Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer

Roel Verdult
Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands.
rverdult@cs.ru.nl

Flavio D. Garcia
School of Computer Science, University of Birmingham, UK.
f.garcia@cs.bham.ac.uk

Barış Ege
Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands.
b.ege@cs.ru.nl

Abstract

The Megamos Crypto transponder is used in one of the most widely deployed electronic vehicle immobilizers. It is used among others in most Audi, Fiat, Honda, Volkswagen and Volvo cars. Such an immobilizer is an anti-theft device which prevents the engine of the vehicle from starting when the corresponding transponder is not present. This transponder is a passive RFID tag which is embedded in the key of the vehicle.

In this paper we have reverse-engineered all proprietary security mechanisms of the transponder, including the cipher and the authentication protocol which we publish here in full detail. This article reveals several weaknesses in the design of the cipher, the authentication protocol and also in their implementation. We exploit these weaknesses in three practical attacks that recover the 96-bit transponder secret key. These three attacks only require wireless communication with the system. Our first attack exploits weaknesses in the cipher design and in the authentication protocol. We show that having access to only two eavesdropped authentication traces is enough to recover the 96-bit secret key with a computational complexity of 2^{56} cipher ticks (equivalent to 2^{49} encryptions). Our second attack exploits a weakness in the key-update mechanism of the transponder. This attack recovers the secret key after 3×2^{16} authentication attempts with the transponder and negligible computational complexity. We have executed this attack in practice on several vehicles. We were able to recover the key and start the engine with a transponder emulating device. Executing this attack from beginning to end takes only 30 minutes. Our third attack exploits the fact that some car manufacturers set weak cryptographic keys in their vehicles. We propose a time-memory trade-off which recovers such a weak key after a few minutes of computation on a standard laptop.

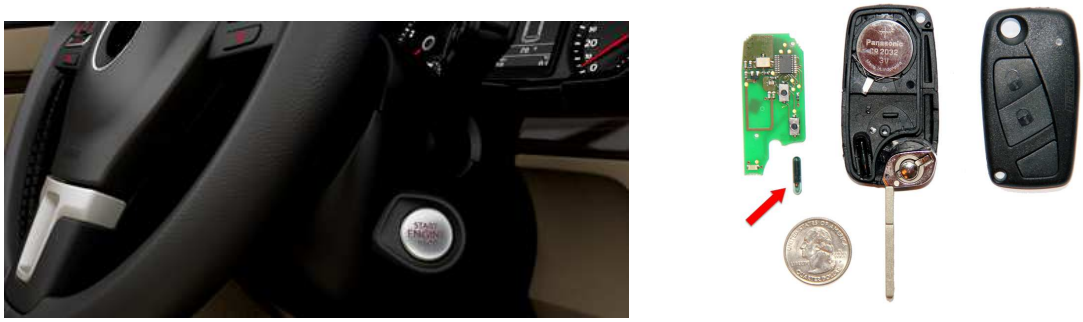
1 Introduction

Electronic vehicle immobilizers have been very effective at reducing car theft. Such an immobilizer is an electronic device that prevents the engine of the vehicle from starting when the corresponding transponder is not present. This transponder is a low-frequency RFID chip which is typically embedded in the vehicle's key. When the driver starts the vehicle, the car authenticates the transponder before starting the engine, thus preventing hot-wiring. In newer vehicles the mechanical ignition key has often been removed and replaced by a start button, see Figure 1(a). In such vehicles the immobilizer transponder is the only anti-theft mechanism that prevents a hijacker from driving away.

A distinction needs to be made between the vehicle immobilizer and the remotely operated central locking system. The latter is battery powered, operates at an ultra-high frequency (UHF), and only activates when the user pushes a button on the remote to (un)lock the doors of the vehicle. Figure 1(b) shows a disassembled car key where it is possible to see the passive Megamos Crypto transponder and also the battery powered remote of the central locking system.

The Megamos Crypto transponder is the first cryptographic immobilizer transponder manufactured by [19] and is currently one of the most widely used. The manufacturer claims to have sold more than 100 million immobilizer chips including Megamos Crypto transponders [22]. Figure 2 shows a list of vehicles that use or have used Megamos Crypto at least for some version/year. As it can be seen from this list, many Audi, Fiat, Honda, Volkswagen and Volvo cars used Megamos Crypto transponders at the time of this research (fall 2012).

The transponder uses a 96-bit secret key and a proprietary cipher in order to authenticate to the vehicle. Furthermore, a 32-bit PIN code is needed in order to be able to write on the memory of the transponder. The con-



(a) Keyless ignition with start button

(b) Megamos Crypto transponder in a car key

Figure 1: Megamos Crypto integration in vehicular systems

crete details regarding the cipher design and authentication protocol are kept secret by the manufacturer and little is currently known about them.

From our collaboration with the local police it was made clear to us that sometimes cars are being stolen and nobody can explain how. They strongly suspect the use of so-called 'car diagnostic' devices. Such a device uses all kind of custom and proprietary techniques to bypass the immobilizer and start a car without a genuine key. This motivated us to evaluate the security of vehicle immobilizer transponders. There are known attacks for three of the four widely used immobilizer transponders, namely DST40, Keeloq and Hitag2. Although, at the time of this research, little was known about the security of the Megamos Crypto transponder.

1.1 Our contribution

In this paper we have fully reverse-engineered all cryptographic mechanisms of Megamos Crypto which we publish here in full detail. For this we used IDA Pro[1] to decompile the software package that comes with the Tango Programmer[2].

Furthermore, we have identified several weaknesses in Megamos Crypto which we exploit in three attacks. Our first attack consists of a cryptanalysis of the cipher and the authentication protocol. Our second and third attack not only look at the cipher but also at the way in which it is implemented and poorly configured by the automotive industry.

Our first attack, which comprises *all* vehicles using Megamos Crypto, exploits the following weaknesses.

- The transponder lacks a pseudo-random number generator, which makes the authentication protocol vulnerable to replay attacks.

[1] https://www.hex-rays.com/products/ida/
[2] http://www.scorpio-lk.com

Make	Models
Alfa Romeo	147, 156, GT
Audi	A1, A2, A3, **A4 (2000)**, A6, A8, Allroad, Cabrio, Coupé, Q7, S2, S3, S4, S6, S8, **TT (2000)**
Buick	Regal
Cadillac	CTS-V, SRX
Chevrolet	Aveo, Kalos, Matiz, Nubira, Spark, Evanda, Tacuma
Citroën	**Jumper (2008)**, Relay
Daewoo	Kalos, Lanos, Leganza, Matiz, Nubira, Tacuma
DAF	CF, LF, XF
Ferrari	California, 612 Schaglietti
Fiat	Albea, Doblò, Idea, Mille, Multipla, Palio, **Punto (2002)**, Seicento, Siena, Stilo, **Ducato (2004)**
Holden	Barina, Frontera
Honda	Accord, Civic, CR-V, FR-V, HR-V, Insight, **Jazz (2002)**, Legend, Logo, S2000, Shuttle, Stream
Isuzu	Rodeo
Iveco	Eurocargo, Daily
Kia	Carnival, Clarus, Pride, Shuma, Sportage
Lancia	Lybra, Musa, Thesis, Y
Maserati	Quattroporte
Opel	Frontera
Pontiac	G3
Porsche	911, 968, Boxster
Seat	Altea, Córdoba, Ibiza, Leon, Toledo
Skoda	**Fabia (2011)**, Felicia, Octavia, Roomster, Super, Yeti
Ssangyong	Korando, Musso, Rexton
Tagaz	Road Partner
Volkswagen	Amarok, Beetle, Bora, Caddy, Crafter, Cross Golf, Dasher, Eos, Fox, Gol, **Golf (2006, 2008)**, Individual, Jetta, Multivan, New Beetle, Parati, Polo, Quantum, Rabbit, Saveiro, Santana, **Scirocco (2011)**, Touran, Tiguan, Voyage, **Passat (1998, 2005)**, Transporter
Volvo	C30, **S40 (2005)**, S60, S80, V50, V70, XC70, XC90, XC94

Figure 2: Vehicles that used Megamos Crypto for some version/year [39]. Boldface and year indicate specific vehicles we experimented with.

- The internal state of the cipher consists of only 56 bits, which is much smaller than the 96-bit secret key.
- The cipher state successor function can be inverted, given an internal state and the corresponding bit of cipher-text it is possible to compute the predecessor state.
- The last steps of the authentication protocol

provides and adversary with 15-bits of known-plaintext.

We present two versions of this attack. First we introduce a simple (but more computationally intensive) attack that recovers the secret key of the transponder with a computational complexity of 2^{56} encryptions. Then we optimize this attack, reducing its computational complexity to 2^{49} by using a time-memory trade-off. For this trade-off, a 12 terabyte lookup table needs to be pre-computed. This optimized version of the attack takes advantage of the fact that some of the cipher components can be run quite autonomously.

Our second attack exploits the following weaknesses.

- Currently, the memory of many Megamos Crypto transponders in the field is either unlocked or locked with a publicly known default PIN code [17]. This means that anybody has write access to the memory of the transponder. This also holds for the secret key bits.
- The 96-bit secret key is written to the transponder in blocks of 16 bits instead of being an atomic operation.

This attack recovers the 96-bit secret key of such a transponder within 30 minutes. This time is necessary to perform 3×2^{16} authentication attempts to the transponder and then recover the key with negligible computational complexity. We have executed this attack in practice and recovered the secret key of several cars from various makes and models. Having recovered the key we were able to emulate the transponder and start the vehicles.

Our third attack is based on the following observation. Many of the keys that we recovered using the previous attack had very low entropy and exhibit a well defined pattern, i.e., the first 32 bits of the key are all zeros. This attack consists of a time-memory trade-off that exploits this weakness to recover the secret key, within a few minutes, from two authentication traces. This attack requires storage of a 1.5 terabyte rainbow table.

We propose a simple but effective mitigating measure against our second attack. This only involves setting a few bits on the memory of the transponder and can be done by anyone (even the car owners themselves) with a compatible RFID reader.

Finally, we have developed an open source library for custom and proprietary RFID communication schemes that operate at an frequency of 125 kHz. We used this library to provide eavesdropping, emulation and reader support for Megamos Crypto transponders with the Proxmark III device[3]. The reader functionality allows the user to send simple commands like read and write to the transponder. In particular, this library can be used to set the memory lock bit and a random PIN code as a mitigation for our second attack, as described in Section 8.

[3] http://www.proxmark.org/

1.2 Related work

In the last decades, semiconductor companies introduced several proprietary algorithms specifically for immobilizer security. Their security often depends on the secrecy of the algorithm. When their inner-workings are uncovered, it is often only a matter of weeks before the first attack is published. There are several examples in the literature that address the insecurity of proprietary algorithms. The most prominent ones are those breaking A5/1 [31], DECT [45, 47], GMR [18], WEP [24] and also many RFID systems like the MIFARE Classic [16, 26, 29, 46], CryptoRF [30] and iClass [27, 28].

Besides Megamos Crypto, there are only three other major immobilizer products being used. The DST transponder which was reverse-engineered and attacked by Bono et.al. in [9]; KeeLoq was first attacked by Bogdanov in [6] and later this attack was improved in [12, 36, 38]; Hitag2 was anonymously published in [60] and later attacked in [8, 13, 35, 52, 53, 57, 58].

With respect to vehicle security, Koscher et. al. attracted a lot of attention from the scientific community when they demonstrated how to compromise the board computer of a modern car [11, 40]. They were able to remotely exploit and control many car features such as tracking the car via GPS and adjust the speeding of the car. In 2011, Francillon et. al. [25] showed that with fairly standard equipment it is possible to mount a relay-attack on all keyless-entry systems that are currently deployed in modern cars.

The scientific community proposed several alternatives [43, 44, 59, 61, 62] to replace the weak proprietary ciphers and protocols. There are several commercial vehicle immobilizer transponders that makes use of standard cryptography, like AES [14]. Examples include the Hitag Pro transponder from NXP Semiconductors and ATA5795 transponder from Atmel. To the best of our knowledge, only Atmel made an open protocol design [1] and published it for scientific scrutiny. The security of their design was analyzed by Tillich et. al. in [54].

2 Technical background

This section briefly describes what a vehicle immobilizer is and how it is used by the automotive industry. Then we describe the hardware setup we use for our experiments. Finally we introduce the notation used throughout the paper.

2.1 Immobilizer

To prevent a hijacker from hot-wiring a vehicle, car manufacturers incorporated an electronic car immobilizer as

an extra security mechanism. In some countries, having such an immobilizer is enforced by law. For example, according to European Commission directive (95/56/EC) it is mandatory that all cars sold in the EU from 1995 are fitted with an electronic immobilizer. Similar regulations apply to other countries like Australia, New Zealand (AS/NZS 4601:1999) and Canada (CAN/ULC S338-98). Although in the US it is not required by law, according to the independent organization Insurance Institute for Highway Safety (IIHS), 86 percent of all new passenger cars sold in the US had an engine immobilizer installed [55].

An electronic car immobilizer consists of three main components: a small transponder chip which is embedded in (the plastic part of) the car key, see Figure 1(b); an antenna coil which is located in the dashboard of the vehicle, typically around the ignition barrel; and the immobilizer unit that prevents the vehicle from starting the engine when the transponder is absent.

The immobilizer unit communicates through the antenna coil and enumerates all transponders that are in proximity of field. The transponder identifies itself and waits for further instructions. The immobilizer challenges the transponder and authenticates itself first. On a successful authentication of the immobilizer unit, the transponder sends back its own cryptographic response. Only when this response is correct, the immobilizer unit enables the engine to start.

The immobilizer unit is directly connected to the internal board computer of the car, also referred to as Electrical Control Unit (ECU). To prevent hot-wiring a car, the ECU blocks fuel-injection, disables spark-plugs and deactivates the ignition circuit if the transponder fails to authenticate.

2.2 Hardware setup

We used the Proxmark III to eavesdrop and communicate with the car and transponder. This is a generic RFID protocol analysis tool [56] that supports raw data sampling at a frequency of 125 kHz. We implemented a custom firmware and FPGA design that supports the modulation and encoding schemes of Megamos Crypto transponders. The design samples generic analog-digital converter (ADC) values and interpret them in real-time in the micro-controller. We have implemented commands to eavesdrop, read and emulate a transponder. Our library is able to decode field and transponder modulation simultaneously and is very precise in timing.

Figure 3: Proxmark 3

2.3 Notation

Throughout this paper we use the following mathematical notation. Let $\mathbb{F}_2 = \{0,1\}$ be the set of Booleans. The symbol $\oplus$ denotes exclusive-or (XOR), 0^n denotes a bitstring of n zero-bits. ε denotes the empty bitstring. Given two bitstrings x and y, xy denotes their concatenation. Sometimes we write this concatenation explicitly with $x \cdot y$ to improve readability. $\overline{x}$ denotes the bitwise complement of x. Given a bitstring $x \in \mathbb{F}_2^k$, we write x_i to denote the i-th bit of x. For example, given the bitstring $x = 0\text{x}03 = 00000011 \in \mathbb{F}_2^8$, $x_0 = 0$ and $x_6 = x_7 = 1$.

3 Megamos Crypto

This section describes Megamos Crypto in detail. We first describe the Megamos Crypto functionality, memory structure, and communication protocols, this comes from the product datasheet [21] and the application note [23]. Then we briefly describe how we reverse-engineered the cryptographic algorithms and protocols used in Megamos Crypto. Finally, we describe these algorithms and protocols in detail.

3.1 Memory

There are two types of Megamos Crypto transponders, in automotive industry often referred to as Magic I (V4070) [20] and Magic II (EM4170) [21]. The EM4170 transponder is the newer version and it has 16 memory blocks of 16-bit words. The contents of these memory blocks are depicted in Figure 4. The older version (V4070) supports exactly the same read and write operations and cryptographic algorithms, but it only has 10 memory blocks. The blocks 10 to 15, which store 64 bits of additional user memory and a 32-bit PIN code are simply not readable. The EM4170 transponder uses the same communication and is therefore backwards compatible with the V4070 transponder. Note that in some cars the new revision is deployed as replacement for the V4070 without making use of, or even initializing the additional user memory blocks and PIN code. The whole memory is divided in three sections with different access rights, see Figure 4.

The transponder identifier *id* is always read-only. The write access over the other memory blocks is determined by the value of the lock-bit l_0. Just as specified, the value of lock-bit l_1 does not have any influence the memory access conditions. Similarly, a successful or failed authentication has no effect on the access conditions.

- When $l_0 = 0$, all memory blocks (except *id*) of a Megamos Crypto transponder are still writable. The key k, PIN code *pin* are write-only and the user memory *um* blocks (which includes the lock-bits l) are read-write. However, after a successful write in block 1, the new value of l_0 determines the access condition for future write operations.

- When $l_0 = 1$, all writing is disabled. However, it does not affect the read access conditions. This means that the key k, PIN code *pin* can not be read out and the user memory *um* becomes read-only. Because the lock-bits l are stored in a user memory block they can always be read out.

The EM4170 allows to set the lock-bit l_0 back to zero using a PIN code *pin*. A valid PIN code resets the access conditions and enables again writing of k, *pin*, *um* and l. The PIN code has to be known or overwritten to the transponder before it is locked, otherwise an exhaustive search of the PIN code is required.

Block	Content	Denoted by
0	user memory	$um_0 \dots um_{15}$
1	user memory, lock bits	$um_{16} \dots um_{29} l_0 l_1$
2	device identification	$id_0 \dots id_{15}$
3	device identification	$id_{16} \dots id_{31}$
4	crypto key	$k_0 \dots k_{15}$
5	crypto key	$k_{16} \dots k_{31}$
6	crypto key	$k_{32} \dots k_{47}$
7	crypto key	$k_{48} \dots k_{63}$
8	crypto key	$k_{64} \dots k_{79}$
9	crypto key	$k_{80} \dots k_{95}$
10	pin code	$pin_0 \dots pin_{15}$
11	pin code	$pin_{16} \dots pin_{31}$
12	user memory	$um_{30} \dots um_{45}$
13	user memory	$um_{46} \dots um_{61}$
14	user memory	$um_{62} \dots um_{77}$
15	user memory	$um_{78} \dots um_{93}$

read-only
write-only
read-write

Figure 4: Megamos Crypto transponder memory layout

3.2 Functionality and communication

The Megamos Crypto transponder supports four different operations: `read`, `write`, `reset` and `authenticate`.

- `read` operations are performed by three different commands, each returns multiple blocks. The transponder returns the concatenation of these blocks in one bitstring. The three available bitstrings are $id_{31} \dots id_0$, $l_1 l_0 um_{29} \dots um_0$ and $um_{93} \dots um_{30}$.
- `write` stores a 16-bit memory block in the memory of the transponder. The arguments for this command are the block number and the data. After receiving the command, the transponder stores the data in memory if the access conditions allow the requested write operation.
- `reset` takes the *id* and 32-bit PIN code as an argument. If the PIN code matches the value that is stored in *pin*, then the lock-bit l_0 is reset, see Section 3.1 for more details about l_0.
- `authenticate` takes three arguments. The first one is a 56-bit car nonce n_C. The second argument is a bitstring of 7 zero bits. The datasheet [21] refers to them as "divergency bits". It seems that these bit-periods are used to initialize the cipher. In Section 3.6 we show that the authentication protocol exactly skips 7 cipher steps before it starts generating output. The third argument is a 28-bit authenticator from the car a_C. If successful, the transponder responds with its 20-bit authenticator a_T.

When the driver turns on the ignition, several back-and-forward messages between the car and transponder are exchanged. It starts with the car reading out the transponder memory blocks that contains the identity, user memory and lock-bits. Next, the car tries to authenticate using the shared secret key k. If the authentication fails, the car retries around 20 times before it reports on the dashboard that the immobilizer failed to authenticate the transponder. Figure 5 shows an eavesdropped trace of a German car that initializes and authenticates a Megamos Crypto transponder.

To the best of our knowledge, there is no publicly available document that describes the structure of Megamos Crypto cipher. However, a simplified representation of the authentication protocol is presented in the EM4170 application note [23] as shown in Figure 6. It does not specify any details beyond the transmitted messages and the checks which the car and transponder must perform. The car authenticates by sending a nonce $n_C = Random$ and the corresponding authenticator $a_C = f(Rnd, K)$. When the car successfully authenticated itself, the Megamos Crypto transponder sends the transponder authenticator $a_T = g(Rnd, f, K)$ back to car.

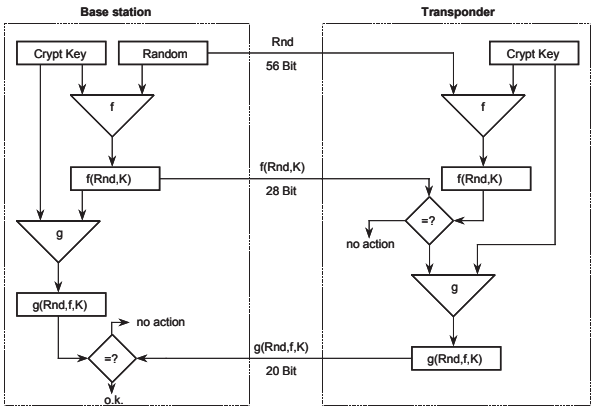


Figure 6: Authentication procedure excerpt from [23]

For communication the Megamos Crypto transponder uses a low frequency wave of 125 kHz and applies amplitude shift keying (ASK) modulation by putting a small resistance on the electro magnetic field. It utilizes a cus-

Origin	Message	Description
Car	`3`	Read identifier
Transponder	`A9 08 4D EC`	Identfier $id_{31} \dots id_0$
Car	`5`	Read user memory and lock-bits
Transponder	`80 00 95 13`	First user memory $l_1 l_0 um_{29} \dots um_0$
Car	`F`	Read large user memory (EM4170)
Transponder	`AA AA AA AA AA AA AA AA`	Second user memory $um_{93} \dots um_{30}$
Car	`6 | 3F FE 1F B6 CC 51 3F |` 0^7 `| F3 55 F1 A`	Authentication, $n_{C_{55}} \dots n_{C_0}$, 0^7, a_C
Transponder	`60 9D 6`	Car authenticated successful, send back a_T

Figure 5: Eavesdropped Megamos Crypto authentication using the 96-bit key `000000000000010405050905`. The structure of the secret key of the car suggests that it has an entropy of only 24 bits.

tom encoding scheme for status bits and a Manchester encoding scheme for transmitting data bits. The Megamos Crypto immobilizer unit signals the transponder to receive a command by dropping the field two consecutive times in a small time interval. Then it drops the field a few microseconds to modulate a zero and leaves the field on to modulate a one.

This way of modulation introduces the side-effect that the immobilizer unit and the transponder could get out-of-sync. When the immobilizer unit sends a bitstring of contiguous ones, there are no field drops for almost 15 milliseconds. The manufacturer realized this was a problem, but instead of proposing an alternative communication scheme they suggest to choose random numbers with more zeros's than ones and especially avoid sequential ones [23]. From a security perspective it sounds like a bad idea to suggest to system integrators that they should effectively drop entropy from the used random numbers.

To get a fair estimate of communication timings we did some experiments. With our hardware setup we were able to reach the highest communication speed with the transponder that is possible according to the datasheet. It allows us to read out the identifier *id* in less than 14 milliseconds and successfully authenticate within 34 milliseconds. These timings confirm that an adversary can wirelessly pickpocket the identifier and all its user memory in less than a second from a distance of one inch. Standing close to a victim for only a fraction of a second enables the adversary to gather the transponder identifier. When this identifier is emulated to the corresponding car, it is possible to gather partial authentication traces. Because the transponder lacks a random generator, this partial traces can later be used to retrieve the responses from the transponder which extends them to successful authentication traces. With a number of successful authentication traces it is possible to recover the secret key as described in Section 5.

3.3 Reverse-engineering the cipher

Recent articles point out the lack of security [11, 40, 41] in modern cars. The software in existing cars is designed with safety in mind, but is still immature in terms of security. Legacy protocols and technologies are often vulnerable to a number of remote and local exploits.

Most car keys need to be preprogrammed, which is also referred to as pre-coded, before they can be associated to a car. During this initialization phase the user memory blocks are filled with manufacturer specific data to prevent mixing of keys. This step adds no security, it just restricts the usage of keys that were meant a specific car make or model.

There are several car locksmith tools[4][5][6] in the after market that can initialize or change such transponder data. Such tools fully support the modulation/encoding schemes and communication protocol of the Megamos Crypto transponder. They implement some publicly available functionality like the `read`, `write` and `reset` commands. However, they do not implement the authentication protocol. To perform a successful authentication, knowledge of the Megamos Crypto cipher is necessary to compute the authentication messages a_C and a_T.

More advanced car diagnostic tools like AVDI[7] and Tango Programmer[8] offer functionality that goes beyond "legitimate" usage. These devices are able to dump the board-computer memory, recover the dealer code, and add a new blank transponder to the car. For this the tools do not require a genuine key to be present but they do need physical access to the can bus.

These diagnostic tools use the Megamos Crypto authentication functionality to speed up the process of adding new transponders to the car. For this, the tool needs the Megamos Crypto algorithm to compute valid

[4] http://www.istanbulanahtar.com
[5] http://www.advanced-diagnostics.co.uk
[6] http://www.jmausa.com
[7] http://www.abritus72.com
[8] http://www.scorpio-lk.com

authentication attempts. We would like to emphasize that non of these tools is able to recover the secret key of a transponder or perform any kind of cryptanalysis. In fact, within the legitimate automotive industry Megamos Crypto is believed to be unclonable.

The software package that comes with the Tango Programmer implements all cryptographic operations of the transponder including the Megamos Crypto cipher. We have analyzed the software thoroughly and extracted the algorithm from it.

Since the application implements several counter measures against reverse-engineering, this task was not trivial at all. It is highly protected with an executable obfuscator that runs a custom virtual machine, as described in [51], and a number of advanced anti-debugging tricks to avoid exposure of its inner workings. To perform our security evaluation of the Megamos Crypto cipher we bypassed all these measures and reverse engineered the cipher in a semi-automatic way by observing the memory state changes and guessing the intermediate cryptographic calculations.

Furthermore, we observed every Megamos Crypto related function call from the program instructions memory segment. When the program counter entered a suspicious memory segment, we invoked our clean-up routine that automatically grouped and dropped all unnecessary instructions (unconditional re-routings, sequential operations on the same variables, random non-influential calculations). After analysing this at run-time, the actual working of the algorithm was quickly deduced from the optimized and simplified persistent instruction set.

3.4 Cipher

This section describes the Megamos Crypto cipher in detail. The cipher consists of five main components: a Galois Linear Feedback Shift Register, a non-linear Feedback Shift Register, and three 7-bit registers. A schematic representation of the cipher is depicted in Figure 7.

Definition 3.1 (Cipher state)**.** *A Megamos Crypto* cipher state $s = \langle g,h,l,m,r\rangle$ *is an element of* $\mathbb{F}_2^{57}$ *consisting of the following five components:*

1. *the* Galois LFSR $g = (g_0 \dots g_{22}) \in \mathbb{F}_2^{23}$*;*
2. *the* non-linear FSR $h = (h_0 \dots h_{12}) \in \mathbb{F}_2^{13}$*;*
3. *the* first output register $l = (l_0 \dots l_6) \in \mathbb{F}_2^7$*;*
4. *the* second output register $m = (m_0 \dots m_6) \in \mathbb{F}_2^7$*;*
5. *the* third output register $r = (r_0 \dots r_6) \in \mathbb{F}_2^7$.

The following definitions describe the successor or feedback functions for each of these components.

Definition 3.2. *The successor function for the Galois linear feedback shift register* $G\colon \mathbb{F}_2^{23} \times \mathbb{F}_2 \times \mathbb{F}_2 \to \mathbb{F}_2^{23}$ *is defined as*

$$\begin{aligned} G(g_0 \dots g_{22}, i, j) = {} & (j \oplus g_{22})g_0 g_1 g_2 (g_3 \oplus g_{22})(g_4 \oplus i) \\ & (g_5 \oplus g_{22})(g_6 \oplus g_{22})g_7 \dots g_{12}(g_{13} \oplus g_{22})g_{14}g_{15} \\ & (g_{16} \oplus g_{22})g_{17} \dots g_{21} \end{aligned}$$

We also overload the function G *to multiple-bit input string* $G\colon \mathbb{F}_2^{23} \times \mathbb{F}_2 \times \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{23}$ *as*

$$G(g, i, j_0 \dots j_n) = G(G(g, i, j_1 \dots j_n), i, j_0)$$

Definition 3.3. *The successor function for the non-linear feedback shift register* $H\colon \mathbb{F}_2^{13} \to \mathbb{F}_2^{13}$ *is defined as*

$$H(h_0 \dots h_{12}) = ((h_1 \wedge h_8) \oplus (h_9 \wedge h_{11}) \oplus \overline{h_{12}})h_0 \dots h_{11}$$

Definition 3.4. *The feedback function for the first output register* $f_l\colon \mathbb{F}_2^6 \to \mathbb{F}_2$ *is defined as*

$$\begin{aligned} f_l(x_0 \dots x_5) = {} & (\overline{x_0} \wedge \overline{x_2} \wedge x_3) \vee (x_2 \wedge x_4 \wedge \overline{x_5}) \vee \\ & (x_5 \wedge x_1 \wedge \overline{x_3}) \vee (x_0 \wedge \overline{x_1} \wedge \overline{x_4}). \end{aligned}$$

Definition 3.5. *The feedback function for the second output register* $f_m\colon \mathbb{F}_2^6 \to \mathbb{F}_2$ *is defined as*

$$\begin{aligned} f_m(x_0 \dots x_5) = {} & (\overline{x_4} \wedge x_1 \wedge \overline{x_2}) \vee (x_5 \wedge \overline{x_1} \wedge x_3) \vee \\ & (x_0 \wedge x_2 \wedge \overline{x_3}) \vee (x_4 \wedge \overline{x_5} \wedge \overline{x_0}). \end{aligned}$$

Definition 3.6. *The feedback function for the third output register* $f_r\colon \mathbb{F}_2^6 \to \mathbb{F}_2$ *is defined as*

$$\begin{aligned} f_r(x_0 \dots x_5) = {} & (x_5 \wedge \overline{x_0} \wedge \overline{x_2}) \vee (\overline{x_5} \wedge x_3 \wedge x_1) \vee \\ & (x_2 \wedge \overline{x_3} \wedge \overline{x_4}) \vee (x_0 \wedge x_4 \wedge \overline{x_1}). \end{aligned}$$

With every clock tick the cipher steps to its successor state and it (potentially) outputs one bit of keystream. The following precisely defines the successor state and the output of the cipher.

Definition 3.7 (Successor state)**.** *Let* $s = \langle g,h,l,m,r\rangle$ *be a cipher state and* $i \in \mathbb{F}_2$ *be an input bit. Then, the* successor cipher state $s' = \langle g',h',l',m',r'\rangle$ *is defined as*

$$\begin{aligned} g' &:= G(g, i, l_1 \oplus m_6 \oplus h_2 \oplus h_8 \oplus h_{12}) \\ h' &:= H(h) \\ l' &:= a l_0 \dots l_5 \\ m' &:= b m_0 \dots m_5 \\ r' &:= c r_0 \dots r_5 \end{aligned}$$

where

$$\begin{aligned} a &= f_l(g_0 g_4 g_6 g_{13} g_{18} h_3) \oplus g_{22} \oplus r_2 \oplus r_6 \\ b &= f_m(g_1 g_5 g_{10} g_{15} h_0 h_7) \oplus l_0 \oplus l_3 \oplus l_6 \\ c &= f_r(g_2(g_3 \oplus i)g_9 g_{14} g_{16} h_1) \oplus m_0 \oplus m_3 \oplus m_6 \end{aligned}$$

We define the successor function suc$\colon \mathbb{F}_2^{57} \times \mathbb{F}_2 \to \mathbb{F}_2^{57}$ *which takes a state* s *and an input* $i \in \mathbb{F}_2$ *and outputs the successor state* s'*. We overload the function* suc *on multiple-bit input which takes a state* s *and an input* $i \in \mathbb{F}_2^{n+1}$ *as*

$$\begin{aligned} \mathrm{suc}(s, i_0 \dots i_n) &= \mathrm{suc}(s', i_n) \\ &\textit{where } s' = \mathrm{suc}(s, i_0 \dots i_{n-1}) \end{aligned}$$

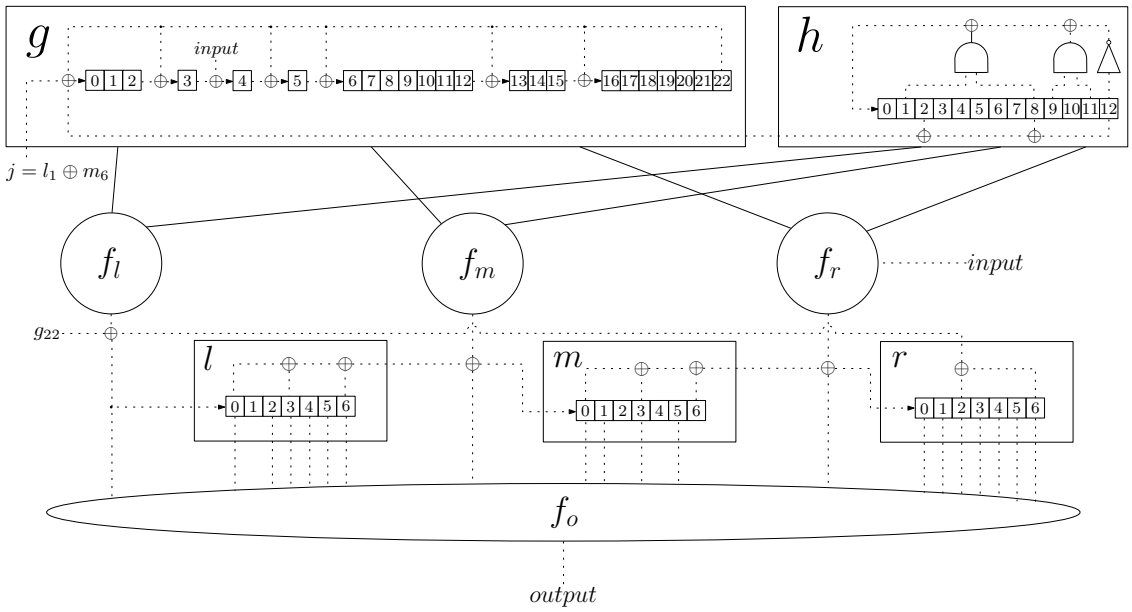


Figure 7: Schematic representation of the cipher

Definition 3.8. *The non-linear output filter function* $f_o\colon \mathbb{F}_2^{20} \to \mathbb{F}_2$ *has been deliberately omitted in this paper.*

Definition 3.9 (Output)**.** *Define the function output*: $\mathbb{F}_2^{57} \times \mathbb{F}_2 \to \mathbb{F}_2$ *which takes as input an internal state* $s = \langle g,h,l,m,r\rangle$ *and an input* $i \in \mathbb{F}_2$ *and returns the bit*

$$f_o(abcl_0l_2l_3l_4l_5l_6m_0m_1m_3m_5r_0r_1r_2r_3r_4r_5r_6)$$

where

$$\begin{aligned} a &= f_l(g_0g_4g_6g_{13}g_{18}h_3) \oplus g_{22} \oplus r_2 \oplus r_6 \\ b &= f_m(g_1g_5g_{10}g_{15}h_0h_7) \oplus l_0 \oplus l_3 \oplus l_6 \\ c &= f_r(g_2(g_3 \oplus i)g_9g_{14}g_{16}h_1) \oplus m_0 \oplus m_3 \oplus m_6 \end{aligned}$$

We also overload the function output on multiple-bit input which takes a state s and an input $i \in \mathbb{F}_2^{n+1}$ *as*

$$\begin{aligned} output(s, i_0 \dots i_n) &= output(s, i_0) \cdot output(s', i_1 \dots i_n) \\ \text{where } s' &= \mathrm{suc}(s, i_0). \end{aligned}$$

3.5 Cipher initialization

The following sequence of definitions describe how the cipher is initialized.

Definition 3.10. *Let* $init\colon \mathbb{F}_2^{23} \times \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{n+24}$ *be defined as*

$$\begin{aligned} init(g, \varepsilon) &:= g \\ init(g, x_0 \dots x_n) &:= init(G(g, 0, x_n), x_0 \dots x_{n-1}) \cdot g_{22} \end{aligned}$$

Definition 3.11. *Let* $p \in \mathbb{F}_2^{56}, q \in \mathbb{F}_2^{44}$ *and* $t \in \mathbb{F}_2^{43}$ *be defined as*

$$\begin{aligned} p &:= n_{C_0} \dots n_{C_{55}} + k_{40} \dots k_{95} \mod 2^{56} \\ q &:= (p_2 \dots p_{45}) \oplus (p_8 \dots p_{51}) \oplus (p_{12} \dots p_{55}) \\ t &:= init(q_{20} \dots q_{42}, q_0 \dots q_{19}) \end{aligned}$$

Then, the initial cipher state $s_0 = \langle g,h,l,m,r\rangle$ *is defined as*

$$\begin{aligned} g &:= t_0 \dots t_{22} \\ h &:= 0p_0 \dots p_{11} \\ l &:= t_{23} \dots t_{29} \\ m &:= t_{30} \dots t_{36} \\ r &:= t_{37} \dots t_{42}q_{43} \end{aligned}$$

3.6 Authentication protocol

This section describes the authentication protocol between a Megamos Crypto transponder and the vehicle immobilizer. This protocol is depicted in Figure 8. An annotated example trace is shown in Figure 5.

Definition 3.12. *Given a key* $k = k_0 \dots k_{95} \in \mathbb{F}_2^{96}$ *and an initial state* s_0 *as defined in Definition 3.11, the internal state of the cipher at step i is defined as*

$$\begin{aligned} s_i &:= \mathrm{suc}(s_{i-1}, k_{40-i}) && \forall i \in [1 \dots 40] \\ s_{i+41} &:= \mathrm{suc}(s_{i+40}, 0) && \forall i \in \mathbb{N} \end{aligned}$$

During authentication, the immobilizer starts by sending an authenticate command to the transponder. This command includes a 56-bit nonce n_C and the 28 bits a_C output by the cipher from state s_7. Then, the transponder responds with the next 20 output bits a_T, i.e., produced from state s_{35}.

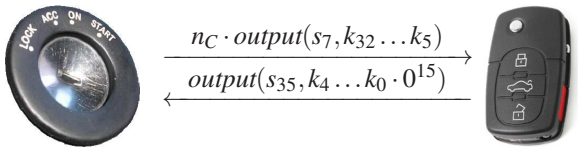


Figure 8: Megamos Crypto authentication protocol

4 Cipher Properties

This section describes several properties of the Megamos Crypto cipher which will be later used in the attacks.

4.1 Rollback

Given a cipher state it is possible to recover its previous state when this exists. Rolling-back the cipher is non-trivial due to the non-linear operations in the suc function. Next we describe precisely how to rollback the cipher to recover a predecessor state.

We start by rolling-back registers g and h. This computation is straightforward as described in the following definitions.

Definition 4.1. *The predecessor function for the non-linear feedback shift register* $H^{-1}\colon \mathbb{F}_2^{13} \to \mathbb{F}_2^{13}$ *is defined as*

$$H^{-1}(h_0 \ldots h_{12}) = h_1 \ldots h_{11}((h_2 \wedge h_9) \oplus (h_{10} \wedge h_{12}) \oplus \overline{h_0})$$

Definition 4.2. *The predecessor function for the Galois linear feedback shift register* $G^{-1}\colon \mathbb{F}_2^{23} \times \mathbb{F}_2 \times \mathbb{F}_2 \to \mathbb{F}_2^{23}$ *is defined as*

$$G^{-1}(g_0 \ldots g_{22}, i, j) = g_1 g_2 (g_3 \oplus b)(g_4 \oplus i)(g_5 \oplus b)(g_6 \oplus b) \\ g_7 \ldots g_{12}(g_{13} \oplus b) g_{14} g_{15} (g_{16} \oplus b) g_{17} \ldots g_{22} b$$

where $b = g_0 \oplus j$

Next we describe how to rollback registers l, m and r. A difficulty in doing that arises from the fact that m_6 in the predecessor state is not determined. To circumvent this issue, we need to first guess the bit m_6 and then check whether this guess is consistent with the rest of the state. For 18.75% of the states this condition is not met for neither $m_6 = 0$ nor $m_6 = 1$, which means that the state has no predecessor. For 62.5% of the states there is only one value of m_6 satisfying this condition, which means that they have only one predecessor state. Finally, 18.75% of the states have two possible predecessor states, one for $m_6 = 0$ and one for $m_6 = 1$. In this case both states have to be considered as potentially being the predecessor state. Given the fact that the average probability of having two predecessors equals the probability of having none the list of candidate predecessor states remains of a constant size.

A precise description of how to compute a predecessor state follows.

Definition 4.3 (Predecessor state)**.** *Let* $s' = \langle g', h', l', m', r' \rangle$ *be a cipher state and* $i \in \mathbb{F}_2$ *be an input bit. Then,* $s = \langle g, h, l, m, r \rangle$ *is a* predecessor cipher state *of* s' *if it satisfies*

$$\begin{aligned} h &= H^{-1}(h') \\ g &= G^{-1}(g', i, h_{12} \oplus h_8 \oplus h_2 \oplus l'_2 \oplus m_6) \\ l &= l'_1 \ldots l'_6 (m'_0 \oplus f_m(g_1 g_5 g_{10} g_{15} h_0 h_7 h) \oplus l'_4 \oplus l'_1) \\ m_6 &= r'_0 \oplus f_r(g_2(g_3 \oplus i) g_9 g_{14} g_{16} h_1) \oplus m'_4 \oplus m'_1 \\ m &= m'_1 \ldots m'_6 m_6 \\ r &= r'_1 \ldots r'_6 (l'_0 \oplus f_l(g_0 g_4 g_6 g_{13} g_{18} h_3) \oplus r'_3 \oplus g_{22}). \end{aligned}$$

4.2 Undoing cipher initialization

In this section we show that the cipher initialization procedure can be reverted. This means that given an initial state it is possible to recover the part of the secret key that was used for initialization. The following describes exactly how this can be achieved. We first introduce some auxiliary functions.

Definition 4.4. *Let* $init^{-1}\colon \mathbb{F}_2^{23} \times \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{n+1}$ *be defined as*

$$\begin{aligned} init^{-1}(g, x_0) &:= g \\ init^{-1}(g, x_0 \ldots x_n) &:= b \cdot init^{-1}(G^{-1}(g, 0, b), x_1 \ldots x_n) \\ &\quad \text{where } b = g_0 \oplus x_0 \end{aligned}$$

Definition 4.5. *Let* $Q^{-1}\colon \mathbb{F}_2^{n+12} \to \mathbb{F}_2^n$ *be defined as*

$$\begin{aligned} Q^{-1}(p_0 \ldots p_{11}) &:= \varepsilon \\ Q^{-1}(p_0 \ldots p_n) &:= (p_2 \oplus p_8 \oplus p_{12}) \cdot Q^{-1}(p_1 \ldots p_n) \end{aligned}$$

Proposition 4.6. *Given an initial state* $s_0 = \langle g, h, l, m, r \rangle$ *it is possible to compute secret key bits* $k_{40} \ldots k_{95}$.

The computation of the key bits is as follows.

$$\begin{aligned} t &:= g \cdot l \cdot m \cdot r \\ q &:= init^{-1}(t_0 \ldots t_{22}, t_{23} \ldots t_{42}) \cdot t_{43} \\ p &:= h \cdot Q^{-1}(h \cdot q) \\ k_{40} \ldots k_{95} &:= p - n_C \mod 2^{56}. \end{aligned}$$

4.3 Entropy of the non-linear feedback shift register

First and foremost, the initialization of the 13-bit non-linear feedback shift register (NLFSR) h is far from ideal. The NLFSR is initialized with only 12 bits by an almost linear function of the random nonce and the secret key. Adding upon the fact that, naturally, as the NLFSR h is not affected by other registers and the input, it is trivial to compute all successor states for a given h. Therefore, the search space for the 13-bit h register drops down to 2^{12}. Moreover, careful observation of the n_C value on the communication channel can leak information on whether the same value has been previously used for initializing h. For instance if the first 13 bits of n_C is the same for two different authentication attempts, depending on the

rest of the bits, the attacker can conclude with a certain confidence that the same state is used for initializing h. This weakness can be later exploited in a differential attack.

5 Cryptanalysis of Megamos Crypto

This section describes a cryptanalysis of the Megamos Crypto cipher. We first introduce a simple cryptanalysis which is easier-to-grasp and recovers the 96-bit secret key with a computational complexity of 2^{56}. Then, in Section 5.1 we reduce its computational complexity down to 2^{48}.

This analysis requires two successful authentication traces $T = \langle n_C, output(s_7, k_{32} \dots k_5), output(s_{35}, k_4 \dots k_0 \cdot 0^{15}) \rangle$ and $T' = \langle n'_C, output(s'_7, k_{32} \dots k_5), output(s'_{35}, k_4 \dots k_0 \cdot 0^{15}) \rangle$. Discarding from all internal states $s_{40} \in \mathbb{F}_2^{56}$ those guesses which produce different 15 output bits than the trace T which leaves $2^{56-15} = 2^{41}$ candidate states for s_{40}. Rolling the cipher backwards for each candidate up to state s_7, as shown in Section 4.1, leaves—on average—the same number of candidate states for s_7, namely 2^{41}. Each step requires guessing one input bit k_i but at the same time the output provides one bit of information. Note that this determines a guess for key bits $k_0 \dots k_{32}$. Rolling further the cipher backwards up to state s_0 requires guessing of $k_{33} \dots k_{39}$ while no output bits are produced. This brings the number of candidate states for s_0 to $2^{41+7} = 2^{48}$. For each candidate s_0, the remaining key bits $k_{40} \dots k_{95}$ can be recovered by undoing the initialization of the cipher as described in Section 4.2. This produces 2^{48} candidate keys $k_0 \dots k_{95}$. On average, there is only one candidate secret key $k_0 \dots k_{95}$ that together with n'_C produces the trace T'. This is because there are only 2^{48} candidates keys and 48 bits of information on the trace.

Time complexity on average, the aforementioned algorithm has a computational complexity of approximately 2^{56} encryptions. We have simulated an FPGA implementation of the algorithm on a Xilinx ISE 10.1 for synthesis and place & route. The results show that our implementation of a Megamos Crypto core covers approximately 1% of the Xilinx Spartan 3-1000 FPGA, the exact same chip that is employed in the COPACOBANA [42]. The maximum frequency that the core can run at is 160.33 MHz, which means we can test a single bit output in 6.237ns. Given this performance and area figures, a rough estimation suggests we can fit at least 50 Megamos Crypto cores in a Spartan 3-1000 FPGA. Considering that there are 120 such FPGA in a COPACOBANA, and since we can run them at 160.33MHz, we can run approximately $2^{39.8}$ tests per second. After every cycle, half of the candidate states are discarded, which means that a search takes less than two days on a COPACOBANA.

5.1 Reducing the computational complexity

Most of the computational complexity of the cryptanalysis described in Section 5 comes from iterating over all 2^{56} internal states s_{40}. In the following analysis we lower this complexity to 2^{48} by splitting the cipher state into two and using a time-memory trade-off. The main idea behind this optimization is to exploit the fact that components g and h are quite independent from components l, m and r. In fact, at each cipher step, there is only one bit of information from l, m, r which affects g, h, namely $l_1 \oplus m_6$. Conversely, there are only three bits of information from g, h that have an influence on components l, m, r.

In order reduce the complexity of the cryptanalysis an adversary $\mathscr{A}$ proceeds as follows.

1. Pre-computation: only once, and for each 2^{12} possible values of h, the adversary computes a table T_h as follows. For each $g \in \mathbb{F}_2^{23}$ and $j \in \mathbb{F}_2^8$ the adversary runs cipher components g and h one step forward. For this, $\mathscr{A}$ uses j_0 as a guess for $l_1 \oplus m_6$. At this stage $\mathscr{A}$ computes $f_0 := f_l(\cdot) f_m(\cdot) f_r(\cdot)$. From the resulting g and h, $\mathscr{A}$ repeats this procedure another 7 times, using j_i as a guess at step i and computing a three bit value f_i. At the end, she creates an entry in the table T_h of the form $< f_0 \dots f_7, j_0 \dots j_7, g >$. When the table is completed $\mathscr{A}$ sorts the table (on f, j).
2. As before $\mathscr{A}$ first eavesdrops one authentication trace between a legitimate transponder and an immobilizer. Thus $\mathscr{A}$ learns n_C, $output(s_7, k_{32} \dots k_5)$ and $output(s_{35}, k_4 \dots k_0 \cdot 0^{15})$.
3. Choose h.
4. Next the adversary will try to recover state s_{40}. For each $l, m, r \in \mathbb{F}_2^7$ the adversary runs these components 8 steps forward. At each step i she needs to guess 3 bits $f_i := f_l(\cdot) f_m(\cdot) f_r(\cdot)$ but she will be able to immediately discard half of these guesses as they will not produce the correct output bit $output(s_{40+i}, 0)$. At each step $\mathscr{A}$ will also compute $j_i : l_1 \oplus m_6$. At the end $\mathscr{A}$ has $2^{21+16} = 2^{37}$ bitstrings of the form $< f_0 \dots f_7, j_0 \dots j_7, l, m, r >$.
5. For each of these bitstrings $\mathscr{A}$ performs a lookup on $f_0 \dots f_7, j_0 \dots j_7$ in the table T_h and recovers g. On average, half of these lookups will not have a match in T_h. In that case the candidate state is discarded, leaving only 2^{36} full candidate states.
6. Each of these candidate states are then rolled forward another 7 steps. Only $2^{36-7} = 2^{29}$ of these states will produce the correct $output(s_{48}, 0^7)$ bits and the rest are discarded.

7. For each of these 2^{29} states the adversary proceeds as in Section 5, undoing the initialization and checking against a second trace.

Time and resource complexity

- Pre-computation: for building the tables T_h the adversary needs to run components g and h of the cipher 8 steps. This has a computational complexity of $2^{23+12+3} = 2^{38}$ cipher steps. The generated tables can be conveniently stored in memory using a structure for compression like `/n/f0/f1/.../f7/j/g.dat`. Storing all these tables require 12 terabyte of memory.
- As before, this cryptanalysis requires two successful authentication traces to recover the secret key. The most time intensive operation of this analysis is performing the 2^{37} lookups in the table for each of the 2^{12} values of h, i.e., 2^{49} table lookups.

The time-memory trade-off proposed in this section requires many indirect memory lookups and is therefore difficult to mount in practice with ordinary consumer hardware.

6 Partial Key-Update Attack

As it was described in Section 3.2, when the transponder is not locked, the Megamos Crypto transponder does not require authentication in order to write to memory. This makes it vulnerable to a trivial denial of service attack. An adversary just needs to flip one bit of the secret key of the transponder to disable it.

Besides this obvious weakness, there is another weakness regarding the way in which the secret key is written to the transponder. The secret key of Megamos Crypto is 96 bits long. As described in Section 3.1, these 96 bits are stored in 6 memory blocks of 16 bits each (blocks 4 to 9), see Figure 4. It is only possible to write one block at a time to the transponder. This constitutes a serious weakness since a secure key-update must be an atomic operation.

Next, we mount an attack which exploits this weakness to recover the secret key. For this attack we assume that an adversary $\mathscr{A}$ is able communicate with the car and transponder. She proceeds as follows.

1. The adversary first eavesdrops a successful authentication trace, obtaining n_C and a_C from the car.
2. Then, for $k = 0$ to $2^{16} - 1$ the adversary writes k on memory block 4 of the transponder, where key bits $k_0 \dots k_{15}$ are stored. After each write command $\mathscr{A}$ initiates an authentication attempt with the transponder, replaying n_C and a_C (remember that the transponder does not challenge the car). For one value of k the transponder will accept a_C and give an answer. Then $\mathscr{A}$ knows that $k_0 \dots k_{15} = k$.
3. The adversary proceeds similarly for blocks $5 \dots 9$ thus recovering the complete secret key.

Attack complexity this attack requires 6×2^{16} key-updates and the same amount of authentication attempts. This takes approximately 25 minutes for each block which adds up to a total of two and a half hours.

6.1 Optimizing the attack

The above attack is very powerful, in the worst case, the attacker needs to update the key on the transponder and make an authentication attempt 2^{16} times. However, the same attack can be applied with only one key-update and 2^{16} authentication attempts, by choosing carefully the value of n_C. The optimized attack can be mounted as follows:

1. As before, the adversary first eavesdrops a successful authentication trace, obtaining n_C, a_C and a_T.
2. then, she writes `0x0000` on memory block 9 which contains key bits $k_{80} \dots k_{95}$.
3. The adversary then increments the observed n_C value and attempts an authentication for each
$$n_C + inc \pmod{2^{56}}, \text{ where } 0 \leq inc < 2^{16}.$$
4. Repeating step 3) at most 2^{16} times, the transponder will accept one a_C value for a particular increment value inc and give an answer. Then $\mathscr{A}$ knows that $k_{80} \dots k_{95} = inc$.
5. The adversary proceeds similarly for blocks 8 and 7. At this point the adversary has recovered key bits $k_{48} \dots k_{95}$.
6. Next, the adversary guesses 15 key bits $k_{33} \dots k_{47}$.
7. Having $k_{33} \dots k_{95}$ the adversary is now able to initialize the cipher, obtain the initial state s_0 and run it forward up to state s_7. At this point the adversary has 2^{15} candidates for state s_7.
8. For each of these candidates, she runs the cipher forward 33 steps up to state s_{40}. While running the cipher forward the adversary is able to determine input bits $k_{32} \dots k_0$ by comparing the output bits to a_C and a_T from the trace.
9. Then, forward each candidate state at s_{40} to s_{55} and produce another 15 output bits to test on, although this time, with the known input of 15 zero bits. On average only one candidate survives this test. The adversary has now recovered the complete key.

Attack complexity This attack requires only one successful authentication trace. In total, we need to write three times on the memory of the transponder and perform 3×2^{16} authentications with the transponder. This can be done within 30 minutes using a Proxmark III. The computational complexity of the last three steps is 2^{15} encryptions which takes less than a second on a laptop.

7 Weak-Key Attack

During our experiments we executed the previous attack on several cars of different make and model. Many of the keys we recovered were of the form $k_0 = \cdots = k_{31} = 0$ and more or less random looking bits for $k_{32} \ldots k_{96}$ (although we have found keys where only ten of the 96 bits were ones). In the remainder of this paper we call such a key *weak*. Figure 9 shows some examples of weak keys we found during our experiments (on the vehicles indicated in Figure 2). To avoid naming concrete car models we use $A, B, C \ldots$ to represent car makes. We write numbers $X.1, X.2, X.3 \ldots$ to represent different car models of make X.

Car	Secret key
$A.1$	00000000**d8 b3967c5a3c3b29**
$A.2$	00000000**d9 b79d7a5b3c3b28**
$B.1$	0000000000 **0001040505090**5

Figure 9: Recovered keys from our own cars. Besides the evident 32 leading zero bits, every second nibble seems to encode a manufacturer dependant value, which further reduces the entropy of the key.

Apparently, the automotive industry has decided to use only 64 bits of the secret key, probably due to compatibility issues with legacy immobilizer systems. If a Megamos Crypto transponder uses such a weak key it is possible to recover this key quickly, even when the memory of the transponder is locked with a PIN code. To be concrete, a weak secret key with the bits $k_0 \ldots k_{31}$ fixed by the car manufacturer allows an adversary know the input bits of the cipher states $s_8 \ldots s_{55}$.

With known input to the cipher at states $s_8 \ldots s_{55}$, it is possible to pre-compute and sort on a 47 contiguous output bits for each internal state at s_8. However, such a table with 2^{56} entries requires a huge amount of storage. There are many time-memory tradeoff methods proposed in the literature over the last decades [2–5, 10, 33, 34, 49]. For example, a rainbow table shrinks the storage significantly, while requiring only a modest amount of computation for a lookup.

Concretely, in order to mount such an attack, an adversary $\mathscr{A}$ proceeds as follows.

1. Pre-computation: only once, the adversary computes the following rainbow table. First, she chooses n random permutations $R_0 \ldots R_{n-1}$ of $\mathbb{F}_2^{56} \to \mathbb{F}_2^{56}$ which she uses as reduction functions (colors). To compute a chain, the intermediate states are generated by

$$s_{i+1} = R_j(output(s_i, 0^{56})).$$

The chain begins with the first reduction function R_0. When a distinguished point (i.e., a state with a specific pattern like a prefix of z zero bits) is reached then the next reduction function R_{j+1} is used, in order to prevent chain merges. The chain is completed once a distinguished point is reached while using the last reduction function R_{n-1}, see Figure 10. The start and end values of each chain are stored in the rainbow table which is sorted on end values.

$$\begin{array}{ccccc} sp_0 & \overset{R_0(\cdot)}{\rightsquigarrow} & d_1 & \overset{R_1(\cdot)}{\rightsquigarrow} \cdots \overset{R_{n-2}(\cdot)}{\rightsquigarrow} & d_{n-1} & \overset{R_{n-1}(\cdot)}{\rightsquigarrow} d_n = ep_0 \\ \vdots & & & & \vdots & \\ sp_m & \overset{R_0(\cdot)}{\rightsquigarrow} & d'_1 & \overset{R_1(\cdot)}{\rightsquigarrow} \cdots \overset{R_{n-2}(\cdot)}{\rightsquigarrow} & d'_{n-1} & \overset{R_{n-1}(\cdot)}{\rightsquigarrow} d'_n = ep_m \end{array}$$

Figure 10: Construction of the rainbow table

2. As before $\mathscr{A}$ first eavesdrops one authentication trace between a legitimate transponder and an immobilizer. Thus $\mathscr{A}$ learns the car nonce n_C and 47 output bits $o_0 \ldots o_{46} = output(s_7, k_{32} \ldots k_5) \cdot output(s_{35}, k_4 \ldots k_0 \cdot 0^{15})$.
3. For each value of $u_0 \ldots u_8 \in \mathbb{F}_2^9$ and each reduction function R_j the adversary looks up $o_0 \ldots o_{46} u_0 \ldots u_8$ in the Rainbow table. In order to look up such a value she sets it as a state and runs the chain until the last distinguished point is reached at the last reduction function. Then, it performs n lookups in the rainbow table (one for each reduction function) to find the corresponding end point ep.
4. When the end point is found then the corresponding start point sp is used to find the previous internal state which generates $o_0 \ldots o_{46} u_0 \ldots u_8$ as output. Since we guessed the last 9 bits $u_0 \ldots u_8$, we should consider this as a candidate state.
5. Then, the adversary rolls back each of those states seven steps, guessing the input $k_{32} \ldots k_{39}$. This produces 2^8 candidate states for s_0. As before, for each candidate s_0 she undoes the initialization of the cipher and recovers the remaining key bits $k_{40} \ldots k_{95}$. These need to be tested with another trace.
6. If the test is passed then we have recovered the secret key. Otherwise the next $u_0 \ldots u_8$ should be considered at step 3).

Attack complexity This attack requires two successful authentication traces. This attack allows for a tradeoff between memory and computational complexity. The longer the chains the smaller the table gets but more computation is needed for each lookup. Just to give an impression of the feasibility of the attack we consider the following configuration. Take z to be 10 bits, therefore our distinguished states have 10 zero bits followed by

other 46 bits. We also take $64 = 2^6$ random permutations $R_0 \ldots R_{63}$. Then, following the computations of Oechslin [49], we get that the size of the rainbow table is $(2^{56})^{\frac{2}{3}} \approx 2^{37}$ entries of 12 bytes which is 1.5 TB. Regarding its computational complexity, we need to compute at step 3) 2^9 candidates for which we compute, for all 2^6 reduction functions and for all offsets, the end point. Since a chain has length at most 2^{16}, this takes at most $2^9 \times 2^6 \times 2^6 \times 2^{16} = 2^{37}$ encryptions. This can be computed within a few minutes on a laptop.

For building the rainbow table (needed only once), computation of the chains is sped up considerably by using FPGAs. Recently Kalenderi et. al. showed in [37] that a single FPGA (similar to the ones used in the COPACOBANA) computes chains 2824 times faster than a single 3GHz processor. They computed rainbow tables for the A5/1 cipher, which is reasonably similar to the Megamos Crypto cipher. Although the internal state of A5/1 is with 64 bits considerably larger than the 56 bits of Megamos Crypto, they are both designed for hardware implementation and both embed a non-linear component that causes some internal states to merge. Their experimental setup generates 345 chains for A5/1 in 830 milliseconds, which is roughly $\frac{345}{0.830} \approx 415$ chains per second. If we compute an estimate with respect to the difference in complexity, the COPACOBANA with a 120 FPGA-array can compute $415 \times (2^8)^{\frac{1}{3}} \times 120 \approx 2^{18.3}$ chains per second. That means it takes only $2^{37-18.3} \approx 2^{18.7}$ seconds, which is less than 5 days, to build the complete rainbow table.

8 Practical considerations and mitigation

Our attacks require close range wireless communication with both the immobilizer unit and the transponder. It is not hard to imagine real-life situations like valet parking or car rental where an adversary has access to both for a period of time. It is also possible to foresee a setup with two perpetrators, one interacting with the car and one wirelessly pickpocketing the car key from the victims pocket.

As mitigating measure, car manufacturers should set uniformly generated secret keys and for the devices which are not locked yet, set PIN codes and write-lock their memory after initialization. This obvious measures would prevent a denial of service attack, our partial key-update attack from Section 6 and our weak-key attack from Section 7.

Car owners can protect their own vehicles against a denial of service and the partial key-update attack, described in Section 6. These attacks only work if the adversary has write access to the memory of the transponder, which means that the lock-bit l_0 is set to zero. It is possible for a user to test for this property with any compatible RFID reader, like the Proxmark III, using our communication library. If $l_0 = 0$, then you should set the lock-bit l_0 to one. It is possible to set this bit without knowing the secret key or the PIN code. When dealing with the more recent version of the Megamos Crypto transponder (EM4170), users should also update the PIN code to a random bit-string before locking the transponder.

On the positive side, our first (cryptographic) attack is more computationally intensive than the attacks from Section 6 and 7 which makes it important to take the aforementioned mitigating measures in order to prevent the more inexpensive attacks. Unfortunately, our first attack is also hard to mitigate when the adversary has access to the car and the transponder (e.g., Valet and car rental). It seems infeasible to prevent an adversary from gathering two authentication traces. Furthermore, this attack exploits weaknesses in the core of the cipher's design (e.g., the size of the internal state). It would require a complete redesign of the cipher to fix these weaknesses. To that purpose, lightweight ciphers like Grain [32], Present [7] and KATAN [15] have been proposed in the literature and could be considered as suitable replacements for Megamos Crypto. Also, immobilizer products implementing AES are currently available in the market.

9 Conclusions

The implications of the attacks presented in this paper are especially serious for those vehicles with keyless ignition. At some point the mechanical key was removed from the vehicle but the cryptographic mechanisms were not strengthened to compensate.

We want to emphasize that it is important for the automotive industry to migrate from weak proprietary ciphers like this to community-reviewed ciphers such as AES [14] and use it according to the guidelines. For a few years already, there are contactless smart cards on the market [48, 50] which implement AES and have a fairly good pseudo-random number generator. It is surprising that the automotive industry is reluctant to migrate to such transponders considering the cost difference of a better chip (≤ 1 USD) in relation to the prices of high-end car models ($\geq 50,000$ USD). Since most car keys are actually fairly big, the transponder design does not really have to comply with the (legacy) constraints of minimal size.

Following the principle of responsible disclosure, we have notified the manufacturer of our findings back in November 2012. Since then we have an open communication channel with them. We understand that measures have been taken to prevent the weak-key and partial key-update attacks when the transponder was improperly configured.

10 Acknowledgments

The authors would like to thank Bart Jacobs for his firm support.

References

[1] Embedded avr microcontroller including rf transmitter and immobilizer lf functionality for remote keyless entry - ATA5795C. Product Datasheet, November 2011. Atmel Corporation.

[2] AVOINE, G., JUNOD, P., AND OECHSLIN, P. Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Transactions on Information and System Security (TISSEC 2008) 11*, 4 (2008), 1–22.

[3] BABBAGE, S. A space/time tradeoff in exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection* (1995), vol. 408 of *Conference Publications*, IEEE Computer Society, pp. 161–166.

[4] BIRYUKOV, A., MUKHOPADHYAY, S., AND SARKAR, P. Improved time-memory trade-offs with multiple data. In *13th International Workshop on Selected Areas in Cryptography (SAC 2006)* (2006), vol. 3897 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 110–127.

[5] BIRYUKOV, A., AND SHAMIR, A. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *6th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2000)* (2000), vol. 1976 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–13.

[6] BOGDANOV, A. Linear slide attacks on the KeeLoq block cipher. In *3rd International Conference on Information Security and Cryptology (INSCRYPT 2007)* (2007), vol. 4990 of *Lecture Notes in Computer Science*, Springer, pp. 66–80.

[7] BOGDANOV, A., KNUDSEN, L. R., LEANDER, G., PAAR, C., POSCHMANN, A., ROBSHAW, M. J., SEURIN, Y., AND VIKKELSOE, C. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007*. Springer, 2007, pp. 450–466.

[8] BOGDANOV, A., AND PAAR, C. On the security and efficiency of real-world lightweight authentication protocols. In *1st Workshop on Secure Component and System Identification (SECSI 2008)* (2008), ECRYPT.

[9] BONO, S. C., GREEN, M., STUBBLEFIELD, A., JUELS, A., RUBIN, A. D., AND SZYDLO, M. Security analysis of a cryptographically-enabled RFID device. In *14th USENIX Security Symposium (USENIX Security 2005)* (2005), USENIX Association, pp. 1–16.

[10] BORST, J., PRENEEL, B., VANDEWALLE, J., AND V, J. On the time-memory tradeoff between exhaustive key search and table precomputation. In *19th Symposium in Information Theory in the Benelux* (1998), pp. 111–118.

[11] CHECKOWAY, S., MCCOY, D., KANTOR, B., ANDERSON, D., SHACHAM, H., SAVAGE, S., KOSCHER, K., CZESKIS, A., ROESNER, F., AND KOHNO, T. Comprehensive experimental analyses of automotive attack surfaces. In *20th USENIX Security Symposium (USENIX Security 2011)* (2011), USENIX Association, pp. 77–92.

[12] COURTOIS, N. T., BARD, G. V., AND WAGNER, D. Algebraic and slide attacks on KeeLoq. In *15th International Workshop on Fast Software Encryption (FSE 2008)* (2008), vol. 5086 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 97–115.

[13] COURTOIS, N. T., O'NEIL, S., AND QUISQUATER, J.-J. Practical algebraic attacks on the Hitag2 stream cipher. In *12th Information Security Conference (ISC 2009)* (2009), vol. 5735 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 167–176.

[14] DAEMEN, J., AND RIJMEN, V. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.

[15] DE CANNIERE, C., DUNKELMAN, O., AND KNEŽEVIĆ, M. KATAN and KTANTANa family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 272–288.

[16] DE KONING GANS, G., HOEPMAN, J.-H., AND GARCIA, F. D. A practical attack on the MIFARE Classic. In *8th Smart Card Research and Advanced Applications Conference (CARDIS 2008)* (2008), vol. 5189 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 267–282.

[17] DIAGNOSTICS, A. AD900Pro transponder duplicating system, operation manual, unlocking technology, March 2011.

[18] DRIESSEN, B., HUND, R., WILLEMS, C., PAAR, C., AND HOLZ, T. Don't trust satellite phones: A security analysis of two satphone standards. In *33rd IEEE Symposium on Security and Privacy (S&P 2012)* (2012), IEEE Computer Society, pp. 128–142.

[19] EM Microelectronic-Marin SA. http://www.emmicroelectronic.com. CH-2074 Marin/Switzerland.

[20] Crypto contactless identification device, V4070. Product Datasheet, Oct 1997. EM Microelectronic-Marin SA.

[21] 125khz crypto read/write contactless identification device, EM4170. Product Datasheet, Mar 2002.

EM Microelectronic-Marin SA.

[22] Custom automotive. retrieved at December 12th, 2014, from http://www.datasheetarchive.com/EM+MICROELECTRONIC-MARIN-datasheet.html, September 2002. EM Microelectronic-Marin SA.

[23] EM4170 application note, AN407. RFID Application Note 407, Sep 2002. EM Microelectronic-Marin SA.

[24] FLUHRER, S., MANTIN, I., AND SHAMIR, A. Weaknesses in the key scheduling algorithm of RC4. In *8th International Workshop on Selected Areas in Cryptography (SAC 2001)* (2001), vol. 2259 of *Lecture Notes in Computer Science*, pp. 1–24.

[25] FRANCILLON, A., DANEV, B., AND ČAPKUN, S. Relay attacks on passive keyless entry and start systems in modern cars. In *18th Network and Distributed System Security Symposium (NDSS 2011)* (2011), The Internet Society.

[26] GARCIA, F. D., DE KONING GANS, G., MUIJRERS, R., VAN ROSSUM, P., VERDULT, R., WICHERS SCHREUR, R., AND JACOBS, B. Dismantling MIFARE Classic. In *13th European Symposium on Research in Computer Security (ESORICS 2008)* (2008), vol. 5283 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 97–114.

[27] GARCIA, F. D., DE KONING GANS, G., AND VERDULT, R. Exposing iClass key diversification. In *5th USENIX Workshop on Offensive Technologies (WOOT 2011)* (2011), USENIX Association, pp. 128–136.

[28] GARCIA, F. D., DE KONING GANS, G., VERDULT, R., AND MERIAC, M. Dismantling iClass and iClass Elite. In *17th European Symposium on Research in Computer Security (ESORICS 2012)* (2012), vol. 7459 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 697–715.

[29] GARCIA, F. D., VAN ROSSUM, P., VERDULT, R., AND WICHERS SCHREUR, R. Wirelessly pickpocketing a MIFARE Classic card. In *30th IEEE Symposium on Security and Privacy (S&P 2009)* (2009), IEEE Computer Society, pp. 3–15.

[30] GARCIA, F. D., VAN ROSSUM, P., VERDULT, R., AND WICHERS SCHREUR, R. Dismantling SecureMemory, CryptoMemory and CryptoRF. In *17th ACM Conference on Computer and Communications Security (CCS 2010)* (2010), ACM, pp. 250–259.

[31] GOLIĆ, J. D. Cryptanalysis of alleged A5 stream cipher. In *16th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1997)* (1997), vol. 1233 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 239–255.

[32] HELL, M., JOHANSSON, T., AND MEIER, W. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing 2*, 1 (2007), 86–93.

[33] HELLMAN, M. E. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory 26*, 4 (1980), 401–406.

[34] HONG, J., AND MOON, S. A comparison of cryptanalytic tradeoff algorithms. *Journal of Cryptology* (2010), 1–79.

[35] IMMLER, V. Breaking hitag 2 revisited. *Security, Privacy, and Applied Cryptography Engineering (SPACE 2012) 7644* (2012), 126–143.

[36] INDESTEEGE, S., KELLER, N., DUNKELMANN, O., BIHAM, E., AND PRENEEL, B. A practical attack on KeeLoq. In *27th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2008)* (2008), vol. 4965 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–8.

[37] KALENDERI, M., PNEVMATIKATOS, D., PAPAEFSTATHIOU, I., AND MANIFAVAS, C. Breaking the gsm a5/1 cryptography algorithm with wainbow tables and high-end FPGAS. In *22nd International Conference on Field Programmable Logic and Applications (FPL 2012)* (2012), IEEE Computer Society, pp. 747–753.

[38] KASPER, M., KASPER, T., MORADI, A., AND PAAR, C. Breaking KeeLoq in a flash: on extracting keys at lightning speed. In *2nd International Conference on Cryptology in Africa, Progress in Cryptology (AFRICACRYPT 2009)* (2009), vol. 5580 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 403–420.

[39] KEYLINE. Transponder guide. http://www.keyline.it/files/884/transponder_guide_16729.pdf, 2012.

[40] KOSCHER, K., CZESKIS, A., ROESNER, F., PATEL, F., KOHNO, T., CHECKOWAY, S., MCCOY, D., KANTOR, B., ANDERSON, D., SHACHAM, H., AND SAVAGE, S. Experimental security analysis of a modern automobile. In *31rd IEEE Symposium on Security and Privacy (S&P 2010)* (2010), IEEE Computer Society, pp. 447–462.

[41] KOUSHANFAR, F., SADEGHI, A.-R., AND SEUDIE, H. Eda for secure and dependable cybercars: Challenges and opportunities. In *49th Design Automation Conference (DAC 2012)* (2012), ACM, pp. 220–228.

[42] KUMAR, S., PAAR, C., PELZL, J., PFEIFFER, G., AND SCHIMMLER, M. Breaking ciphers with COPACOBANA—a cost-optimized parallel code breaker. In *Cryptographic Hardware and Embedded Systems (CHES 2006)* (2006), vol. 4249 of *Lec-*

ture Notes in Computer Science, Springer-Verlag, pp. 101–118.

[43] LEMKE, K., SADEGHI, A.-R., AND STÜBLE, C. Anti-theft protection: Electronic immobilizers. *Embedded Security in Cars* (2006), 51–67.

[44] LEMKE, K., SADEGHI, A.-R., AND STBLE, C. An open approach for designing secure electronic immobilizers. In *Information Security Practice and Experience (ISPEC 2005)* (2005), vol. 3439 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 230–242.

[45] LUCKS, S., SCHULER, A., TEWS, E., WEINMANN, R.-P., AND WENZEL, M. Attacks on the DECT authentication mechanisms. In *9th Cryptographers' Track at the RSA Conference (CT-RSA 2009)* (2009), vol. 5473 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 48–65.

[46] NOHL, K., EVANS, D., STARBUG, AND PLÖTZ, H. Reverse engineering a cryptographic RFID tag. In *17th USENIX Security Symposium (USENIX Security 2008)* (2008), USENIX Association, pp. 185–193.

[47] NOHL, K., TEWS, E., AND WEINMANN, R.-P. Cryptanalysis of the DECT standard cipher. In *17th International Workshop on Fast Software Encryption (FSE 2010)* (2010), vol. 6147 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–18.

[48] MIFARE DESFire EV1 contactless multi-application IC, MF3ICDx21. Product short data sheet, December 2010. NXP Semiconductors.

[49] OECHSLIN, P. Making a faster cryptanalytic time-memory trade-off. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)* (2003), vol. 2729 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 617–630.

[50] Smart MX secure smart card controller IC, P5CC018. Objective Specification, Revision 1.0, April 2003. Philips Semiconductors.

[51] ROLLES, R. Unpacking virtualization obfuscators. In *3rd USENIX Workshop on Offensive Technologies (WOOT 2009)* (2009).

[52] SOOS, M., NOHL, K., AND CASTELLUCCIA, C. Extending SAT solvers to cryptographic problems. In *12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)* (2009), vol. 5584 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 244–257.

[53] SUN, S., HU, L., XIE, Y., AND ZENG, X. Cube cryptanalysis of Hitag2 stream cipher. In *10th International Conference on Cryptology and Network Security (CANS 2011)* (2011), vol. 7092 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 15–25.

[54] TILLICH, S., AND WÓJCIK, M. Security analysis of an open car immobilizer protocol stack. In *10th International Conference on Applied Cryptograpy and Network Security (ACNS 2012)* (2012).

[55] VAN OURS, J. C., AND VOLLAARD, B. The engine immobilizer: a non-starter for car thieves, 2011.

[56] VERDULT, R., DE KONING GANS, G., AND GARCIA, F. D. A toolbox for RFID protocol analysis. In *4th International EURASIP Workshop on RFID Technology (EURASIP RFID 2012)* (2012), IEEE Computer Society, pp. 27–34.

[57] VERDULT, R., GARCIA, F. D., AND BALASCH, J. Gone in 360 seconds: Hijacking with Hitag2. In *21st USENIX Security Symposium (USENIX Security 2012)* (2012), USENIX Association, pp. 237–252.

[58] ŠTEMBERA, P., AND NOVOTNÝ, M. Breaking Hitag2 with reconfigurable hardware. In *14th Euromicro Conference on Digital System Design (DSD 2011)* (2011), IEEE Computer Society, pp. 558–563.

[59] WANG, P.-C., HOU, T.-W., WU, J.-H., AND CHEN, B.-C. A security module for car appliances. *International Journal of World Academy Of Science, Engineering and Technology 26* (2007), 155–160.

[60] WIENER, I. Philips/NXP Hitag2 PCF7936/46/47/52 stream cipher reference implementation. http://cryptolib.com/ciphers/hitag2/, 2007.

[61] WOLF, M., WEIMERSKIRCH, A., AND WOLLINGER, T. State of the art: Embedding security in vehicles. *EURASIP Journal on Embedded Systems 2007* (2007), 074706.

[62] WU, J.-H., KUNG, C.-C., RAO, J.-H., WANG, P.-C., LIN, C.-L., AND HOU, T.-W. Design of an in-vehicle anti-theft component. In *8th International Conference on Intelligent Systems Design and Applications (ISDA 2008)* (2008), vol. 1, IEEE