standard NIST implementation of [Nat01] and on a hardware implementation specified in [Sat04]. AES-128 is a block cipher with a 128-bit key and 16-byte (or 128-bit) input blocks. To perform encryption, the plaintext is first fed into a 16-byte state register. This state register is then manipulated 41 times during the sequence of the 10 rounds of AES-128 to produce the ciphertext. There are 4 types of manipulations: SubBytes, AddRoundKey, ShiftRows and MixColumns, with SubBytes being the only non-linear operation in AES. AddRoundKey is performed 11 times during encryption, first with the supplied secret key and then 10 times with round keys derived from the secret key using a non-linear process which uses the SubBytes process as well.

We assumed that the device leaks the Hamming weight of the 8-bit operand on its data bus. Following the observations of sec. 9.4, we aimed to provide the minimum amount of (errored) measurements to the solver. Thus, we only modeled the first few rounds of AES. The text below assumes the reader is familiar with the workings of AES. The interested reader is referred to [Nat01] for more details. We made the following assumptions:

- Only the **plaintext** is provided to the solver, but not the ciphertext.
- The leaks from the **key expansion process** are not available to the solver, since we assume that the DUT performs round key expansion in advance.[1]
- The **AddKey** and **AddRoundKey** operations leak the Hamming weights of the 16 state bytes after the XOR with the key/round key, as well as the Hamming weights of the key bytes themselves, giving a total of 32 leaks per subround.
- The **SubBytes** operation is implemented as a look-up table (LUT). The equations representing the SubBytes operation use the canonical representation (see [OKPW10, S 5.2]), using a single equation per output bit or a total of 8 equations per invocation of SubBytes. The LUT operation leaks only the

[1] It was already established in [Man02] that the Hamming weights leaked from an 8-bit microcontroller implementation of AES during key expansion are sufficient for full key recovery, even without any additional state information.

Hamming weights of the 16 state bytes after the SubBytes operation (and not any other internal state information), for a total of 16 leaks per subround.

- The **ShiftRows** operation is implemented logically as index shuffling and as such does not leak any information.
- The **MixColumns** operation is implemented using 8-bit XTIME and XOR operations as specified in [Nat01] and as such leaks 36 additional bytes of internal state per round. In addition to the 16 leaks of the final state, this gives a total of 52 leaks per subround.

In total each round of AES (consisting of **SubBytes, ShiftRows, MixColumns** and **AddKey/AddRoundKey**) leaks 100 Hamming weights of 8-bit values.

The experiments were performed under various error rates and set sizes up to the theoretical thresholds calculated in sec. 9.4. We chose to focus on the performance of the solver and not on that of the decoder, assuming that the leaks satisfied the necessary condition for **decoding success** outlined in sec. 9.4. To apply an error rate of e to the Hamming weight measurement vector of length m for a given experiment, em indices were chosen independently at random, and the Hamming weight at each of these selected indices was modified by either ± 1 (for sets of size $k = 3$) or by $+1$ (for sets of size $k = 2$). Sample AES instances created using this method are available online[OW12a] and are described in the Appendix.

Each ASCA or TASCA instance consists of a general description of the cryptographic algorithm as a set of equations, an assignment of any known inputs to the algorithm, a specification of the measurement setup, and finally a set of potentially errored measurements. For TASCA instances we also include a goal function, which instructs the solver to aim for a solution (i.e. key assignment) which minimizes the amount of noise in the measurements. If we change the measurement equations so that they do not admit noise (i.e., let $k = 1$) and eliminate the goal function, we transform the problem from a TASCA instance to an ASCA instance similar in form to that used in [RS09].

Note that the optimizer we used to perform TASCA was not memory-efficient enough to represent the entire AES encryption in equation form. As a result, we provided it with a known plaintext and the cipher equations for the first round of encryption only. By contrast, the SAT solver was provided with a plaintext/ ciphertext pair, and all the cipher equations. In order to have comparable experiments, we only exploited the 100 first round leakages, in both cases.

11.2. An Equation Set for AES

The AES hardware realization can be modified and optimized in a variety of ways. Specifically when dealing with the S-box component of AES, which performs the

Table 11.1.: Instance size and performance of straight encryption

	Keeloq	AES (LUT)	AES (Canright)
Instance file size	553K	32873K	12569K
# of equations	1153	27344	93090
# of variables	13825	171208	229008
# of constraints	13825	173640	231506
Encryption time (sec.)	2.59	61.07	245.45

SubBytes operation, there are a variety of hardware implementations offering various tradeoffs between better speed and more efficient hardware consumption.

Our first TASCA representation of AES implementation was based on a port of an OpenCores VHDL AES code [Sat04]. This implementation models the S-box as a lookup table (LUT), leaving the compiler with the task of optimizing it to a minimal hardware footprint. A second TASCA representation was based on the efficient composite field representation of the S-box designed by Canright [Can05]. In this design, the S-box input is manipulated under a more efficient basis representation.

Tab. 11.1 summarises the performance of the two cipher implementations, with the performance of the Keeloq encryption provided as reference. Since the encryption was described as an equation set, performance was similar whether the plaintext, the ciphertext or any intermediate state was supplied. Similarly, any round key can be substituted for the secret key with no effect on performance. Analyzing the results, it appears that the performance of the solver is dominated by the number of equations under consideration and not by the complexity of the equations themselves. This may be a property of the SCIP solver, and not of PB optimizers in general, since SCIP essentially performs a search over the tree of equations. Specifically, other PB solvers which compile their inputs into SAT instances may show better performance using the Canright S-box, since its reduced hardware complexity should make it easier to simulate.

Surprisingly, the solver had a very hard time inverting the SubBytes and MixColumns operations given their algebraic description. We found out that including equations for both SubBytes and for inverse SubBytes (that is, one equation stating that $S_{i+1} = SubBytes\,(S_i)$ and another stating that $S_i = SubBytes^{-1}\,(S_{i+1})$) sped up the solver dramatically.

11.3. An Attack on the Key Expansion Algorithm

Our first attack on AES was a reconstruction of the SPA attack of Mangard on the key expansion algorithm, as described in [Man02], without any errors. A secret key was recovered from 1, 2 and 3 rounds of expansion (16, 32 and 48 Hamming

Table 11.2.: A TASCA attack on the AES key expansion phase

Rounds	AES (LUT)			AES (Canright)		
	instance size	# of eq'ns	time (sec)	instance size	# of eq'ns	time (sec)
1	765K	164	11	208K	1484	193
2	1529K	308	1341	414K	2948	10800
3	2293K	452	1690	620K	4412	345600

weights[2]). The key expansion was modeled using both S-box representations. The results are summarised in Tab. 11.2.

The time performance of the solver was worse than we estimated. We were also surprised to find that Canright representation yielded longer running times than the LUT representation, despite having instances that are 3 times smaller. Understanding these phenomena is a topic of future work.

11.4. A Full Attack – ASCA with no errors

An ASCA instance typically has a smaller solution space than a TASCA instance, and is thus much easier to solve. We used ASCA instances to determine the minimal amount of side-channel information which can be provided to the solver for an efficient and correct response.

Fig. 11.1 shows the success probability of non-error-tolerant ASCA measurements as a function of the run-time with different amounts of side-channel data. It can be seen that 20%-60% of the problems are solved within about 1 minute while the rest run for a significantly longer amount of time. This agrees with the findings of [RSVC09].

Tab. 11.3 summarizes the success rate of the ASCA solver as a function of time for different amounts of side-channel data. Based on sec. 8.2.3, we consider as success cases where the solver terminates and returns a key which is at most 4 bytes apart from the correct one .

Based on these results we conclude that very few error-free side-channel measurements – a single AES round is sufficient – can usually achieve full key recovery. We discovered that if we reduced the leak count to less than a full round, the probability of success became vanishingly small. Note that the success probability after 2 hours is largely unaffected by the amount of side-channel information. Thus it makes sense to use only a single round of leaks and no more, and thus increase the robustness of the attack.

[2]The published results in [Man02] show that 40 recovered Hamming weights are enough to uniquely determine the secret key

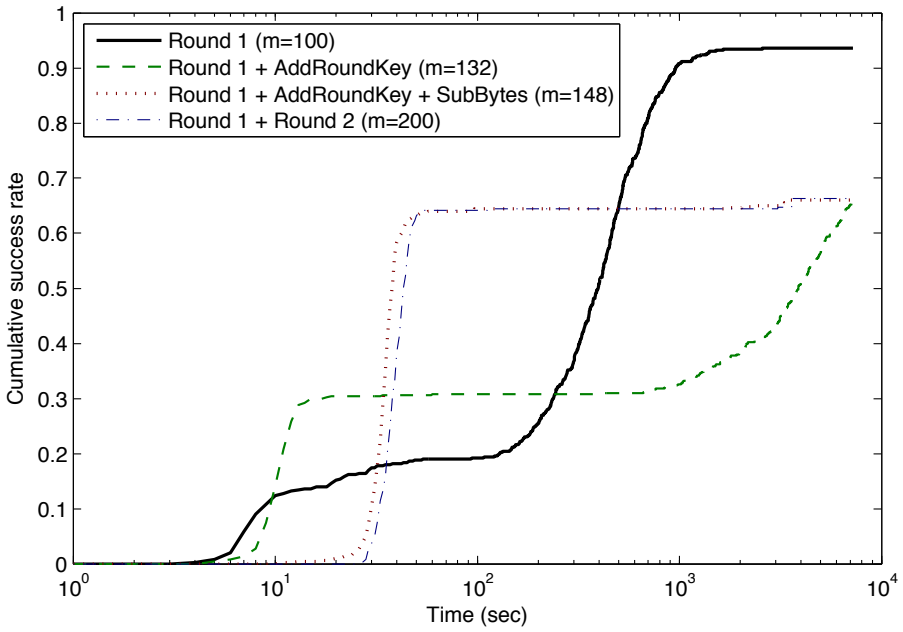


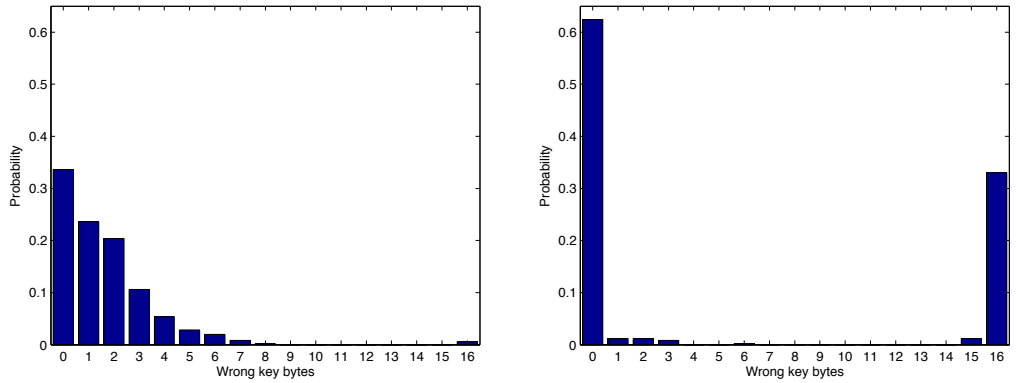
Figure 11.1.: ASCA cumulative success rates for different amounts of side-channel data

Amount of side-channel data	Mean solving	Success rate after...		
	time	10 sec	80 sec	2 hrs
Round 1 (100 leaks)	392 sec	12.4%	19%	93.6%
Round 1 + AddRoundKey(132 leaks)	1950 sec	14.2%	30.8%	65.6%
Round 1 + AddRoundKey + SubBytes (148 leaks)	102 sec	0%	64%	66%
Round 1 + Round 2 (200 leaks)	137 sec	0%	64.2%	66.3%

Table 11.3.: Success rate of the ASCA solver at different times

A possible drawback to using a small amount of side-channel data is the risk of a producing an underspecified problem, causing the solver to return an incorrect answer. To demonstrate the impact of this effect, Fig. 11.2 shows the distribution of incorrect key bytes in the recovered solution for 100 and 132 of side-channel measurements, where instances which did not terminate after 2 hours were assigned the all-zero key.

We can see that with 100 leaks, only 6.4% of the recovered keys had more than 4 incorrect key bytes. With 132 measurements this rate drops to 0.2%, but an additional 35% of the instances fail with a solver time-out. With additional measurements the fraction of wrong bytes drops to nearly 0 (graphs omitted). As discussed previously in sec. 8.2.3, even 3 or 4 incorrect key bytes can be considered a correct result, if we allow the immediate neighborhood of the candidate key to be probed using brute force. Under this assumption, in all cases the key was either recovered correctly or



(a) Round 1 (100 side-channel measurements) (b) Round 1 + AddRoundKey (132 side-channel measurements)

Figure 11.2.: ASCA wrong key histograms

the operation timed out. This leads us to conclude that 100 leaks (one full AES round) are enough to uniquely determine the key in the case of error-free ASCA (i.e., $k = 1$), even if the ciphertext is not provided to the solver.

11.5. A Full Attack – TASCA and Set-ASCA with errors

This section describes the results of TASCA and Set-ASCA runs on simulated power traces of AES which have been corrupted with noise. We tested the average solving time and average success rates for various combinations of side-channel information amounts and error rates. We also measured the effect of the optimizing aspect of the solver (the goal function) on the performance of our solver.

11.5.1. Effect of error rate

The objective of this experiment was to measure the effect of the error rate on the solving time and success probability of our TASCA solver. Based on the conclusions of the previous section, we provided our TASCA solver with one round of side-channel leaks (m =100 measurements) and with a known plaintext. The error rate was chosen to be between 0% and 25%. According to Tab. 9.1, a decoder has at least a 99% probability of success when producing such a set of equations, given the above parameters and a set of size k =3. Fig. 11.3 shows our results. We also ran the experiment with $k = 2$ and error rates of up to 5% (results omitted).

In general, the TASCA approach showed itself capable of recovering the secret key, with a success probability of 30%-80%, from errored traces in 6 to 24 hours for sets

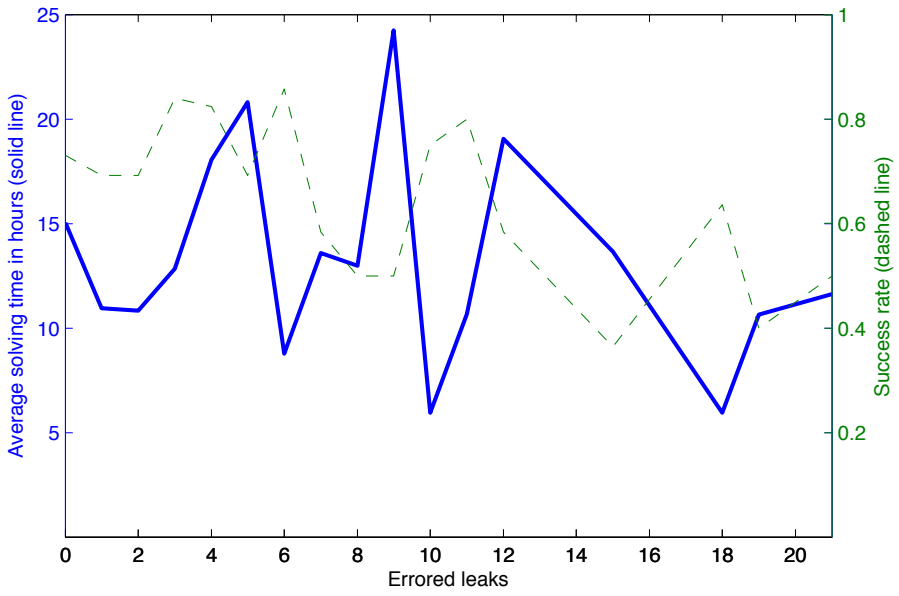


Figure 11.3.: Effect of error rate on success rate and solving time for 3-set TASCA of AES with $m = 100$ leaks (20 experiments per data point)

of size $k = 2$ and $k = 3$, even with error rates all the way up to the theoretical boundary of 19.5% previously calculated in sec. 9.4. Interestingly, we could find no significant correlation between the error rate and the success rate, nor between the error rate and the solving time. This is in contrast to our previous results on Keeloq in chapter 10, where increasing the error rate caused a measurable increase in the running time and a corresponding decrease in the success rate.

11.5.2. Comparing TASCA and Set-ASCA

As discussed in sec. 8.2.4, an alternative approach to TASCA called Set-ASCA was introduced in [RSVC09] and more recently investigated in [ZWG+11]. In this approach the k values in the set are each equally acceptable, that is, there is no incentive for the solver to choose one value over another. The Set-ASCA equation set, which is essentially similar to the straight ASCA equation set, is then submitted to a standard SAT solver. Assuming an identical quantity of leaks is used in both cases, Set-ASCA has been shown in [ZWG+11] to be about 20 times faster than (optimizing) TASCA. However, many keys – potentially an exponential amount – may satisfy the same side-channel leak equations. If a non-optimizing solver is used in this case, it will arbitrarily choose a satisfying solution and terminate, making its success probability exponentially small. In contrast, an optimizing solver will only terminate once it has found a solution which minimizes the goal function. Assuming the goal function has been correctly specified, we hypothesised that an optimizing TASCA approach is more likely to find the correct key than a random satisfying

assignment.

To investigate this scenario, we compare a TASCA and a Set-ASCA solver operating on m =100 leaks of side-channel data with 0% errors, varying the set size from $k = 1$ (standard ASCA) to $k = 4$. Tab. 11.4 describes our results. Each data point in the Table is the average of 200 randomly generated runs.

Set Size k		1	2	3	4
Success probability	(TASCA)	78%	87.5%	73%	72.7%
	(Set-ASCA)	78%	9.2%	0%	0%
Mean solving time in minutes	(TASCA)	6.45	245.2	901.99	1332.07
	(Set-ASCA)	6.45	171.05	7.18	2.92

Table 11.4.: TASCA vs. Set-ASCA performance with 100 leaks

Note that for a set size of k =1, both TASCA and Set-ASCA solvers are reduced to the case of standard ASCA, making their performance identical. As we expected, the solving time of the optimizer is much worse than that of the SAT solver, a difference that only grows as the set size grows. On the other hand, it can clearly be seen that the accuracy of Set-ASCA falls dramatically when we must increase the set size k to overcome decoder failures: the SAT solver used in Set-ASCA may terminate quickly, but it provides the correct key in only 9% of the cases for $k = 2$ and is virtually always wrong for $k = 3$ or $k = 4$. This is caused by the large set of admissible solutions made possible by the lower precision of the leak equations. Since the SAT solver chooses one of these solutions arbitrarily, its success rate falls dramatically as the set size grows. In contrast, the optimizer is motivated to choose the best solution from the available set. As a result, its success rate is largely determined by the error rate (as we saw in sec. 11.5.1) and not by the set size.

11.6. Exploiting probabilistic information

As stated in the previous section, in an ASCA the attacker takes the output of a decoding process and converts it into a series of measurement equations. An example for such a decoding process would be a nearest neighbor decoder, a naïve Bayes decoder [MRS08, S 13.2] or a template decoder [CRR02]. In most cases this decoder does not only output "hard" data (i.e. the most likely leak value) but also some additional "soft" information, such as confidence information, a ranking of several possible leaks by decreasing order of likelihood or, most generally, a full vector listing the aposteriori probability for each possible leak value, conditioned on the received trace. In this section, we discuss an improvement of TASCA which is capable of taking advantage of this soft (probabilistic) information.

For this purpose, let us start from the standard scenario of a Hamming weight-based ASCA. In this context, the tradeoff between robustness and information is

generally achieved by choice of the **set size** k. It defines the number of acceptable values for each individual side-channel leak in the equation set, relative to the apriori selection of a leakage model (e.g. the Hamming weight of the manipulated data). The value of k can be either determined as a global constant for all equations in the set (e.g. as in [OW12b, RSVC09]), or determined on a per-leak basis according to some heuristic (e.g. as in [MBZ+12, ZWG+11]). In the case of a precisely-defined equation set, in which $k = 1$, only the most likely value output by the decoder is accepted. This representation provides the most information, but it cannot tolerate any errors. As the set size k grows, so does the robustness of the equation set, but this comes at the price of a loss of information. The original work on ASCA [RS09] investigated only the case of $k = 1$. Thus, the single value chosen as most likely by the decoder was entered into the equation set. In the set-ASCA experiments of [MBZ+12, RSVC09, ZWG+11], more than one value was listed as possible to the solver, sacrificing information for robustness against errors. In this case, each leakage equation would accept the k most likely values, as output by the decoder. The TASCA attack of [OW12b] also uses a set, and additionally uses a goal function to mark one of the value in the set as likelier than the others, without further quantification of this likelihood. This representation is more informative than in a set-ASCA, but it still does not fully take advantage of the information provided by the decoder.

We now present a more expressive way of representing the probability information provided by the decoding phase. In the most general case, the decoder outputs a full probability vector for each leak, listing the aposteriori probability of the leak having each possible value, conditioned on the specific trace being received. This output is typical for e.g. template decoders [CRR02]. In the case of a Hamming weight-based template decoder, each potential leak will have an associated vector of 9 aposteriori probabilities corresponding to Hamming weights 0 to 8. If we further assume that individual leaks are uncorrelated, then the combined probability of all leaks in the trace is proportional to the product of the individual aposteriori probabilities. Of course, most of these combinations are impossible, since they violate the cipher equations. The goal of the solver in this case would be to find the set of leaks that maximizes the product of aposteriori probabilities while still corresponding to a valid encryption. As shown in [RSVC09], the exact values of the Hamming weight leaks provide enough information to uniquely and efficiently find the correct key of an AES encryption. If we define the exact value of leak i as x_i, we can define the objective of the attack as:

$$x_1 \cdots x_m = \arg \max_{x_1 \cdots x_m} \prod_{i=1 \cdots m} \Pr(x_i | trace) \; s.t. \text{ cipher eq'ns are satisfied.}$$

Since the goal function of the SCIP solver is expressed as a sum of integers which must be minimized, we represent the objective using this equivalent expression:

$$x_1 \cdots x_m = \arg \min_{x_1 \cdots x_m} \sum_{i=1 \cdots m} -\log\left(\Pr(x_i | trace)\right) \; s.t. \text{ cipher eq'ns are satisfied.}$$

The representation of a probability vector $\overline{p_x}$ for a certain leaked Hamming weight x with set size k as a side-channel leak equation is thus split into two parts: the *constraint set* and the *goal term.* The constraint set is very straightforward – it considers k different events called "HW(x) is 0", "HW(x) is 1", etc., describes each event in terms of the relevant combination of bits in the leaked byte, and finally requires that one and only one of these events be true in for each leak in a satisfiable solution. The goal term matches each event with a corresponding probability. Each probability p is represented in the goal term as $-\lfloor C \log p \rfloor$, where C is an implementation parameter. The goal terms of all leaks in the system are then summed together to create the global goal function.

11.6.1. Experimental Validation

In order to compare set-ASCA, basic TASCA and TASCA with probabilities in the Hamming weight leakage model, we designed a first set of experiments, based on simulated leakages from the PIC device illustrated in Fig. 11.5. For each experiment, we list the *decoding success rate* – the proportion of traces for which all 100 correct leaks are included in the 100 k-sized sets provided by the decoder – and the *key recovery success rate* – the proportion of traces for which the solver returned the correct key within a reasonable time. We also report on the (median and maximum) solving time and show the average *number of correct key bytes* in case of successful attacks. For set-ASCA, both the plaintext and ciphertext are included in the equation system, meaning that an attack can only succeed when every 16 key bytes are correct. On the other hand, in TASCA instances only the plaintext is used, meaning that when the set size increases, several keys can be valid according to the algebraic representation. In this latter case the average number of correct key bytes can be below 16. As explained in sec. 11.7.2, the attack is still considered successful when at least 12 out of the 16 key bytes are correct. Our results are summarised in Tab. 11.5.

These experiments lead to a number of interesting observations. First and as expected, they clearly illustrate the information vs. robustness tradeoff. That is, the probability of decoding success grows as the set size grows (better robustness). However, this impacts the performances of the different attacks in different manners. For set-ASCA, the solving time quickly increases to the point of intractability, because of a lack of information. By contrast, the basic TASCA are more resistant to the loss of information: as the set size grows the running time increases, but the key recovery probability is much less affected. Yet, the limited information available to the solver causes parts of the key to be recovered incorrectly in some cases, which then requires an additional brute forcing step. Combining probabilistic information with a set size of 3 finally allowed the optimizer to recover the correct key in virtually all experiments. It also reduced the running time compared to the basic TASCA. Yet, both TASCA approaches are still much slower than the set-ASCA approach

attack	set size	decoding success	key rec. success	med. solving time	max. solving time	# of correct key bytes
set-ASCA	1	0%	0%	N/A	N/A	N/A
set-ASCA	2	83%	83%	2 seconds	6 seconds	16
set-ASCA	3	100%	0%	24+ hours	24+ hours	N/A
basic TASCA	1	0%	0%	N/A	N/A	N/A
basic TASCA	2	83%	75%	43.7 minutes	11.8 hours	14.48
basic TASCA	3	100%	80%	16.8 hours	66 hours	13.25
prob. TASCA	1	0%	0%	N/A	N/A	N/A
prob. TASCA	2	83%	82%	56.7 minutes	10.07 hours	15.88
prob. TASCA	3	100%	100%	8.2 hours	143 hours	16

Table 11.5.: set-ASCA, basic TASCA and probabilistic TASCA experimental results against the PIC microcontroller simulated leakages with Hamming Weight model.

when it succeeds (e.g. for set sizes $k \leq 2$), due to the more complex design of the optimizer. This is also reflected by the larger memory requirements of the TASCA solving phase.

Summarizing, the TASCA approaches allow improved flexibility as they systematically deal with the information vs. robustness tradeoff during the solving phase. By contrast, set-ASCA shift this problem to the decoder phase. In case of low-noise scenarios, or whenever the adversary can average the measurements, set-ASCA is the method of choice because of its reduced memory requirements and solving times. It also allows exploiting all the leaks (i.e. not only the first round ones). By contrast, the more the measurements are noisy and/or hard to interpret by the adversary (e.g. because of countermeasures), the more the TASCA approaches becomes interesting, thanks to its optimizing features.

11.7. Beyond the Hamming weight model

As illustrated in sec. 11.7.2, Fig. 11.6, the leakage of certain devices (e.g. in 65nm and smaller technologies) cannot always be precisely expressed with simple models. As a result, it is interesting to investigate how ASCA/TASCA can be extended towards these more challenging scenarios. In this section, we show how to move from Hamming weight-based models to more generic ones.

For this purpose, let us assume that the attacker has performed template-based profiling of the DUT [CRR02]. Given a power trace, the he can now create a

probability vector for each leak, where each entry in this vector matches a certain possible leak value, and each value in the vector is the *aposteriori* probability of this leak conditioned on the power trace being processed. Assuming the DUT has an 8-bit architecture, each such vector contains 256 entries. The decoding process will output a number of such vectors – one for every leak in the equation set. As for the Hamming weight model, we use this side-channel information to restrict the size of the solution space. In order to do so, we define a parameter called the **support size** k', which is comparable (though not identical) to the set size k in the previous section. It corresponds to the amount of possible values associated to each leak. These values are chosen according to the probability vector: the k' most probable values are considered possible, and the others are rejected. Hence, the value of k' must be carefully chosen in order to avoid rejecting the correct value from the set of possible ones, making the problem unsolvable.

Representing this generic leakage model as clauses or equations is less easy than for the Hamming weight model. For the set-ASCA, the easiest way to represent a set of k' possible values for a leaked byte x is to exclude all impossible values. For example, in order to exclude the value $x = 9 \Leftrightarrow (x_0, ..., x_7) = (0, 0, 0, 0, 1, 0, 0, 1)$, we add to the SAT problem the clause $(x_0 \cup x_1 \cup x_2 \cup x_3 \cup -x_4 \cup x_5 \cup x_6 \cup -x_7)$. Each set of k' possible values is thus translated into $256 - k'$ clauses with 8 literals per clause. In order to speed up the solving process, we additionally apply some simplification techniques, e.g. reducing the length and number of closes. For the TASCA, the representation of a probability vector $\overline{p_x}$ for a certain leaked byte x with support size k' as a side-channel leak equation is again split into two parts: the *constraint set* and the *goal term*. The constraint set describes k' different events called "x is 0", "x is 1", etc., and requires that one and only one of these events is true for each leak. The goal term matches each event with a corresponding probability. An example of such a representation can be found in the Appendix. Since this representation is especially suited for template-based profiling, we call it **template** TASCA and set-ASCA.

11.7.1. Impact of the support size and goal function

A first natural question in this new setting is: how small must the support size be for the attacks to succeed, and what is the impact of the probabilistic information that can be added to the optimizer? To answer it, we designed an experiment in which we compared many pairs of template-TASCA instances of single-round AES with different support sizes. In each pair, one of the instances was provided with an **unweighted** probability vector (that is, all nonzero elements in the probability vector are considered of equal probability), while the other was provided with a **weighted** vector function. The weighted vector function was simulated (independent of any actual leakage model), and chosen such that the single correct byte value always had a higher probability than all the other ones in the support. For the rest, the

instances were identical, with only plaintext provided, such that the solver could potentially output an incorrect key as in the previous section.

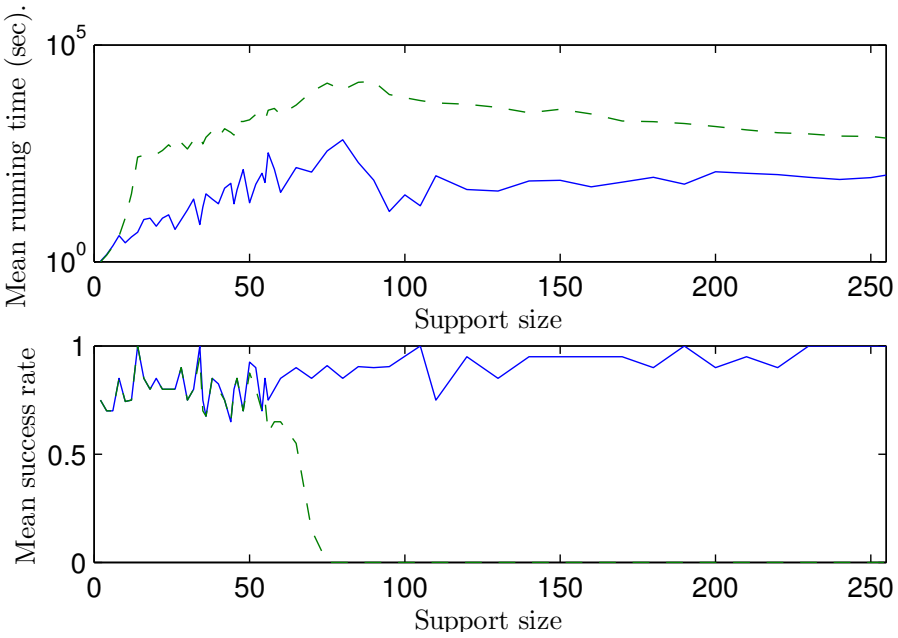


Figure 11.4.: Template-TASCA attacks with weighted (solid line) and unweighted (dashed line) probability vectors: experimental running time and success rate.

The results of this experiment are illustrated in Fig. 11.4. As we can see, they can be divided into four distinct phases. In the first phase (support sizes up to 10), the performance of the weighted and unweighted instances is identical, probably because enough information is available in the support of the function, making the additional information in the goal function redundant. In the second phase (support sizes $10 - 50$) both the weighted and the unweighted instances end in successful key recovery, but the weighted instances are faster by two orders of magnitude. We see that in this range there is still enough information in the unweighted instances to precisely specify the correct key, but the added information of the goal function allows the optimizer to reach the correct answer more quickly. In the third phase (support sizes $50 - 70$) the success rate of the unweighted instances slowly falls to 0, probably because more and more incorrect keys can satisfy the constraint set. However, the additional information in the goal function causes the optimizer to prefer the likeliest solution, which in our case was the correct one. Finally, in the fourth phase (support sizes 70 and up) the large amount of possible keys in the support makes the success rate of the unweighted instances marginally small. Note that even with a full support ($k' = 256$) the performance was still good. This implies that all information about the instances can be encoded into the goal function and not into the constraints, and thus that the correct key will never be excluded from the equation set.

Furthermore, it could be argued that the running time of the weighted templates is faster than that of the unweighted ones because the correct guess is always the highest ranked. To investigate this scenario, we repeated the same experiment, this time setting the rank of the correct key candidate to 2. The change in rank caused the running time of the weighted case to increase, but had no effect on the success rate. We verified this behavior for ranks of up to 14.

11.7.2. Experimental validation

To validate our results, we considered two different scenarios. First, we used the templates obtained from a PIC microcontroller. As illustrated in Fig. 11.5, this device closely follows a Hamming weight leakage model. Next, we used the templates obtained from the AES S-box implemented in a 65-nanometer CMOS technology, previously analyzed in [RSVC+11]. In particular, we selected one of the S-boxes for which the leakage model is not correlated with the Hamming weight of the manipulated data, as illustrated in Fig. 11.6. In both cases, the signal-to-noise ratio was similar and relatively high (with the variance of the signal approximately 10 times larger than the noise variance), yet leading to some decoding errors, as will be investigated next. In both cases, the **experimentally obtained templates** (with their associated apriori probabilities) were fed with **simulated inputs**, and the outputs were used as the basis of the attack. These setups were selected in order to illustrate the efficiency of ASCA in different implementation contexts.

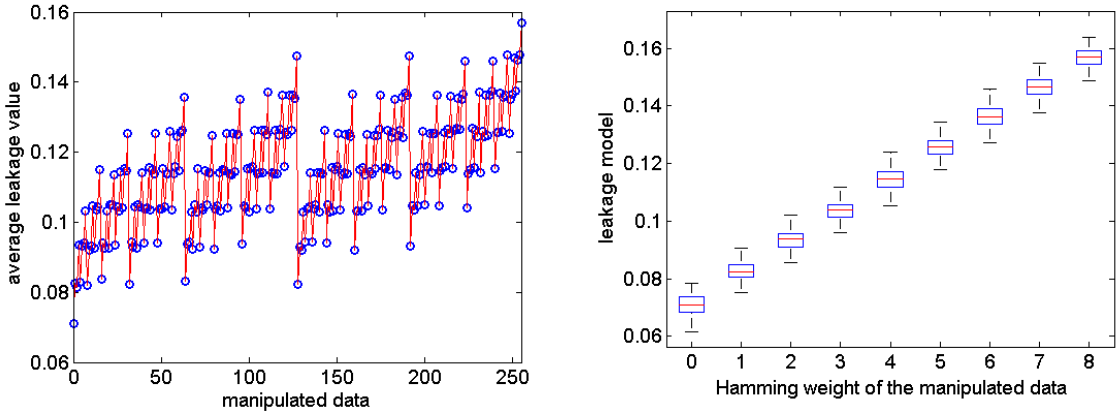


Figure 11.5.: PIC leakage model: average values (left) and grouped by HWs (right).

Finally, we note that because of the previously mentioned memory limitation issue, the success condition is defined differently for set-ASCA and TASCA. In the set-ASCA scenario, the entire encryption operation is included in the equation set, meaning that the solver either outputs the correct key or otherwise runs for an intractable time. By contrast, the TASCA solver only exploits the first round equations and, therefore, can sometimes return an incorrect key. We deal with this condition by measuring the amount of incorrect bytes in the result – if 4 bytes or

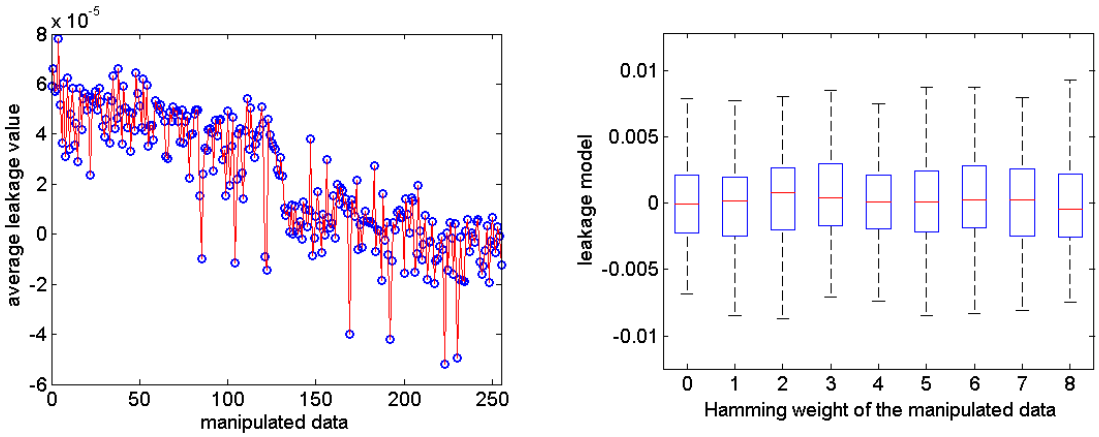


Figure 11.6.: 65nm S-box leakage model: average values (left) and grouped by HWs (right).

less are incorrect, we assume that the correct key can be recovered from this partially correct key by brute force[3] and declare success. We also recall that the amount of leaks exploited (i.e. 100) was the same in all our experiments.

attack	set size	dec. SR	key rec. SR	med. solving time	max. solving time	# of correct key bytes
set-ASCA	64	15.5%	15.5%	2 sec.	2 sec.	16
set-ASCA	90	90%	90%	265 sec.	24+ hours	16
set-ASCA	100	100%	29%	24+ hours	24+ hours	16
prob. TASCA	64	15.5%	15.5%	35.88 sec.	86.03 sec.	16
prob. TASCA	90	90%	90%	245.72 sec.	869.4 sec.	16
prob. TASCA	100	100%	100%	342.76 sec.	21271 sec.	16
prob. TASCA	256	100%	100%	62254 sec.	48+ hours	16

Table 11.6.: Template set-ASCA and probabilistic TASCA experimental results against simulated leakages from a 65nm S-box, with generic template model.

As in the previous section, we verified the effectiveness of our attacks by performing several experiments. This time, we considered a DUT where the simulated leakages are generated according to the model of the 65nm ASIC implementing one AES S-box presented in sec. 11.7.2. As illustrated in Fig. 11.6, the leakage function of

[3] Assume that e of the 16 bytes are incorrect. The attacker must go over all $\binom{16}{e} \approx 2^{4e}/e!$ possible locations for those errored bytes, then try $256^e = 2^{8e}$ possible candidate assignments for these positions, resulting in an approximate total effort of $2^{4e} \cdot 2^{8e} = 2^{12e}$ AES operations. Most modern Intel CPUs have a native implementation of AES (AES-NI), which allows a sustained rate of more than 2^{31} AES operations per second [ADF+10]. Thus, an attacker can use a single machine with an AES-NI implementation to probe the neighborhood of a candidate key and find the correct key within less than 24 hours, even if 4 of the 16 bytes are incorrect.

this device is very different from the Hamming weight model. Thus, it constitutes a perfect target for our template-based set-ASCA and TASCA. In a first step, we profiled the AES S-box, resulting in 256 templates corresponding to the 256 possible transition values. Each univariate template assumes a Gaussian noise and was characterized by a mean value μ and a noise standard deviation σ. In a second step, we used Bayesian inversion to simulate the classification probability $\Pr(x_i|trace)$ from the template output $\Pr(trace|x_i)$.

The results of the attacks are summarised in Tab. 11.7.2, where we selected different support sizes k'. As expected, smaller support sizes lead to more unsatisfiable/unsolvable problems, but these problems are solved faster, meaning a higher success rate for the computation phase. As soon as $k' \geq 100$, all the problems are solvable, but the solving process becomes much longer. We compared two attacks: the set-ASCA and the probabilistic TASCA. Both essentially confirmed our previous observations. Namely, the set-ASCA instances are very fast to solve for low support sizes, but suddenly increase in difficulty between $k' = 90$ and $k' = 100$. By comparison, probabilistic TASCA instances for low support sizes are much slower to solve than set-ASCA ones. Nevertheless, the difficulty of solving TASCA instances increases slower than for set-ASCA ones. In the end, probabilistic TASCA is able to solve problems with support size $k' = 256$, which is totally infeasible for set-ASCA (as $k' = 256$ means no side-channel information for set-ASCA instances). Summarizing, we again observe a tradeoff between efficiency (set-ASCA) and flexibility (probabilistic TASCA).

Tab. 11.7.2 presents a comparison of the Hamming weight model and the template model in terms of set size and support size. For each set size, *i.e.* for each number of possible Hamming weight values, the table details the corresponding average support size $\bar{k}'$ and the minimum and maximum support sizes k'_{min} and k'_{max}. For instance if $k = 2$, the two consecutive Hamming weight values $\text{HW}(x) = \{0 \text{ or } 1\}$ correspond to $k'_{\text{min}} = 9$ possible transition values out of 256. Similarly, the two Hamming weight values $\text{HW}(x) = \{3 \text{ or } 4\}$ correspond to $k'_{\text{max}} = 126$ possible transition values out of 256. On average, a leak that is represented by a set of 2 possible Hamming weight values can also be represented by a set of $\bar{k}' = 95$ possible transition values. Contrarily to the attacks using the template model where the support size is the same for every leak, the attacks using the Hamming weight model present different support sizes. Therefore, some sets of Hamming weight values offer more information than others. In other words, the Hamming weight information is not uniformly distributed over the 100 considered leaks in the first AES round. This table allows us to compare and better understand the results from Tab. 11.5 and Tab. 11.7.2. For example, we observe that solving set-ASCA problems with the Hamming weight model for set size $k = 2$ takes about 2 seconds, while solving set-ASCA problems with the template model for a similar support size $k' = 90$ takes more than 250 seconds. Hence, set-ASCA seems to take advantage of the non-uniform information proposed by the Hamming weight model. This confirms observations already made

in [CFGR12]: the SAT solver usually exploits small parts of the equation system where the information is most concentrated. Interestingly, the same is not true for probabilistic TASCA: template instances with support size 90 or 100 are faster to solve than Hamming weight instances with set size 2. Our hypothesis is that for probabilistic TASCA, the goal function contains more information when using the template model than the Hamming weight model, as the Hamming weight model does not make any distinction between different transition values with the same weight. As a consequence, the advantage offered by non-uniform information is counterbalanced by a less informative goal function.

Set size k	$\bar{k}'$	$k'_{\min}$	$k'_{\max}$
1	50	1	70
2	95	9	126
3	134	37	186

Table 11.7.: Comparison between set sizes (Hamming weight model) and corresponding average $\bar{k}'$, minimum $k'_{\min}$ and maximum $k'_{\max}$ support sizes (template model).

11.8. Comparison with Keeloq results

This report describes an attack on the block cipher AES[Nat01]. The discussion in Chapter chapter 10 applied TASCA to Keeloq[Daw98], a simple stream cipher used in car remote controls and other low-security applications. There are several differences between the Keeloq cipher and the AES cipher, when viewed from the perspective of a TASCA attack. On one hand, the Keeloq cipher seems easier to analyse because of its weak diffusion property – the key bits are shifted into the Keeloq cipher state one bit at a time, whereas in 8-bit AES they are introduced into the cipher state one byte at a time. On the other hand, the Keeloq implementation runs on an ASIC circuit which leaks 32-bit Hamming distances, a side-channel leak which is generally considered more difficult to attack in contrast to the 8-bit Hamming weights leaked by the AES implementation.

Tab. 11.8 contrasts AES and Keeloq instances in terms of their respective sizes and solving times. The Keeloq instances use m =90 subrounds (and hence 90 measurements), while the AES instances use one AES round with m =100 measurements. In both cases the TASCA instances use a set size of $k = 3$ and the leaks are corrupted with a 5% error rate. It can be seen from the table that despite the fact that the AES instance is only 9 times larger than the Keeloq instance, it is harder to solve by three orders of magnitude. Another interesting difference, which we demonstrated in Figure Fig. 11.3 on page 81, is the low correlation between the error rate and the solving time for AES – as shown in [OKPW10], the solving time of Keeloq TASCA instances increases super-linearly with the error rate, while we could observe no such

condition in the case of AES. In our experiments, AES instances took roughly the same amount of time to solve regardless of their error rate.

	Keeloq	AES	Ratio
ASCA instance size	140KB	1.3MB	x9.3
ASCA instance time	0.36 sec	387 sec	x1072
TASCA instance size	140KB	1.3MB	x9.3
TASCA instance time	22 sec	8.7hr	x1436

Table 11.8.: Comparison of AES and Keeloq performance

Both differences in behaviour between AES and Keeloq may be attributed to the different diffusion properties of the two ciphers. While in Keeloq the key is introduced into the equation one bit at a time, in AES it is XORed into the plaintext as soon as encryption begins and further diffused by the following operations. The good diffusion property of AES makes it difficult for the solver to infer the value of one variable from the assignment of another. This both increases the run-time and defeats the "added value" granted to the solver for correctly determining a partial solution. Since diffusion has such a profound effect, finding a more precise power model of the cipher which allows additional leaks to be considered and reduces this diffusion property should AES instances easier to solve.

12. Discussion and Conclusions

12.1. Better PB Solvers

The author does not claim to be an expert in the design and usage of PB solvers. In fact, the SCIP tool which we used has hundreds of configuration options which were left at their default values. It appears that the performance of the solvers can be increased by quite a large factor using careful design – the fact that a simple AES encryption took 60 seconds on our unoptimized platform is especially surprising. Since these solvers rely on heuristics to improve their performance, a set of heuristics for cryptanalysis needs to be developed. With a proper choice of heuristics we hope the performance of these attacks can be increased by several orders of magnitude, either by using SCIP or by evaluating a different PB solver. To this end, we have shared our cryptanalytic instances with the PB design community [MR09].

12.2. TASCA as Part of the Design Tool Chain

The specification language used to define PB optimization problems is rich enough to allow description of arbitrary Boolean circuits. It seems possible to write a compiler that receives a hardware description in a high-level language such as VHDL [IEE09] and outputs a PB-solver instance. Such a tool can be made part of a secure hardware design workflow, allowing designers to evaluate the susceptibility of their designs to side-channel attacks. By performing TASCA attacks with different subsets of the side-channel information, designers can assess the risk caused by exposure of various components of the internal state and so decide which components need a higher level of protection.

12.3. A strategy for successful TASCA attacks

As shown in this report, the choice of operating parameters can have a substantial effect on the running time and success probability of an algebraic side-channel attack. Increasing the amount of side-channel measurements available to the solver has a mixed effect: On one hand, as shown in [ZWG+11], it decreases the space of possible solutions sufficiently to allow the use of non-optimizing solvers with better running

times. On the other hand, it increases the sensitivity of the solver to noise. For a fixed amount of side-channel leaks, we showed that increasing the size of the set of acceptable values per leak (k) increases the running time but does not decrease the success probability of the attack.

In view of these findings, we can recommend that implementers first characterise the expected signal-to-noise ratio of the DUT using standard signal processing techniques; Next, they should choose the minimal amount of side-channel leaks (m) required for a successful key recovery; Finally, they should choose the minimal set size (k) which can tolerate the expected amount of noise with good probability. Specifically in the case of AES, we discovered that using m =100 leaks is a good choice for signal to noise ratios of as low as 20dB for k =1, up to 14dB for k =2, and up to 11dB for k =3.

12.4. Comparing Optimizers and Solvers

Optimizers are less efficient than solvers in terms of running time, but since a solver does not have any efficient way of representing the objective function which contains the aposteriori probabilities, its running time quickly becomes intractable when high robustness is desired. It may be possible to increase the robustness of the solver-based approach by finding a better way of choosing the set size k or support size k'. For example, instead of choosing the k' most likely value, the solver can set a threshold probability and include in its support all values with a higher probability than this threshold. The solver might also be used in an adaptive manner - slowly increasing the support size while the solver returns unsatisfiability, until we reach the minimal sized support for which a solution exists. Quite naturally, the opposite approach would be interesting too. Namely, the search for improved optimizers, allowing to represent more complex problems with reduced memory efficiency would be another way to close the gap between set-ASCA and TASCA. Finally, it would be interesting to carefully investigate the connection between the offline and online phases of a template attack on the success of template-TASCA and template set-ASCA. A better model obtained through better profiling in the offline phase should intuitively allow the use of lower-quality data in the online attack phase, and vice versa.

12.5. Contributed Data

The TASCA instances we created during the course of our research may be interesting to developers of constraint programming tools. Our instances have a highly regular structure due to the structure of symmetric block ciphers. Our experiments indicate that the solving difficulty of these instances is largely independent of the variable/clause ratio, but instead depends on domain-specific properties such as the

signal-to-noise ratio of the power trace. We have made available data sets both for the Keeloq and the AES ciphers, and both for the Hamming weight leakage model and for more elaborate leakages. These data sets are all available on our web site[OW10c, OW12a]. We have also submitted some of our instances to the yearly pseudo-Boolean competition [MR09] under the non-linear optimization category. Quite interestingly, the solving performance of these instances has improved considerably between consecutive PB competitions. For example, our most difficult Keeloq instance, *90_rounds_15_errors.opb*, took us 2990 seconds (49 minutes) to solve using SCIP version 1.2.0. In the 2011 pseudo-Boolean competition the same instance was solved in 450 seconds (7.5 minutes) by SCIP version 2.0.1.4. Finally, in 2012 the same instance was solved in only 50 seconds by SCIP 2.1.1.4 – a 60-fold increase in performance.

12.6. Practical Considerations for Solver Authors

As we evaluated different open-source solvers, we found that there are several properties which make some solvers better suited than others for our specific attack. Our observations may be of interest to writers and designers of constraint solving systems:

1. **Robust input handling**: We wrote our equation set in the variant of the OPB format specified by Manquinho and Rousselin in [MR09]. We chose this format to allow us to easily switch between the different submissions to the PB competition and evaluate their performance. However, we discovered that some solvers crashed or behaved unpredictably when receiving extremely large or improperly formatted input files. In addition, several solvers did not allow us to use meaningful variable names, requiring that all variables be named "x1", "x2", etc. Solver designers should take note that some instances may be hand-written and contain errors and domain-specific variable names, and be prepared to deal with such inputs.

2. **More detailed outputs:** The typical outputs produced by the solvers we evaluated contained either a description of the optimal instance – that is, the satisfying assignment and the corresponding value of the goal function – or an opaque claim of unsatisfiability. We were also interested in more detailed outputs in both cases – for satisfiable instances, we would like the solver to provide descriptions of satisfying but suboptimal assignments it considered and later rejected. In the case of unsatisfiability, we were interested in learning about the minimal clause set required to cause a logical contradiction. Finally, we were interested in any partial assignments discovered by the solver, even if the instance ultimately times out. We would have been able to use these partial outputs as the starting point to a cryptographic brute-force attack which quickly goes over all unassigned variables.

3. **Snapshots and suspend/resume functionality:** Some of our instances took several weeks to run. While these instances were running we had extremely limited information about their progress, mainly based on periodic outputs to a log file. In addition, any intentional or unintentional restart of the solver hardware (due to e.g. security updates, power failures, etc.) caused us to lose many days of computation time. We would have liked the solver to periodically save its current run-time state to disk for detailed analysis. More importantly, we would have wanted the ability to back up these snapshots, then restart the solver based on such a snapshot.
4. **Indication of key variables:** Some of our AES instances were over 5MB of size and contained more than 20,000 variables and clauses. Many of the variables in the instance described internal technical states of the AES cipher which were of no practical interest to us as attackers. In fact, we were only ultimately interested in the 128 variables describing the secret key bits. Since our solver had a "presolving" phase, at which it applied various Boolean optimizations to the problem instance to make it more efficient to solve, we would have liked the ability to indicate these critical variables ahead of time, to ensure they are not "optimized away" during the presolving step.

12.7. Conclusion

This part of my dissertation showed a new attack methodology called *Tolerant Algebraic Side-Channel Analysis (TASCA)*. This methodology transforms a single-trace side-channel analysis problem into a *pseudo-Boolean optimization problem* (PBOPT) form. The PBOPT syntax allows a very natural representation of measurement errors. Using this approach it is possible to mount successful single-trace attacks against real ciphers, even in the presence of realistic error rates. The noise-tolerant TASCA approach was shown to be usable for even for full-strength ciphers such as AES. The secret key can be recovered from 60%-70% of AES instances even when only a single trace is provided, and even when 20% of the trace signal is corrupted by noise. This new cryptanalytic capability may compromise secure systems whose defense against (statistical) side-channel attacks was an aggressive re-keying schedule which results in a small amount of traces per given key – even a single encryption is enough to recover the key, assuming that the device under attack has been properly profiled by the attacker.

Algebraic side-channel attacks can be carried out even if the leakage function does not conform to the Hamming weight model. The solver-based approach (set-ASCA) was shown to be faster than the optimizer-based approach (TASCA) when a high degree of robustness is not required (for example, if the traces can be preprocessed by averaging many traces). However, in cases when robustness is required, the optimizer approach was shown to be both faster and with higher success rate than the solver-based approach. This is due to the additional flexibility afforded by the

optimizer goal function, which allowed us to construct a generic representation of the measured leak as a vector of aposteriori probabilities. The new flexible representation presented here allows TASCA and set-ASCA attacks to be used as a natural match for template attacks. To carry out a combined Template-TASCA or Template-set-ASCA, the attacker should not only create templates for the original key bytes, but also for all intermediate values. The solver step will then replace any traditional post-processing step used in template attacks such as brute-force key enumeration. As a result, we further illustrated how an ASCA can be used effectively as a post-processing step of a template attack, dramatically reducing its data complexity. We believe that attention should be given to this capability when evaluating the security of systems using template attacks.

Part III.

Conclusion

The results described in this dissertation show how important it is to consider not only the algorithmic component of security, but also the physical aspects of building and breaking secure systems. The RFID systems described in the first part of this work are use a highly secure public-key infrastructure which is prohibitively expensive to crack, but our hardware-based attacks remove any notion of security from the system under attack. Similarly, the AES cryptosystem attacked in the second part of this work is all but invulnerable to standard cryptanalytic attacks, but a side-channel attacker with completely reasonable access to the system can break it quickly and effectively.

This work shows that the implementation aspects of any secure system must be considered with the same thoroughness as that of the algorithmic aspects. Indeed, as the science of cryptography advances, tomorrow's attackers will probably focus on hardware attacks and not on attacks on the underlying cryptosystem. This means that in the future the field of hardware attacks and countermeasures should become even more significant. It is important, then, to train researchers and practitioners dealing with security both on the theoretical and on the practical elements of security. Special care should be given to the underlying assumptions on which the system's security is established – what influence over the system does the attacker have? What sort of information does the learn attacker about the system as it operates? Most importantly, what is the goal of the attacker? How much damage must the attacker do before the system is considered broken?

As computers of various shapes and forms find a dominant role in additional areas of our life, making sure these computers are secure and trustworthy becomes more and more important. It is my hope that this dissertation advances the state of research on this significant topic.

Acknowledgments

Carrying out the research detailed in this dissertation required the help of many people, and I am deeply grateful to all of them for making the past few years some of the most enjoyable in my life. First of all I thank my advisor, Prof. Avishai Wool, for his ideas, his focus and his support. I learned much from his scientific insight and his practical advice.

Much of the research dealing with physical attacks on RFID was done in collaboration with Dvir Schirman (who co-authored Subsection 5.2 of this dissertation), and some of the equipment we used in our experiments was hand-manufactured by Ilan Kirschenbaum. The work dealing with side-channel attacks includes work done in collaboration with Mario Kirschbaum and Thomas Popp from Technische Universität Graz (who co-authored Subsection 8.1.2 of this dissertation), as well as work done in collaboration with Mathieu Renauld and François-Xavier Standaert from Université catholique de Louvain (who co-authored Subsections 11.6 and 11.7 of this dissertation). Other parts of my work were done with the help of Alex Arbit, Yoel Livne, Roy Gemer, Amir Mualem, Shay Cohen, Amit Erez and Doron Shutzberg.

I thank the anonymous referees of this dissertation for carefully reviewing my work and suggesting some improvements.

I would like to thank the Check Point Institute for Information Security, the Wallonie-Bruxelles International Research Scholarship and the Helmuth Heinemann Fellowship for their generous financial support during my studies.

I would also like to thank my parents and parents-in-law for helping me handle being both a researcher and as a father. Above all, I would like to thank my wife sincerely, for her help, support and inspiration during my studies.

Appendix: A sample TASCA Instance

The appendix demonstrates some of the equations used in a TASCA attack on AES with set size $k = 3$, following the notation introduced in Part II. We show a sample of the goal function, the plaintext assignment, the round functions and the measurement equations. The equations are given in the OPB format supported by the SCIP solver[BHPW09]. Full instances can be downloaded from our website at [OW12a].

```
* Plaintext Assignment: 16 equations of 8 bits each
* This compact form of assignment, which specifies 8 bits at a time,
* results in smaller instance sizes and
* better performance using SCIP, when compared to
* assigning a single bit per clause.
* s_0_0_[0:7] = 0xC5:
+1 s_0_0_0 +2 s_0_0_1 +4 s_0_0_2 +8 s_0_0_3 +16 s_0_0_4 ...
  +32 s_0_0_5 +64 s_0_0_6 +128 s_0_0_7 = 197 ;

[...]

* Round 1 of AES: up to 10 sets of equations (typically 1)
* s_1_[0:15]_[0:7] = AddKey(s_0_[0:15]_[0:7], k_[0:15]_[0:7]):
* s_1_0_0 = XOR(s_0_0_0, k_0_0)
 -1 s_1_0_0 +1 s_0_0_0 +1 k_0_0 -2 s_0_0_0 k_0_0 = 0 ;
* s_1_0_1 = XOR(s_0_0_1, k_0_1)
-1 s_1_0_1 +1 s_0_0_1 +1 k_0_1 -2 s_0_0_1 k_0_1 = 0 ;

[...]

* s_1_15_7 = XOR(s_0_15_7, k_15_7)
-1 s_1_15_7 +1 s_0_15_7 +1 k_15_7 -2 s_0_15_7 k_15_7 = 0 ;

* s_2_0_[0..7] = SubBytes(s_1_0_[0..7]):
-1 s_2_0_0 +1 ~s_1_0_0 ~s_1_0_1 ~s_1_0_2 ~s_1_0_3 ~s_1_0_4 ...
  ~s_1_0_5 ~s_1_0_6 ~s_1_0_7 +1 ~s_1_0_0 s_1_0_1 ~s_1_0_2 ...
  ~s_1_0_3 ~s_1_0_4 ~s_1_0_5 ~s_1_0_6 ~s_1_0_7 ...
 +1 ~s_1_0_0 s_1_0_1 s_1_0_2 s_1_0_3 s_1_0_4 s_1_0_5 s_1_0_6 s_1_0_7 = 0 ;

[...]
```

```
* s_4_[0:15]_[0:7] = ShiftRows+MixColumns(s_2_[0:15]_[0:7]) [8-bit]:

* [s_4_0, s_4_1, s_4_2, s_4_3]_[0:7] = ...
  ColumnXform([s_2_0, s_2_5, s_2_10, s_2_15]_[0:7]) [8-bit]:
* s_4_0_0_1_0 = XOR(s_2_0_0, s_2_5_0, s_2_10_0, s_2_15_0)
-1 s_4_0_0_1_0 -8 s_2_0_0 s_2_5_0 s_2_10_0 s_2_15_0  ...
  +4 s_2_0_0 s_2_5_0 s_2_10_0  +4 s_2_0_0 s_2_5_0 s_2_15_0 ...
  +4 s_2_0_0 s_2_10_0 s_2_15_0  +4 s_2_5_0 s_2_10_0 s_2_15_0 ...
  -2 s_2_0_0 s_2_5_0  -2 s_2_0_0 s_2_10_0  -2 s_2_0_0 s_2_15_0 ...
  -2 s_2_5_0 s_2_10_0  -2 s_2_5_0 s_2_15_0  -2 s_2_10_0 s_2_15_0 ...
   +1 s_2_0_0  +1 s_2_5_0 +1 s_2_10_0  +1 s_2_15_0 = 0 ;

[...]

* Side channel measurements:
* Measured Hamming weight for byte 0 of subround 0 = 4:
+1 s_0_0_0 +1 s_0_0_1 +1 s_0_0_2 +1 s_0_0_3 +1 s_0_0_4 ...
  +1 s_0_0_5 +1 s_0_0_6 ...
  +1 s_0_0_7 +1 e_s_0_0_p -1 e_s_0_0_n = 4 ;
* Measured Hamming weight for byte 1 of subround 0 = 5:
+1 s_0_1_0 +1 s_0_1_1 +1 s_0_1_2 +1 s_0_1_3 +1 s_0_1_4 ...
  +1 s_0_1_5 +1 s_0_1_6 +1 s_0_1_7 +1 e_s_0_1_p -1 e_s_0_1_n ...
  = 5 ;

[...]

* Measured Hamming weight for byte 15 of subround 3 = 0:
+1 s_3_15_0 +1 s_3_15_1 +1 s_3_15_2 +1 s_3_15_3 ...
  +1 s_3_15_4 +1 s_3_15_5 +1 s_3_15_6 +1 s_3_15_7 ...
  +1 e_s_3_15_p -1 e_s_3_15_n = 0 ;

* Goal term:
* each side channel introduces two error variables:
* "_p" for positive, and "_n" for negative error.
* The goal function minimises the sum of all these variables.
min: +1 e_s_0_0_p +1 e_s_0_0_n +1 e_s_0_1_p +1 e_s_0_1_n ...
   +1 e_k_15_p +1 e_k_15_n;
```

Bibliography

[Ach07] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.

[ADF+10] Kahraman Akdemir, Martin Dixon, Wajdi Feghali, Patrick Fay, Vinodh Gopal, Jim Guilford, Erdinc Ozturc, Gil Worlich, and Ronen Zohar. Breakthrough AES Performance with Intel AES New Instructions. Technical report, Intel Corporation, October 2010. `http://software.intel.com/file/27067`.

[BBWW07] Joachim Biskup, Dominique Marc Burgard, Torben Weibert, and Lena Wiese. Inference control in logic databases as a constraint satisfaction problem. In *Information Systems Security*, pages 128–142. Springer, 2007.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *CHES*, pages 16–29, 2004.

[BHP09] T. Berthold, S. Heinz, and M. E. Pfetsch. Nonlinear pseudo-boolean optimization: Relaxation or propagation? In O. Kullmann, editor, *SAT 2009*, volume 5584 of *LNCS*, pages 441–446. Springer, 2009.

[BHPW09] T. Berthold, S. Heinz, M. E. Pfetsch, and M. Winkler. SCIP – solving constraint integer programs. SAT 2009 competitive events booklet, 2009. `http://www.cril.univ-artois.fr/SAT09/solvers/booklet.pdf`.

[BKL+07a] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES*, pages 450–466, 2007.

[BKL+07b] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES*, pages 450–466, 2007.

[BW05] D. Bertsimas and R. Weismantel. *Optimization Over Integers*. Dynamic Ideas, 2005.

[C1G05] Epcglobal inc., EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz – 960 MHz, version 1.0.9. Online, September 2005.

[Can05] D. Canright. A very compact S-Box for AES. In J. R. Rao and B. Sunar, editors, *CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.

[CB07] Nicolas T. Courtois and Gregory V. Bard. Algebraic cryptanalysis of the data encryption standard. In Steven D. Galbraith, editor, *Cryptography and Coding*, volume 4887 of *Lecture Notes in Computer Science*, pages 152–169. Springer Berlin Heidelberg, 2007.

[CBW08] N. Courtois, G. V. Bard, and D. Wagner. Algebraic and slide attacks on KeeLoq. In K. Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 97–115. Springer, 2008.

[CFGR12] Claude Carlet, Jean-Charles Faugère, Christopher Goyet, and Guénaël Renault. Analysis of the algebraic side channel attack. *J. Cryptographic Engineering*, 2(1):45–62, 2012.

[Com09] Common Criteria Recognition Agreement. Common criteria for information technology security evaluation part 2: Security functional components. Online, July 2009. `http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R3.pdf`.

[Con76] John Horton Conway. *On Numbers and Games*. Academic Press, 1976.

[CRR02] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 13–28. Springer, 2002.

[Daw98] S. Dawson. Code hopping decoder using a PIC16C56. Microchip confidential, leaked online in 2002, 1998.

[DGB87] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In *CRYPTO*, pages 21–39, 1987.

[DM07] Saar Drimer and Steven J. Murdoch. Keep your enemies close: distance bounding against smartcard relay attacks. In *Proceedings of 16th USENIX Security Symposium*, pages 1–16, Boston, MA, 2007. USENIX Association.

[Dol09] Boaz Dolev. Laying the groundwork for electronic elections in Israel (in Hebrew). Invited Talk, CPIIS IDC/TAU Workshop on Electronic Voting, May 2009. `http://www.cs.tau.ac.il/voting/`.

[DR98] J. Daemen and V. Rijmen. AES proposal: Rijndael, 1998.

[EC06] European commission decision c(2006) 2909: Technical specifications on the standards for security features and biometrics in passports and travel documents. Online, June 2006. `http://ec.europa.eu/justice_home/doc_centre/freetravel/documents/doc_freetravel_documents_en.htm`.

[EKM+08] T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the Keeloq code hopping scheme. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 203–220. Springer, 2008.

[EMV09] EMVCo contactless communication protocol specification v2.0.1. Online, July 2009. `http://www.emvco.com/specifications.aspx?id=21`.

[Fin] Klaus Finkenzeller. Personal communication.

[Fin03] Klaus Finkenzeller. *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, 2003.

[FPB11] Klaus Finkenzeller, Florian Pfeiffer, and Erwin Biebl. Range Extension of an ISO / IEC 14443 type A RFID System with Actively Emulating Load Modulation. In *7th European Workshop on Smart Objects: Systems, Technologies and Applications (RFID SysTech)*, May 2011.

[Gov08] Government of Israel, Ministry of the Interior. Public tender 16-2008 for the establishment and operation of a computerized election system, August 2008.

[Gov09] Government of Israel, Prime Minister's Office. Decisions of the ministerial committee on legislation (in hebrew). Online, August 2009. `http://www.pmo.gov.il/PMO/vadot/hakika/2008-2012/08-2009/des663.htm`.

[Han05] Gerhard Hancke. A Practical Relay Attack on ISO 14443 Proximity Cards. Manuscript, February 2005.

[Han06] Gerhard P. Hancke. Practical attacks on proximity identification systems (short paper). In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 328–333, Oakland, CA, 2006. IEEE Computer Society.

[HMM09] Gerhard P. Hancke, K. Mayes, and K. Markantonakis. Confidence in smart token proximity: Relay attacks revisited. *Computers & Security*, In Press, Accepted Manuscript:–, 2009. `http://www.science-direct.com/science/article/B6V8G-4WJBBP3-3/2/936ae8bac0b8b1f86fe246f134b4ea95`.

[HSP08] Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. Rfid and its vulnerability to faults. In *CHES '08: Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, pages 363–379, Berlin, Heidelberg, 2008. Springer-Verlag.

[IEE09] IEEE standard VHDL language reference manual. *IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002)*, pages c1 –626, 26 2009.

[Ins05] Texas Instruments. Multi function reader series 4000. Online, March 2005. http://www.ti.com/rfid/docs/manuals/pdfSpecs/RF-MFR-RNLK-00.pdf.

[Int08] Intel Corporation. Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors. Technical report, November 2008. http://download.intel.com/design/processor/applnots/320354.pdf.

[ISO01] International Organization for Standardization, Geneva. *ISO/IEC 14443-2 Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 2: Radio frequency power and signal interface*, 2001.

[JJ05] Dejan Jovanović and Predrag Janičić. Logical analysis of hash functions. In Bernhard Gramlich, editor, *Frontiers of Combining Systems*, volume 3717 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin Heidelberg, 2005.

[JRS03] Ari Juels, Ronald L. Rivest, and Michael Szydlo. The blocker tag: selective blocking of rfid tags for consumer privacy. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 103–111. ACM Press, 2003. http://doi.acm.org/10.1145/948109.948126.

[KJJ99] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[KW03] C. Karlof and D. Wagner. Hidden Markov model cryptoanalysis. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 17–34. Springer, 2003.

[KW05] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcards. In *International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 47–58, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[KW06] Ilan Kirschenbaum and Avishai Wool. How to build a low-cost, extended-range RFID skimmer. In *Proceedings of the 15th USENIX Security Symposium*, Vancouver, B.C., Canada, 2006. USENIX Association.

[Ltd08] Advanced Card Systems Ltd. ACR122U NFC contactless smart card reader. Online, August 2008. http://www.acs.com.hk/index.php?pid=product&prod_sections=0&id=ACR122U.

[Man02] S. Mangard. A simple power-analysis (SPA) attack on implementations of the AES key expansion. In P. J. Lee and C. H. Lim, editors, *ICISC 2002*, volume 2587 of *LNCS*, pages 343–358. Springer, 2002.

[MBZ+12] Mohamed Saied Emam Mohamed, Stanislav Bulygin, Michael Zohner, Annelie Heuser, and Michael Walter. Improved Algebraic Side-Channel Attack on AES. Cryptology ePrint Archive, Report 2012/084, 2012. http://eprint.iacr.org/2012/084.

[MC09] Mini-Circuits. ZHL-32A coaxial amplifier. Online, August 2009. http://www.minicircuits.com/pdfs/ZHL-32A.pdf.

[Mif03] Easing travel in london's congested public transport network. Online, June 2003. http://mifare.net/showcases/london.asp.

[MM00] F. Massacci and L. Marraro. Logical cryptanalysis as a SAT problem. *J. Autom. Reason.*, 24(1-2):165–203, 2000.

[MM05] Tim "Minime" and Christopher "Mahajivana". RFID zapper. 22nd Chaos Communication Congress, December 2005. https://events.ccc.de/congress/2005/wiki/RFID-Zapper(EN).

[MOP07] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[MR09] V. Manquinho and O. Roussel. Pseudo-boolean competition 2009. Online, http://www.cril.univ-artois.fr/PB09/, July 2009.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[MZ06] Ilya Mironov and Lintao Zhang. Applications of SAT solvers to cryptanalysis of hash functions. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115. Springer Berlin Heidelberg, 2006.

[N3l06] N3ldan. Cheap/free capacitor bank and charger. Online, October 2006. http://www.instructables.com/id/Cheapfree-capacitor-bank-and-charger/.

[Nat99] National Institute of Standards and Technology. *FIPS PUB 46-3: Data Encryption Standard (DES)*. National Institute for Standards and Technology, Gaithersburg, MD, USA, October 1999.

[Nat01] National Institute of Standards and Technology . *FIPS PUB 197: Announcing the Advanced Encryption Standard (AES)*. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[NT08] New-Tronics. mobile HF hustler antenna. Online, October 2008. http://www.new-tronics.com/main/html/mobile__hf.html.

[oFSU10] Ministry of Finance Spokesman Unit. Comments on the Haaretz article about computerized elections (in Hebrew). Online, April 2010. http://www.eng.tau.ac.il/~yash/RFID/tehila-response.pdf.

[OKPW10] Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. Algebraic Side-Channel Analysis in the Presence of Errors. In *CHES*, pages 428–442, 2010. http://iss.oy.ne.ro/TASCA.

[Ore] Yoram Abraham Oren. Personal communication.

[ORM+10] Yoram Abraham Oren, Pinchas Rosenblum, Ofer Margoninsky, Ilan Yom-Tov, and Boaz Dolev. (wo/2010/010564) electronic voting system. Online, January 2010. http://www.wipo.int/patentscope/search/en/WO2010010564.

[ORSW12] Yossef Oren, Mathieu Renauld, François-Xavier Standaert, and Avishai Wool. Algebraic side-channel attacks beyond the hamming weight leakage model. In Patrick Schaumont and Emmanuel Prouff, editors, *Workshop on Cryptographic Hardware and Embedded Systems 2012 (CHES 2012), LNCS 7428*, pages 140–154, Leuven, Belgium, 2012. International Association for Cryptologic Research, Springer. http://iss.oy.ne.ro/Template-TASCA.

[OSW12] Yossef Oren, Dvir Schirman, and Avishai Wool. RFID jamming and attacks on israeli e-voting. In *Smart SysTech 2012 – European Conference on Smart Objects, Systems and Technologies*, pages 0–999, Munich, Germany, 6 2012.

[otISU07] Ministry of the Interior Spokesman Unit. Pilot of computerized elections for regional councils (in hebrew). Online, November 2007. http://www.israel.gov.il/FirstGov/Templates/NewsItem.aspx?NRNODEGUID={6CA2D671-1427-46A1-91D7-9C8957E733EB}.

[OW] Yossef Oren and Avishai Wool. Side-channel cryptographic attacks using pseudo-boolean optimization. *Manuscript submitted to a journal.*

[OW10a] Yossef Oren and Avishai Wool. Israeli e-voting RFID card zapper. Online, April 11 2010. http://youtu.be/wxd3-YodOmM.

[OW10b] Yossef Oren and Avishai Wool. RFID-Based electronic voting: What could possibly go wrong? In *International IEEE Conference on RFID*, pages 118–125, Orlando, USA, 4 2010.

[OW10c] Yossef Oren and Avishai Wool. TASCA-on-keeloq pseudo-boolean instances. Online, 2010. http://iss.oy.ne.ro/TASCA/Instances.

[OW12a] Yossef Oren and Avishai Wool. Template TASCA pseudo-boolean instances. Online, 2012. http://iss.oy.ne.ro/Template-TASCA/Instances.

[OW12b] Yossef Oren and Avishai Wool. Tolerant Algebraic Side-Channel Analysis of AES. Cryptology ePrint Archive, Report 2012/092, 2012. http://iss.oy.ne.ro/TASCA-eprint.

[PHH08] Thomas Plos, Michael Hutter, and Christoph Herbst. Enhancing side-channel analysis with low-cost shielding techniques. In Christoph Lackner, Timm Ostermann, Michael Sams, and Ronald Spilka, editors, *Proceedings of Austrochip 2008, Linz, Austria, October 8, 2008, Proceedings*, pages 90 – 95, 2008.

[PRR+07] N.R. Potlapally, A. Raghunathan, S. Ravi, N.K. Jha, and R.B. Lee. Aiding side-channel attacks on cryptographic software with satisfiability-based analysis. *IEEE Trans. on VLSI Systems*, 15(4):465 –470, april 2007.

[RCT07] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum. Keep on blockin' in the free world: personal access control for low-cost RFID tags. In *Proceedings of the 13th international conference on Security protocols*, pages 51–59, Berlin, Heidelberg, 2007. Springer-Verlag.

[RS09] M. Renauld and F.X. Standaert. Algebraic Side-Channel Attacks. In Dongdai Lin Jiwu Jing Feng Bao, Moti Yung, editor, *Information Security and Cryptology (INSCRYPT) 2009*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 12 2009.

[RSVC09] M. Renauld, F.-X. Standaert, and N. Veyrat-Charvillon. Algebraic side-channel attacks on the AES: Why time also matters in DPA. In C. Clavier and K. Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 97–111. Springer, 2009.

[RSVC+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In *EUROCRYPT*, pages 109–128, 2011.

[RTS] Alon Rosen and Amnon Ta-Shma. Electronic voting in Israel. Online. http://sites.google.com/site/evotingisrael/.

[Run05] Tilman Runge. 22nd chaos communication congress lightning talks, day 1. Online, December 2005. http://youtu.be/uXEJl_I49MQ#t=18m58s.

[Run07] Tilman Runge. Schriftliche arbeit jugend forscht: Der RFID-Zapper (in German). Online, February 2007. http://rfidzapper.dyndns.org/RFID-ZAPPER.pdf.

[sat] SAT 2011 Competition. http://www.cril.univ-artois.fr/SAT11/phase2.pdf.

[Sat04] H. Satyanarayana. AES128 package. Online, December 2004. http://opencores.net/project,aes_crypto_core.

[Shi09] Meir Shitrit. Local authorities bill (elections) (amendment - election systems) (in hebrew). Online, May 2009. http://www.knesset.gov.il/privatelaw/data/18/1180.rtf.

[Sin10] Carsten Sinz. SAT-Race 2010. http://baldur.iti.uka.de/sat-race-2010/results.html, 2010.

[SKS+07] Omer Al Saraereh, Abdul Karem, A Al Sbeeh, Ahmad H Zaid, and Ibrahim M Hruob. Monopole Antenna. *Computer Engineering*, pages 2–29, 2007.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer Berlin Heidelberg, 2009.

[Soo] Mate Soos. CryptoMiniSat2. http://www.msoos.org/cryptominisat2/.

[Str03] R.D. Straw. *The ARRL antenna book: The Ultimate Reference for Amateur Radio Antennas*. Amer Radio Relay League, 2003.

[Str07] J.A. Stratton. *Electromagnetic theory*, volume 33. Wiley-IEEE Press, 2007.

[Tec08] Agilent Technologies. E4438C ESG vector signal generator. Online, May 2008. http://www.home.agilent.com/agilent/product.jspx?nid=-536902340.536880956.

[TSTMM11] P.H. Thevenon, O. Savry, S. Tedjini, and R. Malherbi-Martins. Attacks on the HF physical layer of contactless and RFID systems. In Cornel Turcu, editor, *Current Trends and Challenges in RFID*, chapter 21. InTech, July 2011.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260 – 269, Apr 1967.

[Wei09] Rivka Weil. Big brother and the slippery slope: On the challenges of e-voting (in Hebrew). Invited Talk, CPIIS IDC/TAU Workshop on Electronic Voting, May 2009. http://www.cs.tau.ac.il/voting/.

[Wun96] R. Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, Technische Universität Berlin, 1996.

[ZWG+11] Xinjie Zhao, Tao Wang, Shize Guo, Fan Zhang, Zhijie Shi, Huiying Liu, and Kehui Wu. SAT based Error Tolerant Algebraic Side-Channel Attacks. 2011 Conference on Cryptographic Algorithms and Cryptographic Chips (CASC2011), July 2011.

תוכן העניינים

אנו מתארים כיצד ניתן להשתמש בהתקפות ממסר ובהתקפות אחרות בעלות עלות נמוכה יותר כדי להתקיף את האמינות, המהימנות והאנונימיות של מערכת ההצבעה האלקטרונית המוצעת, ומהם אמצעי הנגד איתם ניתן להגן מפני התקפות אלה.

התוצאות המתוארות בעבודה זו מראות עד כמה חשוב לקחת בחשבון לא רק את המרכיב האלגוריתמי של אבטחה, אלא גם את ההיבטים הפיסיים של בנייה ותקיפה של מערכות מאובטחות.

תקציר

כל פונקציונליות אבטחה, כגון הצפנה או אימות, יש ליישם בעולם האמיתי לפני שניתן יהיה להביאה לשימוש מעשי. יישום זה מגיע בצורה של יישום תוכנה למכשיר למטרות כלליות, כגון מחשב אישי, או לחילופין כהתקן חומרה מאובטחת ייעודי, שמטרתו העיקרית היא לגלם את פונקציונליות האבטחה. דוגמאות להתקני חומרה מאובטחים כוללים כרטיסים חכמים, אזעקות לרכב ופתקי הצבעה ממוחשבים. כדי להעריך את האבטחה של מערכת מאובטחת, חוקרים מחפשים פגמים המאפשרים לתוקף לפרוץ את הנחות האבטחה של המערכת (לדוגמה, כיצד לאפשר לגורם בלתי מורשה להציג או לשנות הודעה שמיועדת למישהו אחר). התקפות פיסיות (המכונות גם התקפות יישום) פוגעות במערכת על ידי ניצול של ההיבטים הפיסיים של היישום של האלגוריתם. התקפות פיסיות מסויימות (כמו, למשל, ניתוח הספק) משחזרות את המפתח הסודי המשמש את המכשיר והמאובטח על ידי ניתוח השפעות פיסיות הנוצרות במהלך השימוש בו, ואחרות (כגון, למשל, התקפות ממסר) משביתות או מגבילות את ההתנהגות הבטוחה של המערכת על ידי ניצול פגמי תכנון או יישום, או על ידי שינוי הנחות היסוד שנעשו על ידי המעצבים של המערכת.

מחקר זה מתמקד בהתקפות פיסיות על התקני חומרה מאובטחים ועל אמצעי-הנגד אשר מאפשרים להגן מפני התקפות אלו. המטרות שלי היו לחקור נקודות תורפה ביישומי חומרה מאובטחים הנוכחיים ולהעריך את היעילות של אמצעי-הנגד הנוכחיים המוצעים נגד נקודות תורפה אלה. שני המסלולים העיקריים של המחקר שלי הם ניתוח ערוצי צד (ובמיוחד ניתוח הספק) ו-RFID מאובטח.

במסלול הניתוח בצד הערוץ, חקרתי דרכים לצמצום דרישות הנתונים להתקפות ניתוח הספק. הראנו איך לבצע התקפות שחזור מפתח במכשיר מאובטח באמצעות כמות נמוכה מאוד של נתוני מדידה. החידוש העיקרי של ההתקפה שלנו היה השימוש בכלי אופטימיזציה פסאודו בוליאני - כלי רב עוצמה לפתרון בעיות אילוץ - כדי לחקור במהירות וביעילות את החלל העצום של פתרונות אפשריים ולמצוא את הפיתרון הנכון.

במסלול ה-RFID המאובטח, חקרתי התקפות פיסיות על מערכות RFID שדה קרוב, במיוחד בהתייחסות למערכת ההצבעה האלקטרונית הישראלית המוצעת. המאפיינים האלקטרוניים של מערכות RFID שדה קרוב מאלצים טווח קצר מאוד בין התג והקורא (בדרך כלל פחות מ 10 ס"מ) כדי שהתג יוכל לפעול. תכונה זו מובילה להנחה המשתמעת (והמוטעית) כי בכל פעם בה קורא יכול לתקשר עם תג, ניתן להניח שהתג הוא מבחינה פיזית קרוב מאוד לקורא. התקפות ממסר מאתגרות הנחת יסוד זו ומאפשרות מרחק בלתי מוגבל כמעט בין התג והקורא.

אוניברסיטת תל – אביב

הפקולטה להנדסה ע"ש איבי ואלדר פליישמן

בית הספר לתארים מתקדמים ע"ש זנדמן-סליינר

חומרה מאובטחת – התקפות פיסיות ואמצעי הגנה

יוסף אורן

חיבור לשם קבלת התואר "דוקטור לפילוסופיה"

הוגש לסנאט של אוניברסיטת תל-אביב

עבודה זו נעשתה באוניברסיטת ת"א בפקולטה להנדסה
בהדרכת פרופ' אבישי וול

אייר תשע"ג

אוניברסיטת תל – אביב

הפקולטה להנדסה ע"ש איבי ואלדר פליישמן

בית הספר לתארים מתקדמים ע"ש זנדמן-סליינר

חומרה מאובטחת – התקפות פיסיות ואמצעי הגנה

חיבור לשם קבלת התואר "דוקטור לפילוסופיה"

יוסף אורן

הוגש לסנאט של אוניברסיטת תל-אביב

אייר תשע"ג