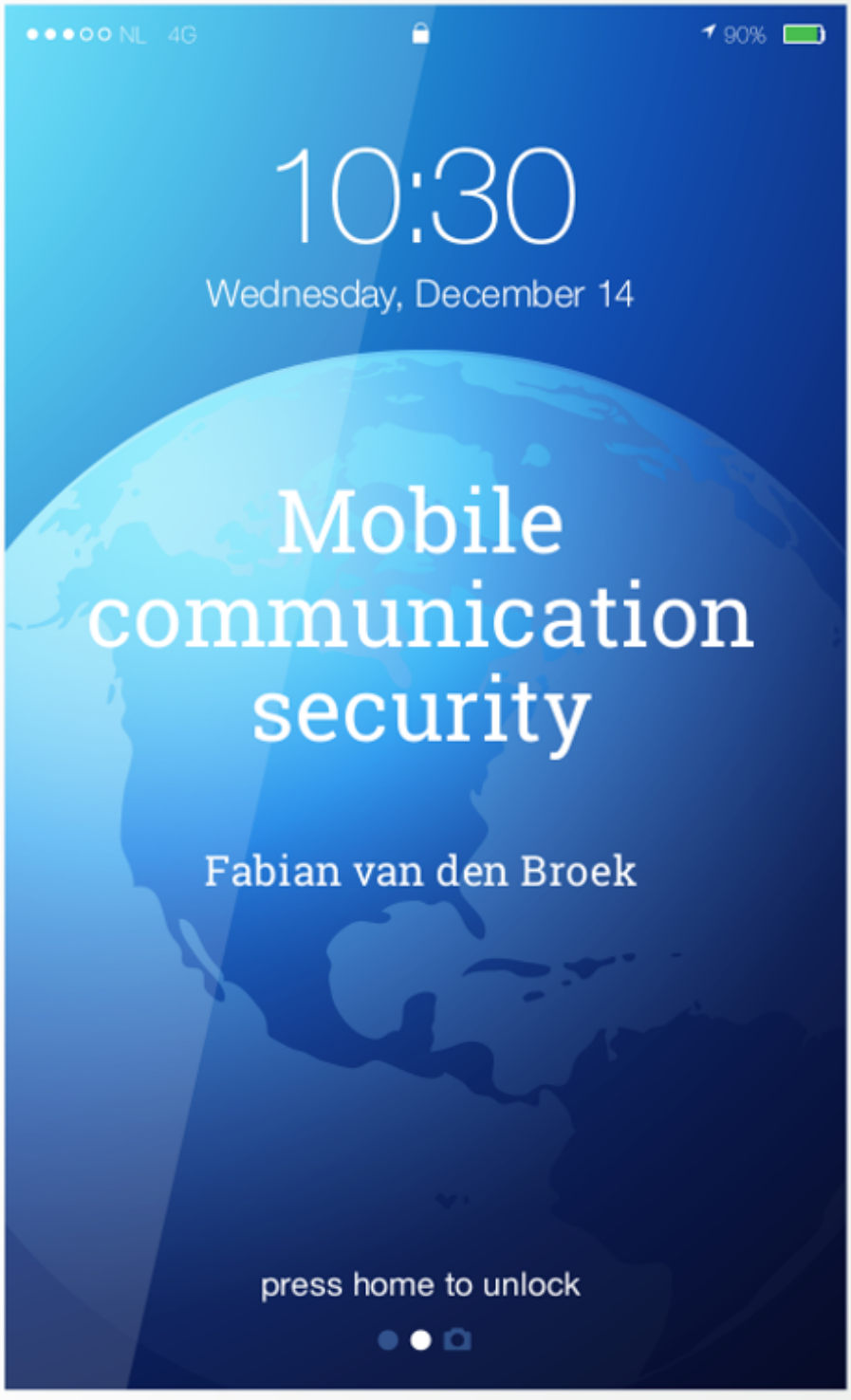

NL 4G
90%
10:30
Wednesday, December 14
Mobile communication security
Fabian van den Broek
press home to unlock

Mobile communication security

Fabian van den Broek, 2016
ISBN: 978-94-028-0427-0
IPA Dissertation Series: 2016-13

Typeset using LaTeX

Cover design
Promotie In Zicht, Arnhem

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/

Mobile communication security

Proefschrift ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.H.J.M. Van Krieken,
volgens besluit van het college van decanen
in het openbaar te verdedigen op

woensdag 14 december 2016,

om 10:30 uur precies

door

Fabian Marinus Jan van den Broek

geboren op 14 augustus 1981

te Beers

Promotor:

prof. dr. B.P.F. Jacobs

Copromotor:

dr. ir. E. Poll

Manuscriptcommissie:

prof. dr. E.R. Verheul
prof. dr. K. Mayes (Royal Holloway University of London, GB)
prof. dr. JP. Seifert (Technische Universität Berlin, DE)
prof. dr. A. Pras (University of Twente, NL)
dr. E. Reitsma (Ministry of Internal Affairs, NL)

Acknowledgements

Over the years a lot of people have contributed either to this thesis directly, or to my personal well-being in some way, and I thank them all. Still, there are some I would like to explicitly mention here.

First of all, I would like to thank my doctoral supervisor, Bart Jacobs, who was also my Master thesis supervisor (and actually suggested the topic) which evolved into this PhD project. After this PhD project, he asked me to come work within a project on the IRMA credential technology. Both were very different, but great experiences, that I would have never had without him.

My utmost thanks as well to my daily supervisor Erik Poll, who is always available for questions and feedback. I am constantly amazed at his ability to do a context switch from whatever he is working on when I come knocking at his office door, to whatever question I have this time.

I greatly enjoyed working with Bart and Erik because they both possess two important qualities: enthusiasm, which is always useful in academia, but also a great sense of humour, which is equally important within security research.

My thanks as well to my reading committee, Eric Verheul, Keith Mayes, Jean-Pierre Seifert, Aiko Pras and Erik Reitsma, for taking the time and effort to read this thesis and for their helpful comments.

Writing a PhD thesis might be a solitary job, but working in science definitely is not. I greatly valued the collaboration with my co-authors over the years: Erik, Ronny, Brinio, Arturo, Bárbara, Brinda, Gergely, Bart, Roel and Joeri. For the last two years I have been working with Wouter and Sietse on the IRMA code base and I really enjoy this collaboration, which has taught me much. The progress we achieved with just the three of us is impressive.

Besides those who I have worked with directly, all the other colleagues of the Digital Security group, both current and past, deserve my gratitude as well for amazing discussions, great advice, hilarious coffee breaks, or just for the fantastic atmosphere.

Het werk binnen de academische wereld en zeker binnen een promotietraject kan af en toe ook enorm frustreren. Dan is het prettig af te kunnen spreken met vrienden voor de nodige relativering. Velen van hen hebben een bijdrage geleverd aan dit werk zonder zich dat zelf te beseffen, maar daarnaast zorgen zij ervoor dat ik een sociaal leven houd, zelfs als een proefschrift of het vaderschap veel energie vergt.

Mijn vrienden laten zich makkelijk indelen in enkele WhatsApp groepen (toepasselijk gezien het onderwerp van dit proefschrift) met de veelzeggende namen: *Everything is awesome*, *Borreltijd!*, *De centrale* en *Merlet*. Dan mis ik alleen nog Tom, wat ook toepasselijk is, want hij laat zich niet indelen. Bedankt allemaal.

In het bijzonder wil ik hier nog mijn paranimfen noemen en bedanken. Freek en Maarten, het is voor mij een eer naast jullie te staan tijdens de verdediging [Mulders07]. En ook daarbuiten, natuurlijk [Verbeek13].

Mijn lieve ouders, zonder jullie zou ik hier nooit hebben gestaan en dat gaat veel verder dan alleen in biologische zin. Jullie lieten mij vrij om mijn eigen keuzes te maken en steunen mij in alles, bedankt.

Lieve Klaartje, tijdens dit promotietraject zijn we getrouwd en samen kregen we twee prachtige kinderen, maar bovenal was en blijf je mijn maatje. Je bent geweldig. Bedankt voor alles.

En dan uiteindelijk Caspar en Simon, ik kan onmogelijk beweren dat jullie positief hebben bijgedragen aan dit werk, maar jullie hebben mijn leven zelf enorm verrijkt. Dankzij jullie heb ik de meest waardevolle titel van mijn leven al binnen: die van jullie papa.

Nijmegen, October 2016
Fabian van den Broek

Contents

Chapter 1
Introduction

In 1882 Almon Brown Strowger bought an undertaking business in Topeka, Kansas. After some successful years business started to decline. When Strowger learnt a friend of him died, he was shocked: not only by the loss, but also to find the funeral arrangements were being made by a competitor. Being a suspicious man by nature, Strowger investigated and would later claim his financial trouble was accountable to a specific switch-board operator who was romantically involved with (according to some stories, she was actually married to) his competitor. Strowger suspected the switch-board operator was connecting calls meant for him to his competitor, thereby providing the first known instance of an impersonation attack on the telephone system [101, 168].

Almon Strowger thought it was ridiculous that callers had to rely on switch-board operators to connect them to the right person and trust these operators to not listen in on phone conversations. He decided to do something about the weakest element in the phone system and set out to remove the human component: the switch-board operator. Together with his nephew Walter Strowger he invented the Strowger switch, an electromechanical phone switch allowing users to select the phone they wish to connect to (initially from a set of a 100 contacts) without the intervention of an operator, by sending electrical pulses over the phone lines. On March 10, 1891 he was awarded the patent (US Patent No. 447918 10/6/1891) for his invention and on November 3 1892 the first Strowger switch was taken into use in La Porte, Indiana. This was quite an occasion, celebrated with a brass band and a special train run from Chicago. Apparently Almon Strowger described his revolutionary system as being *"girl-less, cuss-less, out-of-order-less, and wait-less."* By automating his phone system Strowger removed the untrusted switch-board operators and replaced them with automated machines, trusting the automation instead of the person.

Automation, however, still leaves the customer at the hands of the people installing and maintaining the automated devices. Furthermore, automated systems lack any sense of context or sanity checks, which allows users to trigger behaviour that makes little sense within the current context, but nets the user some benefit. An example of this is phone phreaking, which started when Joe Engressia found out that whistling a tone at 2600 Hz (for piano players the fourth E above middle C) during a phone conversation would drop the current conversation, but keep the line open [147]. This could be exploited by starting a call to a toll-free number, sending the 2600Hz

tone, and then dialling the actual number you wished to call. The resulting call would not be billed to the user, as the phone switch regarded the user's phone line as used for a free call. Joe Engressia was gifted with perfect pitch, but other less or otherwise gifted individuals started looking for reliable ways to generate the required tones. In a more humorous example it turned out that the toy whistle that came in a box of Cap 'n Crunch cereal could produce the exact 2600 Hz tone needed when one hole was covered. In the end several tones were found which influenced switches and some people started making devices, called boxes, to generate the right tones for the required durations. The best known of these were the original blue boxes, which influenced the call routing, but many different boxes appeared, such as red boxes (faking coin drops in a payphone) and black boxes (allowing other people to call you for free).

Phone phreaking had its peak in the late 1960s and early 1970s. The weakness at the heart of these attacks was that the control signals where transmitted over the same connection as the user data and this so-called in-band signalling allowed attackers to insert fake control signals over their phones. When phone companies switched to out-of-band signalling in the 1980s, phone phreaking pretty much went away, although the drop in costs for domestic phone conversation probably also removed a lot of the incentive [148]. Currently, practically all of the switched phone network uses out-of-band signalling; still, it was discovered in 2005 that common police wiretapping equipment was backwards compatible with the old in-band signalling, which could be used to stop the recording of phone conversations by this equipment by transmitting the 2600 Hz signal [153].

The two examples discussed above illustrate that the telephone network has always been under attack and naturally this did not change when mobile telephony was introduced in the latter half of the 1980s. The first generation of cellular telephony was analogue and had no real authentication [148]. Mobile phones simply identified themselves with a serial number transmitted plain text over the air (the U.S. system required an additional number to identify the subscriber). This led to e.g. call-sell fraud, where an attacker would capture the serial numbers of mobile phones off the air and re-program their phones with these serials. The attacker would then sell phone minutes on his phone to people wanting to make cheap phone calls.

In a recurring story for (mobile) telephony, the lack of security and resulting fraud did not hamper the success of the technology. The prognosis of 500,000 subscribers by 1990 was greatly exceeded by the actual number of 10 million subscribers in 1990 and 15 million in 1992 [101]. This enormous and unforeseen popularity caused a degraded service for costumers as the large numbers of subscribers within urban areas caused congestion on the wireless part of the network. This drove the need for a digital mobile telephony system, since digital systems can use data compression and allow for the shared use of frequencies over several phones at the same time. Additionally, digital systems improve transmission quality, so more users can enjoy a better service when switching to a digital system. However, seeing the large amount of fraud occurring in the analog system, the designers of the different digital systems were given the explicit design goal of adding security. Which finally brings us to the subject of this thesis: the security of mobile communication.

1.1 Research question

Now that we have come to the subject of this thesis we come to the research question:

How secure is mobile communication in practice?

Now this is of course a rather vague question, which requires more specificity on what is meant by "secure" and "mobile communication." As we discuss in more detail in the rest of this section, by mobile communication we mean the 3GPP family of networks, and as attacker model we consider an attacker with access to the wireless interface. But first, lets consider the "in practice" part of the research question. This thesis covers some practical work, e.g. by verifying how difficult it is to eavesdrop conversations in practice in Chapter 3, but this qualification signifies that we do not just want to analyse the theoretical strength of security and theoretical improvements, but we also want to consider the practicality. So, for instance, when we look at the encryption used to protect mobile communication in Chapter 4, we look at the encryption used in our local networks at that time, not the much better encryption proposed – but either not supported or simply unused in our local situation. Also, when suggesting possible improvements to the mobile networks in Chapter 8, we take care that these improvements can be added to the current network infrastructure without too much trouble. In fact, we are currently working on getting the improvements from Chapter 8 introduced in the official specifications, so they can be more easily adopted by manufacturers.

1.1.1 Mobile communication

There are many different kinds of mobile communication technologies, such as satellite telephony, DECT, and WiFi. It is therefore necessary to define the scope of this thesis, with respect to mobile communication. In this thesis we will only look at global cellular technology. That is technology in which the geographic area being served is divided up into cells. These cells allow for frequency re-use between non-neighbouring cells. Because of interference, two neighbouring cells can never use the same frequencies. This is schematically shown in Figure 1.1, which also shows that cells can differ in size, for instance to accommodate a densely populated area. This figure is a simplification of the practical situation in which one cell tower often contains three transmitters, each handling 120° angle around the tower. This does not change anything about the general working of a cell tower, it just defines a smaller cell. The advantages of a cellular network come at a cost. The providers need to do extra book keeping to know in which cell a subscriber is currently located, in order to be able to route incoming traffic to the correct cell. The cellular approach also requires extra protocols in order to allow a subscriber a continued uninterrupted service as he moves around between cells. So, the often used term (also in this thesis) of "mobile networks" is thus something of a misnomer, since the networks themselves are not mobile, but they allow the subscribers to be mobile.

Even within cellular technologies there are way too many systems to discuss all of them in this thesis. This is why we will only look at the most popular digital mobile communication technologies, the 3GPP or GSM family of networks, named after the

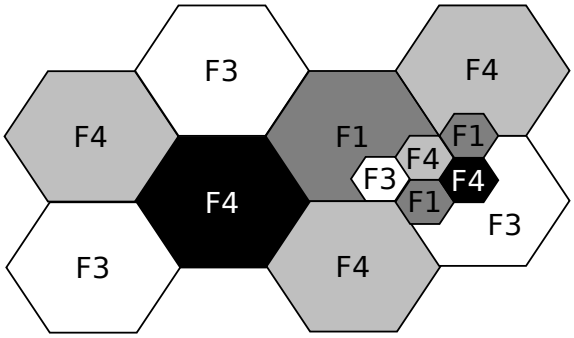


Figure 1.1: *A schematic division of frequencies in a geographical area, with four different frequencies (F1 - F4).*

consortium maintaining the specifications (3GPP – 3rd Generation Partnership Project) and after the first network in the family (GSM – Global System of Mobile communication), respectively. So this excludes the analogue cellular networks and the earlier competing digital systems, such as the Japanese PDC and the American IS-95. As of April 2016, over 7.7 billion subscriptions are using the GSM family of networks for around 4.7 billion unique subscribers [89]. Moreover, the biggest remaining competitor for the GSM family, Qualcomm's CDMA family of networks, halted development of their fourth generation, after Qualcomm favoured GSM's fourth generation LTE instead. This makes the GSM/3GPP family of networks the most popular global digital cellular networks by far.

The 3GPP family of networks consists of several different types of networks which are usually designated to a specific generation, starting with GSM as 2G –the second generation– in order to differentiate it from its analogous predecessors. These networks are summarised in Table 1.1 and are discussed in more detail in Section 2.1.

Not all chapters in this thesis consider all mobile standards within the GSM family, so Table 1.2 shows which chapter discusses which technology.

Mobile telephony networks are operated by Mobile Network Operators (MNOs),

Table 1.1: *Overview of the different networks within the 3GPP family, with their usual generation designation and major improvements they brought*

Generation	Name	Improvement
2G	GSM	First digital mobile phone system of the 3GPP family.
2.5G	GPRS	Support for package switched data is added.
2.75G	EDGE	Higher bandwidth.
3G	UMTS	Higher bandwidth and mutual authentication.
3.5G	HSPA	Higher bandwidth.
4G	LTE	Higher bandwidth, all IP based and forward secrecy.
True 4G	LTE-Adv.	Higher bandwidth.

these are the companies that own and maintain the cellular infrastructure. Users or subscribers have a subscription with a MNO, or a MVNO (Mobile Virtual Network Operator – a provider offering mobile phone services to users, but using the network of a MNO). MNOs and MVNOs are colloquially known as providers or telco's, and we will use the term provider in this thesis, instead of the more vague acronyms.

There are many different components in each of the 3GPP networks, and most have interesting responsibilities from a security perspective. However, in this thesis we will focus on the wireless interface of mobile networks, as this is the part of the network that anyone with a good enough transceiver has access to. This wireless interface is also referred to as the air interface or wireless link.

Figure 1.2 shows the model we will consider for most of this thesis, where the phone and SIM (Subscriber Identity Module — here meaning a generic piece of separate hardware securely storing the subscriber's credentials, not specifically the SIM used in GSM) connect wirelessly to the cell tower which connects directly into the "Network." In essence the cell tower is simply an antenna for the network. The network in turn routes data to the correct recipient, keeps track of subscribers within reach of its cell towers and generated authentication tokens.

Section 2.2 provides a more detailed version of this model.

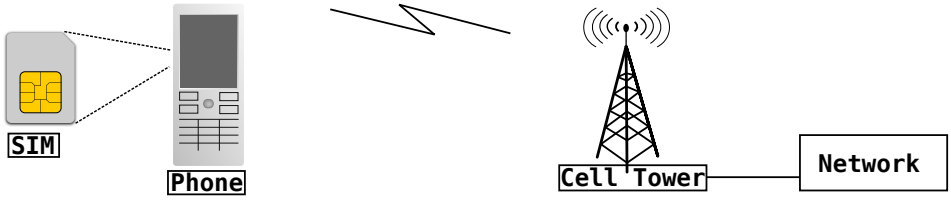


Figure 1.2: *The focus of this thesis lies on the wireless interface of mobile networks.*

1.1.2 Attacker model

Before we address what we mean by security in this thesis we will first define an attacker model.

The previous subsection already established that this thesis concerns attacks on the wireless interface. This means we disregard the providers or other insiders as potential attackers, as well as any attacks directly from or on the network, when they did not originate on the wireless interface or the end points (phones and cell towers). We also consider the phones, SIMs, cell towers and network themselves as trusted, so they are out-of-scope of our attacker model.

When we look at the wireless interface of the 3GPP networks from a security-oriented view, we can distinguish several levels of abstractions which could be a point of attack:

- Radio signals
- Protocols, which make use of cryptographic primitives
- Protocol and crypto implementations

- Additional services

At the lowest level all information transmitted through the air is essentially *radio signals* and an attacker will need to be able to generate and receive these radio signals to attack over the wireless interface. This level mostly represents a "physical" hurdle for other attacks, where it is interesting to see how hard it is to access this level, but where likely the only meaningful explicit attack is a denial of service attack through jamming.

The different 3GPP protocols are specified using these radio signals and cryptographic primitives, so we recognise the *cryptographic primitives* and *protocol* abstractions as a different level, as an attacker might find a weakness in the cryptographic primitives used in the protocols, or a weakness in the protocols themselves. Especially at the protocol level, one typically assumes a Dolev-Yao attacker [41] who has complete control of the wireless interface. An attack found at this level will have the highest impact, as it is applicable on all instances using the specific cryptographic primitive or protocol, and will be vary hard to patch in all implementations.

An attacker might also find weaknesses in *implementations* of these protocols and cryptographic primitives involved. The 3GPP protocols are very complicated and extensive, with lots of options, making it very likely that the actual implementations contain mistakes. An attacker can always choose to try and exploit one of those mistakes in one of the endpoints, even if the protocols themselves are secure. Such attacks would only impact specific implementations, but there are only a few vendors of cell towers and mobile phones, so one would expect a security vulnerability found at this level to still have a major impact. Additionally, getting access to one of these implementations could prevent an attacker from having to fiddle around with the radio signals and directly attack the logical protocol layer.

On top of the implementations of the 3GPP protocol are *additional services* that are being deployed, using the 3GPP protocols as a basis, but which were often not imagined when the 3GPP protocols where designed. Examples of these services can be payments being transferred over SMS (such as in many African countries), or using the mobile phone as an additional factor in online authentication. An attacker could simply be interested in 3GPP networks because the service he is targeting is deployed on top of one of those networks. This means attacks at this level are very application-specific.

This thesis will look at each of these levels of abstraction presented here, as is summarised in Table 1.2.

One level not discussed in this thesis is the design and implementation of the mobile phone operating systems and applications. This is a very interesting and important security subject, but veers closer to traditional operating system security and is easily a large enough subject to fill a thesis on its own.

Another way to look at the attacker model, is to examine the resources required for an attack or to look at the goals of the attacker. These resources can be both in terms of financial resources and time needed, but also in terms of the required level of expertise.

The revelation of the Snowden documents have proven what many already suspected, namely that state actors are capable of attacking mobile communication security in almost any way. State actors can hack into the heart of the providers networks [150] or simply order taps, but also run fake cell towers mounted on drones [104]

or steal the keys from SIM cards [105]. In at least one instance, the NSA was capable of intercepting and storing all telephone communications from a single country in a month [12]. While it is very interesting research to consider ways to protect the mobile networks against state agents such as the NSA, the question of how secure mobile communication currently is, is uninteresting against such a powerful attacker, as the answer is obvious. Therefore this thesis is mostly interested in attackers of limited means, by which we mean motivated attackers that have basic technical skills, but not the financial resources to buy off-the-shelf attack equipment such as fake cell towers.

About the possible goals of the attacker, we do not wish to make too much assumptions. These goals will, for the most part, be simplified to breaking one of the security goals, as doing so is either a goal in itself, or most likely a prerequisite for a higher level goal, e.g. financial gain.

1.1.3 Security goals

Before we continue with the in-depth look at security of mobile communication, it is essential to discuss what it means for a cellular network to be secure. In this thesis we will narrow the definition of what it means to be secure, to several security goals. These are:

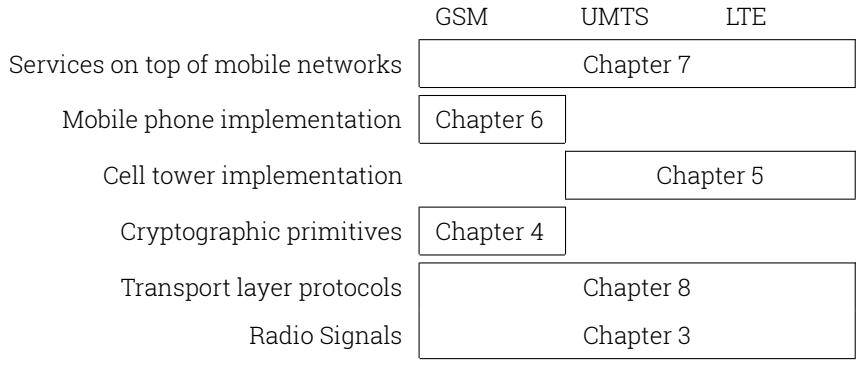
- *Confidentiality*, of data transmitted on the wireless link. This can be explicit user data (e.g. voice or IP packets) or meta data such as identifiers. We recognise two special forms of confidentiality, namely:
  - *Untraceability*, of user activity in the network.
  - *Location privacy*, of the user's geographical location.
- *Integrity*, of user and signalling data on the wireless link.
- *Availability*, of the wireless network and of the user device (phone).
- *Authentication*, of the user to the network and of the network to the user.

The traditional security goals from the well-known CIA acronym, also play an important role in the security of mobile networks. Here it should be noted that one standard attack against availability is always possible on the wireless interface of mobile networks, namely jamming of the frequencies used for communication. This obvious attack is not regarded in this thesis.

Another important security goal in mobile networks is *Authentication*, authentication both of the network and of the subscriber. Authentication of the subscriber by the network is both important to prevent theft-of-service, but also to prevent impersonation attacks. Authentication of the network by the subscriber protects the subscribers against network-impersonation attacks, e.g. Man-in-the-Middle attacks.

Security goals that are more specific for cellular network security are *Location privacy* and *Untraceability*. Essentially, an attacker should not be able to identify which users are currently at a geographic area, nor should he be able to track a subscriber through the network, neither by observing wireless network traffic nor by generating traffic himself. The network needs to know where every SIM is, so that incoming

Table 1.2: *Overview of the thesis's contents divided over mobile technologies and layers within that technology.*



transmissions can be routed to the correct location. As such, location privacy and untraceability are impossible to guarantee if the attacker controls the network.

To achieve these high level security goals listed above, additional, more detailed, security requirements have been specified by the 3GPP consortium, such as integrity of signalling information, as will be discussed in Chapter 5.

1.2 Contributions and outline of this thesis

This section describes the organisation of this thesis and highlights the contributions by the author. Table 1.2 provides an overview which shows for each chapter which level of abstraction within the wireless technology is examined and which wireless 3GPP technology.

Chapter 2 provides some background for the interested reader. It contains information which is not essential for understanding the rest of the thesis, but can provide the reader with a deeper insight. The chapter gives a more detailed overview of the different 3GPP networks, the elements they contain, and their air interfaces. It also gives an overview of the different types of hardware that were used during the more practical parts of this research.

Chapter 3 examines how feasible it is to break confidentiality on the wireless link of mobile telephony in practice. This can either be done passively (eavesdropping) or actively (Man-in-the-Middle attacks) and both approaches are investigated. This means this chapter focuses on the physical and protocol layers of the wireless connections as well as on the security goals confidentiality and authentication.

In the case of GSM the practical feasibility is tested by examining the effort required in catching and interpreting GSM signals using commodity hardware. The choice to focus on GSM in this part was a simple practical matter, as this was the only mobile telephone technology at the time where we could conceivably access the wireless network using generic radio transceivers. Finally,

this chapter also looks at the available countermeasures against these attacks against confidentiality on the wireless link.

This chapter is an updated version of the paper *Eavesdropping on GSM: state-of-affairs* (F. van den Broek [173]), presented at the 5th Benelux Workshop on Information and System Security, WISSec 2010. An earlier version was presented at BruCON 2010 [174]. Since, the paper was over six years old, it needed to be updated to the current situation. I have independently performed and written down the research described in this chapter.

Chapter 4 focuses on Time-Memory Trade-Off attacks against A5/1, the main cipher used in GSM. When using a passive eavesdropping attack against GSM, an attacker will likely end up with A5/1 encrypted data. A Time-Memory Trade-Off attack is currently the best known attack against this cipher, which was at that time by far the most widely used cipher in GSM, which in turn was the most widely deployed mobile network. The much stronger A5/3 cipher has seen additional roll-out in recent years, though at the moment of writing, A5/1 is still most prevalent. The newer mobile telephony technologies use much stronger cryptography, but as phones can be forced to the weaker GSM, these confidentiality attacks are most important.

Confidentiality is the primary security goal investigated in this chapter. Since the attacks discussed in this chapter retrieve the session key, they also impact integrity and authentication. The chapter compares different Time-Memory Trade-Off techniques and delivers the first analysis of the Fuzzy Rainbow Tables used against A5/1. It combines with the previous chapter in looking at the difficulty of breaking confidentiality on the wireless link in practice.

This chapter is based on the paper *A comparison of time-memory trade-off attacks on stream ciphers* (F. van den Broek and E. Poll [176]), presented at the 6th International Conference on the Theory and Application of Cryptographic Techniques in Africa, AfricaCrypt 2013. I was the main contributor in this research and wrote most of the resulting publication.

Chapter 5 looks at possible attacks coming from a corrupted type of cell tower, specifically femtocells. Femtocells are cheap, consumer-owned mobile telephony base stations, which extend the coverage of mobile networks within a small range. These devices have significant implications on all the security goals of mobile networks, which this chapter examines in depth. The choice for looking at the possible dangers of corrupted femtocells is again a practical choice, as it is much simpler for an attacker to obtain a femtocell then to get access to an actual cell tower. Additionally, this chapter examines how hard it is for an attacker to obtain access to an actual femtocell, by testing the security of the, at that moment newest, commercially available femtocell.

This chapter is based on the paper *Femtocell Security in Theory and Practice* (F. van den Broek and R. Wichers Schreur, [178]), presented at the 18th Nordic Conference on Secure IT Systems, NordSec 2013. The practical analysis of the femtocell was performed with the help of Roel Verdult and Joeri de Ruiter. I performed most of the theoretical security analysis independently and wrote most of the resulting paper.

Chapter 6 focuses on the security of the actual protocol implementations on mobile phones. Within mobile phones a specific chip (or computing core within a chip), called the baseband chip, is responsible for the handling of the mobile network protocol. Although the specifications of the mobile network protocols are public, they are very extensive and the implementations of the baseband stacks are closed-source and notoriously complicated. We tested several of these implementations in actual mobile phones, using our own base station and a technique called fuzzing – in essence automated, largely random, security testing – focusing on two parts of the GSM protocol: Short Message Service (SMS) and the Public Warning System (PWS). Besides finding attacks at the protocol level, an attacker can also attempt to find weaknesses in the implementations, which can also have a large impact if said implementations have a large install base. Although, in this chapter we also found that there are many differences in baseband stacks, even from the same vendor, which decreases the impact of flaws found at this level.

This chapter is based on the paper *Security Testing of GSM Implementations* (F. van den Broek, B. Hond and A. Cedillo Torres [175]), presented at the International Symposium on Engineering Secure Software and Systems, ESSoS 2014. The research for this chapter was part of the Master theses of Brinio Hond and Arturo Cedillo Torres. Brinio Hond's master thesis discussed the fuzzing of SMS messages and Arturo Cedillo Torres's master thesis was on the fuzzing of broadcast messages, specifically PWS (Public Warning System) messages. As supervisor to both projects I was heavily involved with the direction of this research, as well as setting up the practical experiments with phones. I wrote most of the resulting publication as a summary of the results from both theses.

Chapter 7 looks at the use of mobile phones as an additional authentication factor and the use of security-sensitive applications on mobile devices. In particular, it looks at so-called server-based signatures, which should also provide non-repudiation. With server-based signatures a user's private key is stored in a server (instead of say a smartcard that is under direct control of the user) and users send documents they want to have signed to this server. Naturally, such an approach requires strong remote authentication. Mobile phones are popular solutions to use as an additional element during authentication. Usually, the authenticating server thereby implicitly trusts the authentication provided by the 3GPP network. This chapter analyses whether mobile phones can provide an appropriate level of security for this assumption.

This chapter is based on the article *Digitale handtekeningen: nieuwe technologie & nieuwe wet- en regelgeving* (F. van den Broek and E. Poll [177]), published in the Dutch journal "Privacy en Informatie". I was the main contributor in this research and wrote most of the resulting publication. Some of the research reported in this chapter was done for a practical risk assessment on a system were users could view their medical data online. For this system SMS messages were considered as an additional authentication factor. This risk assessment was commissioned by the Dutch Ministry of Health [144]. Based on this assessment, this authentication mechanism was judged to be too weak for securing access to this medical data.

Chapter 8 looks at ways to improve the security offered by current mobile networks. Specifically, this chapter proposes a solution against so-called IMSI catching, which is an attack whereby an attacker can wirelessly retrieve the long term identifier inside every SIM card in the vicinity. IMSI catching is among the oldest attacks possible in mobile networks, but it still persists, even in the most modern 3GPP protocols. IMSI catching is clearly an attack against location privacy and traceability, so those security goals are the focus in this chapter. Additionally, our solution for 2G technology actually strengthens the authentication procedure, by providing the SIM the possibility to verify if a challenge was created by its home network. The solutions offered are compatible with the current specifications of the mobile standards and requires only limited changes to the SIM and the authentication server, both of which are under control of the user's network provider. Therefore, any individual (virtual) provider that distributes SIM cards and controls its own authentication server can deploy a more privacy friendly mobile network that is resilient against IMSI catching attacks, without requiring any changes in the infrastructure, or in the intervening massively deployed network equipment. We are currently working with the Fraud and Security Architecture Group (FSAG) of the GSMA to get this solution into the 3GPP specifications.

This chapter is based on the paper *Defeating IMSI Catchers* (F. van den Broek, R. Verdult and J. De Ruiter [179]), presented at the ACM Conference on Computer and Communications Security, CCS 2015. I was the main contributor in this research and wrote most of the resulting paper, except for the formal verification of our solution using ProVerif, which was performed by Joeri de Ruiter.

Chapter 9 draws conclusions and provides directions for future work.

Chapter 2
Background

The background information presented in this chapter is here purely for completeness sake and is not essential for understanding the rest of the thesis. For the interested reader this chapter gives an overview of the different generations within the 3GPP family of mobile networks. It also presents a more detailed, but still very generic overview of mobile networks and gives an overview on the specific hardware that was used for the more practical research presented in the rest of this thesis.

2.1 Generations

Currently, providers are rolling out the LTE network globally, which is also called 4G, where the G stands for generation, to imply that this is the fourth generation of mobile telephony. Most mobile networks have had these designations to a certain generation. This started with GSM and IS-95, which were called 2G networks, thereby retroactively naming the preceding analogue systems 1G.

We will now shortly describe the major generations within the 3GPP family. This information was also summarised in the Chapter 1 in table 1.1.

2.1.1 2G – GSM

GSM was the acronym given to the research group in charge of creating a new digital cellular mobile phone standard, Groupe Spécial Mobile. When GSM started getting world wide popularity GSM's long form was changed to Global System of Mobile communication.

GSM was developed in the late 1980s and deployed in most Western countries in the early 1990s. GSM saw its first deployment in 1991 in Finland. Since then GSM has seen an enormous rise both in its coverage and in the number of subscribers, making it perhaps the most successful technology of the last twenty years.

When GSM was first deployed there was some security research, which mostly focused on the specifications and the reverse engineering of the secret and proprietary encryption algorithm [25]. Several weaknesses in GSM's protocols and cryptographic primitives where quickly identified, though practical exploits of these weaknesses proved complicated because of all the signal processing involved. This changed

around 2010 with the arrival of cheap hardware [49] and open-source software [26] which provided easy access to the GSM spectrum. This immediately led to some high profile attacks, such as the release of the Time-Memory Trade-Off tables for breaking GSM's standard encryption [132]. These attacks are discussed in Chapter 3.

GPRS

GSM has no support for packet-switched data, as the phone network was still circuit-switched when GSM was developed. There are some techniques to actually run an IP stack on top of SMS, but the performance of this is rather dreadful. This is where General Packet Radio Service (GPRS) comes in. GPRS is a backwards compatible update for GSM allowing for packet-switched data, such as IP protocols to be transmitted over GSM. The peak speed of GPRS-based data is 114 kbit/s.

EDGE

Enhanced Data rates for GSM Evolution (EDGE) is an improvement on GPRS and therefore also known as Enhanced GPRS (EGPRS). EDGE can manage a peak bandwidth of 473.6 kbit/s. As you can read below on UMTS, this bandwidth qualifies EDGE as a 3G technology. Still, it is usually still designated within the second generation, saving a generation upgrade for non-backwards compatible technologies. EDGE is being deployed on GSM networks since 2003.

2.1.2 3G – UMTS

For the third generation of mobile telephony the International Telecommunication Union (ITU) — a United Nations agency responsible for telecommunication issues, such as coordinating the global use of the radio spectrum and helping in creating worldwide technical standards — specified some standards for mobile technologies to be considered 3G. Most important in these specifications was an information transfer rate of at least 200 kbit/s.

In order to establish a 3G mobile phone system based on GSM, telecommunications associations started collaborating under the name 3rd Generation Partnership Project (3GPP). Later on they also became responsible for maintaining the GSM, GPRS and EDGE specifications.

The Universal Mobile Telecommunications System (UMTS) was 3GPP's first technology to be classified as 3G. It was introduced in 2002 and started out managing an information transfer rate of 384 kbit/s. UMTS introduced Code Division Multiple Access (CDMA), a channel access method in which multiple parties can use the same frequency simultaneously by assigning codes to each connection used within the modulation. Confusingly, the name CDMA is also used to describe Qualcomm's family of mobile networks, which also use the CDMA channel access method, as opposed to GSM's Time Division Multiple Access (TDMA) where the same frequency is shared by dividing it into different time-slots. This switch to the CDMA channel access method effectively meant that new phones and cell towers were needed for UMTS, as it made UMTS non-backwards compatible with its pre-decessors.

The most important security change incorporated in UMTS was the mutual authentication procedure between phone and cell tower, meaning that, as opposed to the earlier technologies, the SIM now also authenticates the network. Additionally UMTS introduced a longer secret key (now 128 bits) and better and publicly available cryptography. UMTS security is discussed in detail in Section 3.5.

HSPA

The High Speed Packet Access (HSPA) is a family of upgrades possible within existing UMTS networks, which are being deployed since 2006. They are usually subdivided in two upgrades HSDPA (Downlink) and HSUPA (Uplink), with HSDPA achieving downlink speeds of up to 14.4 Mbit/s and HSUPA achieving uplink speeds of up to 5.76 Mbit/s.

After HSPA there was another improvement called Evolved HSPA, or HSPA+. HSPA+ has a theoretical maximum downlink speed of 672 Mbit/s and an uplink speed of 168 Mbit/s.

Neither HSPA or HSPA+ included any security improvements.

2.1.3 4G – LTE

The designation of Long Term Evolution (LTE) as 4G is controversial. Again, for the fourth generation the ITU defined a peak speed requirement. This time the requirement was 100 Mbit/s for fast moving targets such as users in cars or trains and speeds up to 1Gbit/s for slow moving targets, such as pedestrians. LTE manages a peak downlink speed of 300 Mbit/s and a peak uplink speed of 75 Mbit/s. This would mean LTE does not qualify as 4G. However, since LTE switches to an all IP network and uses Orthogonal Frequency-Division Multiple Access (OFDMA) as a channel access method, thereby not being backward compatible with its predecessors, and since LTE was already being marketed as "4G", the ITU decided to name LTE a 4G technology. LTE is being deployed since 2010.

Security-wise the most important change is the introduction of forward security, by deriving extra keys during the authentication procedure, as explained in Section 3.5.

LTE-Advanced

LTE-Advanced improves the speeds of LTE to a peak downlink speed of 1 Gbit/s, with a 100 Mbit/s for fast moving targets. This caused the ITU to consider LTE-Advanced as "true 4G". Some small LTE-Advanced test networks have been rolled out since 2013, but major commercial deployment is not yet underway.

2.2 Network overview

In Chapter 1 we discussed a very simplified model for cellular networks focusing on the wireless interface. That model is shown in Figure 1.2.

Mobile telephony networks have many more components. Figure 2.1 gives a generic but more detailed overview of the 2nd and 3rd generation of 3GPP networks. The

model presented here is still very generic and should not be used as an overview of the actual layout of any 3GPP network, but rather as a generic idea of common elements between these networks. In an attempt for this thesis to use as little terminology as possible, all the names in Figure 2.1 are generic names instead of the actual terms. We choose to use generic terms because the different generations of 3GPP networks introduce new names for each component. For instance, cell towers are called "Base Transceiver Station" (BTS) in GSM, "NodeB" in UMTS and "eNodeB" in LTE. It is therefore simpler to introduce generic, hopefully self-explanatory terms for these components rather than switching between terms when we discuss different networks.

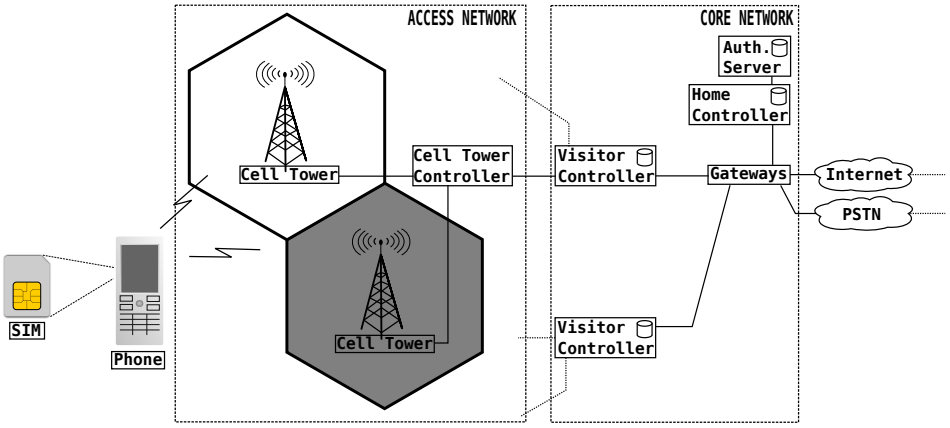


Figure 2.1: *Generic overview of 3GPP style networks.*

The 3GPP networks are usually divided in at least two domains: the access network and the core network, as is shown in Figure 2.1. Sometimes an additional subscriber domain is defined and sometimes the subscriber domain is included in the access network. The subscriber has some 3GPP supporting mobile device, in the figure represented by a phone, and a SIM. The SIM (Subscriber Identity Module), a removable smart card securely storing the subscriber's unique serial number, secret keys and authentication algorithms, was a unique feature for GSM and is often named as one of the main reasons why GSM's popularity outgrew its competitors [101]. It is also the only actual GSM term we use in Figure 2.1, since this term now seems firmly established in common language. Indeed, the SIM's successor, UMTS's Universal Integrated Circuit Card (UICC) containing a USIM application, is still colloquially called a SIM. Mobile phones are recognised within a network based on the serial number within their SIM, which is provided by the subscriber's provider.

The mobile phone connects wirelessly to cell towers, which are relatively dumb devices, mostly forwarding every message to their controllers. These controllers can serve one or more cell towers, managing the radio channel setup and handovers between connected cell towers, and watch the status of the cell tower's hardware.

The real logic of the mobile network starts at the Visitor Controllers. These controllers have a database in which they keep track of all mobile phones within their area. They handle handovers between cell tower controllers and store authentication records generated by the Authentication Server (Auth. server in Figure 2.1). When a mobile phone enters a cell under control by a another Visitor controller, this Visitor

Controller will inform the Home Controller, who will store the current Visitor location of his subscriber and inform the previous Visitor Controller to remove the subscriber from its database.

The Home Controller and Authentication Server are both part of the home network of the subscriber's own provider, while the rest of the network could be part of another provider. If this is the case a subscriber is said to be *roaming*.

If a new authentication is required, the Visitor Controller will request authentication records from the Home network's Authentication Server. The contents of these authentication records depend on the actual 3GPP network, but at least they contain a challenge, a response and a session key, generated with the authentication algorithm and secret key present in both the Authentication server and the SIM. This leaves the provider some freedom to choose his own authentication algorithms since he controls both the contents of the SIM and the Auth server in his back-end. It also allows visiting networks to authenticate the SIM without learning the secret key.

There are several gateways that connect everything together and connect the mobile network to other networks, such as the internet and the Public Switched Phone Network (PSTN).

LTE Network Overview

The 2nd and 3rd generation of 3GPP networks have fairly similar network topologies, but for the fourth generation there were some major changes, which are reflected in Figure 2.2. As before, the names provided for the network entities are self-chosen names and not the actual names as defined by the 3GPP consortium. For the LTE

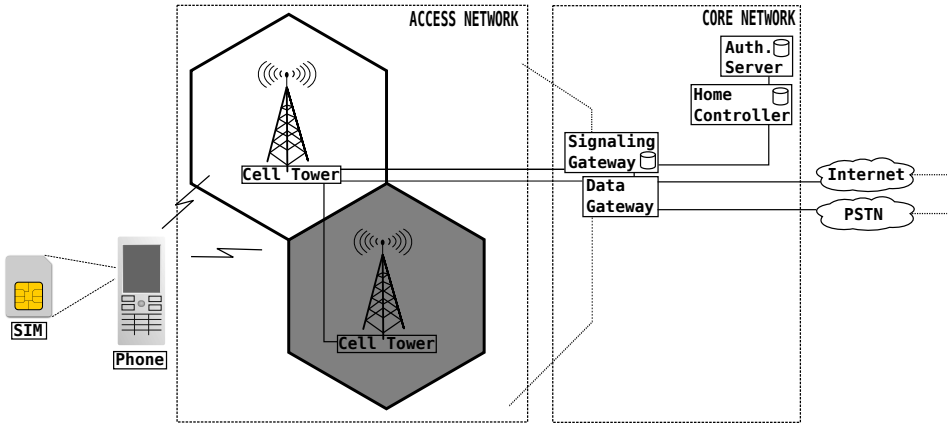


Figure 2.2: *Generic overview of LTE style networks.*

style networks, the cell tower controllers have been incorporated with the cell towers. This makes the cell towers "smarter" devices than their previous generation counterparts, which are now capable of communicating directly with each other when directly connected. In this way handovers within the area of one Signalling Gateway can be handled efficiently between connected cell towers.

The other major difference is the explicit splitting of signalling and user data flows, whereby the former is handled through a Signalling Gateway and the latter by

a Data Gateway. This split of the different information flows also allows for different security measures per flow, as will be explained in the next section. Additionally, all the links within the access and core network are now fully IP based.

2.3 Authentication within the network

Authentication on the different 3GPP networks will be discussed in more detail in Chapter 3. However, there are some interesting interplays between the network topologies, the authentication and the different confidentiality and integrity protected links for each of the 3GPP generations, which did not fit within the scope of Chapter 3, where the focus lies solely on the air interfaces. These interplays are summarised in Figure 2.3, where for the three generations of 3GPP networks the authentication steps are shown, as well as whether the user and signalling data are encrypted (the padlock symbol) and have a MAC (the seal symbol). Please note that this figure, nor this section provides any judgement on the strength of the cryptographic algorithms used for each generation. A discussion on that can be found in Chapters 3 and 4.

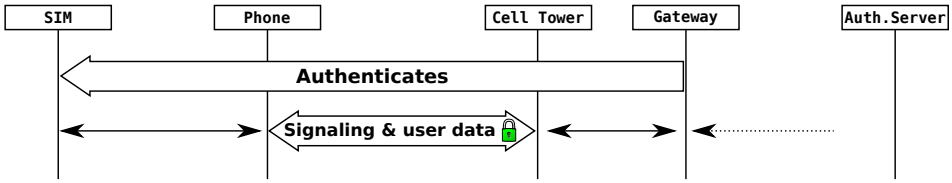
For all three networks shown in Figure 2.3 there are signalling messages that are unprotected. Additionally, all three networks support a null-cipher for the user data. So, only the protected information flows are shown in the figure. The figure illustrates the differences between the 3GPP generations in terms of (mutual) authentication, and what data is protected in between which network entities.

For all 3GPP networks the authentication procedure also establishes session keys. The authentication server and the SIM share a secret key, which is used by the authentication server to compute the response, and session keys based on a fresh random challenge. In the case of UMTS and LTE the authentication server also computes a token which proves knowledge of the secret key. These computed values are them moved on to other network components. It is up to a specific gateway to compare the SIM's response with the response provided by the authentication server and forward the session keys to the correct place in the network.

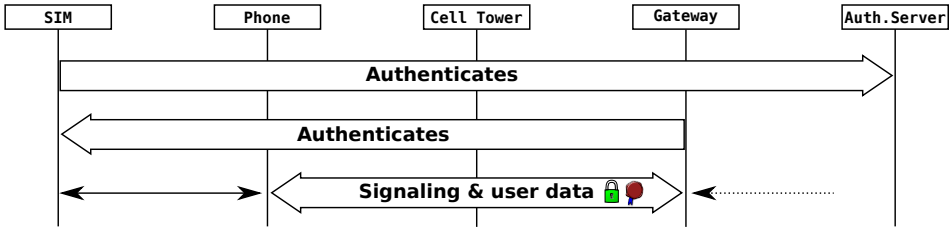
Although UMTS and LTE have mutual authentication between subscriber and network, this authentication does not happen between the exact same entities. The SIM verifies the authentication token it receives, which essentially provides a MAC over the challenge and a sequence number (to prevent replay attacks) using the secret key. The SIM thereby authenticates this token coming from the authentication server, while the SIM's response is verified at the respective gateway.

UMTS and LTE also provide explicit integrity protection on certain messages, through computing a MAC over these messages using an extra session key: the integrity key. GSM only has encryption on its messages, which, since the encryption key results from the authentication process, is assumed to provide some authentication. Since encryption in GSM is provided by XORing keystream with plaintext, there is no integrity protection; simply flipping a bit on the ciphertext will flip the underlying plaintext bit. In other words the cipher is malleable.

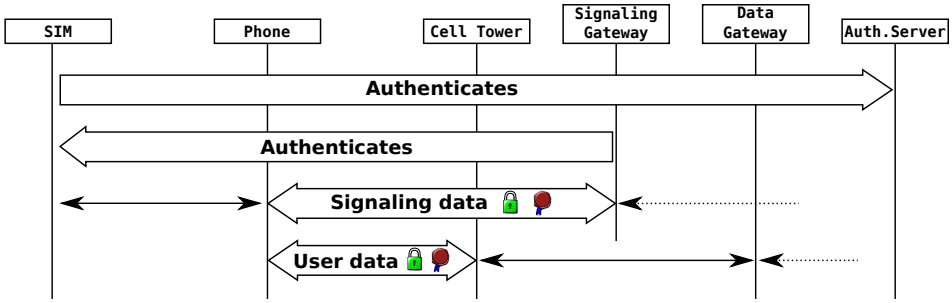
There is also a difference how far into the network the connection is secured. In GSM the encryption key is forwarded to the cell tower, which decrypts messages coming from the phone and encrypts messages going towards the phone. For UMTS the 3GPP consortium chose to secure the user link all the way to the first gateway within



(a) GSM



(b) UMTS



(c) LTE

Figure 2.3: *Authentication, confidentiality and integrity in 3GPP networks*

the core network. LTE has a secured signalling connection between the phone and the signalling gateway and a separately secured user data connection between the phone and the cell tower. Additionally, all connections within the provider's network are to be secured by IPSEC [69]. This setup was chosen to allow cheaper handovers between connected cell towers, as the encryption and integrity key for the user data connection can be forwarded between directly connected cell towers.

2.4 Equipment and software used

During this research we used several different pieces of hardware and different open-source software projects; these are most prominently used in the practical sections of Chapters 3 and 6. This section describes the hardware and software used and some of their limitations. The different combinations are ordered by their capabilities.

2.4.1 Analysis and eavesdropping of the air interface

There are several options for analysis and eavesdropping on the GSM air interfaces. For the other 3GPP protocols, this is more complicated. Hardware is available for catching air interface signals of UMTS or LTE, but we found no software driving this hardware. Also, as will be explained in detail in Section 3.5, the newer generations air interfaces are not as susceptible to eavesdropping as GSM is.

USRP + AirProbe

The Universal Software Radio Peripherals (*USRP*s) are a series of generic transceiver devices developed by Matt Ettus and which can be ordered through his company Ettus Research [49]. Originally, these devices where introduced as open hardware, although it is unclear where the designs of the devices can be found. The different models cost somewhere between 600 and 4000 euros, with the base USRP1 model costing 615,- euros. The devices are transceivers that can be linked to a computer and can be tailored to specific frequencies by extending it with daughter boards and attaching the appropriate antenna. The USRPs contain a programmable FPGA which can be used to perform some signal processing, some digital-to-analog (DAC) and analog-to-digital (ADC) converters and a communication port for the connection with a host computer. Depending on the model, the converters and the computational power of the FPGA range in quality. The communication port also ranges between a USB2.0, a USB3.0 or single to dual Gigabit Ethernet connections. We used the USRP1, then simply known as the USRP, as it was the only model available when we started this research.

The USRP can be extended with a daughter board in order to receive the correct frequency spectrum. We have used several daughterboards:

- a DBSRX, a 800 MHz to 2.4 GHz receive-only board,
- a WBX, a 50 MHz to 2.2 GHz transceiver board, and
- two RFX1800, 1.5 to 2.1 GHz, of which the firmware can be flashed to change them into RFX900s, 750 to 1050 MHz transceiver boards.

For eavesdropping and analysis of GSM frequencies all boards functioned with similar results. In Section 2.4.4, we detail why the USRP was incapable of following GSM signals over multiple frequencies (frequency hopping).

GNU Radio [82] is a free software toolkit licensed under GPL for implementing software-defined radios. It was started by Eric Blossom. Basically GNU Radio is a library containing lots of standard signal processing functions, such as filters and (de)modulations and offers a general interface to a transceiver. GNU Radio mainly functions as the driver of the USRPs. GNU Radio, out-of-the-box, does not offer much in terms of GSM sniffing capabilities. However, GNU Radio can be used by other software packages, such as *AirProbe*, to perform the low level functions of GSM sniffing, such as reception and demodulation.

AirProbe [29] is an open-source project aimed at learning the details of GSM technology, helping people who develop other open GSM technology and demonstrating the insecurity of the current GSM standard. Currently, AirProbe supports sniffing on

single frequency GSM downlink (cell tower to phone) traffic and it can interpret several types (but not all) of bursts (GSM packets). There has not been much development for AirProbe in the last three years to add the missing functionality.

Motorola C123 + OsmocomBB

OsmocomBB [139] offers a new firmware for the Texas Instruments Calypso baseband chip (the closed source component inside phones, handling the 3GPP protocol). One of the phones containing this baseband chip is the Motorola C123. This phone (as well as most Calypso phones) has a 3.3V serial port on its 2.5 mm audio jack. Using a converter cable the phone can be connected to a PC's USB port.

With the OsmocomBB software it is possible to re-flash the phone, with code handling only the lowest layer of the GSM protocol. All higher layers then run on the connected PC. This allows a user to change the phone's behaviour to, for instance, listen to GSM channels destined for other phones. While the OsmocomBB project offers interesting capabilities, it is not an out-of-the-box eavesdropping project, as several necessary parts for this have not been released. Section 3.3.1 details the use of the OsmocomBB project within an eavesdropping attack.

Nokia 3210 + Gammu

We also made extensive use of a Nokia 3210 GSM phone, connected to a computer, also via USB, running the open-source Gammu [2] project. This combination enabled us to force the Nokia 3210 in a debug mode that transparently logs all packets sent to and from the phone.

The Gammu + Nokia phone method has much better reception than the USRP + AirProbe. This makes sense, since after all the mobile phone is specifically made to receive these signals. Since this only uses functionality already present in the Nokia 3210, you only see the messages to or from the specific phone connected to the computer. You cannot see any message for other phones, nor is it possible to change the phone's behaviour in this manner. So, Gammu with a Nokia 3210 is a great practical aid to get a better grasp of the GSM protocol and to fine-tune the USRP, but it lacks the versatility to be useful in a eavesdropping attack.

2.4.2 Running our own GSM network

We ran our own GSM network for several different research goals. Next to the insight provided by running your own network, it allowed for testing the difficulty of eavesdropping on single frequency cells, and our fuzzing research discussed in Chapter 6. Again for the newer generation 3GPP networks there is little availability of open-source software for running your own network. There is commercial LTE software available which requires a USRP [14], but this software is not open-source, making it harder to make changes in its operation for, for instance, fuzzing.

In the Netherlands, the regulatory body for frequency spectrum use within the Dutch ether, the Agentschap Telecom [1], decided to make a small part of the GSM frequency freely available. This spectrum, specifically 1782.1 to 1784.9 MHz for the uplink and 1877.5 to 1880 MHz for the downlink can be used for running you own GSM

network. Doing this does not require a license, but you should register your network use.

USRP + OpenBTS

The USRP was discussed earlier in this section. Using the USRP as a GSM basestation required some adaptation. The USRPs have a 64MHz crystal oscillator internal clock, while most GSM phones use a 13MHz symbol clock with a much better accuracy. Of course the 64MHz samples can be re-sampled to (a multiple of) 13MHz, although this brings an extra computing complexity. Also the USRP's oscillators are much less accurate and can show quite some drift when compared to GSM system clocks, resulting in bad reception. Using a more accurate external clock providing a pulse at (a multiple of) 13MHz solves these issues. We used the programmable Fairwaves Clocktamer for this [71].

OpenBTS [26], founded by David Burgess, offers an open-source cell tower implementation for GSM using the USRP as a transceiver. Some of the logic normally present in a cell tower controller is placed inside OpenBTS. The OpenBTS software can be connected with Asterisk [7], which is an open-source software project implementing a telephone exchange, and has the option to connect to a VOIP network.

2.4.3 SIM cards and the SIM-phone interface

While not directly a subject in this thesis, the phone-SIM interface is interesting, if only to observe. This can for instance tell you when a phone has to re-authenticate to a network without having to observe the air interface. This makes this method of research often the only available option when investigating UMTS or LTE networks. The SIM-Phone interface uses a standard ISO 7816 smartcard interface.

RebelSim Scanner

The RebelSim scanner [146] provides passive eavesdropping between the phone and SIM card. This is achieved by providing a SIM to smartcard holder, which allows the original SIM card to be inserted in a smartcard reader. This smartcard reader then forwards all communication to dummy SIM-shaped connectors which can be inserted in the phone and meanwhile also copies all communication to a serial port connection which can be observed on a PC. This setup is shown in Figure 2.5. These SIM-shaped connectors are used by several other projects aimed at examining the Phone-SIM interface, discussed below. These connectors allow access to the phone-SIM interface, while the phone is in operation. A detailed photo of such a connector is shown in Figure 2.4. The RebelSim connectors are only available in the standard SIM size and not in the currently much used micro or nano SIM shape.

There are some tools out there, such as SimParser.pl [80], which help understand the output given by RebelSim Scanner.

SmartLogic Tool

While the SIM tracer only allows passive eavesdropping on the communication between SIM and Phone, the SmartLogic Tool [37] can actively insert commands. It is a sniffer

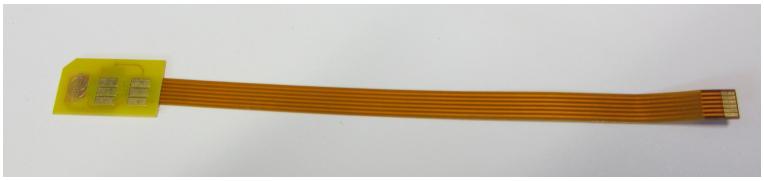


Figure 2.4: *The RebelSim SIM shaped connector.*

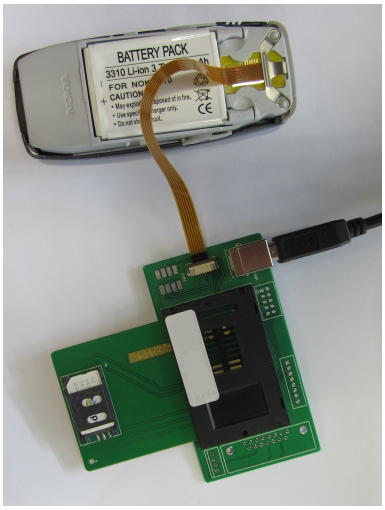


Figure 2.5: *The RebelSim Scanner setup, which uses a SIM shaped connector to eavesdrop the communication between SIM and Phone.*

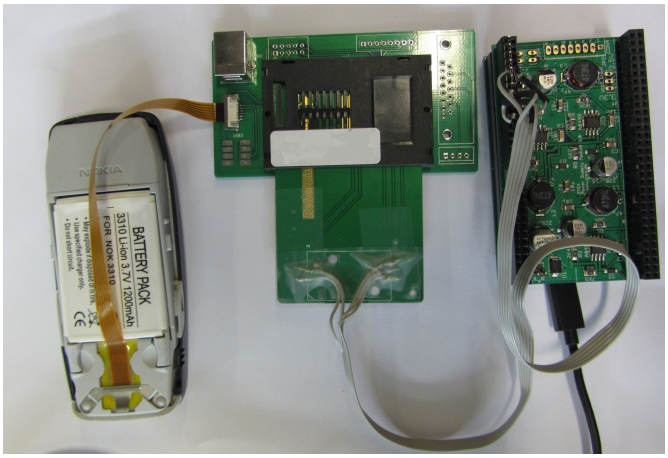


Figure 2.6: *The SmartLogic setup, which uses the RebelSim setup, but extents it by simulating a SIM card. Optionally an actual SIM card can be read through a card reader.*

and MitM tool for smartcard communication, originally designed for bank cards.

The SmartLogic tool uses an FPGA board to essentially emulate a SIM card within the RebelSim setup, as is shown in Figure 2.6. Optionally, an actual SIM card can be read by a card reader connected to a server. The SmartLogic FPGA board can either fully implement a SIM card, relay all communication to the SIM card at the server, or anything in between. This allows for listening to and manipulating of SIM-phone traffic. In our experiments it turned out that this was mostly successful with older phones [81]. It seems the speeds of communication of SIM cards is much higher than of bank cards, which caused unreliable connections with the SmartLogic tool.

Bladox Turbo SIM

The Bladox Turbo SIM [18], is a thin sheet with SIM connectors on both sides and a small chip. It can be inserted inside a phone, between the SIM and the phone, and will Man-in-the-Middle the traffic between both, based on the programming of the chip. A part of the original SIM card will usually need to be cut off, to accommodate the chip, as can be seen in Figure 2.7.

This offers, for instance, the ability to add new functionality to the SIM, without having actual access to the SIM, but it can also change the commands and/or responses between SIM and phone. It is convenient in that the whole setup is contained within a phone and so does not need any wires sticking out. However, programming the Turbo SIM can be quite a hassle and debugging is very hard. It is also not very useful for simple observation of the SIM-Phone interface, due to limited storage.

2.4.4 Problems using the USRP to eavesdrop on GSM

Many GSM cell towers transmit on multiple frequencies. To achieve an average signal reception quality, most data connections with phones will switch rapidly between

Figure 2.7: *Picture from the Bladox website [18] showing how to cut the SIM card to place the Turbo SIM in between SIM and phone.*

these frequencies, a process referred to as "frequency hopping". While not a security measure per se, the hopping sequence, which is often negotiated confidentially, makes eavesdropping on GSM frequencies much harder, especially when using generic equipment like the USRP.

In its standard configuration a USRP1 creates 16 bit I and Q samples when receiving a given frequency. These are complex samples, with the real part (Q) describing the cosine of the signal, and the imaginary part (I) describing the sine of the signal plus 90 degrees. One sample is thus 32 bit long and can be sent to the host computer through the communication port, for further processing. The USRP1 can receive a bandwidth of 32MHz and can transmit on a bandwidth of 64MHz. It transmits the samples to the host computer via a USB2.0 connection, which has a practical maximum data throughput of 32 Mbyte/s. Other USRP models can receive and transmit on a bandwidth of up to 120 MHz, but the costs rise quickly.

As discussed in Section 2.4.2, the internal clock of the more affordable USRP1 has a too imprecise internal clock. An issue that can be solved by using a more accurate external clock.

The main problem for reception is the channel hopping used in most GSM networks. Using the AirProbe software out of the box helps you receive a single carrier on the down link (messages from cell tower to mobile). In order to capture an entire conversation when channel hopping is used, you will need a way to gather all the bursts on all the different frequencies. There are two general approaches to achieve this:

I. Let the USRP follow the hopping sequence.

II. Capture all possible frequencies and attempt to follow the sequence afterwards.

Approach I requires a lot of processing inside the USRP's FPGA. All the parameters for the hopping sequence need to be retrieved from certain bursts, then the hopping sequence needs to be calculated and followed for every burst. There are three possible configurations for mobile networks in which to transmit the hopping sequence parameters. In one of these the parameters are transmitted in the clear. In the other two configurations these parameters are transmitted after encryption has been enabled. During our experiments we only ever observed cell towers that use so-called "early assignment", in which the hopping parameters are transmitted under encryption. Besides, the network can always command a new hopping sequence under encryption, irrespective of which configuration is used. This necessitates breaking the encryption really fast in order to follow the hopping sequence in time. This is currently not possible. The USRP's FPGA only samples a certain frequency at a certain rate and sends these samples to a computer. So having the FPGA decrypt and interpret the

bursts in order to follow the hopping sequence will require a lot of implementation. It is questionable whether even the faster FPGA offered by some USRP models will be able to decrypt messages and then compute the hopping sequence in time. This approach might need additional FPGAs, and thus additional costs, to pull off. Also the tune delays of the USRP's hardware (the time between a "tune" command and the moment the USRP retrieves usable samples from the desired frequency) seem too large at the moment and need to be brought down. However, it is an approach that should work for every cell tower and regardless of the amount of traffic.

Approach II requires the capture of large amounts of data, namely to log all channels and determine the hopping sequence later. The problem here lies in reducing the data the USRP sends on to the computer. A hopping sequence can hop between 64 carrier frequencies at maximum. These carriers are all 200 KHz wide, and can be spread out evenly on the entire GSM spectrum. So, worst case the attacker has to capture the entire GSM band (for GSM900 this is 25MHz for the up or down link). The problem here is data throughput to the PC. Each sample of a USRP is represented by two 16 bit numbers. For the USRP1's USB2.0 connection this means that the maximum bandwidth that can be sent to the computer is around 8MHz (8MHz $\times 2 \times 16 = 256$Mbit/s). The USRPs with a GBE connection can manage around 30MHz, which is enough for one sided capture of GSM900. The top model USRP X-310, with two GBE interfaces could capture the entire uplink and downlink frequency bands with a single device. This would require the host computer to be able to process 100 MByte/s of data (25MHz $\times 2 \times 16$ = 800Mbit/s) and even 200 MByte/s for both up and down link, which is too much for most PCs.

Of course some optimisations are possible. A single cell tower never serves the entire GSM frequency band. This means that you can already discard all carriers above the top frequency and all carriers below the lowest frequency of a specific cell tower. However, that approach will not work for most situations, since the maximal number of carriers (64) for a single cell tower is still too large for the USRP. Also, since the USRP can only receive a continuous frequency band, if the top and bottom frequencies are too far apart, this approach will not work.

Another optimisation would be to have the FPGA discard all channels that have no traffic on them. This will of course only be effective if only a few phones are active –as in calling – at the same time. It would be even better to have the FPGA interpret enough of the bursts, so it can already drop some that are not a part on the conversation the attacker tries to capture. This optimisation does see the same problems as those with the first approach because it requires a lot of FPGA computation – though less so, because the FPGA does not need to crack A5/1 in the second approach.

The objections stated above, however, do not form a problem if the cell tower under attack does not employ channel hopping, or only transmits on a few frequencies in a tight spectrum. On such cell towers eavesdropping using an USRP is a genuine possibility. It is not clear how many, if any, of the cell towers in operation, match one of these conditions. This makes it hard to estimate the risk in the current situation. During this research only a handful of cell towers were observed, but none of those fulfilled these conditions that would allow eavesdropping using the USRP.

2.4.5 Interesting hardware we did not use

There is a lot of hard and software available which we did not use during this research and which looks interesting. The most notable of these include:

- **RTL-SDR [149]** is a project where you can make a transceiver for a software defined radio out of a $20 TV tuner. The website reports success in using this transceiver for sniffing GSM signals.
- **Osmo SDR [162]**, the Osmocom Software Defined Radio, is much cheaper than an USRP, and should be an able replacement for GSM research. Currently, the firmware is not ready.
- **OpenBSC [189]** is an open-source software project for running your own GSM network. In essence it implements the Cell Tower Controller from Figure 2.1. The project requires a cell tower and currently supports several options, amongst which is their own OsmoBTS project, which implements a cell tower and can even run on two OsmocomBB phones.
- **Osmocom SIMtrace [140]** can be used to monitor the phone-SIM communication, but also emulate a phone or SIM, or be a Man-in-the-Middle between the phone and SIM. While the hardware supports all these modes, only the monitoring aspect has been implemented in software. SIMtrace also uses the SIM connector from the RebelSim to connect to the phones, but recently they made their own connectors, including a micro SIM connector.
- **Other USRPs [49]**. As we discussed in Section 2.4 on USRPs, there are currently many different models of USRP. Although a lot more expensive than other transceivers discussed in this list, these USRPs are probably also higher quality and more versatile.

The fact we did not use any of these devices should not be taken as a negative, as these devices where simply not available at the time we started this research. So, these might be useful to consider for any researcher wanting to start with practical research into mobile telephony.

Chapter 3

Security of the wireless interface

This chapter looks at the "physical layer" and protocols used on the wireless interface of mobile telephony networks. We discuss the current state of confidentiality of communication over the wireless interfaces of the 3GPP networks. The bulk of this chapter discusses attacks against GSM, since attacks on the newer technologies UMTS and LTE, can often be reduced to an attack on GSM, by jamming the frequencies used for UMTS and LTE. Some of the attacks against GSM involve a Time-Memory Trade-Off attack against the cryptography most commonly used in GSM, which is discussed at length in Chapter 4.

This chapter discusses different attacks, both passive and active, and assesses the difficulty to perform these attacks using available hardware and software. Although this chapter focuses on confidentiality, authentication is also an important property as weaknesses in the authentication allow for the active attacks. We analyse the weaknesses in GSM that lead to these attacks and show which of these weaknesses are resolved by the newer generation networks: UMTS and LTE.

This chapter provides a first comprehensive account of (often partial) attacks discussed in [10, 13, 132], taking into account both theoretical aspects and the practicalities of actually intercepting or spoofing wireless 3GPP traffic.

This chapter is based on the paper *Eavesdropping on GSM: state-of-affairs*, presented at the 5th Benelux Workshop on Information and System Security, WISSec 2010 [173]. It has been updated extensively to accurately represent the current state of mobile telephony security in early 2016. The original publication only considered the use of universal radio reception equipment (specifically the USRP) for eavesdropping attacks. Since then, these attacks were demonstrated using modified mobile phones, the description of which has been added for this chapter. An overview of the different active attacks was also added compared to the original publication, as was the description of the added security offered by newer generation networks.

3.1 Introduction

At the end of 2009 look-up tables usable for a brute-force attack against the main cipher used in GSM where released [135]. Chapter 4 will discuss the exact method behind these tables in detail. At that time it was thought that these tables could be combined with open-source hardware and software, which were capable of capturing the GSM frequencies, to eavesdrop on GSM conversations. In 2011 the practicality of these tables was demonstrated in an attack, but now using four specific phones and newly written software [13].

We use the common term 'eavesdropping' above, but in this chapter we are interested in any possibility of breaking the confidentiality of the wireless link. Attacks against the confidentiality can be classified in two major categories: active and passive, that is to say eavesdropping or Man-in-the-Middle attacks. The Man-in-the-Middle attacks are possible because of weak authentication, which is therefore the second security goal this chapter focuses on. We solely focus on attacks against the wireless connection in this chapter and for now do not look into the end point security. Both the end points of the wireless connection, the mobile phone and the operator's network, will be discussed in Chapters 5 and 6 respectively.

We review the known passive and active attack against confidentiality on GSM's wireless link in Section 3.2. We then discuss the feasibility of performing these attacks using easily available software and off-the-shelve hardware in Section 3.3. This discussion will introduce some specifics of GSM's air interface, but will not give a complete overview. For such an overview we refer the reader to [172] or 3GPP's specifications [164]. Section 3.4 analyses the weaknesses in GSM that enable the discussed attacks. This section also discusses the possible countermeasures within GSM. We then look at the newer generation 3GPP protocols and how they implement security measures that protect against the confidentiality attacks on GSM in Section 3.5. Finally we draw conclusions in Section 3.6.

3.2 Attacking GSM wireless confidentiality

There are two major ways to attack confidentiality on the GSM wireless link:

1. active,
2. passive.

With active we mean those attacks in which an attacker at some point transmits signals himself. This is in contrast to passive attacks where the attacker only ever receives messages. Active attacks are noticeable due to the attacker's transmissions, while passive attacks are completely undetectable.

It is not a question if it is possible to break the confidentiality on the wireless link, offered by GSM. For years there has been commercial equipment available capable of eavesdropping conversations or text messages [157]. How this equipment achieves this is not publicly known. In 2011 researchers demonstrated an eavesdropping attack [13]. This section focuses on how such attacks can work. In Section 3.3 we examine how feasible these attacks are using readily available equipment and software.

3.2.1 Passive attacks – Eavesdropping

Eavesdropping on GSM, or probably any communication system for that matter, can be broken down into three stages:

1. Capturing the signals.
2. Decrypting the captured signals.
3. Interpreting the decrypted signals.

This section will look at these steps in more detail.

Capturing the signals

This first stage was a major obstacle for many years. Specialised equipment to capture the GSM signals has long been around, but was very expensive and often proprietary. GSM can be used on several frequency bands, but most commonly used are the GSM-900 and GSM-1800 bands. The frequency bands are divided into channels of 200 KHz wide each. In a typical conversation two of these channels will be used at any given time for a mobile phone to communicate with the cell tower, one channel for each direction. These channels are separated by a constant offset.

One part of a phone conversation in a GSM network, e.g. 20ms of speech data, is transmitted in four packets, called *bursts* in GSM. These bursts are modulated radio waves transmitted in a time slot of 576.9 μs. Most GSM networks employ *channel hopping*, which is used as a signal quality measure, and causes the transmission to switch to a new frequency after every single burst. The challenge in capturing the GSM signals lies in receiving the bursts on time and in demodulating them correctly, in other words: to be able to follow the channel hopping sequence with enough precision.

SMS messages, on the other hand, are usually transmitted over control channels which can be easier to capture, as these control channels often do not use frequency hopping.

Basically, capturing these signals requires any form of radio receiver capable of receiving signals on the required frequency and then either following all frequencies currently being used, or being able to follow the hopping sequence fast enough. Originally, thoughts went out to using general reception equipment such as the Universal Software Radio Peripheral (USRP). This failed to defeat the channel hopping problem and so only worked on single frequency cells. However, a solution was found in one specific baseband chip (baseband chips are the closed-source dedicated hardware component running the GSM protocol in mobile phones), of which the source code had appeared online several years before. The knowledge of this code made it possible to reverse engineer this chip and find a way to flash the chip with new software, allowing the capturing of GSM signals and feeding these back to a computer. This attack requires several phones in order to bypass the frequency hopping problem. Usually two phones are used for the up link and two phones are used to monitor the down link frequencies. These hardware approaches are detailed in Section 3.3.1.

Decrypting the captured signals

Assuming that encryption is enabled on the GSM network, decrypting the captured bursts is the next step. Otherwise, the decryption step is unnecessary. There are three encryption algorithms defined for GSM: A5/1 and A5/2, both stream ciphers, and A5/3, a block cipher. Of these three A5/2 is by far the weakest and can be broken in less than a second on a personal computer with only a few dozen milliseconds of ciphertext [10]. The A5/3 algorithm is considered the strongest encryption of the three. A5/3 saw a theoretical break in 2010 by Dunkelman et al. [42]. This attacks requires 2^{26} chosen plaintext messages encrypted under related keys. For now this does not lead to a practical attack on A5/3, though it is cause for concern since this weakness does not exist in MISTY, the cipher that A5/3 was based on. At the moment of writing no attack on A5/3 is feasible, the future will tell whether this remains so.

A5/1 has been the main encryption algorithm used in most Western countries. Even though its successor, A5/3, has been recommended for use since 2004, we are only seeing serious adoption since the public attack on GSM from 2011 [13]. January of 2016 was the first time we saw more than half of the mobile networks in the Netherlands support A5/3 [88].

A5/1 is a stream cipher with three registers that clock irregularly (the internal state is shown in Figure 4.7 on page 69) and have a combined size of 64 bits — which is also the size of the session key. The A5/1 algorithm was the first encryption algorithm used in GSM. It was originally kept secret and was only disclosed to GSM manufacturers under an NDA. In 1999 though, Marc Briceno reverse engineered the design of both A5/1 and A5/2 from a GSM phone [25]. Several attacks against A5/1 have been published since then [86, 10, 9].

The best cryptanalytic attack against A5/1 is by Golic in 1997 [86], which recovers the initial internal state of the A5/1 cipher from 64 successive keystream bits, by guessing several bits of the internal state and finding the rest by solving, on average $2^{40.16}$ equations. The most practical attack available to anyone is a Time-Memory Trade-Off attack, for which the look-up tables are now available on-line. We discuss this attack and the available software in Section 3.3.2.

Interpreting the decrypted signals

After the signals have been captured and deciphered they still need to be interpreted. The payload of the bursts needs to be reordered and can be checked for transmission errors. Besides the cryptography, all the specifications of GSM are public, so this does not require any reverse engineering. Several projects that implement a GSM stack are discussed in Section 3.3.3 and can be used to interpret the decrypted signals.

3.2.2 Active attacks

We described active attacks on the wireless link as attacks in which the attacker transmits signals. These could for instance be jamming the frequencies to achieve a Denial-of-Service, or *fuzzing* attacks, were malicious packets are transmitted to network equipment to trigger undefined behaviour, as is discussed in detail in Chapter 6. However, in this chapter we are focusing on attacks against confidentiality on the

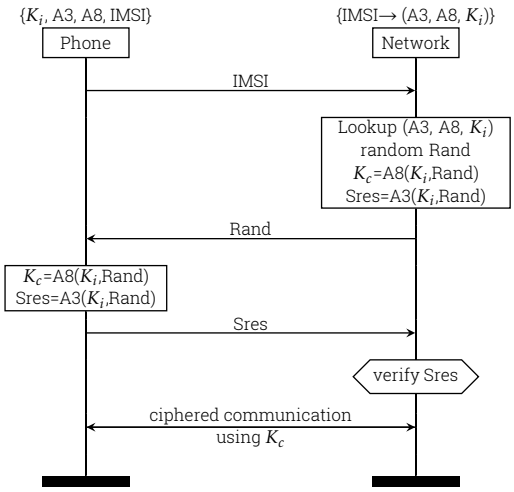


Figure 3.1: *A schematic representation of authentication in GSM. A successful authentication will also result into both sides having the same session key.*

wireless link, which leaves us with three more-or-less separate attacks: a full MitM attack, a sort-of one-way MitM attack and an active key-retrieval attack. These attacks will be detailed in the following sections, where we assume an attacker is capable of interpreting the signals he intercepts, e.g. by way of the open source projects discussed in Section 3.3. We discuss the available software for running an active attack in Section 3.3.4.

All of these attacks are possible because in GSM the mobile phones (actually SIM cards) do not authenticate the cell towers. Figure 3.1 shows the authentication steps between a SIM card in a mobile phone and the mobile phone network. Such an authentication is always triggered by the network and is based on both the network and the SIM card (handed out by the network provider) knowing the same symmetric key (K_i). Before the authentication starts the network needs to know which SIM it is dealing with, so at some earlier point in time the SIM has to have identified itself. Usually the SIM will identify itself by transmitting a temporary pseudonym (called the TMSI, for Temporary *IMSI*), but at least the very first time the SIM identifies itself to the network, or any other time when the network does not recognise the TMSI, the SIM will transmit its unique identifier, the IMSI (International Mobile Subscriber Identity). The network transmits a challenge (Rand) and upon reception of the challenge the SIM computes a response (Res) using the challenge, the secret key and an authentication algorithm (A3). The network verifies the response and if found correct then the SIM card (and by extension the mobile phone containing the SIM card) is authenticated. After a successful authentication both parties also end up with a shared session key (K_c) based on the challenge, which can be used to encrypt further communication.

Both the authentication procedure and command to start encrypting the commu-

nication are initiated by the network alone. The authentication and key-generating algorithms are called A3 and A8, respectively. In essence, both need to be cryptographic hash functions. Providers can choose whatever they want for these algorithms, since they control both the SIM card and the network server generating the challenge and resulting response and session key. Some example algorithms are provided for combined A3/A8 calculation by ETSI, which are called COMP128, which in time-honoured tradition were kept confidential. These were quite popular in the early days of GSM, but after being reverse engineered by Briceno et al. they proved to be simple to break [24]. The reverse engineering of A3/A8 also showed that COMP128 actually delivered a 54 bits session key with ten appended zeros. Now it is assumed most providers stopped using these weak authentication algorithms. Indeed, of all the SIMs that were examined during this research project – around fifteen of them – only one pre-paid SIM still generated 54 bit keys. 3GPP proposed new algorithms for A3/A8 for 3G capable (U)SIMs, leveraging the much better algorithms provided by 3G [60]. These algorithm have all been published and have thus far withstood public scrutiny.

The encrypt command includes the choice of encryption algorithm, which can be a null-cipher (A5/0), the original A5/1 cipher, the much weaker A5/2 or the strongest cipher A5/3. For this section we will assume the network uses strong encryption (i.e. A5/3) as otherwise the whole exercise of an active attack is less useful.

Man-in-the-Middle

In a MitM attack an attacker places himself in between a phone and a cell tower, acting as the cell tower towards the victim's phone and as the phone towards the cell tower. Figure 3.2 gives a schematic overview of the MitM attack.

Assuming the attacker cannot break the authentication algorithm, and the challenges are not repeated, the attacker can not provide the correct response to the challenge himself. All he can do is forward the challenge to the actual mobile phone, and forward its response back to the network. After this step both the network and the phone will have a shared session key for encryption, which remains unknown to the attacker.

For all communication that follows, no explicit authentication step is performed, the identity of the phone is assumed by the network if it knows the shared session key, until the network decides to re-authenticate the phone.

Retrieving the session key can be done by initiating some false incoming communication to the phone, which is encrypted with the very weak A5/2 algorithm. This communication can be anything, e.g. a silent SMS, so the victim remains unaware of the attack. The attacker, acting as the valid cell tower, will issue a command to start ciphering to the phone (CIPHERING MODE COMMAND), which includes the chosen cipher (e.g. A5/2). after which the phone will respond with a success message (CIPHERING MODE COMPLETE) encrypted with the chosen cipher. Since the contents of this success message can be guessed almost completely, the attacker can retrieve the key used for encryption, as described in Section 3.2.1, which is the session key shared between phone and network. After obtaining the key and closing the connection with the phone, the attacker is in a MitM position between phone and network, capable of intercepting any communication until the next authentication command

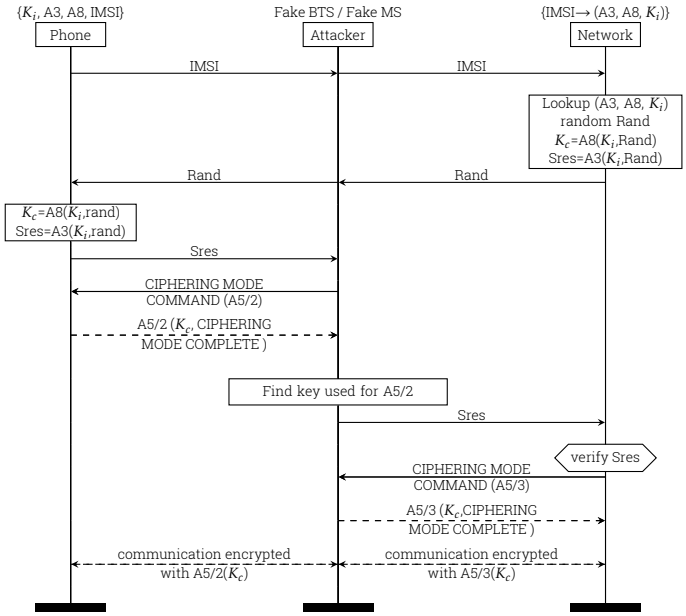


Figure 3.2: *A schematic representation of the MitM attack.*

from the network.

Barkan et al. [10] argued the possibility of attacking the authentication step in much the same way. This attack is illustrated by Figure 3.2, where the specified delay between the network's challenge and the phone's response – a whole 12 seconds – is long enough to retrieve the session key in the same way as in the attack above, before sending the response on to the network. This is a stronger attack, as any authentication command is immediately defeated.

There is another MitM attack possible, which is also presented by Barkan et al. [10], wherein the attacker changes one of the initial messages sent from the phone to the network detailing the phone's capabilities. This so-called class-mark message contains among others, the ciphering capabilities of the phone. By removing the phone's ability to support A5/1 and A5/3, the network will have no choice but to default to A5/0 or A5/2.

One-way MitM

The previous scenario showed a classic Man-in-the-Middle attack. In this easier variant the attacker only acts as a cell tower towards the victim's phone, but not as the victim's phone towards the network. This means that this MitM attack will only help the attacker listen in on outgoing traffic (i.e. traffic instigated by the mobile phone) and not incoming traffic [142]. This attack is represented schematically in Figure 3.3.

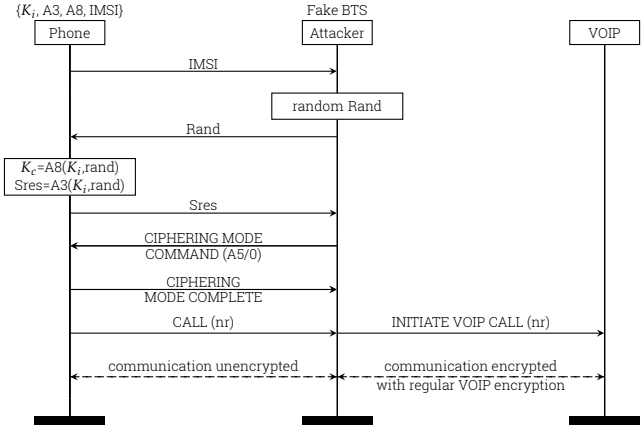


Figure 3.3: *A schematic representation of the one-way MitM attack.*

Here, the attacker acts as a fake cell tower (fake BTS) and when the victim phone connects, starts the standard authentication procedure. Since, the attacker does not know the correct response to his challenge, he will simply accept any response coming from the victim. Then, once the mobile phone wants to start a call (or send out a text message), it will notify the attacker's cell tower of this. The attacker's cell tower simply behaves as a normal cell tower, but tells the mobile phone not to use encryption. The attacker will then need a different channel through which he forwards the call. This can for instance be another mobile phone, or a VOIP connection. The attacker then sets up the connection to the callee, using the encryption required by the medium he is using, encrypting the data stream of the caller and decrypting the data stream of the callee. In this attack the callee could notice the incoming call originating from a different number than normal. Therefore, the attacker should use a hidden number; This could still deviate from what the callee expects, but would be less suspicious.

Active key-retrieval

As a last, partly active attack we consider the active key-retrieval attack. In this attack an attacker obtains both the initial authentication command from the network and the subsequent encrypted communication to and from the victim's phone. The attacker does this either by passive eavesdropping, or through a MitM attack. The attacker can then, at any moment in time, present himself as a cell tower to the victim phone, re-using the captured authentication command. As the resulting session key is derived from just the random and the secret key, this results in the same session key. The attacker can then start any form of encrypted A5/2 communication with the victim phone to retrieve the session key and then decrypt the encrypted communication obtained earlier [10].

This attack is essentially the same as the first MitM attack, except for the timing

of the key-retrieval. Also, this attack reduces the time the attacker has to be active.

3.3 The practical implementation

Eavesdropping on GSM is possible — equipment can be bought that allows for eavesdropping on A5/1 and A5/2 encrypted conversations [157], but this equipment is sold restrictively to law enforcement agencies and military, and at a high price. But attacks using readily available equipment and some open-source software[1] against confidentiality on GSM's wireless link were presented in 2010 [132] and demonstrated in 2011 [13]. This section covers the current landscape of such hardware and software.

As we discussed in the previous section, a passive attack requires the means to (1) capture the signals, (2) decrypt the captured signals and (3) interpret the decrypted signals. For an active attack, the only extra functionality needed is to act as a cell tower and possibly as a mobile phone. We will look at each of these separately. We reviewed most of these hardware components and software projects in the background section 2.4.

3.3.1 Capturing the signals

For capturing the GSM signals, there are currently two open-source options:

- AirProbe combined with a USRP, or
- OsmocomBB combined with specific phones.

Of these two options the second one, OsmocomBB, is currently clearly the best.

Capturing using AirProbe and a USRP

AirProbe [29] is an open-source project implementing an air-interface analysis tool for the GSM (and possibly in the future also later 3G standards) mobile phone standard. It uses parts of the GNU Radio library for its signal processing. One part of the project handles the reception of GSM signals (using the GNU Radio functions) while another part can also be used to interpret the GSM signals, which is why we will get back to AirProbe in section 3.3.3. Currently AirProbe is only able to listen to the down link (cell tower → mobile phone) of conversations, so some development is still required.

GNU Radio works with several different types of RF hardware, such as sound cards, but it is mostly used in combination with the Universal Software Radio Peripheral (USRP). In the background chapter (Section 2.4.4) we already discussed the problems using the USRP as a generic GSM eavesdropper. In short, the standard USRP1 can only support eavesdropping on cell towers that do not use frequency hopping. Improved versions of the USRP might be able to overcome this limitation, but the prices also increase to over $2,500. Using these newer USRPs would also require quite some extra software development effort, but it seems most development on getting the USRPs to work with AirProbe to eavesdrop GSM communications was halted when it turned out eavesdropping also works with a couple of $15 phones and the OsmocomBB software.

[1] With an eye on responsible disclosure not all required software was released to the public.

Capturing using mobile phones and OsmocomBB

Every GSM-capable mobile phone is by definition capable of capturing GSM traces of the air. However, within every GSM capable phone there is a baseband chip which connects to the phone's transceiver and handles the GSM protocol. These baseband chips are all closed source, making it very complicated to direct a standard mobile phone to listen to frequencies and time-slots outside of those designated to that phone.

This problem was overcome when the registry level manuals of the Texas Instruments Calypso baseband appeared online for some time. These manuals helped some researchers to re-flash a functioning Calypso baseband chip with new firmware. The new firmware, called OsmocomBB [139], runs the lowest layer of the GSM wireless protocol stack on the baseband and the layer two and three on a computer connected to the baseband. This allows the computer to order the baseband to tune to specific frequencies and time-slots. The Calypso baseband chip is available through several old and cheap mobile phones, such as the Motorola C115, C117, C118 and C123, the Sony-Ericsson J100i and the Pirelli DP-L10. The OsmocomBB website contains instructions on how to connect several of these mobile phones to a computer. These phones are limited to capturing downlink (cell tower to phone) traffic; hardware filters prevent them from receiving the uplink channels. Again, there are instructions on the OsmocomBB website to remove these filters.

The OsmocomBB software then offers several command line tools to interface with the phones. There are tools to dump the common broadcast channels of cell towers. The actual tools to capture the signals belonging to a specific phone on directed channels are not released within OsmocomBB.

3.3.2 Decrypting the captured signals

A project was publicly announced in August of 2009 suggesting a way to efficiently break the A5/1 cipher. This project runs under the, slightly unimaginative, name *A5/1* [130]. However, in July 2010 the look-up tool for this project was released and named *Kraken*. To avoid any confusion with the cipher we will refer to the A5/1 project by the name Kraken.

The Kraken project mainly consists of creating large tables in a generic time-memory trade off. This had been proposed before [122], but the distinguishing factor of this new project is that instead of computing the tables at one place, everybody on the internet could join in and compute a table and then share them via bit torrent [133]. The code to compute these tables can be downloaded and it runs on certain types of NVIDIA and ATI graphics cards.

These tables and the exact Time-Memory Trade-Off technique used here is discussed and analysed in detail in Chapter 4. In short, the idea behind these tables is as follows. The contents of several bursts that are sent through the air, after encryption is enabled, can for a large part be guessed. This gives known plaintext samples. XOR-ing those plaintext samples with the actually captured ciphertext reveals keystream samples. The tables now function as a code book with 64 bits of keystream that are mapped to internal A5/1 states producing that exact piece of keystream. Since storing all possible 64 bits of keystream would be infeasible, these tables store a subset of these possible samples, making this attack probabilistic.

In 2010 a set of tables, named the 'Berlin set,' was released together with the look-up tool, Kraken. The tables were distributed via bit-torrent in a special transport format. Currently this is around 1.5TB of data. This transport format can then be transcoded into the actual read format, which takes just under 1.7TB disk space. If the key is in the tables, then Kraken will typically find it within a few minutes (around 1 to 4 minutes) on a Intel Core2 Quad 2.33GHz machine with the tables divided over several disks. Using solid state memory instead of conventional hard drives for the table storage significantly improves on this look up time, at the expense of an increase in the financial costs.

In the attack that was demonstrated, the hackers transmitted a single SMS message to a victim and recorded the reception of that message. Recovering the session key from this SMS reception allowed eavesdropping on all following communication using the same session key. It turned out that most providers keep the same "session" key for several hours. Some refresh the session key before transmitting SMS messages, but not before phone calls. It is currently unknown how many providers changed this setting, which we assume should be simple to adjust.

So in this approach the attacker sends an SMS message, making this a more active attack. However, there are several (partly) known plaintext messages sent encrypted over GSM, also during phone conversations, to allow this attack to happen completely passively. A knowledgeable attacker should be able to make fairly strong known plaintext guesses based on observations of the wireless transmissions within a cell. Software which automatically makes known plaintext guesses has not been released for reasons of responsible disclosure.

A major drawback of this approach is that it requires faultless reception of 64 consecutive bits. A single flipped bit in a captured sample will make the sample useless for retrieving the session key using these tables. In GSM the encryption is performed on top of the error detection codes, so the error detection can not be used to find reception errors before the decryption step. Currently the reception from the USRP, while running AirProbe, does not provide the faultless reception needed by Kraken. Reception with the specific TI Calypso based phones using the OsmocomBB software provides reception that is good enough to perform this attack. Still, when generating encrypted bursts yourself and feeding the keystream directly into Kraken, the tables usually find the session key once in every five bursts. With actual captured samples this success rate dropped to about one in twenty in our experiments.

3.3.3 Interpreting the decrypted signals

After the demodulation and decryption steps the bursts need to be interpreted. There are currently several open-source projects that implement at least part of the GSM stack. These are the OpenBTS [26], OpenBSC [189], and AirProbe [29] projects.

The *OpenBTS* and *OpenBSC* projects both aim to offer a functional open-source GSM network. The *AirProbe* project on the other hand aims to create a functional sniffer for GSM traffic. So for eavesdropping activities the AirProbe project seems the most logical choice. However, the AirProbe project is still lacking some essential functionality. For one, the type of a received burst is decided following the standard, most likely, division of the broadcast channel, instead of making this decision based on the structure of the received burst. This means that the results are worse when cell towers

use non-standard division of the broadcast channels. Also, currently the AirProbe sniffers can only interpret some types of bursts. At the moment a lot of development to AirProbe is still necessary in order to be able to receive and interpret all of the GSM bursts. There does not seem to have been a lot of development into AirProbe over the last three years.

The OpenBTS and OpenBSC software on the other hand are actively maintained, but do not offer the functionality to read in decrypted signals from a file and interpreting these. Naturally, all this functionality is present in these software stacks, but an attacker would still need to develop this ability.

3.3.4 Transmitting GSM signals

The transmission of signals is the only component of these attacks specific for active attacks. When we look at the active attacks discussed in Section 3.2.2, we can see that all attacks require an attacker to act as a cell tower and for one specific attack (the MitM attack shown in Figure 3.2) the attacker also needs to be able to impersonate a mobile phone.

For this first capability, impersonating a cell tower there are different options:

- OpenBTS software using a USRP, or
- OpenBSC software using a supported hardware device.

Both options work fine and only require an attacker to implement his own MitM logic. We used the USRP and OpenBTS combination in our research, as it was hard to get the supported hardware for OpenBSC when we started in 2010. Now OpenBSC also supports off-the-shelf hardware.

Acting as a mobile phone is possible using the OsmocomBB software and compatible phones. The "mobile" program within OsmocomBB implements most of the required functions for a mobile phone. Again requiring an attacker to add the logic for his attack.

3.4 Analysis and countermeasures

Reviewing the *passive* attack described in Section 3.2.1 and the practical implementation section above, we can pinpoint the root causes that make these attacks possible:

- re-use of session keys,
- weak ciphers,
- with a too small key length, and
- too much guessable plaintext.

Almost all of the weaknesses of the passive attack can be found in the cryptography primitives. The A5/1 and A5/2 ciphers can both be considered weak. In the case of A5/2 this was by design, but A5/1 was, and still is, the main cipher used in GSM. There are several weaknesses within A5/1, but the main points that made the cipher

breakable by a hacker's cracking project are: (I) the internal state of the stream cipher is made up out of a meagre (for modern eyes) 64 bits (II) the state-space of the internal state collapses after several rounds to around 61 bits. These weaknesses of A5/1, combined with the large number of known plaintext samples led to a workable Time-Memory Trade-Off attack, which is discussed at length in Chapter 4.

The only non-crypto weakness exploited by the eavesdropping attack is the large source of known plaintext in the GSM protocol. Messages transmitted over the wireless interface are required to have a standard length. If the message contained within them is smaller, then the message is padded to the standard length. This padding consists out of a standard pattern of "2b" in hex. There is a revision to the GSM standard to add random padding to messages [59], but this seems to be hardly ever implemented [88]. Another source of known plaintext are the so-called SYSTEM INFORMATION 5, 5ter and 6 messages, which are standard control messages transmitted at a predictable interval and transmitted both unencrypted and encrypted. An attacker would need to observe the cell in which he wishes to perform his attack in order to gain the known plaintext of these control messages and an indication of when they are transmitted. Again, because of reasons of responsible disclosure, no software was released that automates this work.

Looking at the *active* attacks we can identify the weaknesses within GSM that make these attacks possible:

- lack of mutual authentication,
- support for weak or no encryption algorithms,
- each encryption algorithm uses the same key,
- the authentication can be replayed,
- unprotected class-mark message (discussed on page 35).

Most of these weaknesses are on the protocol level. The fact that mobile phones do not authenticate the network is the obvious weakness here and essentially the root weakness of every active attack. This allows an attacker to impersonate the network, and possibly also the phone. The fact that GSM then allows to switch to unencrypted modes, or weaker encryption, is unfortunate.

There are some practical considerations when performing these active attacks.

- The attacker needs to time his attack before the actual data connection he wishes to capture is set-up. Just dropping in mid conversation claiming to be network towards the phone and the phone towards the network would not work as the transmissions of the actual phone and actual cell tower would still reach each other. Also, the attacker would have to transmit in the correct time slot which would interfere with the actual transmissions in that time-slot.
- By posing as a cell tower (of the victim's provider) towards the victim's phone, and ensuring that his transmissions are better received than the actual cell towers (e.g. by also jamming the original cell tower's frequencies), the phone will automatically switch to the attacker's fake cell tower. As these are mobile

phone systems, the whole design is tolerant to phones being mobile. So, the network will not be suspicious if a phone no longer responds and the phone is not suspicious if it sees a new cell tower.

- Since all the active attacks require the attacker to be transmitting, these attacks could be noticed when the correct frequencies are being monitored. Also, all attacks that trick the victim phone into using A5/0 are in principle detectable, as this requires the mobile phone to display an open lock symbol. However, there is an option on the SIM card which prevents mobile phones from showing this symbol [58]. There are now also phones on the market that will warn a user when the phone is forced to use A5/2, or to use no encryption [157].

3.4.1 Countermeasures

As we already discussed, theoretical attacks against GSM are almost as old as the GSM system itself. So the GSM industry has had ample time to prepare for the practical implementations of these attacks. There are several possible countermeasures against these attacks:

1. encrypt content using A5/3,
2. use random padding in GSM packets,
3. randomise control messages,
4. use newer 3GPP protocols.

These points are discussed in more detail below. These countermeasures mostly protect against passive eavesdropping, with only the last countermeasure protecting against active attackers, who then still have the ability to do a fall-back attack to GSM. The GSM Map project tries to track the implementation of these countermeasures by the different providers [88].

Encrypt content using A5/3

In Section 3.2.1 we already briefly discussed the A5/3 cipher. This cipher has been public from its inception, and as yet no feasible attack has been found. So using this cipher to encrypt conversations prevents eavesdropping.

However, using A5/3 will not improve GSM's security much. This is due to the fact that irrespective of the choice of encryption algorithm, the session key used will be the same. This, combined with the presence of a weak cipher, allow an active attacker to retrieve the session key. Basically the session key is created based on the secret key, known only to the SIM card and the home network, and a challenge transmitted by the cell tower. This challenge is transmitted in the clear, so an attacker could replay the authentication, i.e. the "Active key-retrieval attack", or use the "Man-in-the-Middle" attack, both discussed in Section 3.2.2. The GSMA (GSM Association) had been advising the use of A5/3 by providers since 2004. In recent years we have seen a definite move towards A5/3, with 2016 being the first year when more than 50% of the mobile networks in the Netherlands offer this stronger encryption [88].

Use random padding

This defensive strategy specifically makes the Kraken attack discussed in Section 3.3.2 harder. It revolves around the fact that the information in GSM packets is padded to a standard length using a standard pattern of "2b". Some packets consist almost entirely of padding bits – for example the "cipher mode complete", the first message a cell phone transmits enciphered to the cell tower, usually has 144 of its 265 bits filled with padding bits – which gives an attacker a large source of known plaintext. However, the length of the information bits is already described in the packet header, making the standard padding pattern redundant.

These padding bits can thus be randomised, and that is exactly what was specified by ETSI in 2008 [59]. This would remove a large source of known plaintext for an attacker. Without known plaintext there are no known keystream samples which can be looked up in the Kraken tables.

It is questionable how fast this change will be implemented, however. All the low level GSM processing is done by closed source GSM stacks, so it is unknown whether this change would affect the already deployed equipment. The mobile handsets in the field cannot be updated, so this change can only be made in new phones. Also, this change will not completely remove all known plaintext from the system. Some messages can still be guessed, such as system information messages, making the attack described in Section 3.3.2 still feasible for longer conversations.

Randomise control messages

Another large source of known plaintext comes from specific control messages whose content an attacker can learn by observing cell tower traffic. These messages are then also sent encrypted to users, usually in guessable time slots. To prevent this known plaintext source ETSI specified the randomisation of these control messages in 2011 [70]. This specification is fairly recent, so not many providers have switched to using it [88].

Use newer 3GPP protocols

This is kind of a cop-out, but a method that is at least currently available to quite some users. The successors of GSM, both the third generation UMTS as the fourth generation LTE, offer much better security.

In order to fully use this added security, a user should deactivate his phone's GSM reception, and solely use UMTS or LTE. Otherwise an attacker could force a phone to use GSM, by jamming the UMTS or LTE frequencies. Of course the usability of this solution will depend on the availability of a UMTS or LTE network, and might have additional data costs. Additionally, voice calls are often not supported over LTE and only sometimes available over UMTS.

We will examine the extra security offered by newer generation networks in the next Section.

3.5 Countermeasures on next generation networks

Learning from the mistakes made in the design of GSM, the designs of newer 3GPP protocols included several security updates compared to GSM.

UMTS

UMTS provided several major security updates to the 3GPP protocols. For the wireless interface, the major updates were the introduction of better cryptographic algorithms and mutual authentication between SIM and network.

For the new cryptographic algorithms the 3GPP consortium decided to use a by then more fashionable process of openness, publishing the specifications of the new algorithm KASUMI [65]. KASUMI is a block cipher with a 128 bit key and 64 bit input and output. KASUMI is based on the cipher MISTY1, which was developed by Mitshubishi. The name KASUMI is Japanese for "mist." This new cipher was introduced in GSM as A5/3, with the only difference being a support for keys of size 64 bits (by simply doubling the key bits). As we already discussed in Section 3.2.1, the best known attack against KASUMI is a related key attack requiring 2^{26} chosen plaintext messages encrypted under related keys [42]. This is infeasible in a UMTS/GSM setting, but it is cause for concern that this same weakness does not exists in the original MISTY cipher.

3GPP also introduced a new stream cipher SNOW 3G, which is based on SNOW, a stream cipher by Lund University [62]. This cipher also revealed a vulnerability under a related key attack [111]. Again this attack seems infeasible within UMTS. Both algorithms have now been under public scrutiny for over ten years and no practical attack has been found, so both algorithms should offer protection against passive eavesdropping attacks.

In addition to confidentiality on the wireless link UMTS also introduced integrity on several messages by including a MAC. The computation of the MAC is also done by using KASUMI, but in a different mode (a chained mode), than for encryption (an output-feedback mode) and under a different key [66].

The most important security update for UMTS though, was the introduction of mutual authentication between network and SIM, which prevents all active attacks discussed in this chapter. A schematic overview of the so-called Authentication and Key Agreement (AKA) procedure in UMTS is shown in Figure 3.4 [68]. The different f1 to f5 algorithms can be chosen by the providers, just as the A3 and A8 algorithms in GSM. The 3GPP consortium does offer suggestions based on Rijndael, combined with different operator codes that get XORed with the random value, for each of the five algorithms.

Just as with GSM's authentication (as explained earlier in Section 3.2.2), the SIM will have to identify itself before the authentication starts. The network can then start to create so-called authentication vectors, which consist of a freshly generated random which acts as a challenge, the corresponding response (Sres), confidentiality key (CK), integrity key (IK), anonymity key (AK) and an authorisation token (AUTN). The AUTN token, is the authorisation proof by the network. It consists of the sequence number XORed with the anonymity key, the Authentication Management Field (AMF) and a MAC over SQN, AMF and Rand. The masked sequence number protects against

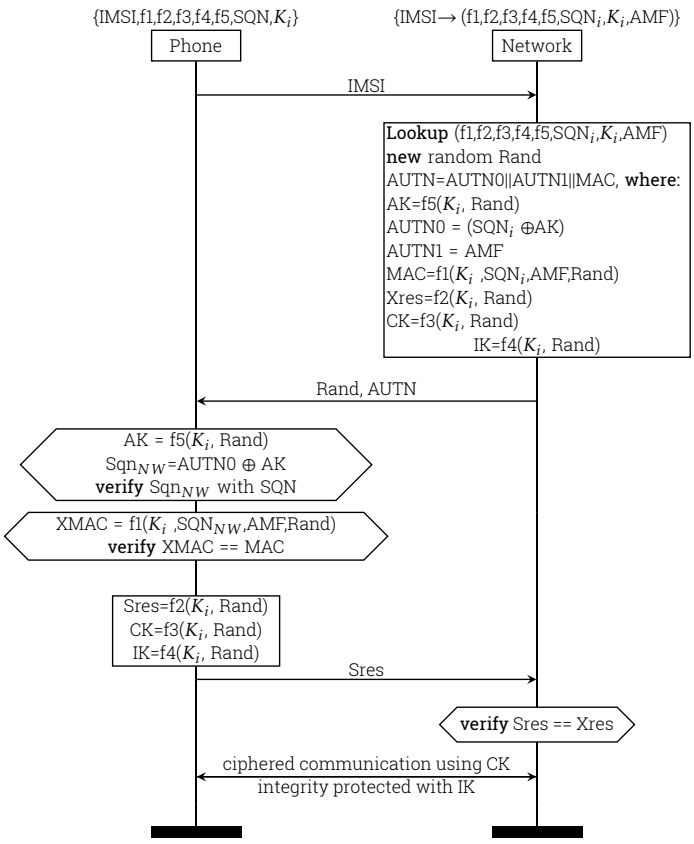


Figure 3.4: *A schematic representation of successful authentication and key agreement in UMTS.*

re-play attacks, the AMF is for the provider to use, for instance to signal a specific algorithm suite, or set a time validity for a key and the MAC authenticates this message coming from the network. The challenge Rand and the AUTN token are then transmitted to the SIM.

The SIM, upon reception of an authentication request, will first verify the sequence number, by computing the anonymity key and retrieving SQN out of the AUTN token. The SIM verifies that the sequence number from the network is higher than its own sequence number. If the received sequence number is lower, or too high, then the SIM will respond with an error message and a genuine network will then start a re-synchronisation setup. By how much the sequence numbers can deviate from each other is a setting chosen by the provider.

If the sequence number falls within the range of allowed sequence numbers, then the SIM will verify the MAC. If the f1 algorithm is strong enough, then only someone who knows the secret key K_i will be able to generate a correct MAC.

Once the AUTN token is found correct, the SIM will compute the confidentiality key, integrity key and response. The response is transmitted back to the network and the two keys are stored in the phone. When the response is found correct by the network, the network can order the use of integrity protection and ciphering. As is explained in the background section 2.3, the authentication parameters and keys are all computed in a separate entity within the providers network and forwarded to different entities that verify the SIM's response and end the confidentiality and integrity protection.

The AKA procedure had been formally verified using enhanced BAN logic and shown to provide both authentication and confidentiality [57].

Since the IMSI is still transmitted before the network authenticates itself, attacks such as IMSI catchers (a fake cell tower retrieves all IMSI numbers in its vicinity) still work. Also, replay attacks of the authentication token are prevented, but replaying this token will break location privacy, as the SIM will respond differently to an out of sync message than to an incorrect MAC message [4]. But both the passive and active attacks discussed in this chapter are no longer possible.

Of the weaknesses we found in our analysis of Section 3.4, a few still remain:

- There is still support for not using encryption; a null-cipher can be chosen for confidentiality (though not for integrity).
- The capabilities of the phone (the class-mark message in GSM) are still transmitted unauthenticated and before integrity protection is possible.

A few other weaknesses are dependent on specific settings by the provider:

- Session key could still be re-used.
- Each encryption algorithm could still use the same key.

There is also the possibility of a fallback attack to GSM, for instance by jamming the UMTS frequencies.

LTE

The security improvements for LTE on the wireless link are less impressive. LTE introduces two new cryptographic algorithms, each with a key length of 128 bits, but with the possibility to extend the keys to 256 bits in the future. These algorithms are an updated version of SNOW 3G and AES (in counter mode for encryption and CMAC mode for integrity).

With the exception of one extra parameter, the AKA procedure from UMTS is kept entirely intact for LTE. This extra parameter allows diversification of the keys. In LTE there are two different security contexts, one between phone and cell tower and one between phone and a core network component called the Mobility Management Entity (MME). The connection to the MME only transports signalling messages and no user data. Both security contexts use diversified keys, derived from CK and IK.

One of the reasons for these separate security contexts is to enable cheaper hand-overs between directly interconnected cell towers, without involving other network components. During such a hand-over the old cell tower can transfer the keys of its

security context to the new cell tower. However, the LTE security design requires two additional security features: backward security and forward security. *Backward security* is a feature where if the security keys leak after a hand-over, for instance through a compromised cell tower, all the information encrypted before that hand-over remains confidential. This feature is easily obtained by transmitting a hash of the security keys to the next cell tower, so a compromised key will only leak a hash of previous keys. *Forward security* is the feature where if the security keys leak before a hand-over, all the information encrypted after that hand-over is again confidentially protected. This requires a fresh key generated after a inner-cell tower hand-over, by the MME and handset. This forward security feature undermines the benefit of not involving the MME during hand-overs. Therefore 3GPP defines n-hop forward security, where any cell tower with knowledge of the current keys is unable to predict the keys used after n handovers. These terms of forward and backward security are defined by 3GPP and are somewhat confusingly named with respect to Perfect Forward Secrecy (PFS). PFS has more in common with backward security, but is not the same. In fact, the LTE hand over security is not PFS, even though it is forward and backward secure [183].

The new key diversification in LTE adds extra security properties, but does not offer additional protection against the weaknesses common between GSM an UMTS described above.

3.6 Conclusions

Passively eavesdropping on GSM remains pretty hard to do using publicly available hard and software. Theoretically, there are no real constraints in breaking conversation confidentiality in GSM using the A5/1 cipher. However, there are still several practical issues making a working implementation of a GSM sniffer using freely available hardware hard to do.

First of all, some essential software has not been released. Although an explanation on how to perform these steps is available, it is still some work to do this. Moreover, when all the required software is available, then the attack is still far away from a catch-all attack, able to eavesdrop on any GSM conversation. This is again due to several practical limitations, such as reception quality and limited coverage of the pre-computations tables.

The release of the rainbow tables and the Kraken tool has made the breaking of the A5/1 encryption much easier. However, this approach does have a few downsides: besides the hard disk size this method also requires perfect samples – putting additional strain on the capturing process – and, as is normal with such tables, they will never give a 100% chance of finding the key. Still, the current coverage is workable given enough samples. It also turns out that it is currently hard to obtain the pre-computation tables online, as only very few people seem to share them.

The presented active attacks all have the same problems on the reception level, but bypass most of the decrypting issues. Their biggest downside is that they are noticeable attacks, as the attacker has to transmit signals, although victims have to be very observant to notice. These issues with active attacks are only present when trying to create these attacks with easily available software and hardware. A well-funded attacker can simply buy practical solutions.

Of the countermeasures that are often referred to by the GSM industry when downplaying the news stories, the most effective one is essentially to by-pass GSM all together and solely use the newer 3GPP protocols UMTS and LTE. However, this could lead to degraded service, as the coverage is not always as good as with GSM and providers are keen to keep voice calls on GSM, keeping their UMTS and LTE frequencies free for high definition video downloads, or other internet traffic. The adoption of voice over UMTS has increased in recent years, often as an additional service called "HD Voice," but voice over LTE is very rare.

UMTS, the successor of GSM introduced, all the added security needed to mitigate GSM's biggest weaknesses: weak encryption and lack of mutual authentication. Regrettably, some minor weaknesses are still present, even in UMTS's successor LTE.

Most of the security of the mobile phone network is dependant on the provider settings, such as which encryption algorithms are supported, and the aforementioned voice over UMTS. These settings differ per location area, per provider. As such, it is hard to make a general assessment of the confidentiality on the wireless interface.

Finally, when the goal of the attacker is to only capture SMS messages, or impersonate the victim, he could perform a much simpler attack: SIM card fraud. In this attack the attacker simply requests a new SIM card at the provider of the victim for the victims mobile phone number. If the provider does not sufficiently verify the victim's identity, then this is a simple, but short-lived, impersonation attack, as the victim will quickly notice a complete lack of service.

Chapter 4

Time Memory Trade-Off attacks

The previous chapter discussed confidentiality attacks on the wireless links of mobile telephony networks, by focusing on the protocols and actually capturing signals. Naturally, the protection for confidentiality is offered by encryption, therefore this chapter will focus on the cryptography used on the wireless link. In a passive eavesdropping attack (and also in some active attacks), an attacker will need to break the encryption in order to obtain his goals. Therefore, we need to know how hard it is to break the actual encryption. So, we focus on the, at that time, most widely used cipher in GSM, A5/1. This chapter looks at the Time-Memory Trade-Off (TMTO) attacks possible against stream ciphers and the specific TMTO attack successfully used against GSM's main cipher. We specifically focus on the time and memory costs of each different attack, with a, for that time, new analysis.

This chapter is based on the paper *A comparison of time-memory trade-off attacks on stream ciphers*, presented at the 6th International Conference on the Theory and Application of Cryptographic Techniques in Africa, AfricaCrypt 2013 [176]. This publication referred to the Fuzzy Rainbow Table attack as the "Kraken" attack, because at the time we were not aware of the earlier work [11] proposing this type of attack. This chapter also contains more information on new publications since our original publication and looks deeper into the practical comparison of TMTO attacks. Finally, this chapter includes our attempts at improving the Fuzzy Rainbow Table attack, which did not fit within the page limit of the original publication.

4.1 Introduction

There are many scenarios in which an attacker wants to reverse a cryptographic function, such as a hash function or a cipher. An attacker trying to break a cryptographic function can always try to either brute force the function, or precompute all possible values beforehand and store them in a large table, so every subsequent attack is a simple look-up. Most cryptographic functions are protected from these attacks by having a large enough key size or state size, which makes the time complexity or the storage requirements of such attacks too large in practice.

In 1980 Hellman caused a breakthrough by suggesting a Time-Memory Trade-Off attack which is probabilistic and falls somewhere in between a brute force attack and a precomputation attack. Hellman showed that using his attack he could reverse an n-bit key cipher, in $2^{2n/3}$ time complexity, by precomputing 2^n values and storing $2^{2n/3}$ of them [95]. The total amount of work done in this attack is more than in a single brute-force attack, but with each subsequent attack the Time-Memory Trade-Off attack is much cheaper. This made ciphers using keys that until then were thought large enough to prevent a brute-force attack suddenly susceptible to this new Time-Memory Trade-Off attack.

Later research into TMTO attacks led to many improvements on Hellman's attack. First came the Distinguished Points method, which reduced the number of disk seeks and is referenced to Rivest [38]. Later Oechslin [136] devised a competing method with a slight speed-up, called Rainbow Table. The Rainbow Table attack seems to be better known, presumably due to its colourful name. Biryukov and Shamir [16] combined Hellman's attack with a specific data-tradeoff attack against stream ciphers [8, 86] resulting in a more efficient TMTO attack for stream ciphers. An attacker can make generic TMTO tables for a stream cipher which can be matched against any large enough sample of keystream, increasing the success chance with every sample. This new understanding directly led to new proposed attacks against one of the most widely deployed stream ciphers in the world: GSM's A5/1 cipher [17, 10].

In 2010 researchers demonstrated a TMTO attack to break the A5/1 cipher of GSM [134]. This attack used a TMTO method which combines two important, but very different TMTO improvements; namely Distinguished Points and Rainbow Tables [130]. This attack was previously suggested by Barkan et al. in 2006 [11], who called it a *Fuzzy Rainbow Table* attack, but which saw very little research attention since its inception. The tool created for breaking the A5/1 cipher was called Kraken.

It seems rather strange for these researchers to have chosen an, at that time, unresearched approach for their attack, so the question arises whether this Fuzzy Rainbow Table attack improves on the already existing attacks. This chapter aims to investigate how much, if any, of an improvement Fuzzy Rainbow Table attack brings to the area of TMTO attacks.

Section 4.2 introduces the general idea of TMTO attacks. Section 4.3 introduces and analyses the four TMTO attacks: Hellman's original attack [95] with Biryukov and Shamir's improvement for stream ciphers [16], Rivest's Distinguished Points approach [38], Oechslin's Rainbow Tables [136], and the first theoretical analysis of the Fuzzy Rainbow Table attack (Section 4.3.4). Most of these attacks have previously been analysed by deriving trade-off curves, which we feel hide too much of the actual costs of these attacks. This is why we performed a new analysis which we expect

provide more insight in the actual costs associated with such attacks. We compare the TMTO attacks in Section 4.4, including an informal analysis on the chances of chain merges, based on our new analysis. Then we attempt to improve the Fuzzy Rainbow Table attack in two separate ways in Section 4.5. Section 4.6 gives an overview and analysis of the practical implementation of fuzzy rainbow tables for Kraken: the attack tool against the GSM cipher A5/1. Finally, Section 4.7 discusses related work and some ideas for future research are given and conclusions are drawn in Section 4.8.

4.2 Typical TMTO

Assume a scenario in which an attacker tries to break a known cryptographic function f for which he has obtained at least one sample of cipher text y. His goal is to reverse the function f, i.e. to find an input x for which $y = f(x)$. This model covers different scenarios:

- Finding the pre-image x of a hash function f for the hash value y.
- Finding the key x used to encrypt a known plaintext p to produce y, i.e. a key x such that $y = f(x) = encrypt_x(p)$, with $encrypt$ e.g. a DES encryption.
- Finding the internal state used to encrypt a known plaintext p with a stream cipher. Here x is the internal state of cipher f and y is the corresponding keystream. So $f(x) = y$ and y is obtained by XORing cipherstream and known plaintext.

This chapter is concerned with the third scenario, finding the "internal state" x of a stream cipher and not the key. Note that in many stream ciphers it is possible to retrieve the key that was used from a given internal state. The essential difference when reversing a stream cipher or a hash function compared with a block cipher is that an attacker can construct tables which are more generic, so they can accept multiple samples from different plaintexts as explained in Section 4.3.1. This shows in the three scenarios above, where you can see that for the block cipher case the computation of $f(x)$ requires a specific plaintext p which the other two scenarios do not need.

When trying to break any cryptographic function there are always two approaches available: exhaustive search and a dictionary attack. In an exhaustive search, an attacker simply attempts all possible values of x to see which $f(x)$ matches the y. In a dictionary attack, the attacker first precomputes a table in which he stores all possible $\langle x, f(x)\rangle$ pairs (or many likely $\langle x, f(x)\rangle$ pairs) and simply looks up the corresponding x for which $f(x) = y$ during the attack. Both approaches quickly become infeasible when the search space of possible x's is too large. For exhaustive search the computation time needed per attack becomes too long, while for a dictionary attack the computation time, as a one time cost, may be manageable, but the memory costs will often be too high.

This is where the Time-Memory Trade-Off (TMTO) attacks come in. Trading off the computational and memory costs may result in an achievable optimum for both. This usually also involves trading off the chances of a successful attack and possible other factors.

A typical TMTO attack consists of two phases: the first is the precomputation phase, often called the *offline phase*, while the second is referred to as the real-time, or *online phase*. In the offline phase, the attacker precomputes a large table (or sets of tables) using the function f he is trying to break, while in the online phase the attacker captures a sample of keystream and checks if this happens to be in his tables. If this attack is successful the attacker can learn the internal state x for which $y = f(x)$. We can evaluate these kinds of attacks by looking at different parameters and costs:

- N: the size of the state space.
- T (Attack time): This can be subdivided between the time for the offline phase, T_{pre}, in orders of magnitude, and the time for the online phase, which in turn can be subdivided into computation time T_c, measured in computation steps of f and seek time T_s, measured in number of disk seeks.
- M (Memory): memory cost of the attack.
- C (Coverage): the number of points from N covered by the tables.
- D (Data): number of usable data samples (y's) during the online phase.
- $\mathbb{P}$ (Chance of success): the chances of a collision between the observed keystream and the precomputed tables.
- ρ (Precomputation ratio): the ratio between the number of precomputed points from N and the total number of points N.

Intuitively, the chance of success $\mathbb{P}$ seems equal to the precomputation ratio, $\rho = C/N$, i.e. the number of points covered by the tables divided by the number of points in the search space. However, this is not exactly true for a number of reasons. Firstly, the tables can contain duplicate values. A certain number of duplicate values is to be expected when the coverage increases, however duplicates within the same table can lead to so-called *chain merges*, which cause large parts of table rows to overlap. These chain merges will be discussed in more detail in the next section, but will for the most part be ignored in the analysis until Section 4.4, which details why it is hard to give an estimate on the occurrences of these chain merges. To stress this difference we introduce $\bar{C}$ and $\bar{\rho}$ as variants of the respective variables that do take chain merges into account.

Secondly, a definition of $\mathbb{P} = \rho$ assumes that all outputs of the cryptographic function f are equally likely, so all points in N have the same chance of occurring. This difference between $\mathbb{P}$ and ρ does not matter for our comparisons, where we assume a perfect cipher, but we will see in the practical example of Section 4.6.2 that this assumption does not always hold in practice, where ciphers can have a bias.

Lastly, if an attacker has multiple samples, as he might have for a stream cipher, then the chance of success increases by a factor D, the number of samples.

4.3 The TMTO attacks

This section compares the costs of the four attacks: Hellman's original attack, Distinguished Points, Rainbow Table, and Fuzzy Rainbow Table. For each we give a theorem

that states the cost for the general case with an arbitrary number of tables, followed by a corollary where we align some of the parameters to allow for an easy comparison. For these corollaries we assume an attacker abides by the $mt^2 = N$ rule, which will be introduced in Section 4.3.1, and precomputes enough points so that $D\rho = 1$. So, the corollaries normalise the attack costs, for easier comparison.

4.3.1 Hellman's original TMTO attack on stream ciphers

TMTO attacks were introduced by Hellman for attacking block ciphers [95]. In Hellman's attack the precomputation tables were created using a single piece of known plaintext. During the online phase an attacker needs to retrieve an encryption of that exact same piece of known plaintext in order to match it against his precomputed tables and have a chance on a successful attack. These precomputed tables are useless for other known plaintext/ciphertext pairs.

In 2000 Biryukov and Shamir [16] combined Hellman's attack against block ciphers with a specific trade-off attack against stream ciphers, found independently by Babbage in 1995 [8] and Golic in 1997 [86]. This combination showed that TMTO attacks against stream ciphers have an extra benefit: an attacker can create tables which are more generic, so any piece of known key stream can be matched to them. These samples can even be overlapping. If an attacker has created TMTO tables to look for keystream occurrences of n bits and he obtains e.g. $n + 6$ consecutive bits, this gives him 7 different keystream samples of length n to match with the results in his tables. Since every sample of known keystream has its independent chance of matching with the precomputed values, every sample increases the success chance of the attack. Alternatively, an attacker can make an estimate, D, on the expected number of samples he will be able to obtain in the real-time phase, this enables him to save a factor D on precomputation (both time and storage) to achieve the same success probability as that obtained with an attack on a cipher with $D = 1$. This effectively transforms the time-memory trade-off into a time-memory-'number of data samples' trade-off.

Hellman's attack when applied to stream ciphers goes as follows. In order to reverse the function f, a table is precomputed in the offline phase. In order to cover as much of the N points of the search space as possible, an $m \times t$ matrix is computed starting from random or sequential start points, where the m rows consist of chains of length t and where each point in the chain is a new iteration of f on the result of the previous point (see Figure 4.1, where iterations of f are denoted by f_i, for reasons that will soon become apparent). Finally, only the begin point and end point of each chain are stored (ordered by the endpoints) as the precomputation table. In the rest of this chapter we will talk about precomputation *matrices* and *table*s, where matrices denote the temporary $m \times t$ precomputation chains and tables refer to the end product, essentially the compressed storage of the matrices.

During the online phase, the attacker obtains keystream samples (e.g. by sniffing a known plaintext encryption, or because he can perform a chosen-plaintext attack). He then makes another chain of at most t iterations of applying the function f and for each iteration checks if the result matches one of the endpoints stored in his table. If this happens, he recomputes the chain starting from the corresponding begin point until the pre-image of the ciphertext, thereby reversing function f in an attack time

$$\begin{array}{ccccccc} x_0 & \rightarrow & f_i(x_0) & \rightarrow & f_i(f_i(x_0)) & \rightarrow \ldots \rightarrow & f_i^t(x_0) \\ x_1 & \rightarrow & f_i(x_1) & \rightarrow & f_i(f_i(x_1)) & \rightarrow \ldots \rightarrow & f_i^t(x_1) \\ x_2 & \rightarrow & f_i(x_2) & \rightarrow & f_i(f_i(x_2)) & \rightarrow \ldots \rightarrow & f_i^t(x_2) \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ x_m & \rightarrow & f_i(x_m) & \rightarrow & f_i(f_i(x_m)) & \rightarrow \ldots \rightarrow & f_i^t(x_m) \end{array}$$

Figure 4.1: *A single $m \times t$ matrix of function f_i. Only the first and last points of each chain are stored.*

of order t in the online phase.

Adding more rows to the matrix computed in the offline phase will eventually cause duplicates, two duplicate points in different chains will cause the rest of these chains to cover the exact same points: the chains merge. Merging chains waste storage and precomputation effort on duplicate points. Hellman shows [95] that the probability of success is bounded by:

$$(1/N)\sum_{i=1}^{m}\sum_{j=0}^{t-1}[(N-it)/N)]^{j+1} \leq \mathbb{P} \leq (mt/N). \tag{1}$$

Hellman proves that this lower bound can be approximated to $3/4$ for tables for which $mt^2 = N$. He argues that increasing m and t beyond $mt^2 = N$ is ineffective, since the chance of overlap only increases as m and t increase. Therefore Hellman continues his analysis of using $m \times t$ matrices satisfying $mt^2 = N$. Most of the subsequent work on time-memory trade-offs copies this choice, although there is no real reason for this.

A single $m \times t$ matrix satisfying $mt^2 = N$ covers only $1/t$-th of the search space N. So, in order to cover a larger part of the search space, Hellman proposed to construct l different $m \times t$ matrices each using a variant of the f function, f_i. The function f_i is defined as $f_i(x) = h_i(f(x))$ where h_i is a simple output modification that is different for each table i. In this way, all l tables only have a small chance of duplicate chains (only within a single table). Naturally there are still chances of duplicate points between different tables, but these will not cause chain merges and are thus not so costly. This comes at a cost for the online attack time, where a separate chain needs to be computed per table.

Theorem 4.1 The general costs for Hellman's attack adapted for stream ciphers are:

M = $2ml$ entries,
T_c = tlD f_i-computations,
T_s = tlD seeks in tables of m entries.

Proof. The memory costs equals the costs of one table, $2m$ since it only stores the starting and endpoints, times the number of tables, l. Having l different tables also carries additional costs in terms of attack time during the online phase, since the attacker will now have to create l different chains of length t for every sample, so both T_c and T_s are in the order of tlD f_i-computations or seeks, respectively. □

In this general case, it might seem that the factor D only has a negative impact on the costs, however, the value for l, the number of tables, can be reduced with a factor D when attacking stream ciphers while the success chance remains the same.

Corollary 4.1 When reversing a stream cipher, using D samples and the $m{\times}t$ matrices satisfy $mt^2 = N$ and precomputing enough points to satisfy $D\rho = 1$, the costs are:

$T_{pre} = \mathcal{O}(N/D)$, $T_c = t^2$ f_i-computations,
$M = 2mt/D$ entries, $T_s = t^2$ seeks in tables of m entries.

Proof. The attacker makes l tables, each with a different f_i. Since each table covers $1/t$-th of the search space ($mt^2 = N$) and the attacker expects D samples, he needs t/D different tables to cover enough points to satisfy $D\rho = 1$. So, there are $l = t/D$ tables each covering mt points, which means the precomputation time T_{pre} is in the order of N/D, since $mt^2 = N$ (assuming that $D \leq t$). The costs for M, T_c and T_s follow by simply substituting l with t/D in Theorem 4.1 □

The memory costs M are measured in entries. We are assuming two entries are needed per chain, which is an overestimate, since some bits can be spared by clever storage methods. The seek time T_s is measured in the number of disk seeks necessary for the attack. In his original analysis Hellman ignores the effect that the size of the tables might have on the time of an individual disk seek. In order to achieve a more accurate measure we take the size of the tables into account, but we ignore the way the tables are organised on disk in our analysis by providing the costs in number of disk seeks.

Hellman's attack provides a time-memory trade-off controlled by choice of the chain length t. The table only stores two points for each chain, the begin and end point. As Theorem 4.1 shows, increasing t reduces the memory cost, but increases the time needed in the online phase, as more time is needed for computing the chain. Conversely, reducing t reduces the time in the online phase at the expense of higher memory cost. Note that if we choose $t = 1$ we have a dictionary attack, while if we choose $t = N$ we have a part of a brute-force attack.

4.3.2 Distinguished Points

The use of distinguished points was the first improvement on Hellman's approach. Hellman's analysis has a practical problem: there is a huge time difference between computing f_i and a disk seek to see if any $f_i(x)$ is stored in the precomputation table. In fact, Hellman's t^2 seeks in the precomputation tables are far more expensive than the t^2 f_i-computations [38]. Since Hellman's analysis counted only the computation steps ($T = t^2$) the difference between theory and practice became very big.

In 1982, a solution was proposed referenced to Ron Rivest [38, page 100], namely to identify a subset of special points, called distinguished points. These points should be easily recognised, usually by a fixed prefix, such as the first k bits being '0'. In the offline phase, chains are computed until such a distinguished point is reached, and that point is then stored as the endpoint. If no distinguished point is reached for a certain number of maximum computation steps, the entire chain is dropped and a new one is computed. In the online phase, the attacker starts developing a chain from captured ciphertext until he reaches a distinguished point, and only then does he need to perform an expensive disk seek. If no distinguished point is encountered in the development of this chain after a predetermined number of steps, than this captured piece of ciphertext is not covered by the tables.

Rivest's approach reduces the number of disk seeks, since now only a single disk seek is needed for every chain that is computed during the online attack, instead of one disk seek for every link. This leads to matrices with chains of varying length. However on average the chain length will be $t = 2^k$.

Using distinguished points has one other benefit. When the precomputation tables are finished it is possible to remove all chain merges from the tables, simply by looking for identical end points. After all, if two chains within a table merge, they will end in the same distinguished point. There is not really an easy way to decide which chain to drop from the table, although an attacker could record the number of points in each chain, while precomputing, in order to keep the longest one. Alternatively, keeping both chains will increase the coverage of the search space (assuming different start points where chosen, then a least a single unique point is added to the coverage by keeping merging chains), at the cost of using storage for duplicate points.

Theorem 4.2 The general costs for a Distinguished Points attack are:
$M = 2ml$ entries,
$T_c = tlD$ f_i-computations,
$T_s = lD$ seeks in tables of m entries.

Proof. The memory costs remain exactly the same as in the previous theorem. The computation costs will also remain the same since a distinguished point will on average be encountered after t steps. The disk-seek cost is now lowered to one disk seek per chain. Since the attacker needs to make l chains —one for each table— for every data sample, the seek time is $T_s = lD$. □

Corollary 4.2 For the Distinguished Points attack, where the $m \times t$ matrices satisfy $mt^2 = N$ and precomputing enough points to satisfy $D\rho = 1$, the costs are:
$T_{pre} = \mathcal{O}(N/D)$, $T_c = t^2$ f_i-computations,
$M = 2mt/D$ entries, $T_s = t$ seeks in tables of m entries.

Proof. The attacker again needs to create $l = t/D$ tables, so both the precomputational work and memory storage remain the same. The costs for T_c and T_s are determined by substituting t/D for l in the preceding theorem. □

This approach can actually save some memory in practice, since k bits of every endpoint are constant and need not be stored. This makes the entries smaller, but the number of entries remains $2mt$. The time cost in the online phase also remains t^2 evaluations of an f_i, but now only t disk seeks are expected, instead of t^2 for Hellman's original attack: a disk seek is only needed when a distinguished point is encountered, which happens once for each chain (on average after per $t = 2^k$ computations), whereas in Hellman's original attack it has to be done for all points in the chain. All this comes at the cost of having variable length chains and probably some extra pre-computation work for chains that do not end in a distinguished point.

4.3.3 Rainbow Table

A different improvement on Hellman's approach, called Rainbow Table, was proposed by Oechslin in 2003 [136], with a factor-2 speed-up in the online phase, for an attack with single samples. Additionally, it has none of the overhead that Distinguished

$$\begin{array}{lllllll}
x_0 & \to_1 & f_0(x_0) & \to_2 & f_1(f_0(x_0)) & \to_3 \ldots \to_t & f_t(f_{t-1}(\ldots f_0(x_0)\ldots)) \\
x_1 & \to_1 & f_0(x_1) & \to_2 & f_1(f_0(x_1)) & \to_3 \ldots \to_t & f_t(f_{t-1}(\ldots f_0(x_1)\ldots)) \\
x_2 & \to_1 & f_0(x_2) & \to_2 & f_1(f_0(x_2)) & \to_3 \ldots \to_t & f_t(f_{t-1}(\ldots f_0(x_2)\ldots)) \\
\vdots & & \vdots & & \vdots & \ddots & \vdots \\
x_{ml} & \to_1 & f_0(x_{mt}) & \to_2 & f_1(f_0(x_{mt})) & \to_3 \ldots \to_t & f_t(f_{t-1}(\ldots f_0(x_{mt})\ldots))
\end{array}$$

Figure 4.2: *A $ml \times t$ rainbow matrix using t different f_i functions. Only the first and last points of each chain are stored.*

Points causes with its variable length, sometimes even unending, chains. However, the Rainbow Table attack is mostly known for its smaller chance of chain merges when less than N points are precomputed.

Oechslin suggested to precompute one large matrix (instead of t different ones) with a different f_i for every link in the chain. The name Rainbow Table stems from the idea of calling each simple output modification h_i a different colour; each column has its own colour, so the entire table looks like a rainbow. This prevents some chain merges when few enough points are pre-computed, since now two chains can only merge if they reach the same value in the same column (i.e. while applying the same f_i). Duplicate points can, of course, still occur, but the penalty for these is not as severe since the chains will not merge if a duplicate happens in a different column.

Theorem 4.3 The general costs for a Rainbow Table attack are:

$$\begin{array}{lcl}
M & = & 2m \text{ entries,} \\
T_c & = & \frac{t(t+1)}{2} D \ f_i\text{-computations,} \\
T_s & = & tD \text{ seeks in a table of } m \text{ entries.}
\end{array}$$

Proof. In a Rainbow Table attack there is a single rainbow table which has m chains, of which only the first and last point are stored, so $2m$ entries. The online attack time becomes $\frac{t(t+1)}{2} D$ instead of tlD, because a different f_i is used for every link in the chain. So instead of computing a single chain $(y, f(y), f^2(y), ..)$ for every data sample an attacker now needs to evaluate t chains of a length ascending from 1 to t f_i calculations, with a different f_i for every link:

$$\begin{array}{lllllll}
 & & & y & \to_{f_t} & f_t(y) & \uparrow \\
 & & y & \to_{f_{t-1}} & f_{t-1}(y) & \to_{f_t} \ f_t(f_{t-1}(y)) & \\
 & \iddots & \vdots & \vdots & \vdots & & \text{t} \\
y & \to_{f_0} \ \cdots & \to_{f_{t-1}} & f_{t-1}(\ldots) & \to_{f_t} & f_t(f_{t-1}(\ldots(f_0(y)\ldots)) & \downarrow
\end{array}$$

For each of these t chains, the end point needs to be looked up in the table, for each of the D data samples, which results in tD disk seeks. □

The memory costs are deceptively low in this theorem, since Rainbow Table attacks use only a single table. This means the dimensions of this rainbow table are larger than those of the previous attacks, so m in the above theorem will be larger

than the values of m for the general costs of the Hellman and Distinguished Points attacks for any meaningful comparison. In order to compare this attack to the other approaches we need the rainbow matrix to cover an equal number of points as the previous attacks. The other approaches use t $m \times t$ matrices. With a rainbow table there is only a single table, so this needs to cover mt^2 points. Keeping the chain length t, means the attacker will need mt entries in his table to cover mt^2 points. So we assume an $mt \times t$ matrix, with t different f_i's, as Figure 4.2 shows.

Corollary 4.3 For the Rainbow Table attack, where the $ml \times t$ matrices satisfy $mt^2 = N$ and precomputing enough points to satisfy $D\rho = 1$, the costs are:

$$T_{pre} = \mathcal{O}(N/D), \qquad T_c = \frac{t(t+1)}{2} D \ f_i\text{-computations},$$
$$M = 2mt/D \text{ entries}, \qquad T_s = tD \text{ seeks in a table with } mt/D \text{ entries}.$$

Proof. In the Rainbow Table case there is little difference in costs between the general case and the case where an attacker chooses m and t to satisfy $mt^2 = N$, since there is only a single table and only the chain size determines the attack time. For comparison's sake, we use a rainbow table of dimensions $(mt/D) \times t$, which covers an equal number of points as the previous TMTO attacks, and keeps the values for T_{pre} and M equal. These mt chains are for comparisons sake, so mt chains of length t have the same coverage as the t $m \times t$ matrices of other attacks. By substituting m of Theorem 4.3 with mt/D, the memory costs are fixed. □

Since D will generally be smaller than t, the number of disk seeks is an improvement when compared to a Hellman style attack, though not as much as the use of distinguished points. Also keep in mind that every table seek could be more costly when using a rainbow table, because of its larger size than the l tables used in the other approaches.

The Rainbow Table attack is most known for a smaller chance of chain merges, but the table defined in Corollary 4.3 will have a similar chance of chain merges than the previous attacks for $D = 1$. Because the same amount of points are precomputed in every f_i (all the points in a single rainbow table column, or all the points covered in 1 Hellman or distinguished point table) and a duplicate between those points causes a chain merge. When assuming more samples, or when precomputing fewer points, i.e. $D\rho < 1$, then the Rainbow Table attack will probably have fewer chain merges than the other TMTO attacks.

The online attack time of the Rainbow Table attack is only dependent on the chain length, ignoring the number of entries in the table, which causes the slight speed up for attacks where $D = 1$.

4.3.4 Fuzzy Rainbow Table

Fuzzy Rainbow Tables combine different time-memory trade-off techniques and where first proposed by Barkan et al. in 2006 [11]. In 2009, researchers started a project to break GSM's standard encryption cipher A5/1 in practice, using fuzzy rainbow tables. They proposed the joint creation of a set of TMTO tables to which everyone could contribute [131]. The idea was to share the intense computing burden of a TMTO's precomputation step by having everyone willing to participate perform a part of the computation on modern GPUs, and share their results over the internet. In the end

$$
\begin{array}{l}
x_0 \to f_0(x_0) \to f_0(f_0(x_0)) \to^* k\|y_{00} \to f_1(k\|y_{00}) \to^* \ldots \to f_s(\ldots) \to k\|y_{0s} \\
x_1 \to f_0(x_1) \to f_0(f_0(x_1)) \to^* k\|y_{10} \to f_1(k\|y_{10}) \to^* \ldots \to f_s(\ldots) \to k\|y_{1s} \\
x_2 \to f_0(x_2) \to f_0(f_0(x_2)) \to^* k\|y_{20} \to f_1(k\|y_{20}) \to^* \ldots \to f_s(\ldots) \to k\|y_{2s} \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
x_m \to f_0(x_m) \to f_0(f_0(x_m)) \to^* k\|y_{m0} \to f_1(k\|y_{m0}) \to^* \ldots \to f_s(\ldots) \to k\|y_{ms}
\end{array}
$$

Figure 4.3: *A Fuzzy Rainbow matrix, where $k\|y$ denotes a distinguished point with the first k bits '0'. Only the first and last points of each chain are stored. Note that each chain consists of s Distinguished Points chains.*

however, the project ended up using a set of tables being computed on a single computer. This set was dubbed "The Berlin Set" and its parameters are discussed in detail later in Section 4.6.2. First we focus on the general approach that was used in this attack.

In order to find the internal state of a generic stream cipher, the Fuzzy Rainbow Table approach combines both distinguished points and rainbow tables in the table layout. This is done by first choosing distinguished points as bit strings starting with k zeros. Then, normal TMTO chains are computed by repeatedly applying f_i to random start points until the output is a distinguished point. The chain is then continued but now with a different f_i; in essence changing the rainbow colour. This is repeated for a predetermined number, s, of rainbow colours ($f_0 \ldots f_s$ functions), until a distinguished point is found while using the final f_s of this chain. This point is the endpoint of a chain and is stored together with the corresponding start point in the TMTO table. Figure 4.3 shows such a precomputation matrix. In order to match a sample y against a table during the online phase, s different chains need to be developed ranging in size from t to st f_i-computations, analogously to the Rainbow Table online attack. On average this will lead to one distinguished point per f_i subchain, assuming that each possible output of f_i is equally likely. While applying f_s, the attacker can see if the resulting distinguished point matches the stored endpoint. The distinguished point found in the chain while applying f_{s-1}, needs to be developed further by applying f_s until the last distinguished point is found. This continues to the distinguished point found using the first f_0, which should require a chain of around st computation steps to match the final distinguished point with the stored endpoint, as Figure 4.4 shows. This approach can boil down to compressing s different Distinguished Points tables

$$
\begin{array}{ccccccccccc}
 & & & & & & y & \to^*_{f_s} & k\|x_{0s} & \uparrow \\
 & & & & y & \to^*_{f_{s-1}} & k\|x_{1s-1} & \to^*_{f_s} & k\|x_{1s} & \\
 & & y & \to^*_{f_{s-2}} & k\|x_{2s-2} & \to^*_{f_{s-1}} & k\|x_{2s-1} & \to^*_{f_s} & k\|x_{2s} & \text{s} \\
 & \iddots & & \vdots & & \vdots & & \vdots & & \\
y & \to^*_{f_0} \ \ldots & & \to^*_{f_{s-2}} & k\|x_{ss-2} & \to^*_{f_{s-1}} & k\|x_{ss-1} & \to^*_{f_s} & k\|x_{ss} & \downarrow
\end{array}
$$

Figure 4.4: *The online phase of a Fuzzy Rainbow Table attack. Here $k\|X$ denotes a distinguished point with the first k bits '0'. Only the last point of every chain is matched against the precomputation table.*

into one: Each chain basically consists of s subchains, depending on the choice of s and t. Intuitively, this means that the memory costs will be lowered by a factor s, but the attack time will increase by a factor s. This attack should keep all other advantages of a Distinguished Points attack, such as the easily identifiable chain merges. When compared with a Rainbow Table attack the number of chain merges should rise with a factor $t = \tau/s$, where τ is the new total chain length, because the same f_i is used for each subchain.

The average length of each subchain, t, can be adjusted by choosing a different length of k for the k-bit distinguished point. The length of one full chain is equal to $\tau = st$.

Theorem 4.4 The general costs for the Fuzzy Rainbow Table attack are:

$M = 2ml$ entries,
$T_c = \frac{s(s+1)}{2} tlD$ f_i-computations,
$T_s = slD$ seeks in tables of m entries.

Proof. The costs for memory use and disk seeks remain the same as in the Distinguished Points case. The computation costs are still based on the costs for matching a single sample against a single table multiplied with the number of tables and the number of samples. The attacker needs to make s chains of sizes increasing from t to st, so in total $\frac{s(s+1)}{2} t$ f_i-computations, to match a single sample against a single table. □

In the Fuzzy Rainbow Table attack we are faced with an additional variable s, which introduces a new Time-Memory Trade-Off within a TMTO attack. It also complicates matters when creating the accompanying corollary by increasing the possible choices. Here we choose three of the most obvious scenario's:

- the full chain length of the fuzzy rainbow table tables is as large as in the previous attacks, which leads to s more tables (Corollary 4.4, more tables),
- the sub chain length of the fuzzy rainbow table tables is equal to the chain length of the previous attacks, the full chains are s times larger than the previous attacks (Corollary 4.5, bigger tables),
- one which falls in between these two scenarios, so where the full chain length is $\sqrt{s}$ larger than in the previous attacks, but the sub chain length is smaller than the chain length in the previous attacks (Corollary 4.6, in between).

Of course these only show three possible choices for s, t and m, of which the first scenario coincides with $m(st)^2 = N$ and the second with the familiar $mt^2 = N$.

Corollary 4.4 (more tables) When reversing a stream cipher using Fuzzy Rainbow Tables, where the $m \times st$ matrices satisfy $m(st)^2 = N$ and precomputing enough points to satisfy $D\rho = 1$, the costs are:

$T_{pre} = \mathcal{O}(N/D)$, $T_c = \frac{(s+1)}{2}(st)^2$ f_i-computations,
$M = \frac{2mst}{D}$ entries, $T_s = s^2 t$ seeks in a tables of m entries.

Proof. A single table will cover $m \times st$ points, or $1/st$ of the key space N (since $m(st)^2 = N$), so st tables are needed to achieve enough coverage for $D\rho = 1$. Given D expec-

ted samples during the online phase, an attacker can reduce the number of required tables to $l = \frac{st}{D}$. This means the memory costs will be $M = 2m \times \frac{st}{D} = \frac{2mst}{D}$ entries.

The attacker needs a total of $\frac{s(s+1)}{2}t$ f_i-computations, to match a single sample against a single table. Since there are D samples and $\frac{st}{D}$ tables, the total attack time T_c equals: $\frac{(s+1)}{2}(st)^2$. The attacker must create s separate chains for each table. During the online attack this comes down to s disk seeks per table, on $\frac{st}{D}$ tables and D samples gives $T_s = s^2t$ disk seeks within tables of m value pairs. □

Corollary 4.5 (bigger tables) When reversing a stream cipher using Fuzzy Rainbow Tables, where the $m \times st$ matrices satisfy $mt^2 = N$ and precomputing enough points to satisfy $D\rho = 1$, the costs are:

$$\begin{array}{lcllcl} T_{pre} & = & \mathcal{O}(N/D), & T_c & = & \frac{(s+1)}{2}t^2 \ f_i\text{-computations}, \\ M & = & \frac{2mt}{sD} \text{ entries}, & T_s & = & t \text{ seeks in a tables of } m \text{ entries}. \end{array}$$

Proof. A single table will cover $m \times st$ points, or s/t of the key space N (assuming $s < t$), so t/s tables are needed to achieve enough coverage for $D\rho = 1$. Given D expected samples during the online phase, an attacker can reduce the number of required tables to $l = \frac{t}{sD}$. This means the memory costs will be $M = 2m \times \frac{t}{sD} = \frac{2mt}{sD}$ entries.

The attacker needs a total of $\frac{s(s+1)}{2}t$ f_i-computations, to match a single sample against a single table. Since there are D samples and $\frac{t}{sD}$ tables, the total attack time T_c equals: $\frac{(s+1)}{2}t^2$. The attacker must create s separate chains for each table. During the online attack this comes down to s disk seeks per table, on $\frac{t}{sD}$ tables and D samples gives $T_s = t$ disk seeks. □

In the original publication on which this chapter is based we only presented the two scenario's above. After this publication Professor Jin Hong from Seoul National University suggested we add the following scenario where the matrices satisfy $mt^2s = N$ and which falls in between the two scenario's discussed above.

Corollary 4.6 (in between) When reversing a stream cipher using Fuzzy Rainbow Tables, where the $m \times st$ matrices satisfy $mt^2s = N$ and precomputing enough points to satisfy $D\rho = 1$, the costs are:

$$\begin{array}{lcllcl} T_{pre} & = & \mathcal{O}(N/D), & T_c & = & \frac{s(s+1)}{2}t^2 \ f_i\text{-computations}, \\ M & = & \frac{2mt}{D} \text{ entries}, & T_s & = & st \text{ seeks in a tables of } m \text{ entries}. \end{array}$$

Proof. A single table will cover $m \times st$ points, or $1/t$ of the key space N, so t tables are needed to achieve enough coverage for $D\rho = 1$. Given D expected samples during the online phase, an attacker can reduce the number of required tables to $l = \frac{t}{D}$. This means the memory costs will be $M = 2m \times \frac{t}{D} = \frac{2mt}{D}$ entries.

The attacker needs a total of $\frac{s(s+1)}{2}t$ f_i-computations, to match a single sample against a single table. Since there are D samples and $\frac{t}{D}$ tables, the total attack time T_c equals: $\frac{(s+1)}{2}t^2$. The attacker must create s separate chains for each table. During the online attack this comes down to s disk seeks per table, on $\frac{t}{D}$ tables and D samples gives $T_s = st$ disk seeks. □

The scenario of Corollary 4.4 (more tables) for Fuzzy Rainbow Tables uses the same sized matrices as that of Corollary 4.2 for the Distinguished Points. It needs

Table 4.1: *Comparison of the different attacks, for* $D\rho = 1$

TMTO technique	M	T_c	T_s
Hellman's attack	$2mt/D$	t^2	t^2 in m entries
Distinguished Points	$2mt/D$	t^2	t in m entries
Rainbow Table	$2mt/D$	$\frac{t(t+1)}{2}D$	tD in mt/D entries
Fuzzy RT (more tables, $t' = st$)	$2mt'/D$	$\frac{(s+1)}{2}t'^2$	st' in m entries
Fuzzy RT (bigger tables)	$\frac{2mt}{sD}$	$\frac{(s+1)}{2}t^2$	t in m entries
Fuzzy RT (in between, $t'' = t\sqrt{s}$)	$\frac{2mt''}{\sqrt{s}D}$	$\frac{(s+1)}{2}t''^2$	$t''\sqrt{s}$ in m entries

s^2 more tables than the scenario of Corollary 4.5 (bigger tables), which is reflected in all the costs. However, keep in mind that the value of t in Corollary 4.4 (more tables) is s times higher than the value of t in Corollary 4.5 and $\sqrt{s}$ higher than the value of t in Corollary 4.6 (in between). We compensate for these differences during the comparison in Section 4.4.

The costs in Corollary 4.5 (bigger tables) confirm our intuition of Fuzzy Rainbow Tables using s compressed distinguished points tables. So it can reduce memory costs a factor s at the price of increasing the computation cost in the online phase by a factor $\frac{s+1}{2}$ in comparison with Distinguished Points approach, which is better than our initial intuition.

4.4 Comparison

The main idea behind Fuzzy Rainbow Table is to combine the benefits of both Distinguished Points (i.e. low number of disk seeks) and Rainbow Tables (i.e. fewer duplicates). The question is whether this really turns out beneficial. The cost of the different attacks are compared in Table 4.1, which lists the costs given in Corollaries 4.1 to 4.6. The three possible Fuzzy Rainbow Table approaches are shown, but Corollary 4.4 has st substituted for t' and Corollary 4.6 has $t\sqrt{s}$ substituted for t', so their t' and t'' respectively are comparable to the value of t for the other attacks.

The three classic attacks

Hellman's attack (adapted for stream ciphers) is added in this table as a baseline, since the other attacks all improve on almost all costs. Between Distinguished Points and Rainbow Table it seems that a Rainbow Table is the best choice for $D = 1$, with only the disk seeks being more expensive due to the larger table. This makes Rainbow Tables the best choice for attacking block ciphers.

However, when attacking stream ciphers with multiple samples, $D > 1$, the comparison is not so simple. The online attack time and the number of disk seeks for Rainbow Table, T_c and T_s, both increase beyond those of the Distinguished Points attack. Of course increasing D also decreases the size of the rainbow table used, making each table search cheaper, but generally seek time will be in the order of the logarithm of the table size, so this benefit is smaller than the increase in the number of disk

seeks. Based on these calculations, the more samples are expected during the online attack, the more attractive the Distinguished Points approach becomes, compared to Rainbow Tables.

The Fuzzy Rainbow Table attacks

Our initial intuition that the Fuzzy Rainbow Table approach is comparable to s Distinguished Points tables stored as one, seems validated when looking at the respective costs of both Fuzzy Rainbow Table attacks and the Distinguished Points attack. If we take $s = 1$, then the Fuzzy Rainbow Table approaches are the same as Distinguished Points, and their costs are identical. Another way to look at the Fuzzy Rainbow Table approach is as a "bloated" rainbow table, with every rainbow colour expanded from one column to t columns (on average). Looking at the costs for the online attack time for the Fuzzy Rainbow Table approach, if we choose $t = 1$ and $s = \tau$ (essentially a rainbow table), it almost compares to the online attack costs of a Rainbow Table attack, where the difference can be explained by a Rainbow Table attack having a single table instead of the $\frac{st}{D}$, $\frac{t}{sD}$ or t tables of the respective Fuzzy Rainbow Table attacks.

It is clear from this comparison that having fuzzy rainbow tables of the more-tables variant is not the best choice. It has more disk seeks than the bigger-tables approach and the Distinguished Points attack, without the benefit of smaller memory costs. Having fuzzy rainbow tables of the "in between" variant are already better with a factor $\sqrt{s}$ less storage and less disk seeks than the more tables variant. The fuzzy rainbow tables of the bigger tables, are in this comparison the clear winner, with another factor $\sqrt{s}$ less storage and less disk seeks opposed to the in between variant. The online costs remain constant over all three variants, since in the end the attacker either computes shorter chains over more tables, or longer chains over fewer tables.

Fuzzy Rainbow Tables vs the classics

The Fuzzy Rainbow Table attack can be tuned further by changing the s parameter. Basically, the Fuzzy Rainbow Table attack moves in between the Distinguished Points and Rainbow Table attacks, guided by the value of s, where a higher choice of s will save memory costs, but increase the online attack costs.

From the three Fuzzy Rainbow Table approaches shown in the comparison table, the bigger-tables approach is the most competitive in this analysis. If the memory costs are the single limiting factor for using the Distinguished Points or Rainbow Table attack, then this Fuzzy Rainbow Table attack seems a good choice.

However, with the continuous drop in the prices of memory such a scenario seems unlikely, so depending on the number of expected samples a Rainbow Table or a Distinguished Points attack is probably the better choice.

Comparing chain merges

The comparison above is based on all attacks satisfying $D\rho = 1$, in other words the costs are compared when all attacks precompute the same amount of points. However, due to duplicates and chain merges not all precomputed points will be unique. It would be more fair if we compared the costs of the attacks when they all satisfy $D\bar{\rho} = 1$, so the number of precomputed unique values would be in the order of N/D.

However, it is hard to estimate the chances on chain merges in a general case for the different approaches. In essence this problem boils down to the expected overlap between two paths (of length t for most approaches) within a digraph consisting of the N points of the search space as nodes and the current f_i as the edges between these nodes. This digraph is a directed pseudo forest, so every node has out-degree 1, meaning there can exist source nodes, but no sinks and from every node there exists a path leading to a cycle. An analysis of the number of expected duplicates, or analogously the number of unique values for a certain TMTO attack seems hard [98, 78, 74, 11] and to our knowledge this is in fact an open problem for most TMTO attacks.

We can however make some assumptions over the chain merges for the different approaches, when they all precompute the same amount of points. We ignore single duplicate points and only look at chain merges, so only duplicates under the same f_i. Then by looking at the number of precomputed points per different f_i function, although ignoring many of the subtle differences between the TMTO attacks, can give an indication on the chances of chain merges.

When we assume that the distinguished points from a Distinguished Points attack are uniformly spread over the iterative function graph, then in general the number of duplicates in the Distinguished Points tables will be about the same as those in the Hellman attack. The number of duplicates in the Rainbow Table attack will in general be smaller than for the Hellman and the Distinguished Points attack, as long as the number of records in a rainbow table is smaller than the $m \times t$ points in a Hellman or distinguished point table.

When we look at the three presented approaches for the Fuzzy Rainbow Table attack, then they are most easily compared to a Distinguished Points attack. The first Fuzzy Rainbow Table approach ($m(st)^2 = N$, Corollary 4.4) uses s different colours inside an almost standard Distinguished Points matrix. So, only $1/s$ th of the points in a single table have the chance of leading to a chain merge, and we would therefore expect s less chain merges in this Fuzzy Rainbow Table approach than in a Distinguished Points approach. The second Fuzzy Rainbow Table approach ($mt^2 = N$, Corollary 4.5) compresses s different Distinguished Points tables into one, but because the end point of one of those Distinguished Points tables is the start point for the next, chain merges in one of these subchains will carry through the rest of the chain. Therefore, we can roughly estimate that this Fuzzy Rainbow Table attack has around $s/2$ more duplicates due to chain merges than a Distinguished Points attack. The final Fuzzy Rainbow Table approach ($mt^2 s = N$, Corollary 4.6) has a full chain length which is $\sqrt{s}$ larger than standard Distinguished Points tables. This means the sub chains are a factor $\sqrt{s}$ smaller than the Distinguished Points chain length, leading to an expected $\sqrt{s}$ less chain merges than the Distinguished Points approach.

When we keep chain merges in mind, the comparison from Table 4.1 becomes more subtle, since it seems that the factor s extra costs in memory and disk seeks of the first Fuzzy Rainbow Table approach compared to the second, is somewhat mitigated by having less duplicates, and thus more unique points in its tables and a higher success chance. The final Fuzzy Rainbow Table approach might offer a nice balance in this case. However, since we have no hard way of quantifying the number of chain merges in these attacks, this analysis remains very tentative.

Both Distinguished Points and Fuzzy Rainbow Table have one extra benefit when comparing chain merges. Both approaches have identifiable chain merges, which

means that every chain merge will automatically end in the same end point. So by simply comparing end points all chain merges can be identified. With extra precomputation effort both approaches are able to replace one chain of every merging chain pair, with a new one, thereby increasing their coverage $\bar{C}$. Naturally, both the increase in coverage and the amount of extra precomputation work are dependent on the number of chain merges.

4.5 Two possible improvements for Fuzzy Rainbow Tables

Given the analysis above it seems that if the Fuzzy Rainbow Table approach is the best choice for a TMTO attack, than it is only marginally so. The analysis also argues that the Fuzzy Rainbow Table suffers from more chain merges than the Distinguished Points or Rainbow Table approach. However, it seems possible to improve the Fuzzy Rainbow Table approach. The Fuzzy Rainbow Table approach stems from the idea to combine two benefits: the reduced disk seek time that comes from Distinguished Points and the reduced chances of chain merges by applying rainbow tables. However, it appears the reduced chances of chain merges of rainbow tables where increased again by combining Rainbow Table with the Distinguished Points approach.

We therefore look into two possible improvements to the Fuzzy Rainbow Table approach, by combining the Rainbow Table and Distinguished Points approaches in a slightly different manner. The two newly suggested approaches contain similarities to the Rainbow Table variants proposed by Barkan et al. [11] and are therefore analogously named *thick Fuzzy Rainbow Table* and *thin Fuzzy Rainbow Table*.

4.5.1 Thick Fuzzy Rainbow Table

The benefit of using the Distinguished Points approach only helps while applying the final colour, since only those values are matched with the pre-computation table. So we propose an improved Fuzzy Rainbow Table approach where distinguished points are only used while applying the final colour of a pre-computation matrix and simple Hellman chains are used while applying all other colours, as shown in Figure 4.5.

$$\begin{array}{ccccccccc}
x_0 & \rightarrow^t & f_0^t(x_0) & \rightarrow^t & f_1^t(f_0^t(x_0)) & \rightarrow^* \ldots & f_s^*(f_{s-1}^t(..(f_0^t(x_0))..)) & \rightarrow & k\|y_0 \\
x_1 & \rightarrow^t & f_0^t(x_1) & \rightarrow^t & f_1^t(f_0^t(x_1)) & \rightarrow^* \ldots & f_s^*(f_{s-1}^t(..(f_0^t(x_1))..)) & \rightarrow & k\|y_1 \\
\vdots & & \vdots & & \vdots & \ddots & \vdots & & \\
x_m & \rightarrow^t & f_0^t(x_m) & \rightarrow^t & f_1^t(f_0^t(x_m)) & \rightarrow^* \ldots & f_s^*(f_{s-1}^t(..(f_0^t(x_m))..)) & \rightarrow & k\|y_m
\end{array}$$

Figure 4.5: *An thick Fuzzy Rainbow Table matrix, where $k\|y$ denotes a distinguished point with the first k bits '0'. It features pre-computation chains of $s-1$ times t f_i-computations and a final round of f_i-computations until a distinguished point is reached. Only the first and last points of each chain are stored.*

The improvement this new approach offers is that the expected cost of chain merges is lowered w.r.t. the Fuzzy Rainbow Table approach, since chain merges that

occur in sub chains will not propagate through the entire chain unless the duplicates occur in the exact same position. Chain merges will therefore be more localised within a single f_i segment. This means that with the same amount of pre-computation costs this thick Fuzzy Rainbow Table approach will likely cover more unique points than the standard Fuzzy Rainbow Table approach.

It does however have some major drawbacks when compared to the standard Fuzzy Rainbow Table attack. The first is that partly merging chains no longer end in the same endpoint, and so it is no longer possible to remove all chain merges from the tables by removing any one of two chains that result in the same end point.

The second drawback of this approach lies in the added costs:

Theorem 4.5 The general costs for Thick Fuzzy Rainbow Table attack are:

$M = 2ml$ entries
$T_c = \frac{(s+1)(st+2)t}{2} lD$ f_i-computations
$T_s = ((s-1)t+1)lD$ seeks in a tables of m entries

Proof. The costs for memory use remain the same as with the Fuzzy Rainbow Table attack. The computation costs are still based on the costs for matching a single sample against a single table multiplied with the number of tables and the number of samples. For every sample the attacker needs to iterate the f_s function, until a distinguished point is reached. These, on average t, steps need to be performed for every possible subchain within the pre-computation matrix, as Figure 4.6 shows.

$$
\begin{array}{ccccccc}
 & & & & y & \rightarrow_{f_s}^{*} & k\|x_{0s} \\
 & & & y & \rightarrow_{f_{s-1}} & \rightarrow_{f_s}^{*} & k\|x_{1s} \\
 & & y & \rightarrow_{f_{s-1}} & \rightarrow_{f_{s-1}} & \rightarrow_{f_s}^{*} & k\|x_{1s} \\
 & & & \vdots & \vdots & \vdots & \\
 & & y & \rightarrow_{f_{s-2}} & \rightarrow_{f_{s-1}}^{t} & \rightarrow_{f_s}^{*} & k\|x_{2s} \\
 & & & \vdots & \vdots & \vdots & \\
y & \rightarrow_{f_1}^{t} \quad \cdots & & \rightarrow_{f_{s-2}}^{t} & \rightarrow_{f_{s-1}}^{t} & \rightarrow_{f_s}^{*} & k\|x_{ss}
\end{array}
$$

Figure 4.6: *The online attack phase of a thick Fuzzy Rainbow Table attack, where a single sample y is compared with a single matrix. Here $k\|X$ denotes a distinguished point with the first k bits '0'. Only the last point of every chain is matched against the pre-computation table.*

So, on average, the attacker needs to make $(s-1)t+1$ chains, and accordingly, also $(s-1)t+1$ seeks per sample per table. The costs for the computation time then come down to $\sum_{j=0}^{s} t(1+jt) = \frac{(s+1)(st+2)}{2} t$ f_i-computations, to match a single sample against a single table. □

Corollary 4.7 When reversing a stream cipher using thick Fuzzy Rainbow Tables, where the $m \times st$ matrices satisfy $mt^2 = N$ and pre-computing enough points to satisfy $D\rho = 1$, the costs are:

$T_{pre} = \mathcal{O}(N/D)$ $\quad$ $T_c = \frac{(s+1)(st+2)t^2}{2s}$ f_i-computations
$M = \frac{2mt}{sD}$ entries $\quad$ $T_s = \frac{(s-1)t^2+t}{s}$ seeks in a tables of m entries

Proof. As in the Fuzzy Rainbow Table approach, this attack requires $l = \frac{t}{sD}$ tables, for comparison's sake. The costs are determined by substituting l accordingly in the previous theorem. □

So the online attack is a factor $t + 2/s$ worse than the Fuzzy Rainbow Table attack, with a factor $(st - t + 1)/s$ more disk seeks.

4.5.2 Thin Fuzzy Rainbow Table

The benefit of the thick Fuzzy Rainbow Table approach comes from preventing chain merges happening in the same f_i subchain (but not on the exact same position) from propagating through the rest of the chain. In the thin Fuzzy Rainbow Table approach we want to keep the identifiable chain merges that result from the Distinguished Points approach and standard Fuzzy Rainbow Table approach, by choosing s f_i functions for a pre-computation matrix, and applying those one after another repeatedly, until a distinguished point is reached:

$$x_0 \rightarrow f_0 f_1 f_2 \dots f_{s-1} f_0 f_1 f_2 \dots f_{s-1} f_0 \dots \rightarrow DP$$

If an attacker chooses the distinguished point such that on average it will take t times the s functions to reach one, the tables will cover around the same amount of points as the original Fuzzy Rainbow Table tables.

This approach keeps the identifiable chain merges, from the original Fuzzy Rainbow Table approach, but at some costs. Chain merges happening while applying the same f_i (while not in the same column) will again propagate through the rest of the chain, so the same amount of chain merges as the original Fuzzy Rainbow Table had are to be expected and the online attack time is slightly worse.

Theorem 4.6 The general costs for Thin Fuzzy Rainbow Table attack are:

$M = 2ml$ entries
$T_c = s^2 tlD$ f_i-computations
$T_s = slD$ seeks in a tables of m entries

Proof. The costs for memory use remain the same as with the Fuzzy Rainbow Table attack. The computation costs are still based on the costs for matching a single sample against a single table multiplied with the number of tables and the number of samples. For every sample the attacker needs to make s different attack chains of average size st, each starting with one of the s different f_i functions:

$$\begin{aligned} y &\rightarrow f_0 f_1 f_2 \dots f_{s-1} f_0 f_1 f_2 \dots f_{s-1} f_0 \dots \rightarrow DP \\ y &\rightarrow f_1 f_2 \dots f_{s-1} f_0 f_1 f_2 \dots f_{s-1} f_0 \dots \rightarrow DP \\ &\vdots \\ y &\rightarrow f_{s-1} f_0 f_1 f_2 \dots f_{s-1} f_0 f_1 f_2 \dots f_{s-1} f_0 \dots \rightarrow DP \end{aligned}$$

So the online attack time becomes $s^2 t$ to match a single sample against a single table, with s disk seeks. □

Corollary 4.8 When reversing a stream cipher using thin Fuzzy Rainbow Tables, where the $m \times st$ matrices satisfy $mt^2 = N$ and pre-computing enough points to satisfy $D\rho = 1$, the costs are:

T_{pre}	=	$\mathcal{O}(N/D)$	T_c	=	st^2 f_i-computations
M	=	$\frac{2mt}{sD}$ entries	T_s	=	s seeks in a tables of m entries

Proof. The total attack time for t/sD tables and D samples becomes: $T_c = st^2$ f_i-computations and the seek time becomes $T_s = s \times \frac{t}{sD} \times D = t$ disk seeks. □

Intuitively we felt improvements on the Fuzzy Rainbow Table approach where possible. Regrettably, both of our suggestions offer no improvement on the original Fuzzy Rainbow Table attack. Our attempt to reduce the number of chain merges within fuzzy rainbow tables (thick Fuzzy Rainbow Tables) caused a big increase in online computation time, while our attempt to keep the identifiable chain merges (thin Fuzzy Rainbow Tables) caused only a slight increase in online computation time, but offered no improvements in return.

4.6 A Fuzzy Rainbow Table Case study: Kraken

The comparison of Section 4.4 remains unfair, due to our inability to quantify the number of chain merges. Therefore, this section will look at the only practical situation where the Fuzzy Rainbow Table approach was used. Naturally, this only provides an insight in the number of chain merges for the Fuzzy Rainbow Table approach with specific parameters on a single practical case. However, this can at least provide us with an insight into the significance of the chain merger problem in practice.

4.6.1 A5/1

The most commonly used cipher within GSM (Global System for Mobile communication) is A5/1. It is a stream cipher that encrypts the communication between a mobile phone and the local cell tower.

The design of A5/1 contains three Linear Feedback Shift Registers (LFSRs) which contain 64 bits in total, as Figure 4.7 shows. The internal states of these registers are instantiated with two variables; a 64-bit session key and the 22-bit current frame number. The session key is a temporary private key that both SIM card and mobile phone network can compute from a shared secret. The frame number is a publicly known value that functions as a counter.

These registers are clocked irregularly through a majority clocking function over the three clock bits (bit 8 of R1 and bit 10 of R2 and R3). So at each step two or all three of the registers will clock, and each register will clock with a chance of $\frac{3}{4}$.

Communication on the GSM-air interface is sent in bursts of 114 bits. For each burst, A5/1 generates 328 bits of keystream. The first 100 of these are discarded. Of the remaining 228 bits, the first half are used for the encryption of a burst on the uplink (mobile phone to cell tower) and the second half is used for encryption on the downlink (cell tower to mobile phone).

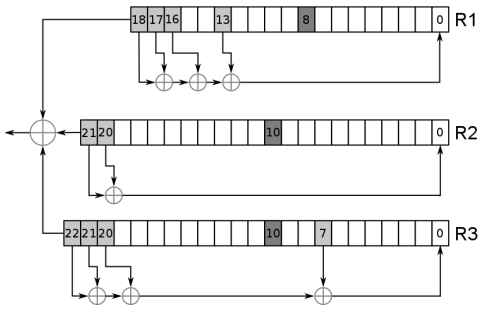


Figure 4.7: *Diagram of the internal design of the A5/1 stream cipher.*

In a standard TMTO attack on A5/1, an attacker will try to retrieve the internal state of the cipher using keystream samples from captured bursts containing known plain text. In an ideal scenario, an attacker can collect 228 bits of consecutive keystream, generated from a single internal state, though it is unlikely that an attacker could predict the required amount of known plain text on both the up and down links. However, since all the communication in one session uses the same session key,[1] the attacker can collect several keystream samples from different bursts, to increase chances of a successful attack. From the internal state the attacker can deduce the possible session key that was used for this encryption, which on average is one per internal state, and thus decrypt an entire session.

The natural assumption here is that the state space has size 2^{64}, but careful examination of the clocking function shows that a large part of the possible internal states are unreachable from any valid state. Several studies have measured the decline of possible states in the A5/1 cipher[86, 130], and all of these find that only around 15% of all possible states are still viable after the initial 100 clockings. This means in practice that an attacker only needs to cover around 15% of the state space: $N \approx 2^{61.26}$.

Note that the initial key generation algorithm provided in GSM generated keys with ten bits set to '0' as was discussed in Section 3.2.2. Together with the state space collapse, this would lead to a state space of only: $N \approx 2^{51.26}$.

4.6.2 The Berlin Set

The Fuzzy Rainbow Table approach, discussed in Section 4.3.4, is used by researchers to perform an attack on the stream cipher A5/1. In this setup the $f_i(x)$ is setting x in the internal state of A5/1, clocking it a 100 steps forward and then producing 64 bits of keystream, combined with some trivial output modification (the rainbow colours). These 64 output bits are then used to set as the new internal state for the next round. Note that these f_i functions step through the same state space as A5/1, but not in the same sequence.

In the pre-computation phase 40 independent tables ($l \approx 2^{5.3}$) where created, dubbed the Berlin set. As distinguished points where chosen those points starting

[1] In practice the same session key is often re-used over multiple sessions.

with 12 zeros ($k = 12$). Eight rainbow colours where used per table ($s = 8$) and they differ for each table, so in total there are 320 different colours.

Every chain consists of eight subchains of average length t. Assuming each possible outcome of an f_i is equally likely: $t = 2^{12}$. So $\tau = 8 \times 2^{12} = 2^{15}$.

Initially, every table was computed with 8662000000, approximately 2^{33}, rows ($m = 2^{33}$). Afterwards, one of every two chains with duplicate end points was removed and the current set contains around 6000000000 entries per table.

This means the researchers pre-computed around $2^{15} \times 2^{33} \times 2^{5.3} = 2^{53.3}$ points of N. Their current set ended up covering around $2^{15} \times 2^{32.5} \times 2^{5.3} = 2^{52.8}$ distinct points, with a set of tables that take up around 1.6 TB. So over 29% of the chains produced in this Fuzzy Rainbow Table attack on A5/1 ended up merging with a previous chain.

With $N \approx 2^{61.26}$, the Berlin set has its parameters around the $mt^2 s \approx N$ option.

4.6.3 Comparison

Table 4.2: *Comparison of the different TMTO attacks*

TMTO technique	M	T_c	T_s
Fuzzy RT ($s = 8, t = 2^{12}, m = 2^{33}, l = 40$)	$2^{39.3}$	$2^{22.5}D$	$2^{8.3}D$
Hellman's attack ($t = 2^{12}, m = 2^{33}, l = 320$)	$2^{42.3}$	$2^{20.3}D$	$2^{20.3}D$
Distinguished Points ($t = 2^{12}, m = 2^{33}, l = 320$)	$2^{42.3}$	$2^{20.3}D$	$2^{8.3}D$
Rainbow Table ($t = 2^{12}, m = 2^{41.3}$)	$2^{42.3}$	$2^{23}D$	$2^{12}D$
Thick Fuzzy RT ($s = 8, t = 2^{12}, m = 2^{33}, l = 40$)	$2^{39.3}$	$2^{34.5}D$	$2^{8.3}D$
Thin Fuzzy RT ($s = 8, t = 2^{12}, m = 2^{33}, l = 40$)	$2^{39.3}$	$2^{23.3}D$	$2^{8.3}D$

Table 4.2 shows a more practical comparison of the different TMTO approaches, by simply substituting the chain length, table size and number of tables with concrete values. The Fuzzy Rainbow Table data show the costs for the exact parameters that were used in the Berlin set. The other approaches have suggested parameters, so that they achieve the same coverage, ignoring duplicates, as the Berlin set. Note that the presented values do not satisfy the $mt^2 = N$ equation and are likely not the most efficient parameters for the techniques.

This comparison clearly shows that the Fuzzy Rainbow Table attack has a benefit in the lowest memory costs of all presented attacks, while having a higher online attack time than the Distinguished Points and Hellman attacks.

The Distinguished Points seems the most favourable attack in this comparison, with only its memory costs higher than the Fuzzy Rainbow Table attack. The comparison is not favourable for the Rainbow Table attack, which is mostly due to its greatest strength, a lower chance of chain merges, not being reflected in these numbers.

Based on these possible parameters for the different attacks each attack precomputes the same number of points, but again it remains unclear what the number of unique points is. The Fuzzy Rainbow Table attack had approximately 2 billion chain merges per table, or 29% of the total number of chains. By the informal analysis from Section 4.4 it can be surmised that the Fuzzy Rainbow Table attack probably has a lot more duplicates due to these chain merges than the Distinguished Points and

Hellman attack, with these parameters. The Rainbow Table attack probably has the least duplicates of all.

4.7 Related work

Some of the discussed TMTO methods have previously been compared with each other. Most of these publications compare the trade-off curves for these attacks [16, 45], which give the rate at which extra memory can be traded in for a reduced attack time. Such as $M^2T = N^2$ for both the Hellman and Distinguished Point attack, with M the memory cost, T the time cost of the online phase, and N the size of the search space. Our comparisons are not based on trade-off curves, because we feel that these curves hide too much of the real costs such attacks have, such as the seek times in the online attack, or the precomputation effort. Biryukov and Shamir compare Hellman's attack with Distinguished Points [16] and Erguler et al. compare Hellman's attack with Rainbow Tables in [45]. Barkan et al. [11] make a comparison within a new theoretic framework and find the Distinguished Points attack better than the Rainbow Table attack, mainly based on the possibility to shorten the stored values of a Distinguished Points attack. In a more elaborate study Hong et al. find the Rainbow Table attack to outperform the Distinguished Points attack by a small margin for single sample attacks [100]. These comparisons seem to contradict each other, though Hong provides an argumentation why the earlier result by Barkan et al. is faulty. Still, the question on which TMTO attack has the lowest costs, in terms of time and memory seems to still be open to debate.

In 2008 Hong et al. [99] already combined Distinguished Points with Rainbow Tables, but in a different way than with Fuzzy Rainbow Tables. Their combination does not improve on just Distinguished Points or Rainbow Table attacks.

We were unaware of any earlier analysis of the Fuzzy Rainbow Table attack at the time of our original publication that this chapter is based on. The attack had been considered in work by Krhovjak et al. [112], where they use it as a practical example for an attack against A5/1, but they provide no analysis and make no comparison with the earlier attacks. Shortly after our publication Hong et al. published an in-depth analysis of the Fuzzy Rainbow Table attack [110] based on an earlier eprint publication [109], where they compare the non-perfect Fuzzy Rainbow Table attack with both the perfect and non-perfect Rainbow Table attack. They find that a Fuzzy Rainbow Table is better than a Rainbow Table on almost all fronts for low success rates. For higher success rates the difference is smaller, but remains in favour of Fuzzy Rainbow Tables, though the Rainbow Table manages to achieve a higher maximum success rate than is possible with Fuzzy Rainbow Tables.

Though we share the conclusion of [110] that Fuzzy Rainbow Table are better than Rainbow Tables, we find the Distinguished Points attack to be better than both. This difference of findings can be partly explained by the somewhat arbitrary parameter choices for the comparison, such as the required chance of success, and our difference in focus; we focused on worst-case multi-sample attacks, while Hong et al. focused on average case single-sample attacks. Still, this would not seem to explain all differences and certainly the comparison by Hong et al. is more extensive than ours, incorporating for instance false alarm costs and storage benefits of the differ-

ent approaches. However, many more factors remain unresearched, such as possible parallelism benefits for the different approaches, and the differences remain small enough that the particularities of specific attack situation could sway the choice in favour of any of the algorithms.

4.8 Conclusions

We have presented our analysis of the cost of the Fuzzy Rainbow Table TMTO attack. This attack is used to break GSM's A5/1 cipher. We have also given a comparison of the costs of Fuzzy Rainbow Table and three older TMTO attacks: Hellman's original attack, Distinguished Points, and Rainbow Tables.

Our comparison is more detailed than earlier work comparing these three older forms of attack. Most earlier work compared the trade-off curves of these well known attacks [16, 45]. This tells us the rate at which extra memory can be traded in for a reduced time, but completely ignores some important costs, namely the precomputation time, seek times, and the number of unique points covered by an attack. We do consider these costs in our comparison: for each attack we give the memory and time costs, split into precomputation time, online computation time, and number of disk seeks.

In our comparison in Section 4.4 the new Fuzzy Rainbow Table attack performed fine, with the lowest memory cost of all attacks and the ability to identify chain merges as its major benefits. Only Distinguished Points seems a better choice in comparison, having a higher memory cost, but the lowest online attack costs. The more well-known Rainbow Tables are only interesting for attacks were only a single sample of plaintext-ciphertext is available, as it is outperformed by Distinguished Points for multiple samples.

Another limitation of comparisons of trade-off curves for the different approaches is that these curves are invariably made under the assumption that the table sizes are always chosen so that $mt^2 = N$. We see no convincing reason to constrain the choice in parameters in this way. Hellman used the constraint $mt^2 = N$ to compute a nice bound for the chance of success of his attack, but other choices for m and t that do not satisfy this constraint might perform better in concrete instances.

Since the publication of our research in 2013 [176], a more detailed analysis of TMTO attacks has been published by Hong et al. [110], which shows the Fuzzy Rainbow Table attack to be the best choice for single sample attacks in the average case. Still, the difference between attacks remain marginal and there remain enough unresearched factors, such as possible parallelism benefits, to keep the question on the best TMTO attack open.

We still abstract away from certain practical costs in our analysis. We count 2 words for each chain entry, while in practice an attacker can store less than this. Also, we count the number of disk seeks as single costs steps, which again in practice can be quite different due to caches. All of these abstractions would be interesting to consider in future work.

One factor that we still have not been able to quantify precisely in our comparison is the chance of duplicates during the precomputation of the tables. We conjecture that the effectiveness of the Fuzzy Rainbow Table attack is in fact lower than our

current results suggest when this number of duplicates values is taken into account. The informal analysis of the expected number of chain merges in Section 4.4 shows that Fuzzy Rainbow Table has a higher chance of chain merges than the other attacks when the number of rainbow colours in the Fuzzy Rainbow Table approach is chosen to achieve lower memory cost.

To compensate for this we attempted two ways to improve Fuzzy Rainbow Tables, which we dubbed thick Fuzzy Rainbow Tables and thin Fuzzy Rainbow Tables. However, both our attempts did not improve on the Fuzzy Rainbow Table approach, in fact our attempts proved to only make matters worse.

Estimating the chance of duplicates during precomputation is the most difficult aspect in achieving a fair comparison. Over 29% of all chains created in the Fuzzy Rainbow Table tables used to break A5/1 ended up merging with existing chains, showing that chain merges can indeed be a significant factor when comparing TMTO attacks. We know no way to compute the expected number of chain merges for the general case, or indeed for any non-trivial practical cipher. Since theoretical analysis of the chance of duplicates seems very difficult, we think that further research which collects empirical data of practical experiments in constructing TMTO tables may be the best way to shed light on this.

Chapter 5

Femtocell security

The previous two chapters looked into attacks on the wireless part of mobile telephony networks. Because of the nature of wireless broadcasts, it is usually easier (or at least less risky) to perform an attack on the wireless connection than attempting to compromise network equipment, which is usually securely stored in hard to reach places, e.g. on top of high masts or buildings. The scenario of attacking network equipment has become much more realistic with the introduction of femtocells. Femtocells are low-powered cellular base stations for mobile telephone networks, meant for home use, but still managed by the provider. They are an increasingly popular solution, with the number of femtocells steadily rising to a market value of $1 billion USD in 2016 [155].

However, femtocells also introduce a number of security concerns. Several earlier femtocells have been hacked to varying degree and analysed. Naturally, the industry has responded and tries to create more secure femtocells.

Femtocells introduce a new attack vector into the mobile networks; one that is easier to exploit than traditional cell towers, but also provides slightly different privileges to attackers. They are the most likely source of an attack, by our non-network attacker (Section 1.1.2) and as such warrant a thorough security analysis.

This chapter looks at what the theoretical security implications are when a possibly compromised femtocell is introduced in a mobile telephony network. Furthermore, this chapter details a practical experiment to assess the security measures modern femtocells take against attacks.

This chapter is based on the paper *Femtocell Security in Theory and Practice*, presented at the 18Th Nordic Conference on Secure IT Systems, NordSec 2013 [178]. Besides some small changes to better embed the article within the rest of this thesis, the publication has been included as this chapter practically verbatim.

5.1 Introduction

In mobile telephony networks such as GSM, UMTS and EV-DO (an American counterpart to UMTS), service is provided through many antennae that each cover a geographic area. These areas are called cells and can range in size based on the transmission power of the signal and the available bandwidth. Within each cell the coverage is influenced differently by local propagation conditions which can result in blind spots where signal reception is so poor that no service is available. To solve this small cells can be created within these blind spots, with a low power antenna that operate on a different frequency from its containing cell.

Small cells can have different sizes, which are usually subdivided into microcell, nanocell and femtocell, from small to smallest. The normal, much larger, cell size is referred to as macrocell. The distinction between the types of small cells is not officially defined, but typically a microcell covers an area the size of a shopping mall or a transportation hub, a nanocell covers a small business or an office floor, and the femtocell a small house or several rooms [155].

Besides the coverage size there is a more important distinction between the femtocell and other small cells. The microcells and nanocells are installed and maintained by the provider and connect directly to the provider's core network, while the femtocell is a consumer-installed (and owned) device and connects to the core network of the provider through the consumer's broadband connection. Naturally this introduces several new security risks for both provider and consumer, since a low-cost device is now placed at the consumer's home, which has the ability to act as an authentic cell tower and to connect to the provider's back end over an untrusted channel.

A femtocell device is a small box with a power and Ethernet connector and at least one antenna. Some of the femtocells have GPS unbar, to verify their geographical location. All of them can listen to neighbouring cells, in order to run on a non-interfering frequency. Usually femtocells contain a dedicated chip that is specifically made for femtocell devices. These chips consist of a base band processor[1], some cryptographic processor and a general purpose processor. All of the femtocells analysed so far run some lightweight form of the Linux operating system.

The rest of this chapter is structured as follows. Section 5.2 gives an overview of femtocells within a cellular network. Section 5.3 gives an overview of the femtocell security model we assume and the most likely attack vectors. In Section 5.4 we discuss possible attacks offered by a compromised femtocell against the 3GPP security goals for UMTS and LTE. A practical security analysis is presented in Section 5.5 where we successfully compromise a modern femtocell (the Vodafone SignaalPlus Plug & Play). Finally, we discuss our conclusions and some ideas for future work.

Related work

3GPP, the standardisation body for the GSM, UMTS and LTE systems, has specified the use of femtocells within mobile telephony networks. Of these specifications 25-467 [64] and 33-320 [63] are the most interesting, and respectively detail the architecture of and the security architecture of the femtocell (called a Home NodeB or HNB).

[1] A dedicated processor for signal processing and real-time transmission operations.

Several books have been written on femtocells. "Femtocell Primer" [28] is a very superficial introduction into femtocells, and focuses more on the economic aspects of introducing femtocells. Two other books, "Femtocells: Technologies and Deployment" [192] and "Femtocells: Design and Application" [106] cover femtocells more extensively. They highlight all the technical difficulties in realising femtocells from an engineering standpoint. Both books contain a small section on security, with only a broad overview of the subject.

Some publications analyse possible security problems that arise when femtocells are introduced in the network [145, 94]. Both are theoretical analyses. Tyler et. al [152] show the economic incentives of possible attackers to use a compromised femtocell to DDoS a telecommunications network.

There have also been practical analyses of a physical femtocell device. Indeed several off-the-shelf femtocells have been hacked with varying consequences. In 2010, a research group that calls itself THC (The Hackers Choice) managed to gain root access to the Vodafone Sure Signal femtocell [163]. This proved a very severe security break, based on an easy to guess root password, which allowed interception of phone calls and allowed attackers to request the current session keys from any handset, from the Vodafone back end. A Samsung Femtocell was rooted by a group of researchers from Trustwave's SpiderLabs in 2011 [169, 72]. We could not find any publications showing the attack capabilities they gained with getting root access to this femtocell.

Researchers from the Technical University of Berlin [23] analysed the security of a femtocell by Ubiquisys. They managed to break its security and run arbitrary code on the femtocell, which also included the functionality to request session keys for connected phones. They conclude with a rather brief list of possible attacks against the femtocell and the core network with their compromised femtocell. A second publication by the same group [84] presents the method used to break this femtocell and shows that this break compromises all security requirements.

Theoretical and practical research are combined in a publication from researchers in Birmingham together with the group from TU Berlin [4]. In this work they formally verified the authentication in the UMTS and LTE systems using ProVerif, discovered an attack on location privacy, and proved the feasibility of this attack by reprogramming a femtocell.

5.2 Femtocell overview

The femtocell idea can be applied to many different cellular communication networks, such as UMTS, LTE and EV-DO. Since each of these has its own terminology for network entities, each network also has their own names for the femtocell and the extra network components required for femtocells. For instance, in UMTS the cell towers are called NodeB, so the femtocell is called HNB (Home NodeB). In LTE, on the other hand, femtocells are called HeNB (Home eNodeB). All these different acronyms can make the different specifications difficult to read. Figure 5.1 shows a femtocell inside a UMTS network. This representation of a UMTS network is a more detailed instantiation of the generic model presented in Section 2.2. Here a UE (User Equipment, the handset) contains a SIM card and connects to the RAN (Radio Access Network)

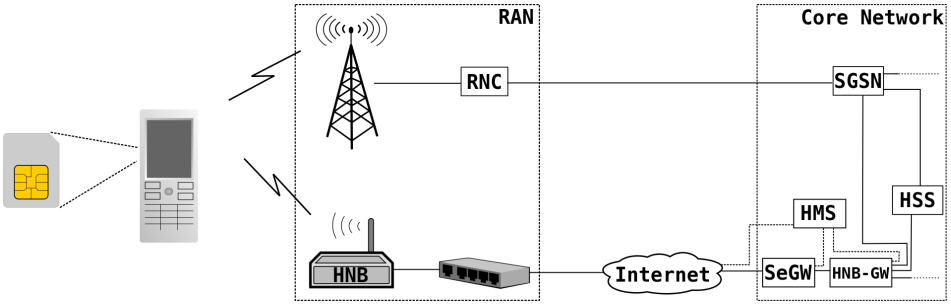


Figure 5.1: *Diagram of a UMTS network that incorporates a femtocell*

either via a cell tower, or through a femtocell called a HNB. For a user who connects to the femtocell, the experience should be indiscernible from connecting to regular cell towers. So a running session should be seamlessly handed-over between the HNB and the NodeBs, dependent on signal strength. From the RAN a connection is made to the Core Network of a provider. The SeGW (Security Gateway) is the entity in the provider's core network where the encrypted connection from the HNB over the untrusted internet connection terminates. The HNB-GW (HNB Gateway) then routes the decrypted traffic inside the provider's core network. The HNB-GW can be combined with the SeGW in a single entity. Communication can be routed to the HSS (Home Subscriber Server), which primarily handles the authentication of SIM[2] cards. There is also a HNB Management Server (HMS), which manages practicalities such as firmware updates and the operational frequencies. The SGSN is also shown as a part of the core network in Figure 5.1, but this is merely there for completeness sake, as the cell tower's equivalent of the femtocell's gateway and not important for any of the further discussion.

Again different terminology is used by the Small Cell Forum, a consortium of industry players that advocate the use of femtocells — they actually started out under the name Femtocell Forum. This terminology is also used in most books on femtocells and is meant as a communication network independent generalisation of the femtocell idea. They mostly describe the same network entities under a different name. Here a FAP (Femto Access Point) is used to designate the actual femtocell radio unit. The Femtocell gateway is often combined with the security gateway and called the FGW. The HMS is now called a FMS (Femto Management Server) and the HSS is replaced by a more generic AAA (Authentication, Authorisation and Accounting) server.

In the rest of this chapter we will attempt to avoid any specific terminology, instead we refer simply to femtocell and handset. However in cases where a specific term is needed we will use the 3GPP terminology.

[2] In GSM terminology, SIM card can mean either the physical smart card or the application which runs on it. For UMTS the physical smart card is called a UICC and the application is called USIM. For simplicity we only speak of SIM cards here.

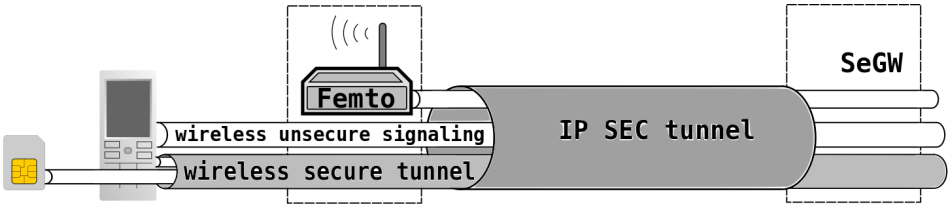


Figure 5.2: *Tunnelled communication between handset and core network over a femtocell*

5.3 The security model

In earlier cellular systems, such as GSM, the handsets did not authenticate cell towers. This allowed attackers to impersonate these cell towers. Modern cellular networks protect against this attack through mutual authentication between handset and network. The network creates a fresh authentication token based on a sequence number and a shared symmetric key (stored in the provider's core network and inside the subscriber's SIM) for each authentication. Handsets will only connect to cell towers that transmit the correct authentication token. So the provider's core network is authenticated and the handset can assume the cell tower is genuine since it was able to obtain this token. The handset then also responds to a challenge sent by the network, so it to can be authenticated by the network.

A femtocell has a wireless connection to a handset and even before the initial authentication some communication is needed. A femtocell is a device that is supposed to function as a small cell tower and is therefore able to relay the authentication tokens from the core network. Both the connection into the provider's core network and the connection a femtocell makes to handsets can be interesting attack vectors that might threaten the cellular network's security model. Therefore there are several security features advised for femtocells in several specifications [64, 63, 55].

This section gives an overview on the security model of a femtocell.

5.3.1 The femtocell security model

Figures 5.1 and 5.2 give an overview of the femtocell inside a cellular network. The communication between the femtocell and the core network of the provider needs to be Tunnelled through an authenticated and encrypted connection. The specifications advice the use of an IPSEC connection between the femtocell and the SeGW, for instance by using IKEv2, which provides authentication based on PKI certificates and integrity and confidentiality on an IP level.

IKEv2 has been formally analysed and shown to be a secure protocol [57] provided both entities keep their secret key hidden. Naturally the keys inside the femtocell need to be stored securely, for example by placing them on a smart card or inside a TPM (Trusted Platform Module).

Figure 5.1 shows that the femtocell can contact the management server directly or through the secure tunnel. Both scenarios are presented in the specifications, although the preferred approach is to use the SeGW. For this chapter we assume network designs where the management server is placed behind the SeGW and all communication between the management server and the femtocell is thus protected by

the IPSEC tunnel. This design seems to make more sense and is also the only behaviour we have encountered in the modern femtocells we have investigated.

All the communication between a femtocell and the core network are routed through the IPSEC tunnel. This communication consists of signalling information and user data. Most of these are again inside their own secure tunnel between handset and the core network.

A femtocell has to pass on the authentication messages from both the handset and the core network unaltered for the handset to connect to the femtocell. In this process the session keys between handset and core network are established, which are used to create the secure tunnel between handset and core network (the inner tunnel in Figure 5.2). These session keys can not be computed or retrieved by the femtocell.

It is possible to design a femtocell in such a way that it also stores the session keys for the secure wireless tunnel. This would seem like a needlessly insecure option, but it does provide some usability benefits, since internet traffic can then immediately be routed onto the internet from the user's router, instead of via the provider's back end. This feature is referred to as *local break-out*. We will use the term "local break-out" to refer to a femtocell that was designed with the possibility to store or request the user session keys for the secure wireless tunnel, regardless of the implementation of the local break-out feature. Our practical experiments were on a femtocell model that did not use local break-out, so we assume a model in which femtocells do not support this feature.

Femtocells can be run in two different modes: open and closed. These modes refer to the femtocell's behaviour with respect to handsets that do not belong to the consumer. In open mode, the femtocell allows any handset (usually only from subscribers to the same provider) to camp on it. In most cases the subscriber who bought the femtocell can manage the femtocells operating mode and its CSG (Closed Subscribers Group). The 3GPP specifications allow for two types of femtocells, a CSG femtocell and a non-CSG femtocell [64]. The difference between these femtocells is whether the femtocell or the core network checks if a handset is a member of the CSG. A CSG femtocell maintains the Access Control List of identities (IMSIs) allowed within the CSG, while a non-CSG femtocell is oblivious to the existence of a CSG, all the CSG management is then handled in the core network of the provider.

5.3.2 Attack vectors

Assuming an attacker has no access to the core network of the provider, the addition of a femtocell into a telco network introduces three new entry points for an attack: the wireless interface, a direct attack on the femtocell device, and an attack on the internet back haul connection.

The first, the wireless interface, is the same as the standard wireless cellular interface, and so femtocells introduce no new threats compared to the normal wireless interface of the telco network.

The last, the untrusted internet back haul, delivers a serious threat to the overall security of a telco network. This is mitigated by using a secure IPSEC tunnel, which provides authenticity, integrity and confidentiality.

This makes a direct attack on the femtocell the most viable entry point for an

attack, especially since the femtocell also stores the secrets that are needed to set up the IPSEC tunnel.

5.4 Theoretical security analysis of femtocells without local break-out

This section looks at possible attacks with a compromised femtocell against the security model of UMTS and LTE. So the weaknesses of GSM and fallback attacks to GSM are not considered. With a compromised femtocell we mean a femtocell on which a hacker can execute arbitrary code. This scenario seems likely, since a femtocell is a reasonably low-cost device that is placed in the care of consumers for an extended period of time and which includes a lot of software on a standard execution platform, running Linux. The hacker might not be able to learn the securely stored secrets, e.g. those on a smart card or in a TPM, but he can access the functionality these provide, like signing or encrypting.

We assume a femtocell design that does not receive any user session keys from the provider's core network. So we assume a femtocell without local break-out, which means the attacker is able to listen to, and influence, messages inside the IPSEC tunnel, but not to messages inside the secure wireless tunnel, nor is he able to influence any decisions made in the provider's core network. This is the main difference with most other analyses [145, 84, 23, 163]. We believe it is more realistic to assume a femtocell without local break-out, since the femtocell we investigated does not support it and it seems the most sensible design choice, security wise. Though certainly femtocells with local break-out exist [163, 84], which are therefore more interesting targets for attackers, it does not seem unreasonable to assume these devices will be phased out in the future.

The 3GPP standardisation organisation has specified several security goals for the UMTS and LTE cellular systems [55, 68] which expand the security goals that were stated for GSM [53]. We will now see what the impact of a compromised femtocell is on all the goals that could conceivably be influenced by femtocells. In some cases this will add new attacks that were previously impossible. In other cases an already existing attack that is currently hard to perform due to the cost of implementing UMTS/LTE signal processing, could be made easier to implement with a compromised femtocell, because it already handles all the signal processing out of the box. This effectively means the introductions of femtocells can lower the costs of an attack.

User data confidentiality and integrity

These two security goals concern the confidentiality and integrity of user data against eavesdroppers and active attackers. Lawful interception is an exception on user data confidentiality.

None of them are weakened by a compromised femtocell, when we assume that no local break-out is implemented in the femtocell, as there is a secure tunnel from the handset to the provider's core network, which is authenticated and provides both confidentiality and integrity. It is infeasible for a compromised femtocell to decrypt or compromise this traffic (assuming strong enough encryption and MACs are used,

such as the KASUMI cipher in UMTS). It is also impossible to influence the encryption choice of the network.

Network authentication

This security goal was specifically added for UMTS security to mitigate an important weakness of GSM. It aims to protect subscribers from fake cell towers through authentication of the network and is unbroken by a compromised femtocell.

This authentication is done by the so-called UMTS-AKA protocol. The network provides a handset with cryptographic proof of knowledge of a shared secret key and a sequence number to prevent retransmission attacks. The UMTS-AKA protocol was formally analysed using enhanced BAN logic and shown to provide both authentication and confidentiality [57]. It is discussed in more detail in Sections 2.1.2 and 3.5.

The femtocell never learns the secret key shared between handset (more specifically SIM card) and network and is as such unable to fake a connection to the real network. Retransmission of an authentication token is infeasible because of the sequence number. When a correct UMTS-AKA run finishes, handset and network communicate through a secure tunnel. This makes it impossible for a compromised femtocell to hijack the session without local break-out.

Subscriber identity authentication

Subscriber identity authentication is meant to protect the network against unauthorised use by ensuring that the subscriber identity transmitted to the provider is the one claimed.

This security goal is ensured through the mutual authentication of handset and core network. This mutual authentication uses the UMTS-AKA protocol, as mentioned above. The authentication itself does not happen on the femtocell, but inside the provider's core network, so insider attacks, such as swapping the authentication tokens inside the network [170], are not feasible from a femtocell without local break-out.

So attacks need to circumvent the UMTS-AKA protocol. A possibility is to place an emergency call at a femtocell and immediately place another call afterwards. An emergency call creates an unauthenticated radiolink, and this link is kept open for the second call. This results in theft of service with a possibly spoofed subscriber identity. However, this threat is detectable by the core network and as such this risk is accepted in the specifications [63]. This attack does not require a compromised femtocell, but could be more easily realised with a compromised femtocell. Due to the detectability the impact is probably small.

Subscriber identity confidentiality

Subscriber identity confidentiality comes down to the secrecy of the IMSI number from eavesdroppers and active attackers. This secrecy is already problematic in current networks due to an *identity request* procedure, which causes the handset to respond with its IMSI in plaintext. So, this attack — often referred to as an IMSI catcher attack — is not introduced by a compromised femtocell, though a compromised femtocell does make the execution of an IMSI-catcher attack a lot easier [84].

The specifications also explicitly state that there should not be any relation between the IMSI number and the subscriber's phone number, other than in a database in the provider's back-end. The check whether a handset (or, more accurately, a SIM card) belongs to the CSG — the Closed Subscribers Group, discussed in Section 5.3.1 — can be made inside the femtocell or within the provider's core network. In the former case a compromised femtocell can uncover the IMSI-phone number relation by adding phone numbers to the CSG, which is standard functionality available to the femtocell owner. The phone numbers need to be translated to IMSIs by the core network and subsequently stored in a CSG femtocell, where they can be uncovered by an attacker. This attack against the subscriber identity confidentiality is more effective than IMSI-catcher attacks, because victims do not need to be connected to, or even near, the attacker. As far as we can tell, this IMSI-*harvest* attack is a new attack, with a higher impact than other attacks against subscriber identity confidentiality.

Signalling confidentiality

The signalling messages between handset and network should remain confidential. Since we assume a femtocell without local break-out, only the signalling that is transmitted outside of the user session tunnel is subject to confidentiality breaches. Since this concerns all unencrypted messages on the wireless link, no new weaknesses are introduced by a compromised femtocell, although some attacks are easier to implement with one.

The most prominent attack here is IMSI catching, which is detailed under the security goal *Subscriber identity confidentiality*. Another attack vector lies in the paging channel. Handsets listen to this channel to see if they have incoming transmissions, by looking for occurrences of their IMSI or, more frequently, their TMSI (a temporary pseudonym for their IMSI). This could lead to a traffic analysis where all incoming transmissions for subscribers connected to a compromised femtocell can be revealed.

Signalling integrity

An attacker should not be able to alter signalling messages between handset and network. A compromised femtocell introduces the attack that makes it possible to alter unencrypted signalling messages.

The user session tunnel guarantees integrity of messages, so any attacks against signalling integrity, have to be made on untunnelled signalling. Attacks against signalling integrity can lead to DoS attacks, which are discussed in the section on Availability shown below. Other possible attacks are to fake paging messages to handsets — which cause a handset to indicate incoming transmissions, when there are none — or abuse of the broadcast messages — such as fake alert messages of the Public Warning System (PWS) [90].

Subscriber location privacy and untraceability

The current or earlier location of a subscriber should not be derivable from transmissions on the air interface.

A recently found location privacy attack [4] unveiled a weakness in the UMTS protocol that can be used to break subscriber location privacy. In short, cell towers send

an *authentication request* message to a mobile phone. This message contains both a proof that the network knows the SIM card's secret key and a sequence number needed for freshness. The mobile phone responds with an error message if the proof of knowledge of the secret key is incorrect, or with a different error message if the sequence number is incorrect. So by replaying any, earlier recorded, correct *authentication request* message for a specific phone, an attacker can see if the target phone is in his current cell. This attack is very similar to a traceability attack against e-passports by Chothia and Smirnov [31], where they also use responses of replayed messages to determine whether an RFID passport is in the vicinity.

The location privacy attack in UMTS was implemented on a femtocell as a proof-of-concept. The implementation showed this attack is indeed viable from any 3G cell tower ranging from femtocell to macrocell.

This attack is also viable from a compromised femtocell without local break-out, since the correct *authentication request* messages are sent plain text from the provider's core network, as no session key is yet established, and can always be replayed without access to the session secrets.

Availability

Attacks against availability, commonly known as DoS, are considered in the UMTS and LTE specifications [56], but availability is not officially stated as security goal [55, 68]. DoS attacks performed from a femtocell can be subdivided into two categories: DoS attacks towards the subscriber and DoS attacks towards the provider.

DoS attacks towards the subscriber are trivially possible by blocking any incoming or outgoing data transmissions on a subscriber camping on a compromised femtocell. Another method to perform a DoS attack is to send malformed packages to the handsets, which attempt to compromise their base-band stack. This could be done at very low layers of the protocol (for example by changing something in the waveforms), below the layer with the integrity checks of the secure tunnel, or by attacking all the layers in the untunnelled signalling messages, such as the broadcast and paging messages. This process of creating malformed packets is called *fuzzing*. To our knowledge this attack has not been attempted on UMTS base band stacks. However, several fuzzing attacks against GSM have shown that older (GSM) base band stacks are vulnerable to this. Chapter 6 will give a detailed overview of fuzzing attacks against GSM baseband stacks.

Another DoS attack that was possible with earlier femtocells [128, 84], is no longer possible on a femtocell without local break-out. In this attack the IMSI detach message of a camped handset is faked. This will cause the network to assume a phone has been switched off and therefore hold all inbound transmissions to this handset. However, this attack only works as long as a handset is connected to the compromised femtocell, and the femtocell needs local break-out to perform it.

DoS attacks towards the provider's core network also seem possible. The most obvious point would be to attack the SeGW, since this entity sets up the IPSEC connections, and therefore needs to do many calculations in order to send and verify received cryptographic messages. If many attacking machines attempt to set up an IPSEC connection with the SeGW it will get overloaded and the connections between the SeGW and genuine femtocells will suffer. An attacker would not need to com-

Table 5.1: *Overview of successful attacks on femtocells; the last entry is the femtocell we attack in this chapter*

	Vendor	Type	Weakness	reference
1	Sagemcom	Vodafone SureSignal	Guessable root password	[163]
2	Samsung	Verizon SCS-24UC4 & SCS-2U01 & Sprint Airave	Adjustable boot loader	[72, 169]
3	Ubiquisys	SFR Home 3G	Insecure update procedure	[23, 84, 4]
4	Sagemcom	Vodafone SignaalPlus Plug&Play	Insecure recovery mode	

promise a femtocell for this, though access to the secrets needed to set up the IPSEC tunnel can make this attack more effective, by causing more computations in the SeGW. Whether it is possible to DoS other entities in the provider's core network, such as the *AAA*/*HSS*, is hard to predict. As we discussed, there have been several successful fuzzing attacks against handsets. So it would seem logical to assume that the base band stack on network equipment is also vulnerable when handling packets just outside of the specifications. Some attacks against the core network were found by a private security company [141], which seems to support this assumption, but it remains impossible to test without access to a test network or by possibly harming the real network.

5.5 Practical security analysis of the Vodafone Plug & Play femtocell

The femtocell itself needs to be a hardened device, since it contains credentials to authenticate to the provider's core network and is placed at the subscriber's home, but it should still be under the management and control of the provider. For a practical security analysis on a modern femtocell we looked at the Vodafone SignaalPlus Plug&Play, the first commercially available femtocell in The Netherlands, available for 80 euros. We first give a high level overview of our attack and discuss its nett effect. We then provide the details for the interested reader.

Overview of the attack

We were able to read out the unencrypted memory of the femtocell, which provided all the secrets needed for our attack. It proved possible to reboot the femtocell in an insecure recovery state, by sending a command over the ethernet connection on a TCP port. The firewall that runs on the femtocell only opens up this port after a secret port-knocking sequence is completed. Once in recovery mode the femtocell has SSH enabled and attempts to retrieve a file via a tftp session to a local network address, which it then executes. We provided the femtocell with a file that gave us a root login on its SSH prompt.

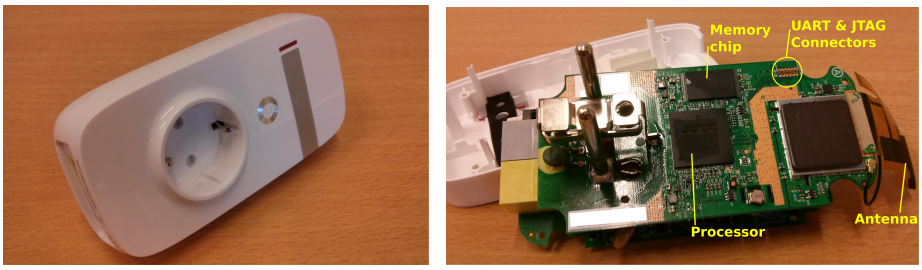


Figure 5.3: *The Vodafone SignaalPlus Plug&Play femtocell.*

The recovery mode of the femtocell runs a different Linux version than the normal mode. From the recovery kernel we can mount all the other partitions, but we cannot get the femtocell into operation, since most programs will not run from the recovery kernel version. So getting the femtocell in operation would require rewriting most of the binaries for this specific kernel version. Also, this implementation would invariably need to be tested, which could result in some non standard behaviour that might be observable inside the Vodafone back end.

However, we were able to compile programs that run on the femtocell in recovery mode. This means we can run arbitrary code on this femtocell, in essence this breaks the security model as detailed in Section 5.4.

The details

We first attempted attacks that were successful on older femtocell models (summarised in Table 5.1):

1. There was no SSH running on our femtocell, so easy guessable root passwords were out.

2. Researchers from Trustwave's SpiderLabs looked for the differences between the source code made available by Samsung[3], to which they were obliged under GPL, and the stock open source versions. They found a key that allowed them to enter the boot loader's menu via the serial port. In our case the femtocell also uses GPL code, but the code placed online by Vodafone was not the same as the code that runs on the device (which we could uncover by dumping the memory chip). Although this is a clear violation of the GPL license, it did hinder our efforts.

3. Holding the reset button does not prompt our femtocell to connect to an insecure update server, like the Ubiquisys did.

So, all previous attacks failed. The Vodafone SignaalPlus Plug&Play, shown in Figure 5.3, is manufactured by Sagemcom. Inside the casing (which can be removed without any physical counter measures) there are two connected PCBs, with a Percello 6000

[3]The source code was made available via the webpage `http://www.samsung.com/global/business/telecommunication-systems/resource/opensource/femtocell.html`

chip and a flash memory chip. The JTAG connectors are logically disabled[4], but there are active UART connectors that show a console log during the boot sequence.

The Percello 6000 chip has a built-in TPM that presumably contains the secrets needed to set up the IPSEC tunnel. The boot logs also show that the integrity of the code which runs on the femtocell is verified: the hashes of some files are compared with signed hashes from the TPM.

The data on the flash memory chip is unencrypted and stored in an UBI format and is divided in several volumes that we were able to dump through direct access to the memory chip. Nearly every volume needed for normal operation has an exact duplicate labelled either with an *A* or *B* post-fix, which seems built-in redundancy (for instance in case an update fails and corrupts the file system). The other partitions also have an exact duplicate, but now with a *_BKP* postfix. Only the recovery partition has no duplicate.

The femtocell runs a firewall that blocks most ports. However, a port-knocking daemon also runs from the *RFS_A* partition. We set up the femtocell on an internal network, without outgoing internet connection. On this internal network we ran our own DHCP server and DNS gateway. After sending packages to the femtocell in the order of the porting-knocking sequence, the final port opened in the firewall for a TCP connection. Reverse engineering the binary that listens to this newly opened port revealed two commands: "reboot recovery" and "switch bank". After the command "switch bank" was sent to the newly opened port, the femtocell boots from the B partitions (or back from the A partitions). More interestingly the "reboot recovery" command causes the femtocell to boot into a recovery mode.

A trace from the WireShark network protocol analyser showed that in recovery mode the femtocell attempts establish a tftp session to the fixed IP address 192.168.1.1 and requests a file called `femto3xx/originalsin`. The *LINUX_R* volume contains the recovery kernel, which is booted in recovery mode, and a compressed recovery file system. This filesystem contains a file `adam` that is called immediately after the boot procedure. This file proved to be a simple script in the execline syntax that tries to tftp the file `femto3xx/originalsin` into a temporary file eve. This eve file then has the executable bit set and is executed. Since `adam` runs with root privileges, the attack file that we offer for the tftp session can put our public key in the SSH `authorised_keys` file of the root account. This gives us root access through SSH on the recovery mode of the femtocell.

We were able to replicate the attack on multiple femtocell devices of the same version. Through the shadow files we found that the root password is the same for every device we gained access to. Running John the Ripper on the root password hash yielded no results.

Using a MIPS compiler we can compile programs that run onto the femtocell, and this gives us arbitrary code execution on the femtocell.

[4] JTAG (Joint Test Action Group) is an industry standard for debugging access to embedded processors and printed circuit boards.

5.6 Future Work

A compromised femtocell without local break-out offers some attack possibilities discussed in Section 5.4, which should be examined further. Most prominently these are the integrity attacks against the untunnelled signalling messages that could offer up new attacks. Also, a compromised femtocell can make fuzzing attacks over UMTS protocols against handsets possible, which to our knowledge have not been attempted before.

We also see some ways to improve our attack against the Vodafone SignaalPlus Plug&Play femtocell. It might be possible to reactivate the JTAG connectors. This would allow a degree of control on the processor that our current attack does not provide.

Our attack could also be extended in using the TPM as an oracle, in order to analyse the data sent through the IPSEC tunnel which are not part of the 3G traffic, so all the management data. We are able to execute arbitrary code on the femtocell, which makes this approach possible, but due to a lack of time we were unable to perform this attack.

5.7 Conclusions

We have provided the first comprehensive security analysis of a femtocell without local break-out in Section 5.4. We have shown that a compromised femtocell enables attacks that directly impact several security goals:

- Subscriber identity confidentiality
- Signalling integrity
- Availability

Several attacks already exist without a compromised femtocell, but we argue that some of these are much easier to exploit with the use of a compromised femtocell. This resulted in easier implementation of attacks against several security goals:

- Subscriber identity authentication
- Subscriber identity confidentiality
- Signalling confidentiality
- Availability

Several attacks using older model femtocells with local break-out, are not possible in our model of a femtocell without local break-out. Of these, the eavesdrop attack on subscriber data probably has the most impact. So, the security of a cellular network with femtocells is improved when the femtocells do not support local break-out; in essence the provider places less trust in a femtocell.

Our analysis resulted in two new attacks, which to our knowledge were not published earlier: (i) the IMSI-harvest attack discussed in the section on Subscriber identity confidentiality (page 82) and (ii) fake Public Warning System messages, discussed in the section on Signalling integrity (page 83).

We also show a practical attack on a modern femtocell without local break-out. A dump of the code of the femtocell enabled us to learn the port-knocking sequence that allows the femtocell to go into an insecure recovery mode, which retrieves a file and executes it. With a couple of days of effort, we were able to gain root access to this device and were able to execute arbitrary code on it. We gained fewer capabilities than previous hacks of older femtocells (which did implement local break-out). Our femtocell was also secured against earlier known attacks.

We made some interesting observations while examining the femtocell. First of all, in accordance with GPL, Vodafone provides a link to source code. However, the provided source code is not the code that actually runs on the femtocell. It appears to be code meant for an older version of different hardware by Alcatel-Lucent, instead of the current version by Sagemcom. This is clearly a violation of GPL and it forced us to dump the contents of the memory chip for analysis.

Secondly, it seems strange to disable SSH access, but to allow access to the femtocell through the secrecy of a port-knocking sequence, which is poor security, since the secret sequence cannot be stored securely. However, the benefit of the port-knocking defence is that this will only work locally, since most devices will be placed in a NAT environment in a subscriber's home, so the router would already block most ports. SSH on the other hand might be accessible over the internet. This would have been especially worrying, since we found that all devices of this type we bought had the same (hashed) root password.

Both our theoretical and practical analysis suggest the security of femtocells is improving. None of the weaknesses from earlier models were present in the new femtocell. Though the main improvement is that the providers place less trust in the femtocell devices, because the femtocells do not provide local-breakout. One should always assume that a femtocell will eventually fall under control of an attacker, so the less trust that is placed in the femtocell, the better. Femtocells without local break-out are a definite improvement, as are femtocells that do not check the membership of the closed subscribers group themselves.

However, femtocells with local break-out are still available on the market and as long as these can connect to the core network, femtocells without local break-out add little security. Even with these femtocells without local break-out some attacks remain possible when a femtocell gets compromised, though these attacks typically have a lower impact.

Responsible Disclosure

We informed Vodafone Netherlands of our findings. They informed us that recent models of their femto cell do not expose the recovery mode. We could confirm that our attack indeed no longer works on these models.

Chapter 6

Fuzz testing GSM implementations

The previous chapters in this thesis have mainly focused on the protocol specifications and cryptography layer of mobile telephony. Finding weaknesses in the specifications can have major impact, because all the equipment has to follow the specifications quite narrowly to allow cooperation between different vendors. However, looking at the implementations of these vendors can reveal completely new vulnerabilities. Even though the specifications are to be followed, these are also notoriously complex, with very many options, of which a large part is hardly ever used. Also specifications tend to only provide the scenarios for correct input, leaving the incorrect cases up to individual programmers to handle.

There are not as many vendors of so-called baseband stacks, the software layers running the wireless part of mobile telephony networks, as you might expect. There are only around 5 major vendors for baseband stacks running on mobile phones, at the time of writing. These are Qualcomm, Broadcomm, MediaTek, NVIDIA and Intel. So, finding a vulnerability in one of their products could potentially threaten many consumer phones. Unfortunately, all these stacks are closed-source, making meaningful security research much harder.

This chapter focuses on our research efforts in fuzz testing the implementations of the mobile telephony protocols. This research showed that basic forms of fuzz testing – effectively random, automated testing – can quickly reveal many bugs in the software implementing the GSM stack in mobile phones. Fuzzing could reveal most of the security vulnerabilities present in this software in an efficient way.

This chapter is based on the article *Security Testing of GSM Implementations*, presented at the International Symposium on Engineering Secure Software and Systems, ESSoS 2014 [175]. Besides some changes in the introduction for a better integration with the rest of the thesis, and moving some information to chapter 2, the background chapter, nothing changed with regards to the original publication.

6.1 Introduction

With current, off-the-shelf, hardware and open-source software it is possible to run your own GSM cell tower to which real phones will connect, since in GSM the network does not authenticate itself to the phones. This opens up the possibility to verify the implementations of the GSM stack of phones by the technique known as fuzzing. Fuzzing has been used a lot to find security holes on internet equipment. Thanks to low level access offered by Ethernet cards it was easy to simply try out all kinds of possible messages, mostly those just outside of the specifications, and see what happens when these are received by network equipment. Fuzzing mobile phones has mostly happened in the hackers scene of security research, with few academic publications.

Naturally, there are many interfaces in mobile phones which can be fuzzed. Just think of every type of input that a phone can receive, such as WiFi, Bluetooth, NFC, installed apps or the SIM interface. All of these inputs can be interesting input vectors for fuzz testing. We focused on fuzzing the GSM baseband stack. This is the part of the phone which handles all the GSM traffic. It is available in every phone, implements a hugely complicated standard and is remotely accessible over the air, which could easily lead to dangerous attacks.

The GSM system comprises many entities, such as the mobile phones and cell towers, but also many more back-end components. Our fuzzing research only focuses on mobile phones. Naturally, fuzzing the network components of a GSM network can have a much larger impact. However, availability of commercially used network components that are not currently running inside an operational GSM network is very limited. Thus we limited ourselves to the readily available mobile phones. In this chapter we discuss our efforts and results in fuzzing two specific parts of the GSM specification: SMS messages and CBS messages.

The well-known Short Message Service (SMS) was added shortly after the initial release of GSM and the first SMS message was sent in 1992 [89]. The first version of SMS allowed the exchange of short text messages between GSM users, but SMS has gone a long way since then. Not only can SMS be used to exchange text messages, but nowadays also pictures, sounds and many other types of data can be sent over the SMS. The current SMS standards also allow segmentation of messages that are too long to fit into a single message, enabling users to transmit much longer messages. The current SMS specification is found in [61, 67].

The lesser-known Public Warning System (PWS) actually started out as the Cell Broadcast Service (CBS), which was developed in parallel to the SMS service as a response of mobile developers to the competing paging services being offered in 1990. It allows providers to broadcast messages to all phones currently connected to a certain cell, i.e. all phones connected to a single transceiver on a cell tower. The original business case was to provide news, weather and traffic information to mobile users, though this never found any wide spread popularity. This lead to both mobile network operators and mobile developers neglecting the implementation of the service in their equipment. However, this service has been gaining importance in the last years, because it can be an ideal method for governments to broadcast information in the event of an emergency to all phones in the vicinity. Several countries define and implement their own warning system that rely on the CBS to deliver emergency information. Due to the diversity of technical specifications of each warning system,

ETSI with the aid of the 3GPP consortium developed a standardised system known as the Public Warning System (PWS). The initial goal of the PWS was to introduce a standard emergency and warning communication infrastructure, as well as specific technical requirements for mobile phones within the European Union to receive these emergency messages. Due to its standardised nature this system and its accompanying protocols can now be implemented worldwide. This allows roaming users to receive broadcast messages no matter what their location is, as long as they are in a GSM coverage area.

In Section 6.2 we discuss the basics of the GSM air interface and provide an introduction into the SMS and CBS protocols. We then discuss fuzzing in general and the specifics of fuzzing mobile phones in Section 6.3. Section 6.4 describes our own fuzzing research, together with the practical details and results. It is here that we also discuss the related work, for comparison and to attempt to provide an overview of the fuzzing research into GSM up to this point. Finally, Section 6.5 presents the conclusions and ideas for future work.

6.2 GSM

The GSM baseband stack is usually described in three layers, where the third layer is again subdivided, as is shown in Figure 6.1. The bottom two layers of the GSM stack show similarities with the OSI model. The first layer, the physical layer, creates a set of logical channels through time division on already divided frequencies. These channels can be used by higher layer functions for many different tasks, as uplink (mobile phone to cell tower), downlink (cell tower to mobile phone) or broadcast (cell tower to all connected phones) communication. These channels can either be a traffic channel, or one of a multitude of control channels. Most control data is transmitted in 184 bit frames which are split up into 4 bursts. These bursts are modulated and transmitted by radio waves.

The signalling protocol used on the second layer, the data link layer, is called LAPDm. The data link layer (and higher layers) is only defined for the signalling channels, not for the speech channels. This is because speech bursts contain no further headers or other meta information, only speech data; during a phone conversation, the traffic on the dedicated speech channels needs no meta information in order to be reconstructed correctly at the receiving end. The LAPDm protocol can provide positive acknowledgement, error protection through retransmission, and flow control.

The third layer is where the match with the OSI model stops. The third layer is subdivided into three layers, of which the last (highest) one is again subdivided into several protocols:

1. Radio Resource management (RR); this concerns the configuration of the logical and physical channels on the air-interface;
2. Mobility Management (MM); for subscriber authentication and maintaining the geographical location of subscribers;
3. Connection management (CM); consists of several sublayers itself, such as:
   (a) Supplementary Services (SS); managing all kinds of extra services that are not connected to the core functionality of GSM;

(b) Short Message Service (SMS); the handling of the SMS messages;

(c) Call Control (CC); creating and ending telephone calls;

(d) Locations Services (LCS); location based services for both the user and the provider;

Layer 3 frames consist of a 2 byte header followed by 0 or more Information Elements (IEs). These IEs can be of several different types: T, V, TV, LV and TLV, where the letters T, L and V denote the presence of a Type, Length and Value field respectively. The type field is always present in non-mandatory IEs. Interesting from a fuzzing perspective are those IEs that contain a length field, LV and TLV, even though they are specified as having a standard length, because these are typical places where a programmer might make a mistake in handling data of non-standard length.

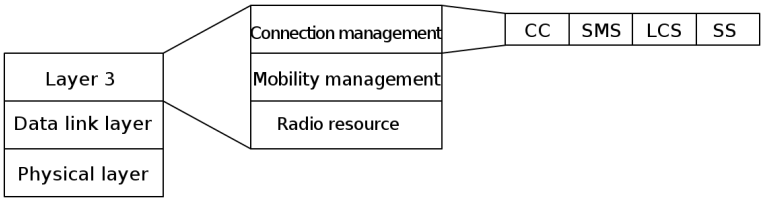


Figure 6.1: *The layers of GSM*

We only fuzz on the third layer of the protocol stack, since this is more likely to trigger observable bugs than fuzzing on the first two layers. That is not to say that the lower layers of the protocol will likely contain less, or less nasty bugs, they are simply harder to observe.

SMS

Before messages can be sent on the SMS sublayer the cell tower needs to notify the mobile phone of an incoming message and set up the channel (Standalone Dedicated Control Channel or SDCCH) with the mobile phone. The delivery of an SMS message then requires four messages exchanged on the SMS sublayer using the SDCCH, as shown in Figure 6.2. The first message is the SMS-DELIVER message sent from the network to the phone. This message contains the actual content (user data) with

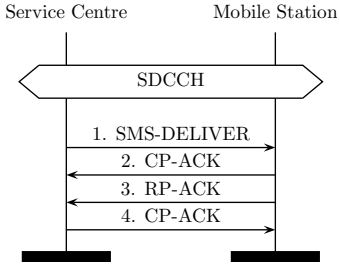


Figure 6.2: *Message sequence chart of delivering an SMS to a mobile phone*

an optional User Data Header (UDH) and mandatory Transfer Protocol (TP), Relay Protocol (RP) and Connection Protocol (CP) headers. The phone first parses the CP header and verifies it. If it is valid the phone (Mobile Station or MS) returns a CP-ACK message, otherwise it returns a CP-ERROR message and releases the connection. If the CP header was correct the MS continues by verifying the RP header and checking if the phone has enough memory to store the message. If either of those checks fails it returns an RP-ERROR and releases the connection. If both checks succeed the MS returns an RP-ACK with a CP header. The final message is sent by the network when the RP-ACK passes the checks for the CP header.

A schematic overview of a correct SMS-DELIVER message can be found in Figure 6.3, where Figure 6.3(b) shows the fields we fuzzed of the SMS-DELIVER message. Of the RP-ACK message we fuzzed practically all header fields.

Public Warning System

For CBS messages no specific traffic channels need to be set up for mobile phones to receive the transmission. The messages are transmitted on the broadcast channel, a specific channel to which all phones always listen to see if they are still in the same cell. So even if a cell is overloaded with regular voice or data traffic, broadcast messages can still be sent to mobile phones. This very feature is what makes them interesting for emergency messages in the first place. Japan's tsunami warning system and the European emergency broadcast (EU-Alert) are examples of implementations of the PWS.

A CBS message is first announced on a broadcast channel and then transmitted in four frames. A schematic representation of a CBS message is shown in Figure 6.4 where all the fields we fuzzed are shown in grey.

6.3 Fuzzing

Fuzzing is the process of transmitting automatically generated, uncommon inputs to a target with the purpose of triggering unexpected behaviour. This unexpected behaviour is typically something like program crashes or failing built-in code assertions. In contrast to human testing, fuzzing can be largely automated and as a result can find (security) errors that likely will not be triggered during normal use or testing. Fuzzing is already a relatively old testing technique, dating from the end of the 70s and start of the 80s [129]. Fuzzing has evolved over the years into several variants:

1. plain fuzzing,
2. protocol fuzzing, and
3. state-based fuzzing.

Plain fuzzing is the original idea behind fuzzing: simply generating lots of test cases, often with random data, and feed this to the program you are testing. These test cases are usually made by mutating correct inputs and is used to test the error-handling routines. It is a highly portable way of fuzzing, but also provides very little assurance on code coverage.

In protocol fuzzing the test cases are generated based on the specifications, especially on specifications of packet formats. Here the fuzzer will try to choose specific test cases which are likely to provide the largest code coverage based on its knowledge of the specifications. Typically, fuzzers will look at each field which can contain more values than are allowed by the specifications and generate test cases with values on the corner cases, values just over the corner cases and some values way out of the range of what is allowed. This is also called "partition fuzz testing", since the possible inputs for a field are partitioned (e.g. a half byte which represents an ID and allows for the values 1-12, would give three partitions: 0, 1 to 12 and 13 to 15). These are often not partitions in the mathematical sense, as they need not be disjoint, but the union of all partitions usually do span the entire input space. Note that although these fuzzers are named "protocol fuzzers" and are in fact mostly used to test protocols, they can also be used to test non-protocol implementations, as long as the input has a format to which it should conform.

The first two fuzz variants discussed here try to find errors by changing the content of individual messages. But there is another part that can often be fuzzed: the state machine. Most protocols have some sort of set sequence in which messages are exchanged and which messages are expected at any one time is tracked in a state machine. When the wrong message is sent at some point in time and still accepted by the implementation it shows a problem with its state machine. The impact of this is hard to estimate, because it depends on what states can be skipped, but for some protocols it might allow one to bypass authentication steps, posing a serious security risk. State-based fuzzers not only change the content, but also the sequence of messages.

Whichever fuzzing approach is used, fuzzing will usually follow three distinct phases:

1. generating the test cases,
2. transmitting the test cases, and
3. observing the behaviour.

The fuzzing approaches discussed above concern the first phase. On traditional computer networks the second phase is trivial. Also, observing the effects after transmitting the fuzz tests is usually easier on traditional computers than on GSM phones, because there is often the possibility of running a debugger, or simply looking for familiar error messages. Fuzzing GSM implementations on baseband chips has many of the same problems as the fuzzing of embedded systems; one cannot easily observe the effect of fuzzed inputs [113].

6.3.1 Fuzzing GSM phones

There are many different GSM-enabled mobile phones. Mobile phones started out with just the ability to make and receive voice calls, but nowadays phones are available that have a wide range of features and possible connections. It is important to realise that the current market contains a wide variety of GSM-enabled devices, not only of different make and model, but also internally: GSM phones can consist of a single processor, which runs the GSM protocol stack and a very limited OS for the

user interface, these are typically cheaper or older GSM phone models. The chip running the GSM protocol stack is referred to as the *baseband chip* [191]. More complicated phones run their OS on a separate general purpose processor, named the *application processor*. Both processors can communicate through a variety of protocols, where the application processor uses the baseband processor like a modem. Most modern phones combine the application and baseband processor in a single SoC (System on a Chip).

Fuzzing mobile phones is challenging compared to e.g. fuzzing network cards, mainly because it is hard to observe undefined behaviour. Most phones are closed devices without any debugging tools, so it is impossible to, for instance, look at the memory during operation. Also most phones run closed source software. This makes it harder to predict where errors will occur. Even Android phones use closed software libraries for low level communication with the baseband chip and if the baseband chip has its own memory a debugger on a rooted Android phone will provide little extra help. Phones usually have limited interaction possibilities, which makes observation a time consuming effort. Generally there are few alternatives to simply using the phone after a fuzz message and observing whether it shows any undefined behaviour, which can lead to false positives. This means internal errors that do not directly lead to observable undefined behaviour, may go unnoticed.

The fuzzed messages need to be introduced to the target phones. This can either be done by transmitting them as actual GSM messages to the phones, or by directly inserting them in the phones, for instance by inserting them on the wire between the baseband and application chip. The latter option is cumbersome to use on many different phones and much harder on modern phones with a single SoC, but this was the only available option when open source GSM networks were not available [126].

For transmitting the messages over actual networks there are several options:

1. you could use a running GSM network, either because you happen to have access to commercial GSM network equipment, or by transmitting fuzzed messages from a modified phone to a target phone over the normal network.

2. A more feasible solution is to change existing GSM equipment, such as a femtocell for transmitting fuzz messages, though the success of this method will depend on the success in breaking the femtocell security. See the previous chapter for more details.

3. Finally there are several open-source projects that allow you to set-up your own GSM network.

Using the existing network (option 1) severely limits the fields you can fuzz and the network operator could change or filter our messages. Adapting a femtocell to do the fuzzing (option 2) could prove unsuccessful, so we chose the third option. The most important of the open-source projects are OpenBTS [26] and OpenBSC [189]. Both systems run on most ordinary PCs and require extra hardware for transceiving GSM signals.

OpenBTS is based on the GNU Radio project [82] and is designed to work with the USRP (Universal Software Radio Peripheral), a generic and programmable hardware radio component. The USRP can be modified through the use of daughterboards

for specific applications and frequencies. Several versions of the USRP are currently available and a typical setup for a local GSM network costs around $1500,- [49].

OpenBSC runs a basestation controller and therefore interfaces with an actual basestation in order to work. OpenBSC started out as a controller for the Siemens BS11, an actual commercial cell tower of which a small batch became available on eBay, and the nanoBTS from ip.access, a corporate solution miniature cell tower. Both cell towers are hard to obtain, so OpenBSC will now also work with a new project, OsmoBTS [190], which in turn implements a cell tower on several devices such as the custom made sysmoBTS and even, experimentally, two modified mobile phones.

The OsmoBTS project was not yet available when we started our research, which led us to choose the OpenBTS option, for full control of the GSM air link.

6.4 Our fuzzing

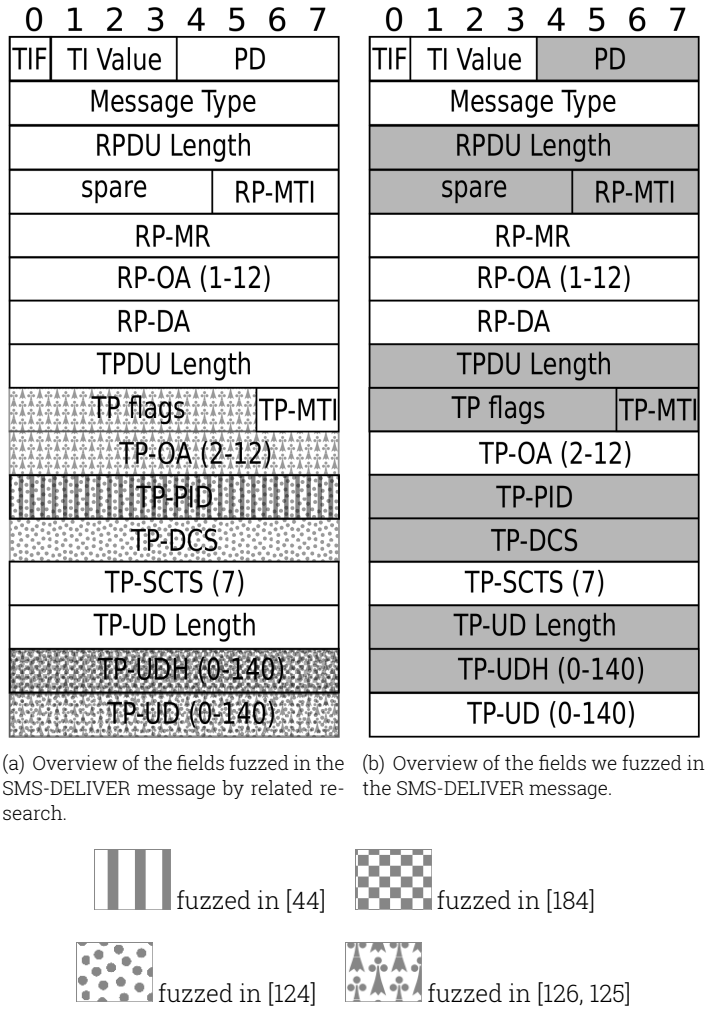
For GSM all the specifications are openly available, but implementations of the baseband stacks are not. Examining the specifications led to the conclusion that although GSM is a very complicated protocol, there are actually very few state changes in the baseband stacks. This is why we mostly resorted to protocol fuzzing, as will be described in Section 6.4.1 and 6.4.2. We attempted some state-based fuzzing on the SMS sublayer by both sending a correct message when it was not expected by the phone and sending a correct message when a different message was expected by the phone. This only showed unexpected results for one phone, the Sony-Ericsson T630, which accepted confirmations of unsent SMS messages, but which did not lead to any exploitable results.

There are several open-source protocol fuzzing frameworks available [121]. However, these frameworks are not made to be used with cell phones. Especially the target monitoring aspects generally work on network interfaces and virtual machines, while we have separate devices connected over a (custom) radio interface. This makes automatic target monitoring with one of these tools impossible. We did end up using one fuzzer, Sulley [138], as the basis for our fuzzer.

6.4.1 How do we fuzz?

For this research we made our own fuzzer GSMFuzz, for the generation of the fuzz messages. It is a fuzzer, with features designed specifically for GSM, but which nonetheless can be used for other protocols as well. The fuzzer is written in Python (version 2.6) and loosely based on Sulley[138], an open-source fuzzing framework. It has the following features added:

- Fuzzing of bit positions within a byte;
- Partition fuzz testing of special fields (type, length), resulting in few cases with maximum impact;
- Innate support for the eight different GSM Layer 3 IEs;
- Length fields can count octets, septets or half-octets (often used in GSM);



(a) Overview of the fields fuzzed in the SMS-DELIVER message by related research.

(b) Overview of the fields we fuzzed in the SMS-DELIVER message.

Figure 6.3: *Overview of the fields fuzzed in the SMS-DELIVER message.*

- Hexadecimal output of fuzz cases to a file, which can be used directly in our extended version of OpenBTS.

GSMfuzz itself is just over 900 lines of code (excluding white space). Besides the source code of the program itself we created 34 files with input to mutate valid messages. The input files are 3601 lines in total (excluding white space and comments). The source code for GSMFuzz is available upon request.

Figure 6.3(b) shows the fields we fuzzed in the SMS-DELIVER message and Figure 6.4 shows the fields we fuzzed in the CBS message.

For the transmission of the fuzzed messages we used the open-source OpenBTS software together with a USRP-1 (where the internal clock was replaced with the

8	7	6	5	4	3	2	1
Spare	LPD		LB	Sequence Number			
Serial Number (2)							
Message Identifier (2)							
Data Coding Scheme (1)							
Page Parameter (1)							
Content of Message (82)							

Figure 6.4: *CBS message fuzzing candidates*

more precise Fairwaves ClockTamer-1.2) and a collection of (a WBX and two RFX1800) daughterboards. Combining this with two Ettus LP0926 900 MHz to 2.6 GHz antennas yielded a setup of around 1500 €.

We tuned the software to only allow a specific set of SIM cards to connect, but this did not prevent several phones in the surroundings to still connect to our cell. This already shows errors in how phones handle connecting to cell towers. Since we did not want to unintentionally harm the phones of our colleagues, we made a Faraday cage around the whole setup, using chicken wire. With a maze size of 12.5mm, which is smaller than ten times the wavelength of GSM signals on 1800MHz, we managed to keep our GSM broadcasts contained.[1] This setup is shown in Figure 6.5.

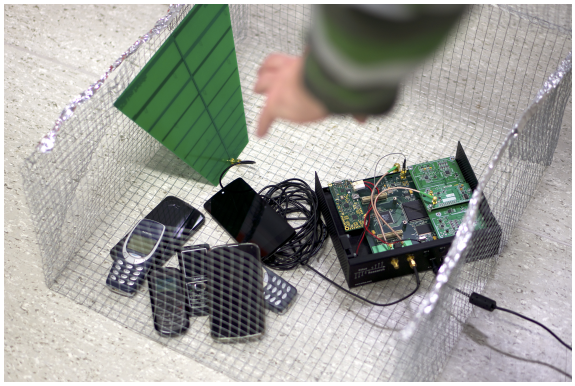


Figure 6.5: *A photo showing our fuzzing setup*

The OpenBTS software does not support emergency broadcasts[2], so for these broadcasts we installed a specific branch of an older OpenBTS version (OpenBTS 2.5.4 - SMS-CB), where this service was already implemented.

Having the ability to generate and transmit fuzzed messages, leaves the third stage: the observation. In our SMS fuzzing case, we alternated each fuzzed SMS message with a correct SMS message to see if the phone still responded by acknowledging the correct message. Then after transmitting a batch of alternating fuzzed and normal SMS messages we quickly tried most functions of the phones.[3] For our CBS fuzz-

[1] There was some leakage through the power cord, but not enough to get phones outside of the cage to connect.

[2] It is likely that this will be included in a newer release [26].

[3] At this stage we also used the phones to empty the SMS memory, which is limited in the older models

Table 6.1: *Overview of cell phones tested in this research and the most noticeable results. Legend: I: irremovable icons, D: DoS message, M: memory bug, N: no notification, R: Reboot S: message handling in violation of specification.*

			fuzzed		
Brand	Type	Firmware/OS	SMS	CBS	Result
Apple	iPhone 4	iOS 4.3.3	yes	no	I,D
Blackberry	9700	BB OS 5.0.0.743	yes	yes	I,S
HTC	Legend	Android 2.2	yes	no	I,D
Nokia	1100	6.64	yes	no	I
Nokia	1600	RH-64 v6.90	no	yes	S
Nokia	2600	4.42	yes	no	I,M,R
Nokia	3310	5.57	yes	yes	I,S
Nokia	3410	5.06	yes	no	I
Nokia	6610	4.18	yes	no	I,N,R
Nokia	6610	4.74	yes	no	I,N,R
Nokia	7650	4.36	yes	no	I,R
Nokia	E70-1	3.0633.09.04	yes	no	I
Nokia	E71-1	110.07.127	yes	no	I
Samsung	SGH-A800	A80XAVK3	yes	no	I,N,R
Samsung	SGH-D500	D500CEED2	yes	no	I,M,R
Samsung	Galaxy S	Android 2.2.1	yes	no	I
Samsung	Galaxy Note	Android 4.1.2	no	yes	S
Sony Ericsson	T630	R7A011	yes	no	I,N

ing we simply used most functions of the tested phones after a batch of fuzzed messages, since there is no acknowledgement of received CBS messages.

Table 6.1 shows the make and models of the phones we used during the fuzzing research. During the research it turned out that many phones did not support the CBS features, so the test set for CBS fuzzing was small.

6.4.2 Fuzzing results

We now give an overview of some of the most interesting results we found during our fuzzing research, which are also summarised in Table 6.1. For a complete overview of the exact fuzzing performed and the obtained results we refer to the Master's theses on fuzzing SMS [97] and CBS [27] on which this chapter is based.

SMS Fuzzing

All tested phones accepted some rarely used SMS variants, such as Fax-over-SMS, which causes strange icons to appear to notify the user of a new message (e.g. a new fax). These SMS variants are so obscure that often the GUI of these mobile phones offered no way for the user to remove these icons, only a message from the network could remove them.

More serious issues were that for five out of sixteen phones we found SMS messages that are received and stored by the phones without any notification to the user.

Figure 6.6: *These two pictures show the strange behaviour when the same SMS is opened twice in a row. Note that the words in the left image are the names of games available on the device.*

This enables attacks of filling up the SMS memory remotely, which causes phones to silently reject new incoming messages with an error message to the network, but all phones notify the user of a full SMS memory. In addition, seven out of the sixteen tested phones could be forced into a reboot with a single SMS message, though each through a different SMS message.

The Nokia 2600 showed strange behaviour where a particular SMS message would display random parts of the phone memory when opened, instead of the SMS message. This behaviour is shown in Figure 6.6.

Both the iPhone 4 and the HTC Legend could be forced in a DoS state were they silently received a message and afterwards could no longer receive any SMS messages, without any notification to the user. Rebooting these phones or roaming to a different network would stop the DoS.

Strangely enough we found no real correlation between specific harmful messages and phone brands. So, a message triggering a reboot in a specific Nokia phone, would have no effect on all other Nokia phones. This is likely due to the large variety in phones as explained in Section 6.3.1

CBS Fuzzing

Our CBS fuzzing research did not reveal any obvious errors such as spontaneous phone reboots. One of the main problems here is that we had no way to tell whether an ignored CBS message was not received by the baseband stack, or that the phone OS did not know how to display it.

The Galaxy Note displayed a fuzzed CBS message which according to the specifications should have been ignored. According to the GSM specification, mobile phones should only receive CBS messages containing Message Identifiers registered in their memory or SIM card. In our initial tests we used the Message Identifier value of 0 and did not register this topic number in the mobile phones. All mobile phones except for the Blackberry received the CBS message. In addition, once we changed the Message Identifier to a value different from 0, all mobile stations did not receive the CBS messages even though this time we did register the topic in the mobile phones.

So we observed that most phones have a lot of trouble to show even correct CBS messages. Since several countries are clearly pushing to get the CBS messages re-supported by phone manufacturers, CBS fuzzing tests should definitely be repeated when wider support is provided.

6.4.3 Related work

One of the most well-known bugs found in SMS implementations is the "Curse of Silence" found by Thomas Engel, though it is not directly clear if he used any systematic way, such as fuzzing, to find the vulnerability. With this bug certain Nokia phones stopped receiving SMS messages after receiving an email as SMS message[4] with a sender's email address longer than 32 characters [44].

The most prolific academic researcher in the fuzzing of GSM phones is Collin Mulliner [127, 126, 125, 124]. In 2006 he fuzzed the Multimedia Messaging Service (MMS) feature of GSM [127]. MMS is an extension to SMS for the exchange of multimedia content. When an MMS message is sent the recipient receives an SMS message with a Uniform Resource Identifier (URI) to a server where the MMS content can be retrieved using the Wireless Application Protocol (WAP). Of the three delivery methods discussed op Page 97 Mulliner et al. used the first method by building a virtual (malicious) MMS server using open source software and retrieving content from it on different cell phones. They found several weaknesses in various implementations, including buffer overflows in the Synchronised Multimedia Integration Language (SMIL) parser, the part that takes care of the presentation of the content on the cell phone to the user. Some of these buffer overflows could be used for arbitrary code execution. Mulliner together with Charlie Miller fuzzed SMS messages on smart phones [126, 125] using the second method of transmission. The three smart phones available for this research were an iPhone, an Android Phone and a Windows Phone. An application was developed for each of the three platforms, which makes it possible to directly generate and inject SMS messages into the phones' modems. Through this application, the researchers were able to make the device believe that an SMS was just received from the GSM network. Finally, Mulliner and Golde fuzzed the SMS implementation on feature phones [124]. This time they used a rogue cell tower based on OpenBSC, so they used the third method of message transmission. Furthermore they used a J2ME3 application for monitoring on the cell phones. Despite the spectacular title of this publication ("SMS of Death") there was no hard evidence that a fuzzed message caused the death of a phone, since this test was not repeated. The researchers did find DoS attacks for six different popular feature phone brands. They formed SMS messages that can even be sent over commercial (real) networks and will cause the phones to reboot, temporarily losing network connectivity. After consultation with the phone manufacturers Mulliner and Golde did not publish the actual messages that cause the DoS.

The company Codenomicon released a white paper detailing a product that fuzzes SMS messages in order to test the whole network chain for delivery of fuzzed SMS messages [184]. This is targeted towards providers as a tool that can be connected inside the core GSM network.

[4] Simply this option of receiving email over SMS is a good illustration of how baroque the SMS standard is!

It is often hard to find out exactly which fields were fuzzed in the studies discussed so far. We have attempted to provide an overview of the fields that have been fuzzed in the SMS-DELIVER message, as far as we can tell from these publications and sometimes through personal communication with the authors. This overview can be found in Figure 6.3(a). We chose to limit this overview to the one message we also fuzz in our research.

Naturally, fuzzing is not the only way to reveal (security) bugs in the GSM baseband stack. Weinmann et al. decompiled baseband firmware updates from two popular baseband chips and performed a manual code inspection which led to several bugs, amongst which one which led to remote code execution [187, 186].

Few researchers have access to the GSM core network. One private security company specialises in fuzz testing GSM core networks and they apparently have a database of possible attacks [141], though the nature of the found vulnerabilities is not public knowledge.

6.5 Conclusions

We have demonstrated that fuzzing is a simpel technique that quickly finds a large variety of bugs in the implementation of GSM stacks on mobile phones. This seems to indicate that manufacturers did not incorporate fuzz testing in their development cycle.[5]

The attacks we found can easily be initiated by an attacker. Some could even be transmitted over the genuine mobile network, as the weaknesses where inside the payload of SMS messages. Providers could, and should, scan for these attacks, and at least some do [184, 141]. We never attempted to transmit these messages over a live network, so maybe our attacks would already be intercepted by the network. Another way to initiate these attacks is by impersonating a fake base station. Setting up a fake base station and sending out malicious messages is, at least for GSM, not that hard nor expensive anymore and the potential damage could be enormous.

The wide diversity of phones makes it harder to find a single bug affecting many different mobile phones. Nevertheless, our fuzzing research in GSM has shown several issues with mobile phones. The most important attacks here led to various types of DoS messages which can usually be solved through a reboot of the phone. Some results show clear buffer overflow errors, such as the SMS message which will show random parts of the phone's memory when read on the Nokia 2600. Although it is not immediately clear how to abuse such an error for remote code execution, it is possible that such an attack will be constructed in the future for a popular brand of mobile phones. Unfortunately, the CBS service seems to be too poorly supported at the moment to achieve any meaningful fuzzing results, or to use it as an emergency broadcast service for that matter.

The hardest part of fuzzing mobile phone implementations is observing the phone's behaviour, which is hard to automate. There are not a lot of other options, other than human testing, for security analysis of the closed source baseband stacks on mobile phones. Then again, with direct access to the baseband stacks fuzz testing these im-

[5] About a year after this research, it became clear that Qualcomm, one of the leading manufacturers of mobile protocol chips, had only recently started incorporating fuzz testing [188].

plementations would be much easier. The manufacturers of the baseband stacks or of the SoC have this access and employing strong security tests on their products could greatly increase the security of their product, which among baseband stacks would be a novel selling point. For future research it would be interesting to focus on fuzzing rooted Android devices, where it may be possible to run debuggers in the memory to better observe strange behaviour.

For now almost all fuzzing research into GSM has focused on fuzzing mobile phones, and then mostly on fuzzing SMS messages, which still leaves many areas open to explore, such as all the other broadcast messages, but also the network side of a GSM network. It seems logical to assume that the baseband stack on network equipment will contain as many bugs as the stacks on mobile phones, and attacks against the GSM network itself would probably have a much larger impact.

Since the 3G and 4G protocols all have mutual authentication, it is not possible to simply deploy a fake base station in order to fuzz the 3G and 4G baseband stacks. A way around this obstacle would be to use self controlled SIM cards, so you can have your cell tower authenticate to the mobile phone. However, as far as we know there is no open source 3G or 4G cell tower software available yet, so this would require a large amount of work to implement.

Most of our effort came from getting the open source GSM base station up and running. After that implementing the fuzzers was only a few weeks of work. The initial effort to set up a base station and incorporate a fuzzer was substantial, but this solution can now be used to fuzz test any GSM phone on SMS or CBS weaknesses in one and a half hour. This makes fuzzing a cost-effective and feasible technique for making implementations of mobile phone stacks more robust and safe.

Responsible Disclosure

We informed the phone manufacturers of the vulnerabilities we found. While most vulnerabilities are likely caused by the baseband chips it was not always possible to find out who the respective baseband manufacturer was.

We did not disclose the found vulnerabilities, as it is highly unlikely that the devices will be patched against these vulnerabilities.

Chapter 7

Mobile phones as second factor in authentication

Until now we only looked at the security of the mobile phone networks themselves. However, there are also many applications running on top of these networks. For instance, a lot of machine-to-machine communication takes place over mobile networks, such as in certain traffic lights, or smart meters. There are also many user applications using the mobile telephony network to communicate. Especially following the success of smart phones, these applications can offer lots of functionality to users.

Now that users can be assumed to have a mobile internet connection all the time, more and more services are offered over the internet. This necessitates the use of strong authentication of users over a distance for security sensitive services; another thing in which mobile phones see a lot of use. An attacker might not be interested in attacks on the mobile networks per se, but rather in attacking this extended use of the mobile phone. This chapter explores the ways in which mobile phones can be used for remote authentication and the, sometimes very security sensitive, applications deployed on top of the mobile telephony network. We first look at mobile phones used as an additional factor in multi-factor authentication and then look at a particular service offering server-based signatures, which also uses the mobile phone as an additional factor during authentication.

This chapter is based on the article *Digitale handtekeningen: nieuwe technologie en nieuwe wet- en regelgeving* which was published in February 2014 in the Dutch journal Privacy & Informatie [177]. The original publication was more focused on legislation and regulations considering electronic signatures. The current chapter is an English version of the publication which takes a broader perspective, including an analysis of multi-factor authentication techniques using mobile phones. Section 7.2 has been added to the original publication. The security assessment of SMS in Section 7.2.2 is based on joint work between the Digital security group of the Radboud University and Price Waterhouse Coopers on a risk assessment for using SMS authentication to secure access to online medical records [144]. This risk assessment was commissioned by the Ministry of Health and led to the cancelling of this online access feature.

7.1 Introduction

More and more services are being offered over the internet. With this transition the need to authenticate users remotely is also growing. With consumer services one popular solution is the use of mobile phones as, at least, an additional factor in remote authentication to the common username and password combination. However, use of these services is no longer limited to traditional PCs, but is moving to mobile devices. When the mobile device itself is used to access a service, it can no longer be an *additional* factor in authentication. Services offered over the internet can be very security sensitive, such as in mobile banking applications, making strong remote authentication important. In 2013 the European Central Bank (ECB) published a set of recommendations for online payments, strongly focusing on authentication [50]. Similar recommendations are being prepared for mobile payments [51].

First we will take a closer look at the remote authentication problem in Section 7.2. In this section we also discuss the current recommendations of the European Central Bank (ECB) concerning strong authentication, as well as look at several popular approaches for using the mobile phones as an additional factor for authentication. Section 7.3 presents a background section into digital signatures and also covers the legal framework for electronic signatures. Signatures themselves of course require authentication, but this section serves mostly as background information for a case-study presented in Section 7.4 where we discuss a server-based signature service. In this service a user's private key is stored in the cloud and the user's mobile phone is used as an additional factor in authentication. We will then draw conclusions in Section 7.6.

7.2 On (remote) authentication

Authentication is a fundamental issue in computer security. Often it is solved with the use of passwords. Authentication can be a local process where a user authenticates himself to (a program on a) computer he has physical access to. With internet services users have to be able to authenticate themselves from their own PC to a remote server: remote authentication.

Remote authentication is more vulnerable to attacks than traditional authentication, not just because authentication parameters are exchanged over an untrusted connection (usually remedied by a secure pipe over the connection), but also because the attacker does not need to be in physical proximity of the authenticating party, making attacks easier to perform. Furthermore, remote authentication may be vulnerable to Man-in-the-Middle attacks, especially since these attacks can be employed at the user end point of a secured connection (e.g. in the case of a Man-in-the-Browser attack [91]).

Authentication is usually meant to define a process of proving or verifying one's identity. But apart from this notion of *user authentication*, we also consider the notion of *transaction authentication*, where we not only authenticate a user, but also check that user's intent in agreeing to some transaction, e.g. an online bank transfer. Transaction authentication is a form of non-repudiation, and typically subsumes a form of user authentication.

In remote multi-factor authentication, the user will usually have a device as an additional factor. In order to restrict the use of this device to a specific user, such a device will often authenticate the user, before performing the authentication calculations as the additional factor in remote authentication. This separate user authentication step is in itself an (assumed) part of the remote authentication. To prevent confusion, we will refer to this authentication step as *local authentication*. Also, we use the term "authentication," to refer to both user and transaction authentication.

7.2.1 ECB's notion of strong authentication

In January of 2013 the European Central Bank (ECB) published its' recommendations for the security of internet payments [50]. These recommendations are the first result of the European Forum on the Security of Retail Payments (SecuRe Pay), a voluntary cooperation initiative including most European central banking authorities. The members of SecuRe Pay have committed to supporting the implementation of the recommendations in their respective jurisdictions and will integrate them in existing supervisory/oversight frameworks. The recommendations should be implemented by payment service providers (PSPs) and Governance Authorities (GAs) of payment schemes by 1 February 2015.

The recommendations provide interesting key considerations and best practises for online authentication. In particular, it defines a notion of *strong user authentication*, which is defined as: "a procedure based on the use of two or more of the following elements:"

1. knowledge, something the user knows, such as a PIN;
2. ownership, something the user has, such as a smart card;
3. inherence, something the user is, e.g. biometric characteristics, such as a finger print.

This definition seems a pretty straight forward definition of multi-factor authentication, but it adds additional demands: "the elements selected must be mutually independent, i.e. the breach of one does not compromise the other(s). At least one of the elements should be non-reusable and non-replicable (except for inherence), and not capable of being surreptitiously stolen via the internet. The strong authentication procedure should be designed in such a way as to protect the confidentiality of the authentication data."

This notion of strong authentication is only defined for online banking and has no legal status (yet). It does provide a starting point for strong authentication that we can use for other authentication problems besides online banking.

Regrettably, the ECB's notion of strong user authentication only concerns authentication of the user, not authentication of transactions carried out by that user. A notion of transaction authentication is only considered in Best Practice 7.3 in the ECB recommendations as an optional extra: "Strong user authentication *could*[our emphasis] include elements linking the authentication to a specific amount or payee."

There is also some haziness in the ECB's definition where it states that at least one element should not be capable of being surreptitiously stolen via the internet. Exactly

which attacks should be protected against by this remark? Simple eavesdropping of HTTP traffic? Or also attacks that use fake bank-sites or even Man-in-the-Browser attacks?

7.2.2 Multi-factor authentication using mobile phones

Multi-factor authentication is an often used technique to add security to authentication. Often "Something the user has" is used as a second factor in authentication, particularly for remote authentication. This "something" the user has can be a hardware token, such as a smart card, an RSA token, or a passport. Especially with consumer services, the mobile phone is a popular device to use for this. Naturally, this assumes the user does not already use the phone itself to access the service he wants to authenticate to, since then it would no longer count as a second factor.

Mobile phones are a convenient choice for an additional authentication factor since practically all users carry a phone anyway. This both saves the costs of creating dedicated tokens and saves users from having to take care of an additional device. Furthermore, because of the many uses a mobile phone offers, users (or at least smart phone users) typically check their phones 150 times a day [119], which results in users noticing the absence of their phones (e.g. through loss or theft) much faster than other hardware tokens. Also, phones have their own connections to the mobile network or the internet. So when using a computer for remote authentication, using a mobile phone as an additional factor automatically provides a separate path to the user. If, for instance, the computer is infected with malware, than this separate path to the user would remain uninfluenced by the malware.

Let's take a closer look at some solutions for multi-factor authentication using mobile phones: SMS mTAN, Google Authenticator, the Twitter app, Mobile PKI, tiqr and Cronto. In this discussion we assume that a user logs in to a server using a computer and uses the provided authentication methods on a phone as an additional authentication step.

SMS mTAN / OTP

A TAN, Transaction Authentication Number, is a code used to authenticate a transaction. Originally, some banks gave lists of TANs to customers which could be used as one-time passwords, to authenticate themselves or to authorise transactions. Several banks now use TAN codes transmitted to the user's mobile phone via SMS for authenticating online banking transactions. These codes are often referred to as "mobile TAN" (SMS mTAN), or as SMS One-Time Password (SMS-OTP) [123]. Note that these two names allude to the two different forms of authentication we distinguish, namely transaction authentication for TAN and user authentication for OTP. With SMS mTAN/OTP the user receives a code via SMS message and enters this code unaltered into his computer (the challenge and response are equal).

SMS mTAN is usable on practically every mobile phone since it only needs a working subscription for the mobile network to receive SMS messages. This option does not store secrets on the phone and the message size offered by SMS provides the possibility to give the user additional information besides the mTAN, such as the receiving account number and amount of money in a banking transaction. This allows a

user to validate the transaction he is authenticating through an independent channel, which is also called out-of-band verification.

A downside to SMS mTAN are the costs involved; every SMS / authentication costs money. Also, this approach is vulnerable to any attack which intercepts the SMS. There are several attacks to intercept SMS messages:

1. Eavesdropping the wireless link (as detailed in Chapter 3),

2. Malware on the phone which intercepts the message and forwards it to the attacker [73, 151], and:

3. SIM-fraud, where an attacker manages to obtain a new SIM card belonging to the phone number of the victim.

In each of these attacks an attacker obtains the TAN code and could use this to authenticate to the service assuming the attacker can already circumvent the other factors involved in authentication [123, 144]. In the first attack, eavesdropping, the victim will also receive the SMS message, which should alarm him if he did not request the message. Alternatively, an attacker could use an active Man-in-the-Middle attack between the cell tower and the victim's mobile phone and simply not transmitting the mTAN on to the victim (Section 3.2.2). In the other two attacks the SMS messages can remain hidden from the user, but in the third attack all telco services to the victim will be discontinued when the new SIM card is activated, which also makes this attack detectable. The malware attack is the attack which scales the best when performed on many targets, since the eavesdropping attack requires the attacker to be in the vicinity of the victim when performing the attack and the SIM-fraud attack requires the attacker to obtain a new SIM card from the victim's provider.[1]

Finally, this authentication method is also vulnerable to a Man-in-the-Middle attack, where the user thinks he is performing a correct authentication, but the attacker modifies or redirects the authentication messages in order to authenticate on behalf of the victim. All the phone-based additional-factor authentication techniques discussed here are vulnerable to such an attack, though some, like the SMS TAN version, still have a benefit because of the out-of-band verification information that can be added in the SMS message. A user can use this information to verify he is performing the expected transaction [144].

Google Authenticator

Google offers a two-factor authentication scheme for its services [87]. First a user authenticates using a user name and password combination, then the users provides an OTP generated on a smart phone app. This smart phone app, called the Google authenticator, is a personalised app containing a secret shared with the Google backend. The app generates a new code every 30 seconds (each code remains valid for one minute), so the challenge consists of the current time and is never transmitted. This

[1] From personal experience we found that some banks have started protecting against this attack through an agreement with the providers. The providers alert the bank when a new SIM card gets activated for a mobile number used for remote authentication to the bank. The bank then simply disallows any transaction from that phone number for a certain period, such as 48 hours.

is also called Time-based One-time Passwords (TOTP) [35]. The response is supplied by the user via the computer.

This time-based authenticator app runs on most smart phones and does not cost any money per authentication. It stores a secret within the app memory, which is not considered very secure. This secret is stored during the personalisation phase, by having the phone scan a QR code which contains the secret.

This additional authentication scheme is vulnerable to phone-side malware which either retrieves the stored secret or forwards authentication codes. If the stored secret ever gets retrieved by an attacker, while the victim remains owner of the phone, then the attacker can defeat the additional authentication step without the victim noticing this. As opposed to SMS mTAN, the Google Authenticator offers no out-of-band channel through which a user can verify the transaction details he authenticates. This makes this authentication method extra vulnerable to a Man-in-the-Middle attack on the victim's computer.

Twitter app

The Twitter app recently added a two-factor authentication option for users signing in via a PC [156]. The Twitter app itself also gives access to a user's account, in which case it no longer counts as a second factor, but here we are interested in the use case of a user using the app solely as a second factor, while authenticating over a PC.

When a user enrols for this two-factor authentication, his app generates an RSA key pair en sends its public key to the Twitter server. Now, when a user tries to log-in to Twitter, the Twitter server will transmit log-in details (time, location and browser information) along with a random challenge to the Twitter app on the user's phone. The app will notify the user of a log-in attempt, who can then approve or deny the action. Upon approving, the app will sign the challenge with its private key and send this back to the Twitter server as a response. After verifying the response the Twitter server allows the user's PC to log-in.

This multi-factor authenticator runs on most smart phones, but will need an active data connection at the time of authentication. This authentication could cost the user money to transmit over a data network, especially when roaming abroad, though the amount of data transmitted is very small.

The app also provides the possibility to generate back-up codes for when the phone does not have a data connection. This is based on the S/KEY password system [93], which in turn is a concrete implementation of Lamport's One-Time Password scheme [114], where the app generates a secret and hashes this n times (in case of the Twitter app n starts out at 10000) and sends this to the Twitter server during enrolment. Back-up codes are generated by hashing the secret $n-1$ times and displaying this to the user. The user can then type this code in his browser during authentication. The Twitter server then hashes the code once and verifies that it equals the value it stored during enrolment. Subsequently the server stores the $n-1$ times hashed value and the app will generate an $n-2$ times hashed value the next time, and so on. Effectively, this provides an OTP.

Both solutions were designed with the additional goal of the back-end server only being able to verify the authentication responses coming from the Twitter app, but not being able to generate said authentication responses itself. So, in the event of

a breach of the back-end server (or at least the database containing the user log-in parameters) an attacker would still not be able to generate valid log-in responses on behalf of users [156]. Although this is an interesting feature, it does raise the question whether an attacker gaining access to the back-end of Twitter would not be able to then impersonate a user by other means, e.g. by changing the stored user log-in parameters to his own?

Both these solutions do have a secret stored in the app memory, which is considered vulnerable. However, the benefit compared to the Google authenticator is that the Twitter server does not store the secret. Both of these Twitter app solutions are vulnerable to mobile malware, which can either retrieve the secrets or forward authentication codes. Again, malware on the PC is also a vulnerability, although the standard public key method of the Twitter app provides the user with out-of-band information which he can use to verify he is logging in to the correct site.

Mobile PKI

Mobile PKI is an interesting variant where secrets are stored on the SIM[2] card [137]. It also makes an explicit separation between user authentication and transaction authentication. As the name suggests, this method relies on public key cryptography, where the SIM stores the user's private key.

Mobile PKI uses the so-called SIM toolkit, an optional application on SIM cards with extensive additional functionality compared to normal SIM card applications. When supported by the mobile phone, the SIM Toolkit can perform actions such as sending SMS messages, or modify numbers called by the user. SIM Toolkit also allows for direct communication between provider and SIM card through service SMS messages, which are regular SMS messages with additional options set in the header indicating that the payload of these SMS messages are to be delivered directly to the SIM card, without notifying the user of the reception. These service messages are used in mobile PKI, where a message contains two data fields, one containing the data to be signed and one containing data to be displayed to the user. After reviewing the data displayed, the user can instruct the SIM card to proceed with signing. This will require an additional local authentication to the SIM card using a PIN code. When successful, the SIM card will sign the appropriate data and send it back to the server using service SMS messages.

Mobile PKI is the most costly of the techniques discussed in this section, since it both requires SIM cards that support Mobile PKI, which are not commonly provided to users, and requires SMS messages (both transmitting and receiving) per authentication. The use of the SIM card also involves an organisational hurdle, since the SIM card is under the sole control of a provider. This means that you both need the permission and cooperation of the provider, as well as have the required trust in the provider.

While it is possible to eavesdrop on the wireless transmission of the service SMS, this does not matter for the Mobile PKI solution. Not only can the contents of this message have extra encryption, the contents of these messages are useless to an attacker without access to the secrets stored in the SIM. After all, under the assumption

[2]We use the term SIM card, as this is the more commonly used term, though most modern phones actually have a UICC with a (U)SIM application.

that challenges are not re-used, simply sniffing the challenge sent to the SIM card does not help an attacker and sniffing the response means the user already agreed to the authentication. The storage of secrets in the SIM is a huge security benefit compared to the other approaches since SIMs offer protected storage, which is not (yet) available in the phone memory.[3]

This approach is susceptible to malware attacks on the phone. Although the specifications of SIM Toolkit require a phone to give sole control over the keyboard and screen to the SIM card, when input/output is exchanged between user and SIM Toolkit [54], this requirement is most likely no longer supported in modern smart phones, where the OS is needed to provide the user with a keyboard.

This authentication method is vulnerable to a Man-in-the-Middle attack on the victim's computer, although like in the SMS mTAN case, the added transaction details that can be shown to the user, could mitigate these vulnerabilities.

tiqr

Tiqr is an authentication method developed by SurfNet [180]. Again, it involves an app running on a smart phone, which performs a challenge/response protocol. In this case the challenge arrives via the PC; it is presented in the form of a QR code on the computer screen, which the user needs to scan using the tiqr app and his smart phone's camera. The user is then presented with the domain name of the website of the log-in attempt, and an option to continue with the authentication. When the user chooses to continue, the response is computed using a shared secret and then transmitted over a wireless data connection of the phone when available. If no data connection is available, the tiqr app will show a response on the screen, which the user can type into the PC.

This approach requires a smart phone with a camera. The response could cost the user money to transmit over a data network, though the amount of data transmitted is very small. Otherwise, this approach does not cost any additional money per authentication and is considered relatively user-friendly, as the online case does not require the user to copy either the challenge or response by manual typing.

Tiqr, like the Google authenticator, stores a secret in the phone memory, making this approach vulnerable to attacks that retrieve the secret from the phone's memory. Tiqr is also vulnerable to phone-side malware which generates and forwards authentication codes.

This authentication method is also vulnerable to a Man-in-the-Middle attack on the victim's computer, though again the shown details (domain name) offer some protection.

Cronto

Cronto is a smart phone app by Vasco, specifically intended for banks [182]. Cronto also uses scanning of a QR code[4] as a means of getting the challenge into the phone application. The difference with standard tiqr is that the Cronto app generates a TAN code displayed to the user, who can then enter this code on his computer. So, Cronto is

[3] Trusted Execution Environments (TEEs) might provide secure storage on mobile phones [181].

[4] In this case a coloured QR code in stead of the standard black-and-white QR code.

very similar to tiqr without a data connection, but with one other important difference: Cronto also shows transaction details to the user.

The Cronto solution does not use any additional channel afforded by the mobile phone.[5] Again, as with all the previous smartphone app approaches, the Cronto app stores a secret in the phone memory, making this approach vulnerable to attacks that retrieve the secret from the phone's memory.

This authentication method can be vulnerable to a Man-in-the-Middle attack on the victim's computer, although the added transaction details could mitigate this vulnerability.

Comparison

Comparing the discussed alternatives, we see that they all use different channels for receiving the challenge and/or transmitting the response and where or if they store a secret. Figure 7.1 summarises these differences by showing the different information flows after a user started the authentication procedure, e.g. by providing username and password.

Three of these approaches use the separate channel offered by the use of a mobile phone for receiving the challenge. In the case of the Google authenticator the challenge is public knowledge (time) and tiqr receives its challenge over the PC used for the authentication. Tiqr uses the separate channel for transmitting the response, just as Mobile PKI and the on-line Twitter app, while the other two remaining approaches use the PC connection for transmitting the response. The separate channel helps to assure not all communication can be influenced from the PC.

Mobile PKI stores its secret in the most secure way. SMS mTAN does not involve a secret, but this results in a severe weakness, since this means the challenge and response are the same and thus the challenge needs to remain confidential. This is something the SMS channel cannot provide strong guarantees for.

Interestingly, the mobile PKI and SMS mTAN/OTP both explicitly recognise user authentication and transaction authentication, while Google Authenticator, the Twitter app and tiqr seem mainly focused on user authentication and Cronto specifically targets transaction authentication.

Of these approaches Mobile PKI is by far the most secure, though the costs and necessary involvement of the provider can be a huge downside. The online version of Twitter authentication offers essentially the same scheme as Mobile PKI, only with the secrets stored in the phone memory instead of in the SIM. This removes most of the downsides with Mobile PKI, but also its greatest strength, since the secrets are no longer stored in trusted hardware. SMS mTAN seems the weakest solution as it has an extra weakness the other approaches do not share: interception of the SMS challenge. This problem could be mitigated by storing a secret on the phone, so the incoming SMS challenge can be used for a true challenge response protocol, but then again the storage of the secret would introduce weaknesses. We discussed intercepting SMS messages on the air interface in Chapter 3. Additionally, SMS messages can be intercepted on the phone itself using malware, but most other authentication methods are also vulnerable to phone malware.

[5] A variant of the Cronto mobile phone app involves a special hardware token, with a camera and a display and two buttons, which functions exactly as the Cronto app, but on dedicated hardware.

All approaches are still vulnerable to malware on the PC that fools a user into authenticating for something else than what he thinks. However, several of these approaches use the possibility to add extra (out-of-band) data showing the user authentication details pertaining to the actual authentication being performed. The online Twitter app and tiqr (both, online and offline) both show additional user authentication information, while Cronto shows additional transaction authentication information. SMS mTAN and Mobile PKI can provide both details in case of user and transaction authentication, while Google Authenticator and the Twitter app in offline mode provide neither. This feature can be very powerful in preventing MitM attacks, and is arguably more important when approving transactions than during log-in attempts. Bank transfers are a good example of a scenario where it is relatively easy to give transaction details to the user. However, as we will see in the case-study presented in Section 7.4, for more complicated transactions, such as the signing of documents, this is not so easy.

According to the ECB definition

We can see how the different second factor authentication solutions hold up against the ECB definition of strong authentication. We still assume users authenticate over a PC, using user name and password as the primary authentication factor. This means the mobile phone solution is the second factor and is mutually independent of the primary factor. Furthermore, we assume the challenge transmitted to the mobile phone is never re-used. So, in order to comply with the ECB's notion of strong security, the second factor techniques need to (I) not be able to be surreptitiously stolen via the internet and (II) should protect the confidentiality of the authentication data.

We already discussed how the requirement of "not being able to be surreptitiously stolen via the internet" is problematically vague. However, we can also argue that the requirement of "confidentiality of the authentication data" already subsumes this vague requirement.

For most techniques, this requirement is highly dependant on the implementation of the authentication process on the primary channel, in our case a PC. Only the Mobile PKI solution offers the possibility of confidentiality of its authentication data, within its own solution. The Google Authenticator uses the current time, and as such part of the authentication data can not be considered confidential.

All techniques where the response is transmitted via the user's PC are vulnerable to Man-in-the-Browser attacks, whereby the authentication response can leak. SMS mTAN/OTP has an additional problem since the transaction data could also be stolen from the SMS transfer. Because with SMS mTAN the challenge and response are equal, any confidentiality leak of the challenge immediately presents a critical vulnerability. This also shows an additional problem with the ECB recommendation, as the channel of which the factor gets surreptitiously stolen presumably does not matter when evaluating strong authentication.

7.3 Digital signatures

This section will look into the practice of digital signatures and the legal framework around the electronic signatures. This serves mostly as a background for the inter-

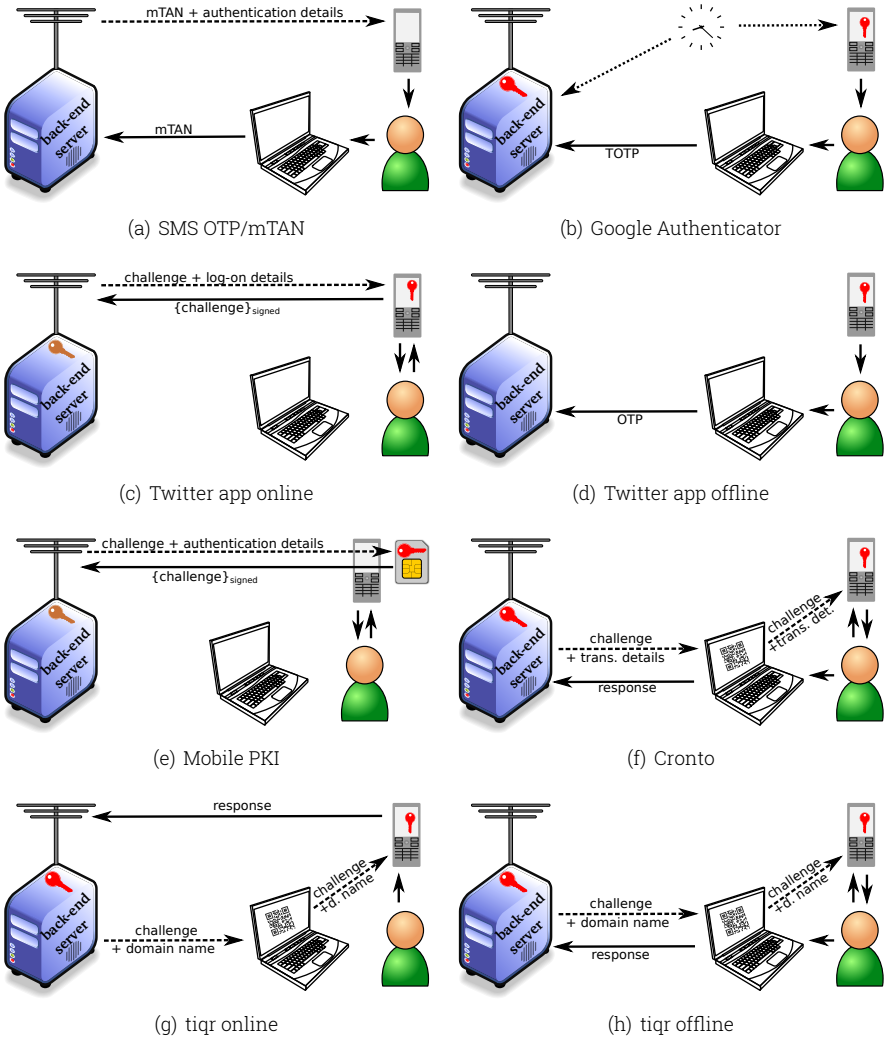


Figure 7.1: *Schematic representation of information flows and stored secrets with different 2nd factor authentication solutions. Challenges are denoted with dotted arrows and responses are denoted with a straight arrow. Keys of the same colour represent symmetric keys and keys of different colours represent asymmetric keys.*

ested reader, introducing the particularities concerning electronic signatures before looking into a server-based service for digital signatures in Section 7.4. Most of the information in this section became outdated on July 1, 2016, when the eIDAS regulation went into effect. A short section (7.3.3) was added here to to describe the biggest change, with respect to this chapter, introduced by the eIDAS regulation. Since the rest of this chapter mostly considers the situation prior to July 1, 2016, the information here is left as-is.

For centuries now handwritten signatures on a document provide both a prove of

identity and a proof of intent from the signer. So, we see a signature combining both user authentication and transaction authentication. Although the actual security of a handwritten signature is debatable, most countries give strong legal status to signatures, and signatures are often required for legal documents, such as contracts and official written statements. Also, legal systems usually have procedures in place in case of a contested signature.

In the digital world we also have digital signatures. Most digital signature schemes are built on a public-private key pair, where the signer makes a hash of the message to sign, encrypts it with his private key, and attaches the result to the message. The validity of the digital signature can than be verified using the public key, which is usually bound to the identity of the signer via a public key infrastructure (PKI).

'Digital signature' is a somewhat confusing name, as there are several important differences between hand-written signatures and their digital homonym, besides the obvious difference in mediums on which they are used.

A difference is that digital signatures are a more widely used concept in the digital world, than hand-written signatures are in the physical world. Completely automated messages in a protocol, for instance, are regularly signed, messages without a natural or legal entity as the signer, in contrast to signed documents in the physical world.

There is a related difference in the extended functionality offered by digital signatures, compared to hand-written signatures. Digital signatures provide integrity of the document, which hand-written signatures do not.

A final difference between handwritten and digital signatures is that we only need a pen and ink for placing a handwritten signature. For digital signatures though, we need some kind of computing device to generate a signature and a computing device to verify a digital signature, as this involves mathematics that most people can not perform by themselves. Also a device is required that stores the private key, this can be the same device that generates the signature or a separate device. This device will need to provide some local (user) authentication to control access to the signing functionality it provides. Depending on the strength of the local authentication, the loss of this device could completely invalidate the security offered by the digital signatures. In this sense a digital signature is more the digital equivalent of a seal (placed on a see-through envelope holding a document), since a seal also protects integrity and is made with a device instead of by the signer.

So, whereas hand-written signatures are always used to prove intent of the signer, digital signatures are mostly used to achieve integrity protection and sender authentication. For the specific form of digital signatures that are meant to fulfil the same function as hand-written signatures the legal term is *electronic signatures*.

7.3.1 Electronic signatures – Legal framework up to July 2016

Many countries now recognise electronic signatures. The European Directive on a Community framework for Electronic Signatures, Directive 99/93/EG [165] was passed through the European Council and Parliament and went in effect on the 13th of December 1999. This European directive handles the legal status of electronic signatures in commerce and on legal documents and proceedings. It can be said to distinguish three types of signatures. These are, in order of reliability:

1. electronic signatures
2. advanced electronic signatures
3. qualified electronic signatures

Electronic signatures are defined in article 1 of the European directive as: "data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication". So, an electronic signature can be anything from a digital signature to a scan of a handwritten signature or even a typed name on a digital document, and the reliability of an electronic signature can be very poor.

For an electronic signature to be considered as an *advanced electronic signature* it has to meet the requirements listed in article 2 of the European directive:

1. it is uniquely linked to the signatory;
2. it is capable of identifying the signatory;
3. it is created using means that the signatory can maintain under his sole control[6]; and
4. it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable;

Where a *signatory* is "a person who holds a signature-creation device and acts either on his own behalf or on behalf of the natural or legal person or entity he represents". Basically, advanced electronic signatures are more trustworthy versions of electronic signatures. Digital signatures, such as PGP signed e-mail, can be advanced electronic signatures when they are created by means under the sole control of the signer (requirement 3).

Qualified electronic signatures are not explicitly named in the European directive (the name stems from ETSI standard TS 101 456), but it is the name commonly given to advanced electronic signatures that also meet the two additional demands of article 5.1 of the European directive: "electronic signatures which are based on a qualified certificate and which are created by a secure-signature-creation device". These are the highest level of electronic signatures and are considered to have the same legal power as hand-written signatures do. In fact, the Directive states that these qualified electronic signatures "(a) satisfy the legal requirements of a signature in relation to data in electronic form in the same manner as a handwritten signature satisfies those requirements in relation to paper-based data; and (b) are admissible as evidence in legal proceedings." The next two subsections will look into the demands set on the qualified certificates and secure signature-creation devices required for generating qualified signatures.

This European directive had to be implemented in the local law of the European member states by the 28th of December 2009. In the Netherlands this happened in

[6]The upcoming eIDAS standard [167] actually relaxes this requirement so a user can entrust a third party with his means to create a signature, as long as the user keeps "sole control over the use of his electronic signature creation data"

May of 2003 with the "Wet Elektronische Handtekening (WEH)", the law on electronic signatures, which incorporates several modifications of existing laws. The Dutch law does not name the different signatures (i.e. electronic, advanced and qualified), but states that any electronic signature made through a method that is considered "sufficiently reliable" is considered to have the same legal status as a hand-written signature. It then follows this by listing six demands, which when met, yield a signature which is assumed to be sufficiently reliable.[7] These six demands are the same six demands for qualified signatures in the European directive.

It is important to realise that these laws and directives do not mean that qualified electronic signatures are the only way a legally binding electronic signature can be made. There is a lot of leeway to accept other forms of electronic signatures, but in those cases it is necessary to make a risk analysis for the specific context, to see if the signature method provides enough guarantees on the identity of the signer and on the integrity of the document for the specific case. The main difference is that for a qualified electronic signature this assessment is not needed, as such a signature is considered sufficiently reliable by default and has to be accepted as a valid signature.

7.3.2 Qualified certificates

Digital certificates are essential for digital signatures, since they provide the link between a signature and the signer. As noted in the previous section, in order to create qualified signatures the signer needs a qualified certificate. Annex I of the European directive supplies demands on the contents of these qualified certificates, such as an indication that the certificate is issued as a qualified certificate and the name or pseudonym of the signer.

Qualified certificates are issued by Certification Service Providers (CSPs). In order for a CSP to be allowed to issue qualified certificates, the CSP should conform to requirements stated in Annex II of the European directive. These requirements come down to the CSP using techniques and procedures that satisfy the current standards for security and reliability and employing competent personnel. CSPs issuing qualified certificates are also liable for damages suffered by entities or natural or legal persons who reasonably relied on that certificate, unless the CSP can prove that he has not acted negligently.

All member states of the European Union have to have a supervisory body for supervision of CSPs established on its territory issuing qualified certificates. Furthermore, member states may introduce a voluntary accreditation scheme by which CSP can get accredited. Although this accreditation is voluntary, it seems the easiest way to show conformation to the requirements and once accredited in a member state, issued qualified signatures have to be accepted in all member states. These accreditation scheme's will usually seek compliance with the requirements for CSPs stated in ETSI TS 101 456 and with CWA 14167-1 and CWA 14167-2, as these are the standards for electronic signature products recognised by the European Commission in Commission Decision 2003/511/EC [166].

Matters get even more complex, as the organisations performing the CSP accreditation require periodic accreditation themselves (they should conform to the requirements stated in NEN EN 45011 and EN 45012).

[7] Article 3:15a, Clauses 1 and 2 of the Dutch Civil Code.

7.3.3 Secure signature-creation devices

Qualified electronic signatures are created using a Secure Signature Creation Device (SSCD). SSCDs need to confirm to requirements (Annex III of the European directive) :

1. Secure signature-creation devices must, by appropriate technical and procedural means, ensure at the least that:

   (a) the signature-creation-data used for signature generation can practically occur only once, and that their secrecy is reasonably assured;

   (b) the signature-creation-data used for signature generation cannot, with reasonable assurance, be derived and the signature is protected against forgery using currently available technology;

   (c) the signature-creation-data used for signature generation can be reliably protected by the legitimate signatory against the use of others.

2. Secure signature-creation devices must not alter the data to be signed or prevent such data from being presented to the signatory prior to the signature process.

These requirements are hard to fulfil in practice. Although software solutions as SSCDs are not explicitly excluded in the Directive, it seems impossible for software solutions to conform to these requirements with the current state of PC security.

The most well-known SSCDs are probably smart cards. They are used, for instance, in the Belgian and German e-ID cards, allowing citizens to sign communication with the government using the smart card in their ID document. Other solutions include USB-tokens and Hardware Security Modules (HSMs). All these solutions are dedicated hardware solutions with protected memory. They all have the same problems in that they need additional hardware in order to communicate to users. In the case of smart cards this can be an independent terminal with a screen and keypad, but also a terminal connected to a computer which in turn provides the screen and keyboard to the user. Naturally, the latter option is less secure than the former, due to possible malware corrupting the data flows. USB-tokens almost always require a computer for communication to the user. HSMs communicate over a network connection and are placed within a server domain and manage things like crypto keys, authentications and signing of automated messages, such as in DNS SEC. They are not generally used for signing personal digital signatures, though Section 7.4 discusses a setup wherein this is exactly their use.

Just as the certificate service providers, the SSCDs can also achieve accreditation. When an SSCD Conforms to the demands set in CEN Workshop Agreement 14169 (CWA 14169) it is assumed to fulfil the demands of the European directive stated above [166]. CWA 14169 defines a Protection Profile for a Common Criteria evaluation of SSCDs. Interestingly, CWA 14169 allows for the end user's computer to be placed out-of-scope of the evaluation, as evaluating every user's computer would clearly be infeasible. However, this also means the local authentication of the user to the SSCD is effectively removed from the scope of the evaluation on the assumption the local authentication is carried out over a trusted path, even though the use of the user's computer is clearly the weak spot for most SSCDs.

Electronic signatures – current legal framework

The new eIDAS (electronic identification and trust services for electronic transactions in the internal market) standard [167] repeals the European Directive, and went into effect on July 1, 2016. This loosens the sole control demand, by allowing a user to "entrust qualified electronic signature creation devices to the care of a third party, provided that appropriate mechanisms and procedures are implemented to ensure that the signatory has sole control over the use of his electronic signature creation data". This was specifically changed in the new regulation to allow for server-based signatures.

The eIDAS regulation remains vague on how *electronic signature creation data* can remain under the sole control of a user, when his SSCD is placed on a server. The organisation offering server-based signature services is expected to "apply specific management and administrative security procedures and use trustworthy systems and products, including secure electronic communication channels, in order to guarantee that the electronic signature creation environment is reliable and is used under the sole control of the signatory." Which is not helping much. In fact, the eIDAS seems to acknowledge that this is currently hard to achieve and thus hard to get accredited, stating: "However, innovative solutions and services such as mobile signing and cloud signing rely on technical and organisational solutions for qualified electronic signature creation devices for which security standards may not yet be available". In those cases alternative processes for evaluation should be used, which have a security level equivalent to standard security evaluations.

7.4 The Digidentity setup

Digidentity eSigning offers a subscription service for creating server-based electronic signatures. These are digital signatures where the user's private key is stored in the cloud. Documents are uploaded to this service in order to be signed. The idea of using a server in support of digital signing is known in the literature as a "delegated server" approach [143]. The more specific idea of having the server signing documents on behave of users is known as "server-based signatures" [6]. Both approaches require a deep confidence in the server maintainers. Naturally in a service such as this, remote authentication is paramount.

Digidentity offers four levels of these signatures, where levels one and two conform to 'regular' electronic signatures, level three conforms to advanced electronic signatures and level four to qualified electronic signatures. It is this fourth level that has our interest, since signatures made with it automatically have the same legal status as handwritten signatures.

7.4.1 Enrolment

Signing up for the Digidentity service is possible via the Digidentity website. A user chooses a user name and password and provides personal information, such as a bank account, postal address, email address, mobile phone number and the document number of an identity document. Much of this personal data is then verified: the bank account through the transfer of 1 euro cent, the email address through email,

the mobile phone number through an SMS OTP and the identity document through a face-to-face check. The user gets a letter containing a PUK code which needs to be entered during a phone call originating from the supplied mobile phone to an automated phone service. During this phone call the user chooses a PIN code.

After these checks the user can log on to the Digidentity website and can there generate two certificates, one for logging on to the Digidentity account from now on and one for digital signatures. Generating these certificates requires the user to supply his PIN code and a SMS mTAN.

7.4.2 Signing

Digidentity offers an iPad app for signing PDF documents: the SignPad app [39]. Currently, Digidentity does not offer the possibility to upload documents using the website. After installation of the app it needs to be personalised. The user supplies his user name and password for his Digidentity account, as well as an SMS mTAN sent to his mobile phone. The user then has to choose a password for accessing the iPad app. This password exists out of 4 letters between A and J, presented on a wheel that rotates for every password attempt. This prevents leaking of the password through fingerprints on the iPad screen.

The SignPad app has a built-in PDF viewer and documents can be added to the app directly from the iPad's mail client or browser or via DropBox.[8] The thus added files can now be viewed and signed. Additionally, the signature of signed documents can be verified. The SignPad app shows some strange behaviour when signing PDF forms. The built-in PDF viewer is incapable of displaying these documents, but the document itself can still be signed!

When a user indicates the wish to sign a document, he can decide on a location within the document to place the signature and provide his location and a signing reason. The document is then uploaded to the Digidentity back-end through an SSL connection, where the user's private key is stored in a HSM [40]. The creation of the signature then needs to be confirmed by the user, by supplying his PIN code and a fresh SMS mTAN. If both are correct, the signature is computed and transmitted back to the iPad, where it is added to the document.

7.4.3 Analysis

The Digidentity setup is an interesting case, since it takes a traditional problem of local authentication (the user towards the SSCD) and changes it into a remote authentication problem by placing the SSCD in the cloud. To illustrate this Figure 7.2 shows the traditional approach of using a smart card as SSCD, while Figure 7.3 shows the Digidentity approach with a cloud-based SSCD. The numbering of steps in both figures has been kept the same as much as possible. It is important to note that we have no information on the setup of the back end of Digidentity, so we present a possible scenario in Figure 7.3 where the SSL tunnel terminates within the HSM and also the authentication challenges are generated and the responses are verified within the HSM. We feel this presents the setup most secured against insider attacks. However, these attacks cannot be ruled out. For a simple example of this remember that the

[8] DropBox is a free, cloud-based, storage service: www.dropbox.com

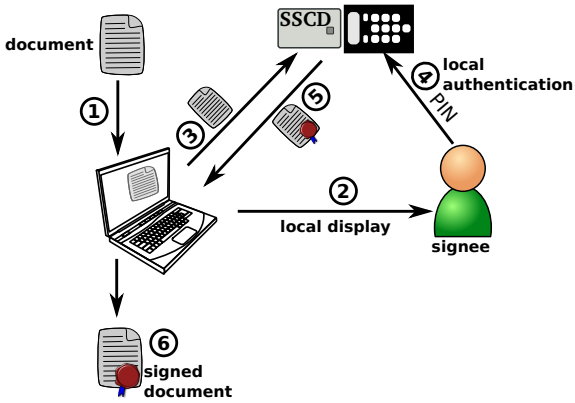


Figure 7.2: *Schematic representation of traditional qualified electronic signatures using a smart card as SSCD, e.g. in the case of the Belgian and German e-ID card.*

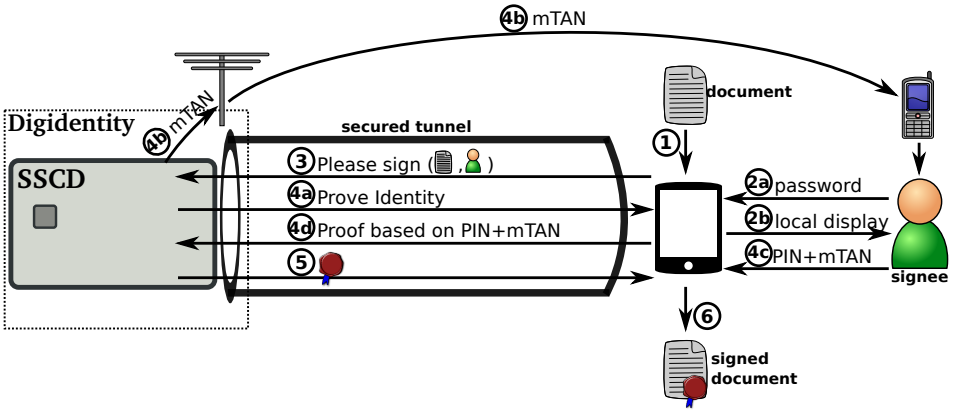


Figure 7.3: *Schematic representation of qualified electronic signatures with the Digidentity approach, i.e. a HSM as SSCD placed in the cloud.*

PIN code chosen by the user is provided to Digidentity via a phone call, requiring an extra process to place the PIN code inside the HSM.

Digidentity is accredited as a CSP able to issue qualified certificates and the e-Signing level 4 method is accredited as a means of generating qualified signatures.[9] It is unclear which parts of the system fell within the scope of this evaluation, as the evaluation report has not been made public. Digidentity stated in a meeting that the authentication step of the user towards the SSCD has been included in this evaluation, but is not clear what this exactly entails. The remote authentication messages are sent over an SSL tunnel, although this does not protect the end points of the tunnel (especially the iPad app).

[9] Both accreditation certificates can be found on the Digidentity website: `https://www.digidentity.eu/static/nl/privacy-veiligheid/privacy.html`

The authentication

When evaluating an authentication scheme, we could use the definition provided by the ECB for strong authentication, presented in Section 7.2.1. The remote authentication when signing a document using the Digidentity service uses two factors:

1. knowledge, namely the user's PIN and
2. ownership, namely the user's iPad and phone (ownership of the iPad is proven because the iPad app is personalised and ownership of the phone is proven using an SMS mTAN).

The ownership of the iPad, however, is not mutually independent with the other factors, since the iPad is the end point of the connection with the Digidentity backend, so if the iPad gets compromised, then the whole authentication is defeated. This leaves the user's PIN and SMS mTAN. Although we refer to SMS mTAN several times when discussing Digidentity's authentication, their mTAN's are actually sent over an SMS variant, which in the 3GPP standards is referred to as 'class 0' or 'flash' SMS. In essence this is simply a normal SMS with a header option enabled which causes the SMS to be displayed immediately on the phone's screen and not be stored in the SMS database, unless the used desires it. The important part for this discussion is that these messages can still be intercepted through one of the methods discussed in Section 7.2.2.

Continuing with the ECB definition, both the mTAN and the PIN code are mutually independent and the mTAN is non-reusable. Whether the mTAN can be surreptitiously stolen via the internet depends again on the security of the iPad, but as we discussed in Section 7.2.2, SMS mTAN does not conform to the ECB definition of strong authentication.

In Section 7.2.2 we concluded that using SMS has a benefit when used for transaction authentication, namely that there is a separate channel through which not only the mTAN is transmitted, but possibly also some transaction details. In the Digidentity case the mTAN is accompanied by a 64 character string, probably showing the SHA256 hash of the document ready to be signed. This transaction detail has little added value as most users will not be able to calculate and verify this hash. The SignPad app shows the same transaction detail when signing, however in our tests comparing all 64 hexadecimal characters between app and SMS took too long and the transaction expired. This might have been because of delays caused by the GSM network.

Given the presented weaknesses and conflicts with the definition of strong authentication, we claim that SMS mTAN, even using the flash SMS, is too weak a path to use in remote authentication for something as important as qualified electronic signatures. In particular, this means that the accreditation scheme allowing qualified signature accreditation for such a system is not strict enough. In October of 2016 Digidentity announced that it will move away from SMS mTAN to its own app. At the time of writing, this app was not yet available for review.

We can than wonder whether any of the other methods for two factor authentication using mobile phones discussed in Section 7.2.2 would offer adequately strong authentication. Most of the discussed methods offer only slightly better security by protecting against eavesdropping the wireless transmission, which is one of the main weaknesses here.

The strongest guarantees are offered through the Mobile PKI solution, which was the only presented method that could possibly adhere to the ECB definition of strong authentication. Interestingly, Mobile PKI comes very close to a smart card based method, with the SIM functioning as a smart card and the phone functioning as a PC and card reader in one. The biggest difference between the Mobile PKI solution and the classical smart card solution is that with Mobile PKI the SIM card is continuously present in the phone, while in smart card solutions the smart card will normally not be present in the card reader when the user is not using it. This introduces an extra weakness, by giving attackers (e.g. via malware) a much longer time frame in which to perform an attack. As a mobile phone solution, the Mobile PKI method does offer the benefit of users noting the disappearance of the mobile phone much faster than of a smart card.

So, of the discussed methods for two-factor authentication using mobile phones, the Mobile PKI method would seem most acceptable for authenticating to a signing service.

Server-based signatures in general

If we assume for a moment that the remote authentication of a server-based signature scheme adheres to the ECB definition of strong authentication, then we can still wonder if a server-based signature solution is acceptable in general, when looking at the European directive on electronic signatures discussed in Section 7.3.1. The demands 1, 2 and 4 for advanced electronic signatures seem to be met, as are the additional demands for qualified electronic signatures, since the Digidentity solution could get the required accreditation.The problem seems to be with demand 3, requiring the signer to have sole control over the means creating the signature. In general, there seems to be no technical means through which a signer can keep sole control over his SSCD when this is placed in the cloud. This can be amended by strict procedures, but it would be hard to argue that a user still has the sole control in these situations, as the SSCD is kept under control by a company who use it on the user's behalf.[10]

In 2005 the Forum of European Supervisory Authorities for Electronic Signatures (FESA) concluded that server-based signatures could, in principle, be able to meet the demands for qualified signatures set by the European directive, although they deem this rather unlikely [79]. Furthermore, they conclude that the usual accreditation for SSCDs would be extremely complex for a signing service and that such signatures would be unusable in Germany, where sole control implies physical control. This opens up a legal issue on the acceptability of server-based signatures accepted and generated in other EU countries being used in Germany.

Server-based signature schemes offer an explicit advantage over traditional smart card-based systems in that they allow for additional security through monitoring of transactions. Such transaction monitoring proved very successful in preventing online banking fraud. However, this will be very dependent on the type of transactions: online banking transactions can be profiled to scan for abnormal transactions, but for generic signatures this could be a lot harder or even impossible. Additionally, transaction monitoring could also introduce privacy or confidentiality issues.

[10] As discussed in Section 7.3.3 the current legal framework was specifically changed to allow server-based signing.

The fundamental problem with the classical, smart card-based solutions, is the lack of a trusted display: the user sees the document on his computer and has to trust that this will be the exact document signed by the smart card. An attacker in control of an infected computer, could replace the document with any document he wants to have signed by the user. For accredited smart card solutions this problem is apparently acceptable.

If the user also uses the computer to input his PIN code to the smart card, then the attacker can even sign multiple documents, for as long as the smart card stays connected to the computer. This risk can be avoided by entering the PIN code on a numeric keypad on a smart card reader, as is for instance required in the German e-ID solution. An attacker in control of the user's computer is then only capable of changing the documents signed by the user and not add new documents to be signed.

Server-based signature schemes also do not have a trusted display, unless specific hardware is rolled out. However, there is the option of adding an out-of-band channel to the user, which not only adds an additional factor to the authentication, but can also offer the option of displaying transaction details to the user on what he is signing [185]. This works well for small transactions such as online banking transactions, where the transaction details can contain the account numbers and amounts involved in all the transactions. For electronic signatures though it is not so clear what to add as transaction details; A typical document to sign will often contain too much information to be simply summarised in these transaction details. The Digidentity solution presented above uses a hash of the document as transaction details, but this requires the user to compute the hash of the document he wishes to sign on a different device than the iPad he currently uses to sign a document. A user not willing or unable[11] to perform this check runs the same risk of the document swap as users of smart card based solutions.

If an attacker also wants to control this out-of-band channel then that will necessitate an additional attack on the second hardware platform. In the Digidentity example an attacker then has to also infect the user's mobile phone as well as the iPad, to achieve this. This is an extra step for the attacker, but it will also increase his capabilities. In a server-based signature solution the SSCD is always online, so an attacker controlling all authentication channels can sign an unlimited number of documents. This shows the fundamental necessity of having trusted hardware in the hands of the user. In the case of Digidentity this trusted hardware for the user is missing and the whole user side security is realised in software on generic and complex operating systems.

A solution for server-based signatures could be to provide users with smart cards and card readers as trusted hardware, but this would defeat the major benefit of server-based signatures, namely that users do not require an additional card reader.

7.5 Related work

There have been many proposals for new authentication schemes using mobile phones as additional factors [118, 3]. Some of these treat the mobile phone as a trusted device

[11] A user can also simply be ignorant, as in the current Digidentity implementation there is no explanation of what the transaction details represent, or how a user could reproduce them.

[118], an assumption we are not ready to make in this chapter. Others provide, in essence, a combination of two methods that were discussed above, such as a combination of SMS mTAN and the Google Authenticator app [3].

Van Rijswijk and Van Dijk presented the tiqr approach (also discussed above) in a paper where they also compare other two factor approaches such as hardware tokens and mobile phone solutions [180]. Their comparison is broader, not only in what they compare, but also where they compare on; e.g. ease-of-use, security and costs, than the comparison above, which focuses solely on security.

Mulliner et al. [123] reviewed the security of SMS-OTP (discussed in Section 7.2.2) in-depth and proposed a solution to improve SMS-OTP security w.r.t. malware, by having the OTP SMS-es being delivered directly to a specific app. This solution hinges on the reliability of the mobile OS to be able to deliver the SMS directly to the specified app, without malware being able to intervene. Additional encryption could then also be applied on the SMS messages, removing most of the weaknesses caused by the use of the SMS channel.

7.6 Conclusions

We have looked into different multi-factor authentication approaches using a mobile phone as an additional factor. Of the presented approaches the Mobile PKI is the most secure, although it is also the most costly solution and requires permission and cooperation of the providers involved. The other approaches all have major security concerns, when used as authentication to sensitive applications. If the SMS mTAN approach would not use an unaltered challenge as a response, it might be an acceptable choice. However, such a solution would then need to store a secret on the user device, which introduces new weaknesses. Perhaps in the future banks can take advantage of smart phones offering NFC combined with RFID bank cards or national identity cards [102]. Given the advent of mobile banking apps, the primary log-in device would then probably be the mobile device, and the RFID bank card would then be the second factor.

We have presented a first look at the new ECB recommendations on the security of internet payments, focusing on their definition of strong authentication. While these are a usable definition, we would like to see some extra clarification on the demand that at least one element used in the authentication "should not be capable of being surreptitiously stolen via the internet." It is not clear which attacks are meant in this part of the definition. Also, placing this demand only to the internet seems too restrictive as many banks use SMS as an extra channel. Furthermore, we regret seeing that the ECB included transaction authentication only as an optional feature: this would seem to be the main point for online banking transactions.

We analysed a server-based signature service called Digidentity. We feel their choice of using SMS mTAN as an additional factor for authentication is too weak for qualified electronic signatures. Digidentity has since announced that they will move away from SMS mTAN as second factor for authentication. Digidentity achieved all necessary accreditation for their SMS mTAN solution. It is unfortunate that the value of this accreditation cannot be judged, since so little is publicly known of the process of accreditation of the SSCD for Digidentity. Since these types of evaluations allow

the user's computer to be placed out-of-scope, it is unclear if for instance the iPad app was included in the accreditation. Potential users should be able to rely on the accreditation scheme, and it is therefore not good for the faith and trust placed in such schemes, that so much of the accreditation process and details can remain confidential.

Finally, we conclude that server-based signature services in general had a great difficulty conforming to the legislation on qualified electronic signatures. With the current state-of-the-art we cannot imagine a setup where a user's private key is stored in an SSCD in the cloud, but the user can still be said to have the sole control over his key. The eIDAS regulation [167], relaxes the legislation enough to allow for server-based signature services, though in the current wording it is not clear how to evaluate such solutions. Hopefully, a scientific debate on the acceptability of server-based signatures will arise now that this new standard has gone into effect.

Chapter 8

Defeating IMSI catchers

So far, this thesis mostly attempted to assess the security of mobile telephony in practice, pointing out several weaknesses in the process. Now we will look into whether it is possible to improve the current situation. To that end, this chapter attempts to protect against an attack that is prevalent in all generations of 3GPP technology: IMSI-catching.

We propose a solution which defeats the IMSI catching attacks and increases the credibility of the mutual authentication with the home network. The latter has significant impact against man-in-the-middle attacks such as presented in [120], and provides additional security to 2G networks which currently only support a unilateral authentication procedure. This solution was formally verified using ProVerif and does not interfere with the workings of the networks as they are defined today, and is in fact backwards compatible with current implementations. The only party that would need to make a change would be the mobile providers, as they provision the user with an IMSI and the SIM that contains it and are also the only party in control of the authentication server, where the IMSIs are linked to the authentication parameters (keys and algorithms). Every provider can independently decide whether to implement this solution, and as our solution changes nothing in the message definitions a change happens transparently for any intermediate providers.

This chapter is based on the article *Defeating IMSI Catchers*, presented at the ACM Conference on Computer and Communications Security, CCS 2015 [179]. Compared to the original publication, this chapter contains some changes in the introduction and some information was moved to chapter 2, the background chapter, for a better integration with the rest of the thesis.

We are currently working with the Fraud and Security Architecture Group (FSAG) of the GSMA (the GSM Association) to get this solution into the current cellular specifications.

8.1 Introduction

In mobile telephony it is hard to avoid providers knowing the approximate location of their users, because they have to be able to route incoming traffic to the mast currently nearest to the user. Actually, there have been proposals for a TOR-like network among mobile phones, in order to obscure the location of users for the network [15], but such a scheme seems impractical with regard to the reliability we have come to expect from the mobile network. This continuous location monitoring can lead to serious (location) privacy issues for users. However, if we accept this issue, for instance because we trust the providers, then there are still many other privacy issues related to mobile telephony, such as apps sharing private data [115] or eavesdropping phone conversations (which was discussed in Chapters 3 and 4). One of the first practical privacy attacks against mobile phones was the so-called *IMSI catching attack*, and this attack persists even in today's mobile network standards. This term refers to the unique identifier present in every SIM card, called the IMSI for International Mobile Subscriber Identifier. This identifier is transmitted in plain-text over the wireless network as (initial) identification and can therefore easily be intercepted.

We continue with some background information about mobile networks in Section 8.2. This section not only discusses identification in mobile networks, but also looks at authentication, since our solution requires a change of a parameter of one of the authentication messages. Section 8.3 describes our solution against IMSI catching for the current technology and provides the general idea of our solution. Section 8.4 describes our solution modified for the older, but still heavily used 2G technology. We formally verify our solution in Section 8.5, as well as analyse the effectiveness and consequences of implementing it. Finally, we review related work in Section 8.6 and draw conclusions in Section 8.7.

IMSI Catching

IMSI catching was one of the first practical attacks on GSM, leading to the development of devices called *IMSI catchers*, which gather all IMSIs that are active in a geographic area. An IMSI catcher can achieve this in two different ways: passive and active. The passive way is by simply observing the wireless traffic and storing all IMSIs observed. For the more effective active attack a fake base station is set up, to which cell phones in the neighbourhood will attempt to connect. The fake base station then simply commands each phone to identify itself. This way IMSIs can be retrieved at any time, while with the passive attack the attacker has to wait for phones to send out their IMSI. IMSI catchers are commercially available, though they are expensive and usually sold restrictively to government officials. However, in recent years cheap and precise enough equipment has become available that can be used to create an (active) IMSI catcher. Likewise, cheap base stations called femtocells are commercially available, several of which have been rooted, making them into very cheap (below $100,-) IMSI catchers, as we discussed in Chapter 5.

Over time, the commercial IMSI catchers were extended with a lot of additional functionality such as eavesdropping on wireless calls. However, they are still, rather euphemistically, called IMSI catchers. This leads to a lot of confusion on what is meant by an IMSI catching attack. For this article, we refer only to the gathering of

IMSI numbers from the air waves (either passive or active) as IMSI catching.

Recent news stories uncovered widespread use of unregulated IMSI catchers. In an article from The Washington Post researchers found 18 IMSI catchers in Washington D.C. within two days [33]. These IMSI catchers were present at airfields and embassies and could be detected since they actually ran active Man-in-the-Middle attacks, possibly to eavesdrop on mobile connections. So, traditional IMSI catchers, as discussed in this article, might be even more widespread. The FCC started an internal task force to study the use of IMSI catchers by criminals and foreign intelligence agencies [32], indicating that these attacks are considered a serious issue.

Location privacy

IMSI catching attacks mostly relate to the issue of location privacy, as the transmission of your IMSI reveals your approximate location. Location privacy attacks attempt to link an identity to a location. By keeping one of these (identity or location) fixed and trying to recover the other, we identify two different goals of location privacy attacks:

Retrieving identities at a location (monitoring)
A list of caught IMSIs can reveal who came at what time within, for example, the near vicinity of a specific building, or was present at a certain rally. Such monitoring is also used for commercial goals such as customer monitoring. We have seen that shop keepers already collect WiFi signals from phones to determine statistics such as returning customers or the effectiveness of their shop front [154]. IMSI catching could be used not only to invade privacy but also for actual physical attacks. Consider, as (dramatic) example, automated terrorist attacks that trigger bombs to explode when high-value targets come in range of an IMSI catcher [85, 21].

Retrieving a person's location (tracking)
Recovering a person's geographic movements can reveal a lot about what they do and who they meet. As IMSI catching requires the operation of rogue cell towers or passive listening antennas in the vicinity of the victim(s), it seems most useful for monitoring attacks. In GSM a cell tower can service an area up to 34 kilometres in diameter, so the vicinity is in the order of several kilometres. For actual tracking over a larger area, an attacker would need a mobile setup – such as one connected to a drone [104] – or a network of antennas. This is not unthinkable, for example the city of London initiated a project where trash cans monitor WiFi signals of mobile phones to profile people's behaviour in order to send them targeted advertisements [107].

Traditionally, IMSI catching is associated with a more hybrid attack where police forces use IMSI catching to recover information about the mobile subscription of a target [159]. They follow a target and gather a list of active IMSI numbers in several unrelated crowds on independent locations and intersect the recovered sets of IMSI numbers. The legitimacy of this method is debatable, especially since they often transmit signals from fake cell-towers which interfere with genuine cell-towers. Furthermore, this technique seems to be used unregulated by several entities, such as intelligence agencies and malicious adversaries [33, 84].

8.2 Background

IMSI catching is an issue in what is often called the 3GPP or GSM family of cellular technology; world-wide this is by-far the most popular of the mobile telecommunication systems. This family of algorithms started in the early 90s with the introduction of GSM, which was then 2nd generation mobile technology. Currently, we are seeing the deployment of the 4G networks (LTE and LTE+). Within each generation there have been incremental improvements mostly to the up- and down-link speed (e.g. HSDPA+ over UMTS in 3G), but the protocols themselves only receive real changes with the move to a newer generation. As previously stated, the IMSI catching problem exists in all generations. This following sections give some background into specifics of the 3GPP networks that are relevant for the IMSI catching dilemma. In Section 8.2.4 we describe 3G authentication in more detail, both as an example and because we use the authentication messages in our solution.

8.2.1 Identification within 3GPP networks

Cell towers in mobile networks identify themselves by broadcasting identifiers. Mobile phones pick up these signals and decide whether to connect to a network or not. This decision is based on data from the SIM, which instructs the phone to look out for certain networks by both frequency and identifier. When a mobile phone connects to a network, it first requests a channel to exchange information on with the cell tower. On this channel the cell tower can always request the SIM's identity. Identification is performed after a simple command from the cell tower to a mobile phone. This command, *Identity request*, specifies a specific identifier (IMSI, TMSI, IMEI or IMEI(SV), see Section 8.2.2), to which the phone responds with a so-called *Identity response* containing the requested identifier [52, 48]. Authentication can only take place after identification, because the authentication is based on a symmetric key shared between the SIM and provider.

Interestingly, the specifications of the mobile standards acknowledge the problems of IMSI catching. In [68] several security goals for mobile networks are stated, among which are confidentiality of the IMSI (user identity confidentiality), user location confidentiality and user untraceability (Section 5.1.1 of [68]). The same document acknowledges the breach of user identity confidentiality introduced with the request identification message (Section 6.2 of [68]), though no breach of the location privacy issues is mentioned here. The specifications further mandate that a SIM does not answer Identity request messages asking for any identifier, other then the IMSI, when no encryption context is yet established (Section 4.4.4.2 of [48]). This, of course, would not prevent IMSI catching, but does prevent the leaking of the other identifiers to IMSI catchers. However, our experiments show that all of the current 3G or 4G enabled phones and SIM cards we tested also transmit the TMSI and IMEI unprotected when requested.

8.2.2 3GPP identifiers

While this article is mostly concerned with protecting the IMSI, many more identifiers exist in the 3GPP networks. We discuss the most important of these briefly below.

International Mobile Subscriber Identifier (IMSI) The IMSI is the main identifier in 3GPP networks and belongs to one specific SIM card. It is a 15 digit number where the first three digits identify the home country (MCC, Mobile Country Code), the following two or three digits identify the home network (MNC, Mobile Network Code). The remaining nine or ten digits identify the specific user/SIM within the provider's database.

Temporary Mobile Subscriber Identifier (TMSI) The TMSI is introduced to protect against traceability of users. The TMSI is a temporary pseudonym provided to the mobile device by the network, to use instead of the IMSI, essentially masking the IMSI against passive attacks. The TMSI is only valid within a certain geographical area. When a mobile phone moves to another area it initiates a location update procedure, which should provide it with a new TMSI. The time a single TMSI remains valid is configurable by the access network. Since all communication with the mobile phone should be based on the TMSI[1], phones are traceable via the TMSI during a validity period. TMSIs do not provide adequate protection against IMSI catching attacks though, since a cell tower can always request a phone's IMSI. TMSIs are therefore easily defeated by active IMSI catching attacks. Furthermore, research shows that in practice TMSIs remain valid for far too long and are re-used over different areas [5], making them even usable in passive IMSI catching attacks.

International Mobile Equipment Identifier (IMEI) The IMEI is a 15 digit number that identifies the mobile device itself. It is included to make black-listing of stolen phones possible. There is a closely related alternative to the IMEI, often referred to as IMEI(SV), which is one digit longer and also identifies the software version running on the phone.

Other identifiers
There are several other ways to identify a mobile device based on its transmissions, for instance the phone's answer to authentication requests. Because these requests are answered based on a shared secret key, the same challenge always invokes the same response. There are also several ways to identify a mobile device outside of the 3GPP protocols. Examples of these include the MAC address of the WiFi or Bluetooth adaptor.

8.2.3 Authentication within 3GPP networks

For all 3GPP systems the customer's SIM card shares a (set of) secret key(s) with the authentication server of his provider. Any authentication and encryption of messages is performed with temporary keys derived from these shared secret keys. The link between a SIM's unique identifier (IMSI) and its secret keys is made either through a diversified key solution or a simple look-up table. Once the SIM has been identified, the network can look-up the accompanying secret key and initiate authentication.

It is important to note that the party authenticating the mobile device does not need to be the user's own provider. A mobile device can be 'roaming,' i.e. using the

[1] Phones can also still be paged using their IMSI numbers, and listening on paging channels shows that this occurs frequently.

network of another provider. We call the network with which the mobile device is currently connected the *access network*, and the network of the user's provider the *home network* (see Figure 8.1). Earlier, in the Background Chapter (Page 16) we already introduced this division, but there we used *core network* instead of home network. In this chapter we use the term 'home network,' to stress that this is the user's own provider, possibly introducing the new features we propose, while other core networks possibly keep everything unchanged.

Figure 8.1: *Schematic overview of the division between the access network and the home network.*

The access network starts a challenge-response protocol with the SIM, which authenticates the SIM and establishes (a) sessions key(s). When using 3G or 4G technologies, the challenge sent by the access network also authenticates the home network, for 2G technologies there is only SIM-only authentication. Since only the authentication server within the home network and the SIM know the secret key, the access network is not capable of authenticating (to) the SIM. Obviously, it would be unacceptable for the different providers to share the secret keys among each other. Therefore, all 3GPP protocols have the access network contact the SIM's home network to request authentication parameters for the IMSI. This is possible because part of the IMSI identifies the home network. The authentication parameters contain a challenge, the associated response, resulting session key(s) and, in case of 3G and 4G technology, an authentication token. This authentication token is used for the SIM to be able to authenticate the home network and is essentially a sequence number shared between the SIM and home network and a MAC over this sequence number and the challenge. The access network forwards the challenge and authentication token to the SIM, which verifies the validity of the authentication token and responds to the challenge. The SIM also uses the random challenge to compute the session key(s) and forwards these to the phone. The access network compares the response from the SIM with the response from the home network. If they are equal, then the SIM is authenticated and both sides of the wireless interface share a (set of) session key(s). In Section 8.2.4 the authentication protocol for the 3G technology is discussed in more detail.

This set-up leads to an interesting a-symmetry in the authentication, whereby the *access* network authenticates the SIM and (in case of 3G and 4G) the SIM authenticates the *home* network. It is also curious that for establishing the session key(s), no input or freshness of the SIM is required.

8.2.4 3G Authentication and key establishment

Since our solution against IMSI catching requires a change in the authentication procedure we will now take a detailed look at authentication in 3G networks. This so-called Authentication and Key Agreement (AKA) protocol provides both mutual authentication, between SIM and home network, and establishes session keys. The AKA protocol for 4G is almost identical, with only one additional parameter used to diversify the session keys. A difference that has no impact for this chapter.[2] This AKA protocol is defined in [68] and we present an overview in Figure 8.2. The bit lengths of important variables are summarised in Table 8.1.

To be more precise we first introduce the variables and functions used in this chapter to formalise the authentication protocol. The set of all available IMSI numbers is denoted by I and is available to the home network of the provider. The home network stores the properties of a SIM card, IMSI (i), secret key ($\mathcal{K}$) and sequence number (SQN), as a tuple $s = \langle i, \mathcal{K}, SQN \rangle$ for each $i \in I$ in the set of all available SIM cards S. To simplify the notation, we use subscript on a tuple to denote a single element from the corresponding tuple, e.g. s_i denotes the IMSI number of the SIM card s. Encryption with key k is denoted by $E_k()$ and decryption by $E_k^{-1}()$. Consequently, generation of a MAC with key k is denoted by $M_k()$. We do not specify a specific algorithm, but several standardised cryptographic primitives and methods are suitable for encryption [75, 22, 36] and generating MACs [43, 76, 77]. The AKA protocol relies on five encryption functions referred to as f_1 to f_5. The implementation of these functions is provider-specific and not fully standardised. However, the standard that defines the security architecture [68] suggest that the provider may use the example algorithm implementation set, defined in [46], for authentication and key generation. This implementation set is based on Rijndael, combined with different provider codes that get XORed with the random challenge, for each of the five algorithms. We assume in this chapter that the operator uses functions with similar characteristics as those proposed in [46].

The SIM has at some point identified itself before the authentication starts. The access network can then request authentication parameters from the home network. The home network computes the authentication parameters, which consist of a freshly generated random which acts as a challenge (RAND), the corresponding response (SRES), confidentiality key (CK), integrity key (IK), anonymity key (AK) and an authorisation token (AUTN). The AUTN token, is the authorisation proof by the home network. It consists of the sequence number XORed with the anonymity key, the Authentication Management Field (AMF) and a MAC over SQN, AMF and RAND. The sequence number protects against re-play attacks, the AMF is for the provider to use, for instance to signal a specific algorithm suite, or set a time validity for a key and the MAC authenticates this message as coming from the home network. The authentication parameters are then transmitted to the access network. The challenge RAND and the AUTN token are forwarded to the SIM.

The SIM, upon reception of an authentication request, first retrieves the sequence

[2] In 4G AKA the access network is not trusted to verify the response to the challenge, as opposed to 2G and 3G. However, this difference also has no influence on the rest of the chapter, as the access network still requests the IMSI to find the home network.

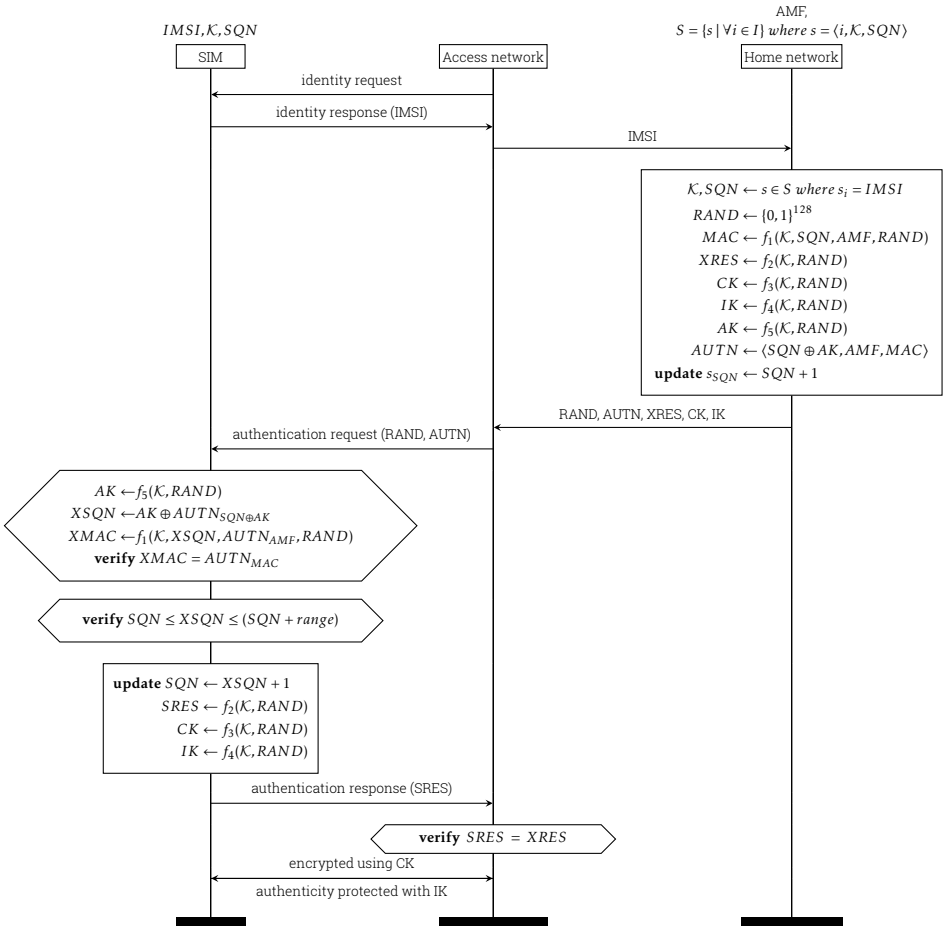


Figure 8.2: *A schematic representation of successful SIM authentication in 3G networks [47]*

number (XSQN), by computing the anonymity key and XORing this with the sequence number from the AUTN token. Then the SIM verifies the MAC over the authentication token. If the authentication token proves genuine, the SIM verifies that the sequence number from the network (XSQN) is higher than its own sequence number (SQN). If the received sequence number is lower, or too high, then the SIM responds with an error message and a genuine network will then start a re-synchronisation setup. By how much the sequence numbers can deviate from each other is a setting chosen by the home network.

If the sequence number falls within the range of allowed sequence numbers, the SIM computes the confidentiality key, integrity key and response. The response is transmitted back to the access network and the two keys are stored in the phone. The access network compares the response of the SIM (SRES) with the response of the home network (XRES), and when found correct, the network can order the use of

Table 8.1: *Bit length overview, starred lines are newly added in the presented solution.*

variable	bit length
IMSI	60 (bcd encoded)
PMSI	34 ⋆
$\mathcal{K}$	64 (2G) / 128 (3 and 4G)
RAND	128
SQN	62 ⋆ (2G) / 48 (3 and 4G)
MAC	64 (3 and 4G)
M_κ	32 ⋆ (2G)

integrity protection and ciphering.

The AKA protocol had been formally verified using enhanced BAN logic and shown to provide both authentication and confidentiality [57]. However, using ProVerif, a location privacy attack was found by Arapinis et al. [4]. Replay attacks of the authentication token are prevented by the sequence number, but replaying this token will break location privacy, as the SIM responds differently to an out of sync message than to an incorrect MAC message. Of course, IMSI catching is an even simpler way of breaking location privacy.

8.3 Solution for 3G/4G

The underlying weakness enabling IMSI catching attacks is that the authentication is based on symmetric cryptography. The use of a shared secret key, means a SIM has to be identified before it can be authenticated. Identification prior to (mutual) authentication is a problem that crops up in other systems as well, such as in e-passports [96] and RFID tags [116]. However, common solutions to this issue do not help for the IMSI catching case. For instance randomising the IMSI is no solution, because the IMSI needs to be identifying for the provider in order to provide cellular service. Other solutions, such as encrypting the identifier with the public key of the home network, would require changes in the messages between the phone and network, as the resulting ciphertext would not fit inside the currently defined identity response messages. Furthermore, some additional randomness would need to be added to every encryption to ensure that the SIM does not simply use another long term identifier (the encryption of the IMSI) instead of the IMSI. The space for the IMSI in identity response messages leaves too little room to add the randomness to the encryption without changing the message size. Since it is unrealistic to expect such changes to core message sizes being implemented in the current mobile phone technology, we present a solution that works within the current implementations.

We propose a solution where the IMSI is replaced with a changing pseudonym that only the SIM's home network can link to the SIM's identity. This hiding of the IMSI is done without changing any of the system messages, thus making it transparent to the access network. This allows our solution to be deployed by providers, on an individual basis, on top of the currently available 3GPP networks.

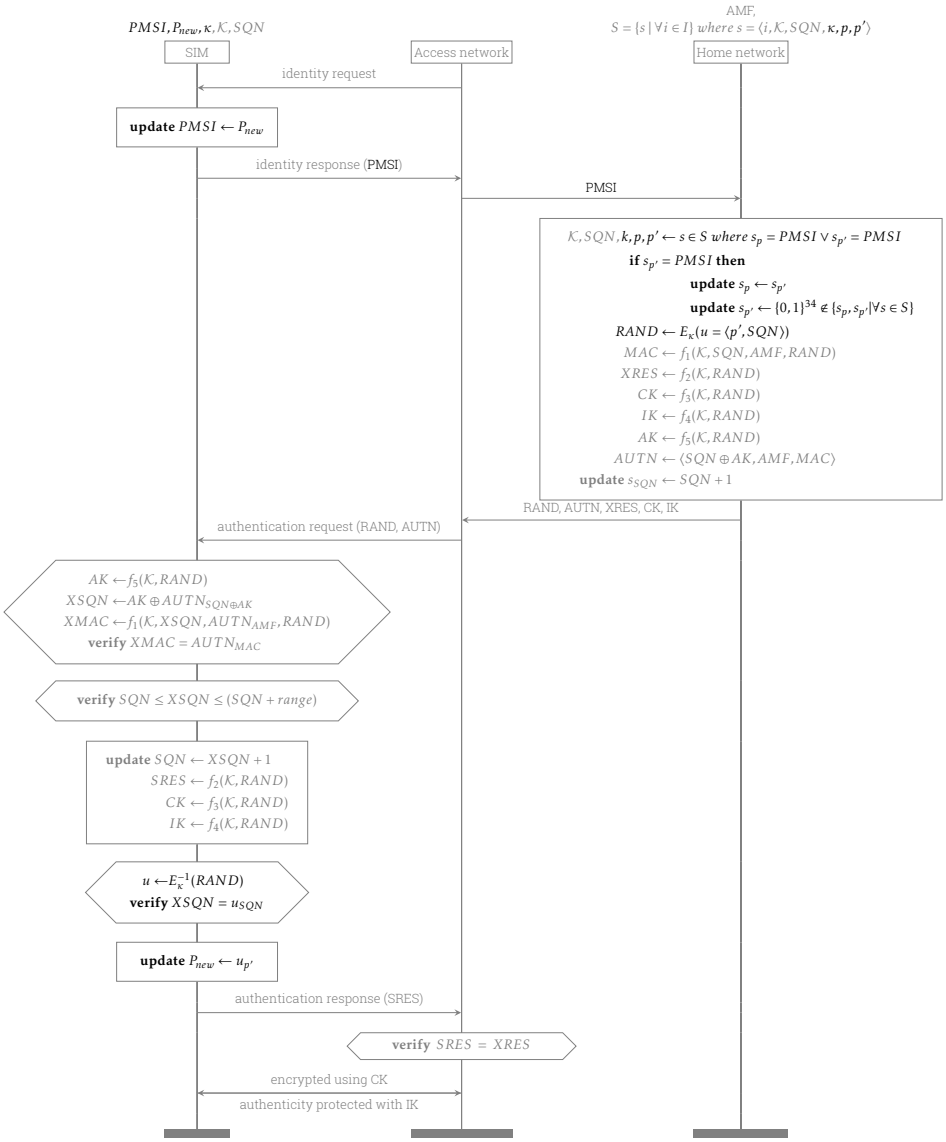


Figure 8.3: *Solution proposed for 3G and 4G compatible authentication protocols. The black text shows our additions to the standard protocols.*

During authentication, the authentication server supplies the user's SIM with a random new IMSI, which we refer to as *Pseudo Mobile Subscriber Identifier* (PMSI). The SIM uses the new PMSI the next time it is requested to reveal its IMSI. As discussed in Section 8.2.4, the user's provider operates an authentication server which generates a random challenge-response pair and the corresponding session keys. We propose to

use the random challenge (RAND) to provide the SIM with the PMSI. The PMSI has to be encrypted in a semantically secure way, in order to keep it confidential between SIM and authentication server. The resulting ciphertext should be sufficiently random and unpredictable to still serve as the challenge.

The changes we make in the authentication protocol for 3G are illustrated in Figure 8.3, though these changes can be applied the same way to the authentication in the other generations of mobile networks. Comparing Figures 8.2 and 8.3 shows that no changes are made in the messages that are transmitted, but only in the end points, i.e. the SIM and home network. This makes our solution compatible with all current implementations of the different generations of mobile networks. The changes needed for the authentication protocol used for 3G and 4G are discussed in detail in the next section, the changes for the 2G authentication protocol are discussed in Section 8.4.

8.3.1 Authentication server

In our solution, authentication servers have to be extended to store three additional values for each SIM: the new shared secret key κ and the two PMSI values p and p'. Here p is used to store the PMSI value the SIM s is currently using and p' stores the new PMSI value that the authentication server designates as the successor PMSI for that SIM. A provider implementing this solution would change the normal routine of its authentication server, when composing an authentication request for PMSI as follows:

1. *Validate if PMSI is known by the home network*

$$\exists s \in S,\ s_p = PMSI \vee s_{p'} = PMSI$$

2. *Update the PMSI when the successor $s_{p'}$ was used*

   if $s_{p'} = PMSI$ **then**
   $s_p \leftarrow s_{p'}$
   $s_{p'} \leftarrow \{0,1\}^{34} \notin \{s_p, s_{p'} | \forall s \in S\}$

3. *Compute challenge RAND by encrypting $s_{p'}$ and SQN*

$$RAND \leftarrow E_\kappa(u), \text{ where } u = \langle s_{p'}, SQN \rangle$$

4. *Compute other authentication parameters: MAC, XRES, CK, IK, AK and AUTN*

5. *Increase sequence number SQN and update $s \in S$*

$$s_{SQN} \leftarrow SQN + 1$$

6. *Transmit authentication parameters to access network*

Steps 1 and 2 are new for our solution, while the computation of RAND in step 3, was changed from generating a random number in the standard procedure. The other steps remain unchanged. The check in steps 1 and 2 has to be done for each

message parameters with an PMSI arriving at the authentication server, such as the location update message[3], essentially using this as confirmation that the SIM card is now using the pending PMSI $s_{p'}$ as replacement for the previous PMSI s_p. Because of efficiency, access networks often request multiple authentication parameters for a SIM at the same time. So, to keep a check on the number of pending PMSIs, the authentication server will keep sending the same new PMSI number (with a different sequence number) to the SIM, as long as a SIM identifies itself with the same PMSI. This means the authentication server needs to keep a running record of at most two PMSI-numbers per SIM card.

8.3.2 SIM card

The SIM is extended to store the new shared secret key κ next to two PMSIs; the currently active PMSI (PMSI) and the future PMSI (P_{new}). Upon receiving the challenge, after normal verification steps (e.g. verifying the MAC and the sequence number) a SIM can decrypt the challenge and verify if the sequence number from the decrypted challenge (u_{SQN}) is equal to the sequence number in the authentication token (SQN). If so, the SIM retrieves the new PMSI (P_{new}). Any future IMSI identification request can then be answered with the newly received PMSI, instead of the previous PMSI by updating PMSI with P_{new}. It is possible for PMSI and P_{new} to temporarily have the same value, if PMSI is updated to its successor (P_{new}) and no new PMSI has yet been received.

Additionally, the SIM could have a policy which determines the amount of time it will wait for a new identity request, so it can refresh its PMSI, before forcing a refresh itself. Forcing a PMSI update is performed by, for instance, signing on to the access network as a freshly arrived SIM with the new PMSI.

To support our solution a SIM card's handling of authentication requests should be changed to include steps 5 and 6 in the following procedure:

1. *Perform existing authentication steps to recover XSQN*
2. *Use existing authentication procedure to verify MAC*
3. *Verify sequence number and update SQN*

$$SQN \leftarrow XSQN + 1$$

4. *Compute CK, IK and SRES*
5. *Decrypt RAND and verify sequence number* u_{SQN}

$$u \leftarrow E_{\kappa}^{-1}(RAND)$$
$$\textbf{verify } XSQN = u_{SQN}$$

6. *Update the future PMSI* P_{new} *to the supplied successor* $u_{p'}$

$$P_{new} \leftarrow u_{p'}$$

[3]Technically the location update is directed to a logically separate entity: the Home Location Register. However in practice this entity is always combined with the authentication server.

8.4 Solution for 2G

In 2G authentication, the SIM authenticates itself to the network, but the network does not authenticate itself to the SIM. Figure 8.4 shows the 2G AKA protocol, with our changes to hinder IMSI catching highlighted. These changes are primarily meant to prevent IMSI catching attacks by supplying the SIM with new PMSI numbers through the challenge, same as before. However, as a side effect our solution gives the SIM the capabilities to verify if a challenge presented to him actually came from his home network.

Since standard 2G authentication has no sequence number, this has to be added for our solution. Compared to 3G and 4G, the set of acceptable SQN values for the SIM has to be much larger as there is no separate check on the SQN before the PMSI is retrieved from the challenge. Furthermore, in 3G/4G their exist protocols to sync SQN when the SIM and home network get out of sync. In the 2G environment we do not have the room to implement a syncing protocol within the current specifications. An attacker could therefore attempt to let the network issue many different challenges for the same SIM, thereby increasing its SQN value, hoping to get the network out of sync with the SIM. The SIM thus has to accept a much larger set of SQN values, e.g. every u_{SQN} value higher than its current value for SQN, which still prevents replay attacks.

Since in 2G there is no guarantee that the challenge presented to the SIM is authentic, our solution requires an additional integrity check on the encrypted PMSI. Otherwise, an attacker could act as a base station and transmit a random number as a challenge. In turn the SIM would decrypt this, and if the SQN value is higher than the current value, it would accept the first part of the decoded challenge as the new PMSI and the SIM will start to identify itself with a number unknown to the home network.

A cryptographic MAC (M_κ), computed by the home network, counters a Denial-of-Service attack which aims to desynchronise the SQN numbers known by the SIM and the home network. Such an authenticity check makes it very difficult for an attacker to forge a valid *authentication request* without knowledge of the secret key κ. This essentially introduces an authentication of the home network to the SIM card, which is not available in the default 2G authentication protocol.

8.5 Analysis

The previous sections presented our solution against IMSI catching, in this section we analyse the effect of the proposed solution. It provides new pseudonyms to the SIM to be used instead of the IMSI. The pseudonyms are provided in a confidential manner. An attacker, either active or passive, is unable to learn said pseudonym before it is used, as long as the attacker does not know the secret key κ. This provides unlinkability between consecutive pseudonyms. Furthermore, this protocol changes nothing in the messages as they are currently defined for 3GPP mobile telephony. The change is transparent for the access network and the challenge used to transmit the PMSI should still be random due to the encryption and fresh due to the increasing sequence number.

In the case of 2G there is an extra benefit to our approach, as it adds a message

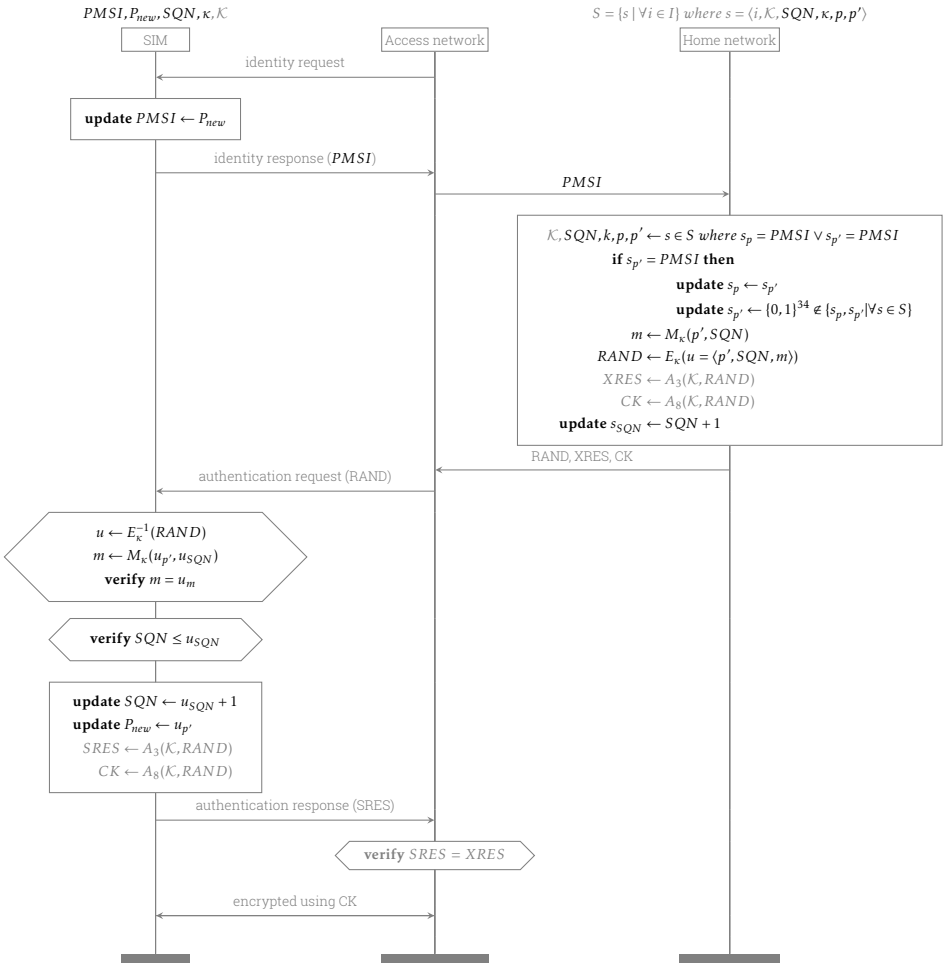


Figure 8.4: *Solution proposed for 2G compatible authentication protocols. The black text shows our additions to the standard protocols.*

authentication to the challenge. This does not prevent a Man-in-the-Middle attack, whereby an attacker simply passes on the challenge (though without learning the PMSI it contains), but it does prevent the replaying of challenges or the insertion of false challenges. Essentially achieving the same level of network authentication as in the standard 3G and 4G AKA algorithms.

The presented approach does not completely remove IMSI catching as an attack. After all, a SIM receives a new PMSI when authenticating, and will only start using it on the next identity request message. So, after switching to a new PMSI, a SIM will keep using that same PMSI for some time. Which means traceability remains until the switch to a new PMSI is made. In practice though, this remaining traceability mostly coincides

with the already existing traceability of the TMSI. Our solution even alleviates the problems of TMSIs – too long validity periods over multiple geographical areas – as switching PMSIs, will automatically refresh the TMSI, since the SIM will then appear as a new SIM to the access network.

The country and home network code of the IMSI will need to remain intact for our PMSI numbers, because these are needed to route messages to the user's home network. This means, that even using our PMSI pseudonyms, there are still some privacy issues, since a SIM will still reveal its home country and home network, when transmitting their PMSI. In essence the use of a PMSI provides k-anonymity to users [161], where k is the size of the group of expected users in this geographic area who have the same home country and home network. In most countries there are only a small number of mobile providers operational, they will mostly have fairly large consumer bases and thus a large k can be expected [171]. In other words, this solution provides stronger anonymity to users from a provider with a large consumer base, within their home country. If those users use their phone abroad, their traceability will likely increase dramatically, as there will be few other users transmitting an IMSI starting with the same country code.

The sequence number is added to the encryption for three reasons: (I) it provides semantically secure encryption, which is important because the same PMSI can be encrypted into several challenges, (II) it prevents replay attacks, and (III) it provides a way for the SIM to check the integrity of the decrypted data (in 3G and 4G), effectively preventing fake challenge attacks.

The use of the sequence number SQN suggests the possibility of the SIM and authentication server getting out-of-sync. However, in the 3G and 4G case the sequence number is already verified before the decryption of the challenge. Furthermore, there is a protocol already in place to re-sync the sequence number. In the 2G case the sequence number is only needed to prevent replay attacks, so the set of acceptable sequence numbers can be big enough to assure that the home network's SQN does not get out-of-sync with the SIM's. An attacker could attempt to start many fake sign-on sessions with a victim's current PMSI in a different cell, to force the home networks SQN out-of-sync with the SIM. However, this can be detected by the home network (many incoming requests for authentication parameters without a location update following it), so this can be counteracted by delaying the handing out of authentication parameters, when such an attack is detected.

In the original protocol for 2G networks no sequence numbers are used during authentication. Even worse, there is no mutual authentication in 2G networks, which means an active attacker can simply insert authentication challenges for the phone. Our approach has the added benefit of preventing such attacks, as both replay attacks on challenges or insertion of fake challenges can be detected (due to the sequence number and MAC).

As it is not possible for an attacker to create a correct PMSI update without the secret key κ, and the sequence numbers cannot be forced out-of-sync, there is no increased risk for a Denial-of-Service attack. A DoS attack could still be used to prevent the SIM from getting new PMSIs. However, this is essentially the same as simply preventing all service to the user, and as soon as the SIM connects to a genuine network, this protocol provides it with a new PMSI.

When the secret key κ is compromised, an adversary is able to track users in the

future. Moreover, our method does not provide perfect forward secrecy [103] and lacks protection against analysis of historical recordings of previous PMSI updates if the secret key κ is known. However, recovery of key κ requires considerable computational power, which can most-likely better be utilised to attack other cryptographic primitives used in 2G and 3G protocols.

Even with our solution in place, users might still have to take additional measures to make tracking harder, e.g. by disabling their WiFi and Bluetooth services, as these are also uniquely identifying.

Finally, our approach only protects from IMSI catching and not eavesdropping or Man-in-the-Middle attacks, which are also often referred to as IMSI catching. A powerful adversary might be able to forward and relay messages between the victim's phone and the genuine home network, while mounting various well-known cryptographic attacks on 2G networks [10, 9, 17, 25], as we discussed extensively in Chapter 3. In such scenario, we consider the security to be compromised since all 2G communication traffic can be observed, and location privacy attacks can be re-introduced through identification based on the contents of data transmissions. However, our solution hinders these attacks and others, such as sending malicious messages over the air, as these can no longer rely on the IMSI to identify their target.

8.5.1 Parameter choice

As was explained in Section 8.2.2 the IMSI is a 15 digit number, containing a three digit country code and a maximum of three digits for the home network code. This means there can be at most 10^{10} different IMSIs per provider. Therefore, we need at most 34 bits for the PMSI in a challenge. Phones send the IMSI in BCD encoding (4 bits per digit), but there is no reason to encode a PMSI encrypted inside the RAND in such an inefficient way. As the challenge is 16 bytes for all generations of mobile networks, this leaves us 94 bits to use for the counter SQN, which is 48 bit, in the case of 3G/4G. In the case of 2G, the remaining 94 bits would need to accommodate both the counter SQN and the MAC (M_κ). This means the 2G case can accommodate a SQN of 62 bits while allowing a M_κ of 32 bits. Table 8.1 provides a short overview of the bit lengths of different variables.

We introduced a new shared key for the encryption of the PMSIs: κ. While the existing shared key $\mathcal{K}$ could be reused for κ, we do not recommend this. It is good security design to use different keys for different functions. In fact, while we use κ for both encryption and MAC generation in the 2G solution, it is again good design to use two separate keys for this. The eventual choice for κ is naturally dependent on the choice of encryption and MAC scheme and as such it is not included in Table 8.1.

Since asking the authentication server for multiple challenges for an IMSI will give several encryptions of the same PMSI, with only an increased counter, the cipher used for encryption should be secure against related plaintext attacks. For the 3G and 4G networks, the AES blockcipher is advised for the implementation of the authentication functions [46]. AES is secure under related plaintext attacks, so simply reusing AES here would be enough. For 2G networks, the authentication functions are called A3/A8 and can be chosen by the provider. The suggested algorithm, called COMP128, is secret and proprietary. At least the first version is known to be weak [83]. Also in the 2G case, the choice for a the MAC algorithm (M_κ) is not trivial, with a 32 bit output and

a 96 bit input. One option would be to reuse 2G's authentication algorithm A3, which takes a 128 bit challenge and computes a 32 bit response, as M_K. So, a provider would have to see whether their implementation for A3/A8 would be secure for both the encryption of the PMSIs (so, related plaintexts) and the MAC generation. Then again, our fix demands an update of the SIM cards anyway, so updating the 2G authentication algorithms could be added at minimal cost.

8.5.2 Roll-out scenario & overhead

We discussed our solution with a major telecom provider, both in terms of feasibility and possible roll-out scenarios. They considered our solution as a possible fix to prevent (long term) tracking of customers.

Swapping all SIMs is a costly operation for a provider. The SIMs themselves are quite cheap, but the process of handing them out is costly. However, many SIM cards currently out in the field can be updated remotely through Over-The-Air (OTA) commands. Not all SIMs in the field that support such updates, accept updates to the authentication procedure, but we could not obtain the numbers of updateable SIMs. Still, since the presented solution is backwards compatible – a non-updated SIM will simply answer the challenge as it always did, without ever switching to the new PMSI – there are no big issues in rolling out our improvement to only a set of SIMs.

The overhead introduced by the additional computations in our solutions is expected to be very small for both the home network and SIM. We introduce one MAC generation and verification (only for 2G) and a symmetric encryption and decryption operation. Even for SIM cards with limited computational power this should not present a problem, as this is functionality that is already used in the original protocols.

Our solution introduces a small overhead for the access network, because a SIM that switches to a new PMSI will look like a completely new phone to the access network. Naturally, the SIM cannot sign off with the old PMSI first, as this would defeat the purpose of unlinkability. Networks are used to phones that do not sign off properly, and occasionally check whether phones registered in their area are still present. Wide adoption of our solution is likely to increase these numbers. The exact influence of this overhead is hard to estimate without field trials, though access networks are already resilient against high number of fast moving SIMs, making it unlikely that our change would significantly impact their workings.

8.5.3 Formal verification

To verify our solution formally we make use of ProVerif, an automatic cryptographic protocol verifier [19]. This tool is used to check for secrecy and authentication properties of security protocols, but can also be used to check privacy related properties [20]. The usual Dolev-Yao attacker model is assumed, where the attacker has complete control over the network but cannot break cryptography [41]. The analysis by ProVerif is sound but not complete, i.e. ProVerif returns no false positives (claiming that a property holds, though an attack still exists), but it might find invalid attacks. We modelled the original protocols and our proposed modified versions for both 2G and 3G/4G in the typed applied pi-calculus.[4] For these models we check whether an

[4] The models are available at `http://www.cs.bham.ac.uk/~deruitej/`

attacker can link different sessions of the AKA protocol that belong to the same user. This is done by starting two sessions by different users. After this, a third sessions is started and the attacker has to distinguish which of the two users started it. For the original protocols for 2G and 3G/4G, no unlinkability is provided and a valid attack is returned by ProVerif. Unlinkability is however proved by ProVerif for our solutions for these mobile technology generations. In addition to unlinkability, we also check authentication between the SIM and home network for 2G. As expected this fails for the original protocol, but holds when our solution is used. These proofs give confidence that our solutions for both the 2G and 3G/4G protocols actually introduce protection against IMSI catching – an attacker is no longer able to determine which sessions belong to which users – while keeping the original functionality of the protocols (authentication) intact.

8.6 Related work

We present a solution against one of the oldest practical attack against 3GPP networks: IMSI catching.

Independently from us, and largely concurrent, researchers from Royal Holloway developed a very similar solution [108] as was presented in this Chapter. Their work also uses the random challenge to transmit new IMSI values to the SIM, although secured in a different fashion where the new IMSI value is XORed with a session key, and the sequence number is included in a MAC. Both are then concatenated along with a 16 bit random number to create the random challenge. Their solution is only tailored towards 3G and 4G networks, without a similar solution for 2G networks or the formal verification, but otherwise seems to achieve the same results as the solution presented in this chapter.

In [160] Sung et al. propose a solution to prevent location tracing by the user's home provider. They achieve this by having the home provider offer sets of "virtual" SIM instances via some medium separate of the mobile network (e.g. via internet which the user can connect to via WiFi). Users choose a successor instance for their current SIM instance from this set, and at some point switch to the successor. Naturally, this solution does not support standard phone services, and can only use flat rate subscriptions. Furthermore, roaming is likely impossible for this solution, since the roaming charges cannot be billed to a specific individual. Still, it is an impressive solution, which even though it is not aimed at preventing IMSI catching attacks, it will hinder these in exactly the same way as our solution.

Besides preventing IMSI catching attacks, there are also approaches aimed at detecting IMSI catchers.

Dabrowski et al. [34] listed several indicators for the possible presence of IMSI catchers. They also created both a network of stationary measurement devices and an Android app, each capable of detecting IMSI catchers. Karsten Nohl and others similarly introduced an Android app capable of detecting IMSI catchers [158]. This app warns the user for the likely presence of IMSI catchers, where the term IMSI catchers refers to the more inclusive meaning of the word and also includes eavesdropping. Warning users of these attacks is very valuable and provides a basis for our claims that these attacks are prevalent, but it does not prevent such attacks as our solution

does.

We know of no other work specifically targeting IMSI catching, though there is a lot of work regarding privacy issues in mobile telephony in general.

While not addressing IMSI catching, there has been research into other location privacy issues caused by the 3GPP protocols. Arapinis et al. [4] used ProVerif to formally verify the 3G specifications. This revealed two new privacy issues; linkability of the IMSI to the TMSI using paging of mobile phones and a traceability attack that was detailed in Section 8.2.4. They also present solutions for both attacks; encrypting the IMSI in a paging command with a shared session key, and encrypting the response of a failed authentication request with a public key of the provider. Interestingly, our solution would diminish the effects of the IMSI - TMSI linkability, as a PMSI refresh will appear to the access network as a new SIM arriving, which causes the assigning of a new TMSI. Our presented solution therefore negates the need for encrypting the IMSI. The solution for the traceability attack is still required whether or not our solution is implemented.

Hahn et al. [92] suggest a different solution for the Arapinis traceability attack. The response of a failed authentication request is essentially encrypted with the new symmetric session key instead of the public key solution offered by Arapinis et al. This solution might be more efficient, though the consequences of switching to the session key provided by a re-played challenge are not deeply explored.

In other work Arapinis et al. looked specifically at the TMSI reallocation protocol [5], both formally and experimentally. Both the specifications and common implementations were found to be having problems leading to privacy attacks. These privacy attacks stem from possible linkability between different TMSIs or recovering the link between an IMSI and a TMSI. These attacks mostly have even simpler counter measures than our solution against IMSI catching. However, again the implementation of our solution would also prevent these TMSI attacks. If the SIM changes its IMSI (PMSI), the TMSI will get changed as well.

Han et al. [94] and Chen et al. [30], describe some issues with untrusted femtocells. Femtocells are essentially small, low power cell towers consumers can buy to enhance the reception of cellular communication in specific spots. They were discussed in detail in Chapter 5. Both publications introduce additional mutual authentication steps to assure the communication with a registered femtocell and prevent a MitM attack. However, both approaches can only be used after a phone has already identified itself to the femtocell, and thus does not prevent IMSI catching.

8.7 Conclusions

We present a solution against IMSI catching attacks that fits within the current standards, making the change transparent for intermediate networks and backwards compatible. We propose the introduction of changing pseudonyms (PMSIs), to use for identification. This solution can be deployed within the current architecture by an individual provider, as it controls the only two entities that need adapting: the SIM cards providing the IMSI and the authentication server within the home network. Additionally, this solution provides the SIM with a way to verify whether a given chal-

lenge was generated by its home network, adding a form of mutual authentication to the traditionally weak SIM-only authentication of 2G networks. Using the protocol verifier ProVerif, we verified that our solution indeed provides unlinkability between succeeding pseudonyms without harming the original verification.

In essence we bring the effectiveness of IMSI catching down to the effectiveness of TMSI catching: learning a temporary id, which is unlinkable to future temporary ids. This prevents long term tracking of individuals, as well as tracing individuals returning to specific locations. Essentially, our proposal provides k-anonymity for users [161], where k is the expected number of users from a specific home network provider and country in a specific location.

Our solution need not interfere with lawful interception uses of IMSI catching, as authorities can still go to a provider with a caught PMSI together with the time of the catching. Based on this information the provider should be able to retrieve the corresponding account from its logs. Our solution, however, does interfere with unlawful IMSI catching.

For future cellular communication standards (5G and on) the issue of IMSI catching could be easily tackled using a-symmetric cryptography, which we could not use because the increase in resulting ciphertext sizes would not fit inside the current message specifications. However, it is still unclear whether the newer standards will introduce such a fix and even if they do, eventual roll-out is still far away. Even worse, the current roll-out of second, third and fourth generation cellular communication will not simply be replaced, and will likely remain functional for the foreseeable future. The solution presented in this chapter could remedy the issue of IMSI catching in the current systems. We are currently working with the Fraud and Security Architecture Group (FSAG) of the GSM Association (GSMA) to see if we can get this solution into the current 3GPP specifications. Hopefully, our fix will contribute to finally solve the privacy and traceability attacks present in over 25 years of 3GPP protocols.

Chapter 9

Conclusions and future work

In this thesis we investigated how secure mobile communication is in practice. More in particular, we looked at the security on the wireless link of the 3GPP-style of cellular networks from various perspectives, ranging from the abstract protocols, the cryptography used, down to the actual software in mobile phones and – on the network side – in femtocells and the services deployed on top of the networks. Finally, we also looked at the possibility to improve the current situation, when it comes to location privacy with respect to IMSI catching.

9.1 Overview and discussion

Most chapters focused on one specific abstraction layer of the security of mobiles communication. Specifically, Chapters 3 and 4 looked at the protocol and cryptography layer, Chapters 5 and 6 looked at the security of the actual implementations of said protocols, Chapter 7 looked at the security offered by using mobile phones as an additional factor in authentication and Chapter 8 investigated the possibility of improving the current security status.

Starting on the layer of the protocols and wireless signals, Chapter 3 discussed confidentiality and authentication on the wireless link. After reviewing well-known attacks on GSM we discussed the difficulty of performing these attacks in practice using readily available hardware and software. The relevance of this discussion is dependent on the attacker model: listening in on mobile phone calls is clearly possible for attackers with the financial resources to buy equipment that performs this attack, or by the telecom operator. However, this discussion focused on the feasibility of these attacks by an attacker with limited financial resources. All of these attacks do not work out-of-the-box, using readily available hardware and software and typically require some investment on the programming side. So, while definitely feasible for a skilled attacker, these attacks are quite complicated to perform for an attacker of 'script kiddy' level.

Chapter 3 also analysed the weaknesses at the core of the attacks on GSM, possible countermeasures against these weaknesses, and the countermeasures employed by the newer generation protocols. We saw that despite substantial improvements on

the security offered by UMTS and LTE protocols – mainly the use of better cryptography and mutual authentication – some weaknesses remain, such as the support for a null-cipher. Irrespective of exactly which 3GPP protocol is used, much of the offered security is dependent on provider settings, which makes it dangerous to apply general statements on the security of these 3GPP protocols to actual networks.

Overall, disabling GSM on your mobile phone will protect against the attacks against confidentiality on the wireless link.

Chapter 4 focused on the cryptography used in mobile networks, specifically discussing Time-Memory Trade-Off (TMTO) attacks on stream ciphers. The publicly demonstrated eavesdropping attack against GSM in 2011 used a TMTO attack called Fuzzy Rainbow Tables. We provided the first costs analysis (in both time and memory) of the Fuzzy Rainbow Tables for multiple data samples. We also performed our analysis on well established TMTO methods and compared the results.

Our analysis showed an advantage of the so-called Distinguished Point attack, especially for its speed. However, many aspects of these attacks remain unresearched, such as possible parallelism benefits, and the differences are small enough that the particularities of a specific attack scenario and cipher can make other TMTO attacks the better choice.

Chapters 5 and 6 focused on the implementation layer of the wireless protocols. If the attacker can compromise an end point there is no need to break the secure link. Even worse, a serious compromise of an end point will often give the attacker more capabilities than he ever had on the link alone. These chapters focused on the different sides of the wireless connection, femtocells in Chapter 5 and handsets in Chapter 6.

Chapter 5 looked into the security of femtocells; low-margin, consumer-owned cell towers. Introducing these devices in consumers' homes could suddenly make attacks against a provider's core network much easier, though some of these dangers could be alleviated by designing the 3GPP networks to not naively trust the cell towers. Therefore the theoretical analysis in this chapter assessed the damage that could come from a compromised femtocell in a setting where the provider puts minimal trust in said femtocell. We showed that in this setting some high impact attacks, such as eavesdropping, could be avoided, while other attacks, such as retrieving the secret relation between phone number and unique identifier, are still possible. In a practical analysis we assessed the security offered by an actual femtocell, through attempts of — and success in — compromising it.

Chapter 6 looked into the protocol implementation of mobile handsets through the testing technique known as fuzzing. With protocols as complicated and baroque as the 3GPP protocols, the chances of mistakes, including security vulnerabilities, in the specific implementation are high. We found that it is very easy to find a surprisingly large variety of bugs with fuzzing, which seems to indicate that manufacturers did not incorporate fuzz testing in their development cycle. Indeed, about a year after this research the biggest manufacturer of mobile protocol chips, Qualcomm, started using fuzz testing and ended up rebuilding its' software stack from scratch [188].

Despite the small number of manufacturers making baseband implementations we found a surprisingly high diversity in behaviour between baseband implementations even from the same manufacturer. This means that attacks found against a specific implementations likely will not affect a large portion of the available mobile

phones. However, finding an attack in a particularly popular phone would still have a high impact, especially since these attacks can be performed wirelessly.

In Chapter 7 we focused on services deployed on top of mobile networks. Specifically, we discussed the different ways in which services bootstrap an additional authentication factor with the use of mobile phones, thereby implicitly trusting the authentication of the 3GPP protocols. Mobile phones offer an obvious improvement as an additional factor in authentication, especially for parties unwilling or unable to roll-out their own trusted hardware. However, this added security is not as strong as dedicated trusted hardware, and we conclude that it is not sufficient to solely rely on the security of the mobile network when authentication for security-sensitive services, such as server-based signing. Also, more and more services are offered from the mobile devices themselves; when the same device is used both to authenticate to a service and to access that service, then this device can no longer be considered an *additional* factor in the authentication.

The security shortcomings discussed in earlier chapters raise the question if some of these shortcomings can be alleviated without completely changing the infrastructure. In Chapter 8 we answer this for one specific attack, namely IMSI catching, an attack against location privacy prevalent in all currently active 3GPP protocols. The solution we propose can be introduced by individual providers, only requiring a change in their authentication server and SIM cards. This change would be completely transparent to all other parties and deployed network equipment. Additionally, the proposed solution even introduces mutual authentication to GSM's authentication protocol, thereby preventing several of the attacks discussed in Chapter 3.

9.2 Conclusions

In this thesis we have seen that members of the 3GPP family of cellular networks (GSM, GPRS, UMTS and LTE) have serious security issues on each of the abstraction levels, such as weaknesses in the abstract protocols and security vulnerabilities in the implementation of these protocols. This is not really surprising; the same has happened when looking at other communication technologies, such as the internet. However, in the case of the internet, vulnerable protocols are updated or replaced, weaknesses in implementations of these protocols are patched with some regularity, and additional security layers are deployed on top of the network stack to achieve security on the higher layers. In contrast, 3GPP networks only see major security updates with the introduction of a new generation, implementations are hardly ever patched, and only recently in the post-Snowden upheaval are we seeing a serious interest in deploying secure apps over the insecure networks. Moreover, internet protocols have seen a lot of public scrutiny over the last thirty years, while cellular networks are just starting to get serious attention.

Still, the security updates within the 3GPP protocols, while far between, have definitely addressed the most serious weaknesses. Especially with the move to UMTS, all eavesdropping and MitM attacks on the wireless connection seem thwarted. For security-sensitive transmissions it is therefore advisable to only use UMTS or LTE.

In recent years there has been a lot of research interest in GSM, especially from the hacker community, because of the relatively easy access afforded by open-source

software and a couple of cheap mobile phones. However, setting up a successful eavesdropping attack still requires a lot of effort and knowledge, as essential parts of the software were never released. For now, the community scrutinising GSM has remained small compared to the old phone phreaks. GSM's successors have received even less attention, which is not just due to the increased security offered by UMTS and LTE, but mainly due to a lack of available software to access their air interfaces. A fake cell tower in UMTS or LTE could for example still fuzz several messages before having to authenticate itself to the phone. As was the case with GSM, all it will take is an open-source implementation of UMTS or LTE cell towers for researchers to start looking hard at those protocols.

It would be tempting to disregard GSM completely, as an 20-odd year old technology that will be replaced by the newer technologies. This however might not be the case: GSM/GPRS coverage is far more extensive than the coverage of the newer protocols and GSM uses less power and is more efficient for voice calls. Also, a lot of wireless machine-to-machine communication still relies on GSM/GPRS, for example in some smart electricity meters. All this has prompted providers to speculate that newer protocols such as LTE will replace their direct predecessor (in the case of LTE, UMTS), but will still run alongside an active GSM/GPRS network. So, for the foreseeable future, GSM is here to stay. Within the Netherlands GSM could be getting some competition by the deployment of a nation-wide CDMA network in the 450 MHz band specifically for machine-to-machine communication in the critical infrastructures, such as the electricity grid [117].

Attacks against the network side would have much higher impact than attacks against single phones, and yet network equipment, such as cell towers, but also core network devices seem to have received little scrutiny. The relative ease with which fuzzing techniques reveal vulnerabilities on mobile phone implementations raises serious concerns on the reliability and security of the network equipment.

Finally, there is the possibility of improving the current situation, without waiting for the next generation. The network settings maintained by the providers offer room for improvement inside the current deployment. For instance, we are seeing a definite move to stronger encryption inside GSM, which prevents eavesdropping attacks. Security updates are also possible in the protocol layer, as long as they do not change anything in the core wireless protocols, so updates to the massively deployed cell phones and cell towers are not necessary. It is nice to observe that all the attacks we started with in Chapter 3 can be prevented just by only supporting the strongest encryption for GSM and by introducing our mutual authentication presented in Chapter 8. This last improvement recently received interest from the GSMA (the GSM Association) and might see actual rollout in the future.

9.3 Directions for future work

There are many areas within the field of security of mobile communication that are not covered within this thesis, but are definitely interesting to look at, such as satellite telephony, GSM-R, or CDMA which have not received much attention from the security community.

When just focusing on the security of 3GPP networks still many topics remain

uncovered by this thesis. Nearly all security research until now has focused on the wireless access network and handsets, which everyone has easy access to. The weaknesses found there inspire little confidence in the security of the core-network components. These might be harder to access, but successful attacks against them will have a much larger impact. With the advent of LTE and its all-IP back-end, access to these components might come within easy reach of attackers.

The many components within a modern smart-phone, such as a GPS module, WiFi, Blue Tooth, and NFC, all bring their own security implications. There can also be several security(capable) components within a phone, such as the SIM card and a secure element. There is a clear need to better understand the phone architecture, and how all these components are linked, to assess the security of mobile phones.

Finally, more and more security-sensitive services are usable from mobile devices, such as mobile banking apps and ticketing apps (which provide the possibility to buy tickets that provide access to venues by displaying a bar code or via the NFC-interface). Looking into strong security architectures for such apps, even with the untrusted hardware and software below it, is interesting to research.

Bibliography

[1] Agentschap Telecom. Homepage of the Dutch regulatory body for use of ether frequencies. http://www.agentschaptelecom.nl/. Last accessed April 2016. Cited on page 21.

[2] Airprobe. Project page on GSM debugging using a Nokia 3310. https://svn.berlin.ccc.de/projects/airprobe/wiki/tracelog and http://www.wammu.eu/. Last accessed April 2016. Cited on page 21.

[3] F. Aloul, S. Zahidi, and W. El-Hajj. Multi factor authentication using mobile phones. *International Journal of Mathematics and Computer Science*, 4:65–80, 2009. Cited on pages 127 and 128.

[4] M. Arapinis, L. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *CCS '12*, New York, NY, USA, 2012. ACM. Cited on pages 46, 77, 83, 85, 139, and 149.

[5] M. Arapinis, L. I. Mancini, E. Ritter, and M. Ryan. Privacy through pseudonymity in mobile telephony systems. In *NDSS*, 2014. Cited on pages 135 and 149.

[6] N. Asokan, G. Tsudik, and M. Waidner. Server-supported signatures. In *Computer Security – ESORICS'96*, volume 1146 of *LNCS*, pages 131–143. Springer, 1996. Cited on page 122.

[7] Asterisk. Open-source project implementing a software telephone exchange. http://www.asterisk.org/. Last accessed April 2016. Cited on page 22.

[8] S. Babbage. Improved "exhaustive search" attacks on stream ciphers. In *Security and Detection, 1995., European Convention on*, pages 161–166. IET, 1995. Cited on pages 50 and 53.

[9] E. Barkan and E. Biham. *Conditional Estimators: An Effective Attack on A5/1*, pages 1–19. Springer Berlin / Heidelberg, 2005. Cited on pages 32 and 146.

[10] E. Barkan, E. Biham, and N. Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. In *Advances in Cryptology - CRYPTO 2003*, volume 2729/2003, pages 600–616. Springer Berlin / Heidelberg, 2003. Cited on pages 29, 32, 35, 36, 50, and 146.

[11] E. Barkan, E. Biham, and A. Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 1–21. Springer Berlin / Heidelberg, 2006. Cited on pages 49, 50, 58, 64, 65, and 71.

[12] Barton Gellman and Ashkan Soltani for The Washington Post. NSA surveillance program reaches 'into the past' to retrieve, replay phone calls. http://wapo.st/0sV5pZ, 2014. Last accessed April 2016. Cited on page 7.

[13] BBC. A filmed demonstration of the eavesdropping attack on GSM. http://www.bbc.co.uk/news/technology-13066662, 2011. Last accessed April 2016. Cited on pages 29, 30, 32, and 37.

[14] F. Bellard. LTE Base Station Software Website. http://bellard.org/lte/. Last accessed April 2016. Cited on page 21.

[15] S. K. Belle, O. Haase, and M. Waldvogel. Callforge: Call anonymity in cellular networks. Technical report, KOPS - The Institutional Repository of the University of Konstanz, 2010. Cited on page 132.

[16] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin / Heidelberg, 2000. Cited on pages 50, 53, 71, and 72.

[17] A. Biryukov, A. Shamir, and D. Wagner. Real time cryptanalysis of A5/1 on a PC. In *Fast Software Encryption (FSE 2000)*, pages 1–18. Springer Berlin / Heidelberg, 2000. Cited on pages 50 and 146.

[18] Bladox. Website offering the turbo sim tools. http://www.bladox.com/. Last accessed April 2016. Cited on pages 24 and 25.

[19] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th Computer Security Foundations Workshop*. IEEE, 2001. Cited on page 147.

[20] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*. IEEE, 2005. Cited on page 147.

[21] M. Böck. Simulation chamber and method for setting off explosive charges contained in freight in a controlled manner. US Patent 14345697, Sept. 19 2012. Cited on page 133.

[22] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *CHES 2007*, volume 4727 of *LNCS*. Springer-Verlag, 2007. Cited on page 137.

[23] R. Borgaonkar, K. Redon, and J.-P. Seifert. Security analysis of a femtocell device. In *SIN '11*, New York, NY, USA, 2011. ACM. Cited on pages 77, 81, and 85.

[24] M. Briceno, I. Goldberg, and D. Wagner. An implementation of the GSM A3A8 algorithm. (specifically, COMP128.). http://www.scard.org/gsm/a3a8.txt, 1998. Last accessed April 2016. Cited on page 34.

[25] M. Briceno, I. Goldberg, and D. Wagner. A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms. http://cryptome.org/gsm-a512.htm (originally on www.scard.org), 1999. Last accessed April 2016. Cited on pages 13, 32, and 146.

[26] D. Burgess. Homepage of the OpenBTS project. http://openbts.org/. Last accessed April 2016. Cited on pages 14, 22, 39, 97, and 100.

[27] A. Cedillo Torres. GSM cell broadcast service security analysis. Master's thesis, Technical University Eindhoven, Kerckhoff's Master, The Netherlands, 2013. Cited on page 101.

[28] D. Chambers. *Femtocell Primer (2nd Edition)*. Lulu Enterprises Inc., 2010. Cited on page 77.

[29] Chaos Computer Club, January 2010. website for the airprobe project: https://svn.berlin.ccc.de/projects/airprobe/wiki. Cited on pages 20, 37, and 39.

[30] C.-M. Chen, Y.-H. Chen, Y.-H. Lin, and H.-M. Sun. Eliminating rouge [sic] femtocells based on distance bounding protocol and geographic information. *Expert Systems with Applications*, 41(2):426 – 433, 2014. Cited on page 149.

[31] T. Chothia and V. Smirnov. A traceability attack against e-passports. In *Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 20–34. Springer Berlin Heidelberg, 2010. Cited on page 84.

[32] Craig Timberg for The Washington Post. Feds to study illegal use of spy gear. http://www.washingtonpost.com/blogs/the-switch/wp/2014/08/11/feds-to-study-illegal-use-of-spy-gear/, 2014. Last accessed April 2016. Cited on page 133.

[33] Craig Timberg for The Washington Post. Tech firm tries to pull back curtain on surveillance efforts in washington. http://wapo.st/1qgzImt, 2014. Last accessed April 2016. Cited on page 133.

[34] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl. Imsi-catch me if you can: Imsi-catcher-catchers. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 246–255. ACM, 2014. Cited on page 148.

[35] David M'Raihi and Salah Machani and Mingliang Pei and Johan Rydell. *TOTP: Time-Based One-Time Password Algorithm*. Internet Engineering Task Force, 2011. RFC6238. Cited on page 112.

[36] C. De Canniere, O. Dunkelman, and M. Knežević. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *CHES 2009*, volume 5747 of *LNCS*. Springer-Verlag, 2009. Cited on page 137.

[37] G. de Koning Gans and J. de Ruiter. The SmartLogic Tool: Analysing and Testing Smart Card Protocols. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, pages 864–871, April 2012. Cited on page 22.

[38] D. Denning. *Cryptography and Data Security*, page 100. Addison-Wesley, Boston, 1992. Cited on pages 50 and 55.

[39] Digidentity. Download page of the signpad app. https://itunes.apple.com/nl/app/signpad-by-digidentity/id436783895. Last accessed April 2016. Cited on page 123.

[40] Digidentity. Certification practice statement. https://www.digidentity.eu/static/nl/voorwaarden/Certification%20Practice%20Statement%20L3.pdf, April 2011. Cited on page 123.

[41] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. Cited on pages 6 and 147.

[42] O. Dunkelman, N. Keller, and A. Shamir. A practical-time attack on the A5/3 cryptosystem used in Third Generation GSM telephony. *Report 2010/013*, 2010. http://eprint.iacr.org/. Cited on pages 32 and 44.

[43] M. Dworkin. Recommendation for block cipher modes of operation: The CMAC mode for authentication. *NIST Special Publication (800-38B)*, 38B:1–25, 2005. Cited on page 137.

[44] T. Engel. S60 Curse of Silence. *CCC Berlin*, 2008. Last accessed April 2016. Cited on pages 99 and 103.

[45] Erguler, Imran, Anarim, and Emin. A new cryptanalytic time-memory trade-off for stream ciphers. In *Computer and Information Sciences - ISCIS 2005*, volume 3733 of *Lecture Notes in Computer Science*, pages 215–223. Springer Berlin / Heidelberg, 2005. Cited on pages 71 and 72.

[46] ETSI. *Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm specification*, 2014. (3GPP TS 35.206 version 12.0.0 Release 12). Cited on pages 137 and 146.

[47] ETSI. *Digital cellular telecommunications system (Phase 2+);Universal Mobile Telecommunications System (UMTS);LTE;Mobile radio interface Layer 3 specification;Core network protocols; Stage 3*, 2015. (3GPP TS 24.008 version 12.8.0 Release 12). Cited on page 138.

[48] ETSI. *Universal Mobile Telecommunications System (UMTS);LTE;Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3*, 2015. 3GPP TS 24.301 version 12.7.0 Release 12. Cited on page 134.

[49] Ettus. Website of the Ettus company, selling USRPs. http://www.ettus.com/. Last accessed April 2016. Cited on pages 14, 20, 27, and 98.

[50] European Central Bank. Recommendations for the security of internet payments. Final version after public consulation. http://www.ecb.europa.eu/pub/pdf/other/recommendationssecurityinternetpaymentsoutcomeofpcfinalversionafterpc201301en.pdf, January 2013. Last accessed April 2016. Cited on pages 108 and 109.

[51] European Central Bank. Recommendations for the security of mobile payments. Draft document for public consulation. http://www.ecb.europa.eu/paym/cons/pdf/131120/recommendationsforthesecurityofmobilepaymentsdraftpc201311en.pdf, November 2013. Last accessed April 2016. Cited on page 108.

[52] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Mobile radio interface layer 3 specification*, 1998. EN 300 940 / GSM 04.08. Cited on page 134.

[53] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Security aspects*, 1998. EN 300 920 / GSM 02.09. Cited on page 81.

[54] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Specification of the SIM application toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface*, 1999. (3GPP TS 11.14 version 8.18.0 Release 1999). Cited on page 114.

[55] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS); 3G Security; Security Principles and Objectives*, 2001. 3GPP TS 33.120 version 4.0.0 Release 4. Cited on pages 79, 81, and 84.

[56] European Telecommunications Standards Institute, France. *Universal Mobile Telecommunications System (UMTS); 3G Security; Security Threats and Requirements.*, 2001. 3GPP TS 21.133 version 4.1.0 Release 4. Cited on page 84.

[57] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS); Formal Analysis of the 3G Authentication Protocol*, 2001. 3GPP TR 33.902 version 4.0.0, Release 4. Cited on pages 46, 79, 82, and 139.

[58] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface*, 2007. (3GPP TS 11.11 version 8.14.0 Release 1999). Cited on page 42.

[59] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2);Mobile Station - Base Stations System (MS - BSS) interface Data Link (DL) layer specification*, 2010. TS 44.006 v9.1.0. Cited on pages 41 and 43.

[60] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Specification of the GSM-MILENAGE algorithms: An example algorithm set for the GSM Authentication and Key Generation Functions A3 and A8*, 2012. 3GPP TS 55.205 version 11.0.0 Release 11. Cited on page 34.

[61] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Point-to-Point (PP) Short Message Service (SMS) support on mobile radio interface*, 2012. (3GPP TS 24.011 version 11.1.0 Release 11). Cited on page 92.

[62] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+);Universal Mobile Telecommunications System*

(UMTS);LTE;Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2;Document 1: UEA2 and UIA2 specifications, 2012. 3GPP TS 35.215 version 11.0.0 Release 11. Cited on page 44.

[63] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS); LTE; Security of Home Node B (HNB) / Home evolved Node B (HeNB)*, 2012. 3GPP TS 33.320 version 10.5.0 Release 10. Cited on pages 76, 79, and 82.

[64] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS); UTRAN architecture for 3G Home Node B (HNB); Stage 2*, 2012. 3GPP TS 25.467 version 11.0.0 Release 11. Cited on pages 76, 79, and 80.

[65] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS);LTE;3G Security;Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification*, 2012. (3GPP TS 35.202 version 11.0.0 Release 11). Cited on page 44.

[66] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS);LTE;3G Security;Specification of the 3GPP confidentialityand integrity algorithms;Document 1: f8 and f9 specification*, 2012. 3GPP TS 35.201 version 11.0.0 Release 11. Cited on page 44.

[67] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);Technical realization of the Short Message Service (SMS)*, 2013. (3GPP TS 23.040 version 11.5.0 Release 11). Cited on page 92.

[68] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+);UMTS;LTE;3G security;Security architecture*, 2013. 3GPP TS 33.102 version 11.5.0 Release 11. Cited on pages 44, 81, 84, 134, and 137.

[69] European Telecommunications Standards Institute. *Universal Mobile Telecommunications System (UMTS);LTE;Network Domain Security (NDS);Authentication Framework (AF)*, 2013. 3GPP TS 33.310 version 11.2.0 Release 11. Cited on page 19.

[70] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Mobile radio interface layer 3 specification; Radio Resource Control (RRC) protocol*, 2014. (3GPP TS 44.018 version 11.7.0 Release 11). Cited on page 43.

[71] Fairwaves. Google code site of the ClockTamer, an external clock compatible with the USRP. Originally on `https://code.google.com/p/clock-tamer/`, currently taken down, but still available via `https://web.archive.org/web/*/https://code.google.com/p/clock-tamer/`. Last accessed April 2016. Cited on page 22.

[72] Z. Fasel and M. Jakubowski. Website detailing how to root the Samsung femtocell. `http://rsaxvc.net/blog/2011/7/17/Gaining%20root%20on%20Samsung%20FemtoCells.html`. Cited on pages 77 and 85.

[73] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A Survey of Mobile Malware in the Wild. In *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '11, pages 3–14, New York, NY, USA, 2011. ACM. Cited on page 111.

[74] A. Fiat and M. Naor. Rigorous time/space tradeoffs for inverting functions. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 534–541. ACM, 1991. Cited on page 64.

[75] P. FIPS. Advanced encryption standard (AES). *National Institute for Standards and Technology (NIST)*, 197(1), 2001. Cited on page 137.

[76] P. FIPS. The keyed-hash message authentication code (hmac). *National Institute for Standards and Technology (NIST)*, 1:1–13, 2008. Cited on page 137.

[77] P. FIPS. Secure hash algorithm-3 (SHA-3) standard: Permutation-based hash and extendable-output functions. *National Institute for Standards and Technology (NIST)*, 202(0), 2014. Cited on page 137.

[78] P. Flajolet and A. Odlyzko. Random mapping statistics. In *Advances in Cryptology – EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 329–354. Springer Berlin Heidelberg, 1990. Cited on page 64.

[79] Forum of Europe and Supervisory Authorities for Electronic Signature (FESA). Public statement on server based signature services. `http://www.fesa.eu/public-documents/PublicStatement-ServerBasedSignatureServices-20051027.pdf`, October 2005. Last Accessed April 2016. Cited on page 126.

[80] S. Gielen. Source code repository of the simparser.pl tool. `https://github.com/sgielen/simparser`. Last accessed April 2016. Cited on page 22.

[81] S. Gielen. Sim toolkit in practice. Bachelor's thesis, Radboud University Nijmegen, The Netherlands, 2012. `http://www.cs.ru.nl/bachelorscripties/2012/Sjors_Gielen___3003698___SIM_Toolkit_In_Practice.pdf`. Cited on page 24.

[82] GNU Radio. Homepage of the GNU Radio project. `http://gnuradio.org/`. Last accessed April 2016. Cited on pages 20 and 97.

[83] I. Goldberg and M. Briceno. Gsm cloning. `http://www.isaac.cs.berkeley.edu/isaac/gsm.html`. Cited on page 146.

[84] N. Golde, K. Redon, and R. Borgaonkar. Weaponizing femtocells: the effect of roque devices on mobile telecommunication. In *NDSS '12*. The Internet Society, 2012. Cited on pages 77, 81, 82, 84, 85, and 133.

[85] S. O. Goldman, R. E. Krock, K. F. Rauscher, and J. P. Runyon. Mobile forced premature detonation of improvised explosive devices via wireless phone signaling. US Patent 7552670, June 30 2009. Cited on page 133.

[86] J. Golic. Cryptanalysis of alleged a5 stream cipher. In *EUROCRYPT 1997*, pages 239–55. Springer, 1997. http://jya.com/a5-hack.htm. Cited on pages 32, 50, 53, and 69.

[87] Google. Google authenticator, open source project. http://code.google.com/p/google-authenticator/. Cited on page 111.

[88] GSM Map. Project tracking the implementation of GSM eavesdropping countermeasures across Europe. http://gsmmap.org/. Last accessed April 2016. Cited on pages 32, 41, 42, and 43.

[89] GSMA. GSM-Association: data and analysis for the mobile industry. https://gsmaintelligence.com/. Last accessed April 2016. Cited on pages 4 and 92.

[90] GSMA. Mobile network PWS and the rise of cell-broadcast. www.gsma.com/mobilefordevelopment/wp-content/uploads/2013/01/Mobile-Network-Public-Warning-Systems-and-the-Rise-of-Cell-Broadcast.pdf. Cited on page 83.

[91] P. Gühring. Concepts against man-in-the-browser attacks. http://www.cacert.at/svn/sourcerer/CAcert/SecureClient.pdf, 2007. Cited on page 108.

[92] C. Hahn, H. Kwon, D. Kim, K. Kang, and J. Hur. A privacy threat in 4th generation mobile telephony and its countermeasure. In *Wireless Algorithms, Systems, and Applications*, volume 8491 of *LNCS*. Springer, 2014. Cited on page 149.

[93] N. Haller. The S/KEY one-time password system. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*. Springer, 1995. Cited on page 112.

[94] C.-K. Han, H.-K. Choi, and I.-H. Kim. Building femtocell more secure with improved proxy signature. In *GLOBECOM IEEE*, December 2009. Cited on pages 77 and 149.

[95] M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401 – 406, Jul 1980. Cited on pages 50, 53, and 54.

[96] J.-H. Hoepman, E. Hubbers, B. Jacobs, M. Oostdijk, and R. Wichers Schreur. Crossing borders: Security and privacy issues of the european e-passport. In *Advances in Information and Computer Security*, volume 4266 of *LNCS*, pages 152–167. Springer Berlin Heidelberg, 2006. Cited on page 139.

[97] B. Hond. Fuzzing the GSM protocol. Master's thesis, Radboud University Nijmegen, Kerckhoff's Master, The Netherlands, 2011. Cited on page 101.

[98] J. Hong. The cost of false alarms in Hellman and rainbow tradeoffs. *Designs, Codes and Cryptography*, 57:293–327, 2010. Cited on page 64.

[99] J. Hong, K. Jeong, E. Kwon, I.-S. Lee, and D. Ma. Variants of the distinguished point method for cryptanalytic time memory trade-offs. In *Information Security Practice and Experience*, volume 4991 of *Lecture Notes in Computer Science*, pages 131–145. Springer Berlin / Heidelberg, 2008. Cited on page 71.

[100] J. Hong and S. Moon. A comparison of cryptanalytic tradeoff algorithms. *Journal of Cryptology*, 26(4):559–637, 2013. Cited on page 71.

[101] A. Huurdeman. *The Worldwide History of Telecommunications*. Wiley-IEEE Press, August 2003. Cited on pages 1, 2, and 16.

[102] IBM Labs. Made in IBM labs: Two-factor security for mobile transactions. http://www-03.ibm.com/press/us/en/pressrelease/42181.wss. Last accessed April 2016. Cited on page 128.

[103] D. P. Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, 1996. Cited on page 146.

[104] Jeremy Scahill and Glenn Greenwald for The Intercept. The NSA's Secret Role in the U.S. Assassination Program. https://theintercept.com/2014/02/10/the-nsas-secret-role/, 2014. Last accessed April 2016. Cited on pages 6 and 133.

[105] Jeremy Scahill and Josh Begley for The Intercept. The great SIM heist. https://theintercept.com/2015/02/19/great-sim-heist/, 2015. Last accessed April 2016. Cited on page 7.

[106] M. R. Joseph Boccuzzi. *Femtocells: Design & Application*. McGraw Hill Professional, 2010. Cited on page 77.

[107] Kadhim Shubber for Wired magazine. Tracking devices hidden in London's recycling bins are stalking your smartphone. http://www.wired.co.uk/news/archive/2013-08/09/recycling-bins-are-watching-you, 2013. Last accessed April 2016. Cited on page 133.

[108] M. S. A. Khan and C. J. Mitchell. Improving air interface user privacy in mobile telephony. In *Security Standardisation Research*, pages 165–184. Springer, 2015. Cited on page 148.

[109] B.-I. Kim and J. Hong. Analysis of the non-perfect table fuzzy rainbow tradeoff. *Report 2012/612*, 2012. Cited on page 71.

[110] B.-I. Kim and J. Hong. Analysis of the non-perfect table fuzzy rainbow tradeoff. In C. Boyd and L. Simpson, editors, *Information Security and Privacy*, volume 7959 of *Lecture Notes in Computer Science*, pages 347–362. Springer Berlin Heidelberg, 2013. Cited on pages 71 and 72.

[111] A. Kircansk and A. M. Youssef. On the sliding property of SNOW 3G and SNOW 2.0. *Information Security, IET*, 5(4):199–206, December 2011. Cited on page 44.

[112] J. Krhovják, O. Siler, P. Leyland, and J. Kur. Tmto attacks on stream ciphers – theory and practice. *Security and Protection of Information 2011*, pages 66–78, 2011. ISBN 978-80-7231-777-6. Cited on page 71.

[113] R. Kuipers and A. Takanen. Fuzzing embedded devices. *GreHack 2012*, page 38, 2012. Cited on page 96.

[114] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM*, 24:770–772, November 1981. Cited on page 112.

[115] S. Lee, E. L. Wong, D. Goel, M. Dahlin, and V. Shmatikov. πbox: A platform for privacy-preserving apps. In *NSDI*, 2013. Cited on page 132.

[116] Y. K. Lee, L. Batina, and I. Verbauwhede. Privacy challenges in rfid systems. In *The Internet of Things*. Springer New York, 2010. Cited on page 139.

[117] Liander. CDMA: the wireless part of Alliander's third net (telecom). YouTube video: `https://youtu.be/JlL0WDK8YKk`. Last accessed April 2016. Cited on page 154.

[118] M. Mannan and P. Oorschot. Using a personal device to strengthen password authentication from an untrusted computer. In *Financial Cryptography and Data Security*, volume 4886 of *LNCS*, pages 88–103. Springer, 2007. Cited on pages 127 and 128.

[119] M. Meeker and L. Wu. Internet trends 2013. Technical report, KPCB, 2013. `http://www.kpcb.com/insights/2013-internet-trends`. Cited on page 110.

[120] U. Meyer and S. Wetzel. A man-in-the-middle attack on UMTS. In *The 3rd ACM workshop on Wireless security*. ACM, 2004. Cited on page 131.

[121] Michael Sutton and Adam Greene and Pedram Amini. Collection of fuzzing software. `http://fuzzing.org/`. Last accessed April 2016. Cited on page 98.

[122] S. Muller and D. Hulton. The A5 cracking project. In *Presentation at Chaos Communication Camp 2007*, 2007. `http://video.google.com/videoplay?docid=8955054591690672567`. Cited on page 38.

[123] C. Mulliner, R. Borgaonkar, P. Stewin, and J.-P. Seifert. SMS-based one-time passwords: Attacks and defense. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 7967 of *LNCS*, pages 150–159. Springer, 2013. Cited on pages 110, 111, and 128.

[124] C. Mulliner, N. Golde, and J.-P. Seifert. SMS of Death: From Analyzing to Attacking Mobile Phones on a Large Scale. In *USENIX Security Symposium*, 2011. Cited on pages 99 and 103.

[125] C. Mulliner and C. Miller. Fuzzing the Phone in your Phone. *Presentation at Black Hat USA*, June 2009. Cited on pages 99 and 103.

[126] C. Mulliner and C. Miller. Injecting SMS Messages into Smart Phones for Security Analysis. In *Proceedings of the 3rd USENIX Workshop on Offensive Technologies (WOOT)*, Montreal, Canada, August 2009. Cited on pages 97, 99, and 103.

[127] C. Mulliner and G. Vigna. Vulnerability Analysis of MMS User Agents. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Miami, FL, December 2006. Cited on page 103.

[128] S. Munaut. IMSI detach DoS, April 2001. Cited on page 84.

[129] G. J. Myers. *The Art of Software Testing*. John Wiley & Sons, 1979. Cited on page 95.

[130] K. Nohl. A5/1 decrypt website. http://opensource.srlabs.de/projects/a51-decrypt/. Last accessed April 2016. Cited on pages 38, 50, and 69.

[131] K. Nohl. Cracking A5 GSM encryption. Presented at HAR 2009 in Vleuten The Netherlands, https://har2009.org/program/events/187.en.html, August 2009. Cited on page 58.

[132] K. Nohl. Attacking phone privacy. *Presented at Blackhat 2010*, 2010. https://srlabs.de/blog/wp-content/uploads/2010/07/Attacking.Phone_.Privacy_Karsten.Nohl_1.pdf. Cited on pages 14, 29, and 37.

[133] K. Nohl. website with tracker files for the A5/1 tables, January 2010. http://reflextor.com/torrents/. Cited on page 38.

[134] K. Nohl and S. Munaut. Wideband gsm sniffing. presented at 26C3 in Berlin, http://events.ccc.de/congress/2010/Fahrplan/events/4208.en.html, December 2010. Cited on page 50.

[135] K. Nohl and C. Paget. Gsm - srsly? presented at 27C3 in Berlin, http://events.ccc.de/congress/2009/Fahrplan/attachments/1519_26C3.Karsten.Nohl.GSM.pdf, December 2009. Cited on page 30.

[136] P. Oechslin. Making a faster cryptanalytic timememory trade-off. *Advances in Cryptology (CRYPTO'03)*, 2003. Cited on pages 50 and 56.

[137] M. Oostdijk and M. Wegdam. Mobile PKI, A technology scouting for security and use of mobile authentication technologies. In *Novay & SurfNet*, December 2009. http://www.terena.org/news/community/download.php?news_id=2528. Cited on page 113.

[138] OpenRCE. Code archive of the sulley fuzzing framework. https://github.com/OpenRCE/sulley. Last accessed April 2016. Cited on page 98.

[139] Osmocom. Homepage of the OsmocomBB project. http://bb.osmocom.org/. Last accessed March 2016. Cited on pages 21 and 38.

[140] Osmocom. Homepage of the SIMtrace project. http://bb.osmocom.org/trac/wiki/SIMtrace. Last accessed April 2016. Cited on page 27.

[141] P1Security. website detailing a fuzzing product for telco core-networks. http://www.p1sec.com/corp/products/p1-telecom-fuzzer-ptf/. Last accessed April 2016. Cited on pages 85 and 104.

[142] C. Paget. Practical Cellphone Spying. *Presented at Blackhat 18*, 2010. http://www.tombom.co.uk/blog/?p=262. Cited on page 35.

[143] T. Perrin, L. Bruns, J. Moreh, and T. Olkin. Delegated cryptography, online Trusted Third Parties, and PKI. In *1st Annual PKI Research Workshop*, pages 97–116, 2002. Cited on page 122.

[144] Radboud Universiteit Nijmegen en Price Waterhouse Coopers. Risicoanalyse EPD-DigiD naar aanleiding van de A5/1 kwetsbaarheid in GSM, 2010. http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2010/06/30/risicoanalyse-epd-digid/meva-3019667b.pdf. Cited on pages 10, 107, and 111.

[145] R. Rajavelsamy, J. Lee, and S. Choi. Towards security architecture for Home (evolved) NodeB: challenges, requirements and solutions. *Security and Communication Networks*, 4(4):471–481, 2011. Cited on pages 77 and 81.

[146] RebelSim. Phone to simcard communication monitoring system. http://rebelsimcard.com/fully-assembled-gsm-umts-cdma-simcard-and-mobile-phone-hex-scan-en.html. Last accessed April 2016. Cited on page 22.

[147] G. D. Robson. The origins of phreaking. In *Blacklisted! 411*, volume 6, April 2004. Cited on page 1.

[148] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems, Second Edition*, chapter 20. Wiley Computer Publishing, 2008. Cited on page 2.

[149] RTL-SDR. Homepage of the RTL-SDR project. http://www.rtl-sdr.com. last accessed April 2016. Cited on page 27.

[150] Ryan Gallagher for The Intercept. Operation socialist. https://theintercept.com/2014/12/13/belgacom-hack-gchq-inside-story/, 2014. Last accessed April 2016. Cited on page 6.

[151] S21sec. Blog post detailing the ZEUS Man-in-the-Mobile malware. http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html. Last accessed March 2016. Cited on page 111.

[152] V. Segura and J. Lahuerta. Modeling the economic incentives of ddos attacks: Femtocell case study. In *EISP 2010*. Springer US, 2010. Cited on page 77.

[153] M. Sherr, E. Cronin, S. Clark, and M. Blaze. Signaling vulnerabilities in wiretapping systems. *Security Privacy, IEEE*, 3(6):13–25, Nov 2005. Cited on page 2.

[154] Siraj Datoo for The Guardian. How tracking customers in-store will soon be the norm. http://gu.com/p/3ym4v/sbl, 2014. Last accessed April 2016. Cited on page 133.

[155] Small Cell Forum. Homepage of the Small Cell Forum. http://www.smallcellforum.org/. Last accessed April 2016. Cited on pages 75 and 76.

[156] A. Smolen. Twitter Blog post introducing the new 2nd factor authentication features of the Twitter app. https://blog.twitter.com/2013/login-verification-on-twitter-for-iphone-and-android. Last accessed March 2016. Cited on pages 112 and 113.

[157] G. S. Solutions. website offering several mobile telephony surveilance products. http://www.global-security-solutions.com/GSAudioSurv.html. Last accessed March 2016. Cited on pages 30, 37, and 42.

[158] SR Labs. Snoopsnitch website. https://opensource.srlabs.de/projects/snoopsnitch. Last accessed April 2016. Cited on page 148.

[159] D. Strobel. IMSI catcher. *Chair for Communication Security, Ruhr-Universität Bochum*, page 14, 2007. Cited on page 133.

[160] K. Sung, B. N. Levine, and M. Liberatore. Location privacy without carrier cooperation. In *IEEE Workshop on Mobile Security Technologies, MOST*, 2014. Cited on page 148.

[161] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002. Cited on pages 145 and 150.

[162] Sysmocom. Webshop of Sysmocom products. http://shop.sysmocom.de/. Last accessed April 2016. Cited on page 27.

[163] THC. THC website detailing an attack against a Vodafone SureSignal femtocell. http://wiki.thc.org/vodafone. Cited on pages 77, 81, and 85.

[164] The 3GPP consortium. Specification portal. http://www.3gpp.org/specifications. Last accessed April 2016. Cited on page 30.

[165] The European Parliament and Council. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures, 1999. OJ L 13 of 19.1.2000. Cited on page 118.

[166] The European Parliament and Council. Commission Decision 2003/511/EC on the publication of reference numbers of generally recognised standards for electronic signature products in accordance with Directive 1999/93/EC of the European Parliament and of the Council, July 2003. (notified under document number C(2003) 2439). Cited on pages 120 and 121.

[167] The European Parliament and Council. Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC, 2014. OJ L 257, 28.8.2014. Cited on pages 119, 122, and 129.

[168] Tollfreenumber.ORG. Website on the life and invention of Almon Strowger. http://www.strowger.net/. Last accessed March 2016. Cited on page 1.

[169] Trustwave. Announcement of the Samsung femtocell. https://www.trustwave.com/pressReleases.php?n=012810. Cited on pages 77 and 85.

[170] J.-K. Tsay and S. F. Mjølsnes. A vulnerability in the UMTS and LTE authentication and key agreement protocols. In *MMM-ACNS'12*. Springer-Verlag, 2012. Cited on page 82.

[171] TSB - Telecommunication Standardization Bureau of ITU. *Mobile Network Codes (MNC) for the international identification plan for public networks and subscriptions*, 2014. (According to Recommendation ITU-T E.212 (05/2008))(Postition on 15 July 2014). Cited on page 145.

[172] F. van den Broek. Catching and Understanding GSM Signals. Master's thesis, Radboud University Nijmegen, 2010. Cited on page 30.

[173] F. van den Broek. Eavesdropping on GSM: state-of-affairs. In *5th Benelux Workshop on Information and System Security (WISSec 2010)*, November 2010. Cited on pages 9 and 29.

[174] F. van den Broek. GSM security: Fact and fiction. Presented at BruCON 2010 `http://2010.brucon.org/`, September 2010. Cited on page 9.

[175] F. van den Broek, B. Hond, and A. Cedillo Torres. Security Testing of GSM Implementations. In *Engineering Secure Software and Systems*, volume 8364 of *Lecture Notes in Computer Science*, pages 179–195. Springer International Publishing, 2014. Cited on pages 10 and 91.

[176] F. van den Broek and E. Poll. A Comparison of Time-Memory Trade-Off Attacks on Stream Ciphers. In A. Youssef, A. Nitaj, and A. Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *Lecture Notes in Computer Science*, pages 406–423. Springer Berlin Heidelberg, 2013. Cited on pages 9, 49, and 72.

[177] F. van den Broek and E. Poll. Digitale handtekeningen: nieuwe technologie en nieuwe wet- en regelgeving. *Privacy & Informatie*, February 2014. Cited on pages 10 and 107.

[178] F. van den Broek and R. W. Schreur. Femtocell Security in Theory and Practice. In H. Riis Nielson and D. Gollmann, editors, *Secure IT Systems*, volume 8208 of *Lecture Notes in Computer Science*, pages 183–198. Springer Berlin Heidelberg, 2013. Cited on pages 9 and 75.

[179] F. van den Broek, R. Verdult, and J. de Ruiter. Defeating IMSI catchers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 340–351. ACM, 2015. Cited on pages 11 and 131.

[180] R. van Rijswijk and J. van Dijk. tiqr: a novel take on two-factor authentication. In *Proceedings of the 25th international conference on Large Installation System Administration*. USENIX Association, 2011. Cited on pages 114 and 128.

[181] R. van Rijswijk-Deij and E. Poll. Using Trusted Execution Environments in Two-factor Authentication: comparing approaches. In *Proceedings Open Identity Summit 2013. Open Identity Summit (OID-2013), September 9-11, Kloster Banz, Germany*, volume 223 of *Lecture Notes in Informatics, LNI*, pages 20–31. Gesellschaft für Informatik, Springer, 2013. Cited on page 114.

[182] Vasco. Website showing the cronto multi-factor authentication solution by vasco. `http://www.cronto.com/`. Last accessed April 2016. Cited on page 114.

[183] C.-E. Vintila, V.-V. Patriciu, and I. Bica. Security analysis of lte access network. In *ICN 2011, The Tenth International Conference on Networks*, pages 29–34, 2011. Cited on page 47.

[184] M. Vuontisjärvi and T. Rontti. SMS Fuzzing. *Codenomicon whitepaper*, 2011. http://www.codenomicon.com/resources/whitepapers/codenomicon_wp_SMS_fuzzing_02_08_2011.pdf. Cited on pages 99, 103, and 104.

[185] T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch. The zurich trusted information channel – an efficient defence against man-in-the-middle and malicious software attacks. In *Trusted Computing - Challenges and Applications*, volume 4968 of *Lecture Notes in Computer Science*, pages 75–91. Springer Berlin Heidelberg, 2008. Cited on page 127.

[186] R.-P. Weinmann. The baseband apocalypse. *Presentation at 27th Chaos Communication Congress Berlin*, 2010. Cited on page 104.

[187] R.-P. Weinmann. Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks. In *WOOT*, pages 12–21, 2012. Cited on page 104.

[188] R.-P. Weinmann. Baseband Exploitation in 2013. *Presentation at 30th Chaos Communication Congress Berlin*, 2013. Cited on pages 104 and 152.

[189] H. Welte. Homepage of the OpenBSC project. http://openbsc.osmocom.org/trac/wiki/OpenBSC. Last accessed April 2016. Cited on pages 27, 39, and 97.

[190] H. Welte. Homepage of the OsmoBTS project. http://openbsc.osmocom.org/trac/wiki/OsmoBTS. Last accessed April 2016. Cited on page 98.

[191] H. Welte. Anatomy of contemporary GSM cellphone hardware. http://laforge.gnumonks.org/papers/gsm_phone-anatomy-latest.pdf, 2010. Cited on page 97.

[192] J. Zhang and G. de la Roche, editors. *Femtocells: Technologies and Deployment*. John Wiley & Sons, Ltd, 2009. Cited on page 77.

Summary

This thesis examines the security of mobile communication technology, with a theoretical and practical perspective. Specifically it examines the wireless interface between cell tower and mobile phone for GSM and its successors (GPRS, UMTS and LTE). In GSM the wireless interface can be eavesdropped with relative ease, making it an unsuitable technology for sensitive information. While such eavesdropping is much harder on GSM's successors, it turns out that all current mobile communication technology still has potential weaknesses. In practice though, these weaknesses are often hard to exploit without expensive equipment or specific knowledge and skills. Besides, a lot of these potential weaknesses are dependent on provider specific settings, which makes it hard to make generic statements on the security of the mobile network. In practice we do see a clear trend of moving to more secure settings (such as the use of strong encryption algorithms).

While we first examined the security of the wireless interface, we later also looked at the end-points of this interface: the mobile phone and the cell tower. For mobile phones we looked at the quality of the software that processes the mobile signals. Through a testing technique known as "fuzzing" (essentially automated, random testing) we can quickly find a large number of bugs. These bugs prove to be surprisingly divers; software of the same vendor on different devices reveals different mistakes. Since we did this research, at least one major vendor of this type of software rewrote its entire software stack to remove most buggy behaviour.

Normally it is hard for an attacker to obtain prolonged physical access to the other end-point of the wireless connection, cell towers. This changes with the introduction of so-called "femtocells," cheap low-powered cell towers for consumers, providing attackers with easier access to a provider's core network. To prevent an attacker from doing too much damage once he gains full control of a femtocell, providers should place as little trust as possible in these devices, by treating them essentially as a relay to an actual cell tower and never sending them any confidential information. Still, even a low-trust femtocell introduces security risks for the mobile network, as they offer attackers a cheap device which can handle a lot of the mobile phone protocols.

Mobile phones are often used as a second factor in authentication. During authentication users then prove that they still have control over their mobile phone. This can be a cheap way to quickly increase the security of a system. However, for high-security services it is not sufficient to only trust in the security offered by the mobile network, as happens with the use of transaction numbers sent over SMS.

Where most of this thesis covers different security analyses, the final part introduces an improvement to mobile networks which significantly increases their security, while requiring only minimal changes. This improvement makes it impossible to track a user's location based on his permanent identity in the mobile network, by replacing the permanent identities with temporary identities, which get renewed in secret. A side effect of this process allows GSM phones to verify whether they are communicating with an authentic cell tower, thereby fixing the biggest security flaw in GSM. This improvement can be introduced by individual providers without the rest of the network needing any changes or even noticing the change. We are currently working with the GSM Association (GSMA) to get this improvement into the mobile specifications.

Samenvatting

Dit proefschrift behandelt de beveiliging van mobiele telefonie in zowel theoretisch als praktisch opzicht. Meer specifiek kijken we naar de draadloze verbinding tussen telefoon en mast voor GSM en haar opvolgers (GPRS, UMTS en LTE). De draadloze verbinding van GSM is vrij eenvoudig af te luisteren, waardoor het onverstandig is deze technologie nog te gebruiken voor gevoelige informatie. Hoewel het veel moeilijker is om de opvolgers van GSM draadloos af te luisteren, blijken ook alle huidige opvolgers potentiële zwakheden te bevatten. Het is in de praktijk echter wel lastig om deze zwakheden te benutten, zonder dure apparatuur en voldoende kennis. Daarnaast zijn veel van deze potentiële zwakheden afhankelijk van individuele instellingen van het netwerk, wat het lastig maakt om generieke uitspraken te doen over het beveiligingsniveau van een mobiel netwerk. We zien echter een duidelijke trend naar veiligere instellingen (zoals het gebruik van sterkere encryptie).

Waar de focus in het begin alleen ligt op de draadloze verbinding, kijkt dit proefschrift verder ook naar de eindpunten van deze verbinding: de mobiele telefoon en de mast. Voor de telefoons hebben we gekeken naar de kwaliteit van de software die de mobiele signalen verwerkt. Gebruikmakend van een testtechniek die bekend staat als "fuzzing" (in essentie geautomatiseerd willekeurig testen) vinden we gemakkelijk een grote hoeveelheid bugs. De gevonden bugs blijken wel verrassend divers; software van dezelfde fabrikant op verschillende telefoons, toont vaak verschillende fouten. Sinds dit onderzoek heeft ten minste één software fabrikant besloten zijn gehele mobiele telefoon-implemenatie opnieuw uit te voeren om het aantal fouten te verminderen.

Normaliter is het lastig voor kwaadwillenden om langdurig fysieke toegang te krijgen tot het andere eindpunt van de draadloze verbinding, de zendmasten. Met de invoering van zogenaamde femtocells, goedkope lokale zendmasten voor consumenten, krijgen aanvallers een makkelijkere methode om toegang te krijgen tot het kernnetwerk van een provider. Om te voorkomen dat een aanvaller veel schade aan kan richten als hij eenmaal volledige controle heeft over een femtocell, is het verstandig dat providers zo min mogelijk vertrouwen in deze femtocells stellen Dit kan door ze als het ware als verlengde antenne te gebruiken van een echte zendmast en ze nooit vertrouwelijke informatie te sturen. Toch is zelfs een weinig vertrouwde femtocell nog steeds een veiligheidsrisico, doordat het kwaadwillenden een goedkoop apparaat biedt dat veel van het mobiele telefonie verkeer afhandelt.

Mobiele telefoons worden ook vaak als tweede factor gebruikt in authenticatie. Gebruikers bewijzen tijdens inloggen te beschikken over hun mobiele telefoon. Dat

is vaak een goedkope manier om de beveiliging van een systeem behoorlijk op te hogen. Voor beveiligingskritische applicaties is het echter niet voldoende om alleen te vertrouwen op de veiligheid van het mobiele netwerk, zoals bij het gebruik van transactienummers over SMS.

Na alle beveiligingsanalyses, introduceren we in het laatste deel van dit proefschrift een verbetering waardoor de huidige mobiele netwerken, met minimale aanpassingen, significant beter beveiligd kunnen worden. Deze verbetering zorgt ervoor dat gebruikers niet meer gevolgd kunnen worden op basis van hun permanente identiteit in het mobiele netwerk, door deze te vervangen door een tijdelijke identiteit, die heimelijk kan worden vernieuwd. Een mooi zijeffect van deze methode is dat het GSM telefoons in staat stelt om te controleren of zij in verbinding staan met een authentieke zendmast, waarmee de grootste beveiligingsfout van GSM wordt opgelost. Deze verbetering kan ingevoerd worden door individuele providers zonder dat de rest van het netwerk daar iets van merkt. We werken nu samen met de GSM Association (GSMA) om deze aanpassing te realiseren in de standaarden.

Index

Curriculum Vitae

Fabian van den Broek

Born 1981, Beers, the Netherlands

Education

VWO (Pre-university secondary education),
Merlet college Land van Cuijk,
August 2000

B.Sc. in Computer Science, Radboud University,
November 2008

M.Sc. in Computer Science, Radboud University,
Specialisation: Digital Security, April 2010, *cum laude*

Work Experience

Teaching assistant Radboud University,
2003 - 2008

Programmer First8 Java Consultancy,
2008 - 2010

Scientific Programmer Laboratory for Quality Software,
2010 - 2011

PhD student Radboud University,
2011 - 2015

Post doc Radboud University,
within the C-DAX project, 2015

Post doc Radboud University,
within the OYOI project, 2015 - present

Titles in the IPA Dissertation Series since 2013

H. Beohar. *Refinement of Communication and States in Models of Embedded Systems*. Faculty of Mathematics and Computer Science, TU/e. 2013-01

G. Igna. *Performance Analysis of Real-Time Task Systems using Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2013-02

E. Zambon. *Abstract Graph Transformation – Theory and Practice*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2013-03

B. Lijnse. *TOP to the Rescue – Task-Oriented Programming for Incident Response Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2013-04

G.T. de Koning Gans. *Outsmarting Smart Cards*. Faculty of Science, Mathematics and Computer Science, RU. 2013-05

M.S. Greiler. *Test Suite Comprehension for Modular and Dynamic Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2013-06

L.E. Mamane. *Interactive mathematical documents: creation and presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2013-07

M.M.H.P. van den Heuvel. *Composition and synchronization of real-time components upon one processor*. Faculty of Mathematics and Computer Science, TU/e. 2013-08

J. Businge. *Co-evolution of the Eclipse Framework and its Third-party Plug-ins*. Faculty of Mathematics and Computer Science, TU/e. 2013-09

S. van der Burg. *A Reference Architecture for Distributed Software Deployment*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2013-10

J.J.A. Keiren. *Advanced Reduction Techniques for Model Checking*. Faculty of Mathematics and Computer Science, TU/e. 2013-11

D.H.P. Gerrits. *Pushing and Pulling: Computing push plans for disk-shaped robots, and dynamic labelings for moving points*. Faculty of Mathematics and Computer Science, TU/e. 2013-12

M. Timmer. *Efficient Modelling, Generation and Analysis of Markov Automata*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2013-13

M.J.M. Roeloffzen. *Kinetic Data Structures in the Black-Box Model*. Faculty of Mathematics and Computer Science, TU/e. 2013-14

L. Lensink. *Applying Formal Methods in Software Development*. Faculty of Science, Mathematics and Computer Science, RU. 2013-15

C. Tankink. *Documentation and Formal Mathematics — Web Technology meets Proof Assistants*. Faculty of Science, Mathematics and Computer Science, RU. 2013-16

C. de Gouw. *Combining Monitoring with Run-time Assertion Checking*. Faculty of Mathematics and Natural Sciences, UL. 2013-17

J. van den Bos. *Gathering Evidence: Model-Driven Software Engineering in Automated Digital Forensics*. Faculty of Science, UvA. 2014-01

D. Hadziosmanovic. *The Process Matters: Cyber Security in Industrial Control Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-02

A.J.P. Jeckmans. *Cryptographically-Enhanced Privacy for Recommender Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-03

C.-P. Bezemer. *Performance Optimization of Multi-Tenant Software Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2014-04

T.M. Ngo. *Qualitative and Quantitative Information Flow Analysis for Multi-threaded Programs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-05

A.W. Laarman. *Scalable Multi-Core Model Checking*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-06

J. Winter. *Coalgebraic Characterizations of Automata-Theoretic Classes*. Faculty of Science, Mathematics and Computer Science, RU. 2014-07

W. Meulemans. *Similarity Measures and Algorithms for Cartographic Schematization*. Faculty of Mathematics and Computer Science, TU/e. 2014-08

A.F.E. Belinfante. *JTorX: Exploring Model-Based Testing*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-09

A.P. van der Meer. *Domain Specific Languages and their Type Systems*. Faculty of Mathematics and Computer Science, TU/e. 2014-10

B.N. Vasilescu. *Social Aspects of Collaboration in Online Software Communities*. Faculty of Mathematics and Computer Science, TU/e. 2014-11

F.D. Aarts. *Tomte: Bridging the Gap between Active Learning and Real-World Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2014-12

N. Noroozi. *Improving Input-Output Conformance Testing Theories*. Faculty of Mathematics and Computer Science, TU/e. 2014-13

M. Helvensteijn. *Abstract Delta Modeling: Software Product Lines and Beyond*. Faculty of Mathematics and Natural Sciences, UL. 2014-14

P. Vullers. *Efficient Implementations of Attribute-based Credentials on Smart Cards*. Faculty of Science, Mathematics and Computer Science, RU. 2014-15

F.W. Takes. *Algorithms for Analyzing and Mining Real-World Graphs*. Faculty of Mathematics and Natural Sciences, UL. 2014-16

M.P. Schraagen. *Aspects of Record Linkage*. Faculty of Mathematics and Natural Sciences, UL. 2014-17

G. Alpár. *Attribute-Based Identity Management: Bridging the Cryptographic Design of ABCs with the Real World*. Faculty of Science, Mathematics and Computer Science, RU. 2015-01

A.J. van der Ploeg. *Efficient Abstractions for Visualization and Interaction*. Faculty of Science, UvA. 2015-02

R.J.M. Theunissen. *Supervisory Control in Health Care Systems*. Faculty of Mechanical Engineering, TU/e. 2015-03

T.V. Bui. *A Software Architecture for Body Area Sensor Networks: Flexibility and Trustworthiness*. Faculty of Mathematics and Computer Science, TU/e. 2015-04

A. Guzzi. *Supporting Developers' Teamwork from within the IDE*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-05

T. Espinha. *Web Service Growing Pains: Understanding Services and Their Clients*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-06

S. Dietzel. *Resilient In-network Aggregation for Vehicular Networks*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-07

E. Costante. *Privacy throughout the Data Cycle*. Faculty of Mathematics and Computer Science, TU/e. 2015-08

S. Cranen. *Getting the point — Obtaining and understanding fixpoints in model checking*. Faculty of Mathematics and Computer Science, TU/e. 2015-09

R. Verdult. *The (in)security of proprietary cryptography*. Faculty of Science, Mathematics and Computer Science, RU. 2015-10

J.E.J. de Ruiter. *Lessons learned in the analysis of the EMV and TLS security protocols*. Faculty of Science, Mathematics and Computer Science, RU. 2015-11

Y. Dajsuren. *On the Design of an Architecture Framework and Quality Evaluation for Automotive Software Systems*. Faculty of Mathematics and Computer Science, TU/e. 2015-12

J. Bransen. *On the Incremental Evaluation of Higher-Order Attribute Grammars*. Faculty of Science, UU. 2015-13

S. Picek. *Applications of Evolutionary Computation to Cryptology*. Faculty of Science, Mathematics and Computer Science, RU. 2015-14

C. Chen. *Automated Fault Localization for Service-Oriented Software Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-15

S. te Brinke. *Developing Energy-Aware Software*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-16

R.W.J. Kersten. *Software Analysis Methods for Resource-Sensitive Systems*. Faculty of Science, Mathematics and Computer Science, RU. 2015-17

J.C. Rot. *Enhanced coinduction*. Faculty of Mathematics and Natural Sciences, UL. 2015-18

M. Stolikj. *Building Blocks for the Internet of Things*. Faculty of Mathematics and Computer Science, TU/e. 2015-19

D. Gebler. *Robust SOS Specifications of Probabilistic Processes*. Faculty of Sciences, Department of Computer Science, VUA. 2015-20

M. Zaharieva-Stojanovski. *Closer to Reliable Software: Verifying functional behaviour of concurrent programs*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-21

R.J. Krebbers. *The C standard formalized in Coq*. Faculty of Science, Mathematics and Computer Science, RU. 2015-22

R. van Vliet. *DNA Expressions – A Formal Notation for DNA*. Faculty of Mathematics and Natural Sciences, UL. 2015-23

S.-S.T.Q. Jongmans. *Automata-Theoretic Protocol Programming*. Faculty of Mathematics and Natural Sciences, UL. 2016-01

S.J.C. Joosten. *Verification of Interconnects*. Faculty of Mathematics and Computer Science, TU/e. 2016-02

M.W. Gazda. *Fixpoint Logic, Games, and Relations of Consequence*. Faculty of Mathematics and Computer Science, TU/e. 2016-03

S. Keshishzadeh. *Formal Analysis and Verification of Embedded Systems for Healthcare*. Faculty of Mathematics and Computer Science, TU/e. 2016-04

P.M. Heck. *Quality of Just-in-Time Requirements: Just-Enough and Just-in-Time*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2016-05

Y. Luo. *From Conceptual Models to Safety Assurance – Applying Model-Based Techniques to Support Safety Assurance*. Faculty of Mathematics and Computer Science, TU/e. 2016-06

B. Ege. *Physical Security Analysis of Embedded Devices*. Faculty of Science, Mathematics and Computer Science, RU. 2016-07

A.I. van Goethem. *Algorithms for Curved Schematization*. Faculty of Mathematics and Computer Science, TU/e. 2016-08

T. van Dijk. *Sylvan: Multi-core Decision Diagrams*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2016-09

I. David. *Run-time resource management for component-based systems*. Faculty of Mathematics and Computer Science, TU/e. 2016-10

A.C. van Hulst. *Control Synthesis using Modal Logic and Partial Bisimilarity – A Treatise Supported by Computer Verified Proofs*. Faculty of Mechanical Engineering, TU/e. 2016-11

A. Zawedde. *Modeling the Dynamics of Requirements Process Improvement*. Faculty of Mathematics and Computer Science, TU/e. 2016-12

F.M.J. van den Broek. *Mobile Communication Security*. Faculty of Science, Mathematics and Computer Science, RU. 2016-13