Gone in 360 Seconds: Hijacking with Hitag2

Roel Verdult   Flavio D. Garcia

Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands.
{rverdult,flaviog}@cs.ru.nl

Josep Balasch

KU Leuven ESAT/COSIC and IBBT
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
josep.balasch@esat.kuleuven.be

Abstract

An electronic vehicle immobilizer is an anti-theft device which prevents the engine of the vehicle from starting unless the corresponding transponder is present. Such a transponder is a passive RFID tag which is embedded in the car key and wirelessly authenticates to the vehicle. It prevents a perpetrator from hot-wiring the vehicle or starting the car by forcing the mechanical lock. Having such an immobilizer is required by law in several countries. Hitag2, introduced in 1996, is currently the most widely used transponder in the car immobilizer industry. It is used by at least 34 car makes and fitted in more than 200 different car models. Hitag2 uses a proprietary stream cipher with 48-bit keys for authentication and confidentiality. This article reveals several weaknesses in the design of the cipher and presents three practical attacks that recover the secret key using only wireless communication. The most serious attack recovers the secret key from a car in less than six minutes using ordinary hardware. This attack allows an adversary to bypass the cryptographic authentication, leaving only the mechanical key as safeguard. This is even more sensitive on vehicles where the physical key has been replaced by a keyless entry system based on Hitag2. During our experiments we managed to recover the secret key and start the engine of many vehicles from various makes using our transponder emulating device. These experiments also revealed several implementation weaknesses in the immobilizer units.

1 Introduction

In the past, most cars relied only on mechanical keys to prevent a hijacker from stealing the vehicle. Since the ’90s most car manufacturers incorporated an electronic car immobilizer as an extra security mechanism in their vehicles. From 1995 it is mandatory that all cars sold in the EU are fitted with such an immobilizer device, according to European directive 95/56/EC. Similar regulations apply to other countries like Australia, New Zealand (AS/NZS 4601:1999) and Canada (CAN/ULC S338-98). An electronic car immobilizer consists of two main components: a small transponder chip which is embedded in (the plastic part of) the car key, see Figure 1; and a reader which is located somewhere in the dashboard of the vehicle and has an antenna coil around the ignition, see Figure 2.

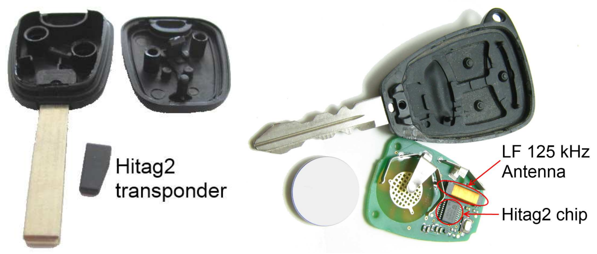


Figure 1: Car keys with a Hitag2 transponder/chip

The transponder is a passive RFID tag that operates at a low frequency wave of 125 kHz. It is powered up when it comes in proximity range of the electronic field of the reader. When the transponder is absent, the immobilizer unit prevents the vehicle from starting the engine.

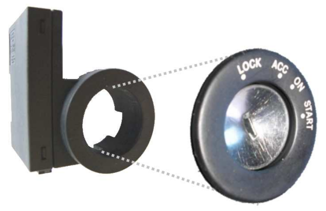


Figure 2: Immobilizer unit around the ignition barrel

A distinction needs to be made with remotely operated central locking system, which opens the doors, is battery powered, operates at a ultra-high frequency (UHF) of 433 MHz, and only activates when the user pushes a button on the remote key. More recent car keys are often deployed with a hybrid chip that supports the battery powered ultra-high frequency as well as the passive low frequency communication interface.

With the Hitag2 family of transponders, its manufacturer NXP Semiconductors (formerly Philips Semiconductors) leads the immobilizer market [34]. Figure 4 shows a list containing some of the vehicles that are deployed with a Hitag2 transponder. Even though NXP boosts "Unbreakable security levels using mutual authentication, challenge-response and encrypted data communication"[1], it uses a shared key of only 48 bits.

Since 1988, the automotive industry has moved towards the so-called keyless ignition or keyless entry in their high-end vehicles [26]. In such a vehicle the mechanical key is no longer present and it has been replaced by a start button like the one shown in Figure 3. The only anti-theft mechanism left in these vehicles is the immobilizer. Startlingly, many keyless ignition or entry vehicles sold nowadays are still based on the Hitag2 cipher. In some keyless entry cars Hitag2 is also used as a backup mechanism for opening the doors, e.g., when the battery of the remote is depleted.

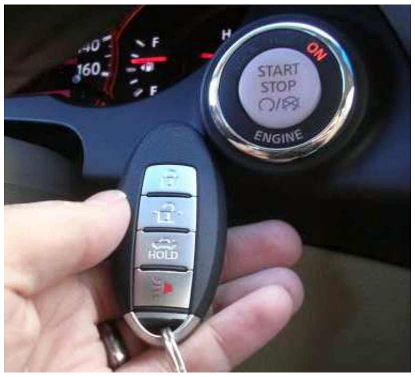


Figure 3: Keyless hybrid transponder and engine start/stop button

Related work

A similar immobilizer transponder is produced by Texas Instruments under the name Digital Signature Transponder (DST). It is protected by a different proprietary cryptographic algorithm that uses a secret key of only 40 bits. The workings of these algorithms are reversed engineered by Bono et al. in [10]. Francillon et al. demonstrated in [18] that is possible to relay in real-time the (encrypted) communication of several keyless entry systems. The article shows that in some cases such a communication can be intercepted over a distance of at least 100 meters.

[1] http://www.nxp.com/products/automotive/car_access_immobilizers/immobilizer/

Make	Models
Acura	CSX, MDX, RDX, TL, TSX
Alfa Romeo	156, 159, 166, Brera, Giulietta, Mito, Spider
Audi	A8
Bentley	Continental
BMW	**Serie 1**, 5, 6, 7, all bikes
Buick	Enclave, Lucerne
Cadillac	BLS, DTS, Escalade, SRX, STS, XLR
Chevrolet	Avanlache, Caprice, Captiva, Cobalt, Equinox, Express, HHR Impala, Malibu, Montecarlo, Silverado, Suburban, Tahoe Trailblazer, Uplander
Chrysler	300C, Aspen, Grand Voyager, Pacifica, Pt Cruiser, Sebring Town Country, Voyager
Citroen	**Berlingo**, C-Crosser, C2, **C3**, **C4**, C4 Picasso, **C5**, C6, C8 Nemo, Saxo, Xsara, Xsara Picasso
Dacia	Duster, **Logan**, Sandero
Daewoo	Captiva, Windstorm
Dodge	Avenger, Caliber, Caravan, Charger, Dakota, Durango Grand Caravan, Journey, Magnum, Nitro, Ram
Fiat	500, Bravo, Croma, Daily, Doblo, Fiorino, Grande Punto Panda, Phedra, Ulysse, Scudo
GMC	Acadia, Denali, Envoy, Savana, Siera, Terrain, Volt, Yukon
Honda	Accord, **Civic**, CR-V, Element, Fit, Insight, Stream, Jazz, Odyssey, Pilot, Ridgeline, most bikes
Hummer	H2, H3
Hyundai	130, Accent, Atos Prime, Coupe, Elantra, Excel, Getz Grandeur, **I30**, Matrix, Santafe, Sonata, Terracan, Tiburon Tucoson, Tuscanti
Isuzu	D-Max
Iveco	35C11, Eurostar, New Daily, S-2000
Jeep	Commander, Compass, Grand Cherokee, Liberty, Patriot Wrangler
Kia	Carens, Carnival, Ceed, Cerato, Magentis, Mentor, Optima Picanto, Rio, Sephia, Sorento, Spectra, Sportage
Lancia	Delta, Musa, Phedra
Mini	Cooper
Mitsubishi	380, Colt, Eclipse, Endeavor, Galant, Grandis, L200 Lancer, Magna, Outlander, Outlander, Pajero, Raider
Nissan	Almera, **Juke**, **Micra**, Pathfinder, Primera, Qashqai, Interstar Note, Xterra
Opel	Agila, Antara, Astra, Corsa, Movano, Signum, Vectra Vivaro, Zafira
Peugeot	**106**, **206**, 207, **307**, 406, 407, 607, 807, 1007, 3008, 5008 Beeper, Partner, **Boxer**, RCZ
Pontiac	G5, G6, Pursuit, Solstice, Torrent
Porsche	Cayenne
Renault	**Clio**, Duster, **Kangoo**, **Laguna II**, Logan, Master **Megane**, Modus, Sandero, **Trafic**, Twingo
Saturn	Aura, Outlook, Sky, Vue
Suzuki	Alto, Grand Vitara, Splash, Swift, Vitara, XL-7
Volkswagen	Touareg, Phaeton

Figure 4: Vehicles using Hitag2 [29] – boldface indicates vehicles we tested

The history of the NXP Hitag2 family of transponders overlaps with that of other security products designed and deployed in the late nineties, such as Keeloq [8, 13, 27, 28], MIFARE Classic [12, 19, 22, 35], CryptoMemory [4, 5, 23] or iClass [20, 21]. Originally, information on Hitag2 transponders was limited to data sheets with high level descriptions of the chip's functionality [36], while details on the proprietary cryptographic algorithms were kept secret by the manufacturer. This phase, in which security was strongly based on obscurity, lasted until in 2007 when the Hitag2 inner workings were reverse engineered [47]. Similarly to its predecessor Crypto1 (used in MIFARE Classic), the Hitag2 cipher consists of a 48 bit Linear Feedback Shift Register (LFSR) and a non-linear filter function used to output keystream. The publication of the Hitag2 cipher attracted the interest of the scientific community. Courtois et al. [14] were the first to study the strength of the Hitag2 stream cipher to algebraic attacks by transforming the cipher state into a system of equations and using SAT solvers to perform key recovery attacks. Their most practical attack requires two days computation and a total of four eavesdropped authentication attempts to extract the secret key. A more efficient attack, requiring 16 chosen initialization vectors (IV) and six hours of computations, was also proposed. However, and as noted by the authors themselves, chosen-IV attacks are prevented by the Hitag2 authentication protocol (see Sect. 3.5), thus making this attack unfeasible in practice.

In [42], Soos et al. introduced a series of optimizations on SAT solvers that made it possible to reduce the attack time of Curtois et al. to less than 7 hours. More recently, Štembera and Novotný [45] implemented a brute-force attack that could be carried out in less than two hours by using the COPACOBANA[2] high-performance cluster of FPGAs. Note however, that such attack would require about 4 years if carried out on a standard PC. Finally, Sun et. al [44] tested the security of the Hitag2 cipher against cube attacks. Although according to their results the key can be recovered in less than a minute, this attack requires chosen initialization vectors and thus should be regarded as strictly theoretical.

Our contribution

In this paper, we show a number of vulnerabilities in the Hitag2 transponders that enable an adversary to retrieve the secret key. We propose three attacks that extract the secret key under different scenarios. We have implemented and successfully executed these attacks in practice on more than 20 vehicles of various make and model. On all these vehicles we were able to use an emulating device to bypass the immobilizer and start the vehicle.

Concretely, we found the following vulnerabilities in Hitag2.

- The transponder lacks a pseudo-random number generator, which makes the authentication procedure vulnerable to replay attacks. Moreover, the transponder provides known data when a read command is issued on the block where the transponder's identity is stored, allowing to recover keystream. Redundancy in the commands allow an adversary to expand this keystream to arbitrary lengths. This means that the transponder provides an arbitrary length keystream oracle.

- With probability $1/4$ the output bit of the cipher is determined by only 34 bits of the internal state. As a consequence, (on average) one out of four authentication attempts leaks one bit of information about the secret key.

- The 48 bit internal state of the cipher is only randomized by a nonce of 32 bits. This means that 16 bits of information over the secret key are persistent throughout different sessions.

We exploit these vulnerabilities in the following three practical attacks.

- The first attack exploits the malleability of the cipher and the fact that the transponder does not have a pseudo-random number generator. It uses a keystream shifting attack following the lines of [16]. This allows an adversary to first get an authentication attempt from the reader which can later be replayed to the transponder. Exploiting the malleability of the cipher, this can be used to read known plaintext (the identity of the transponder) and recover keystream. In a new session the adversary can use this keystream to read any other memory block (with exception of the secret key when configured correctly) within milliseconds. When the key is not read protected, this attack can also be used to read the secret key. This was in fact the case for most vehicles we tested from a French car make.

- The second attack is slower but more general in the sense that the same attack strategy can be applied to other LFSR based ciphers. The attack uses a time/memory tradeoff as proposed in [3, 6, 7, 11, 25, 38]. Exploiting the linear properties of the LFSR, we are able to efficiently generate the lookup table, reducing the complexity from 2^{48} to 2^{37} encryptions. This attack recovers the secret key regardless of the read protection configuration of the transponder. It requires 30 seconds of communication with the transponder and another 30 seconds to perform 2000 table lookups.

- The third attack is also the most powerful, as it only requires a few authentication attempts from the car immobilizer to recover the secret key (assuming that

[2]http://www.copacobana.org

the adversary knows a valid transponder *id*). This cryptanalytic attack exploits dependencies among different sessions and a low degree determination of the filter function used in the cipher. In order to execute this attack, an adversary first gathers 136 partial authentication attempts from the car. This can be done within one minute. Then, the adversary needs to perform 2^{35} operations to recover the secret key. This takes less than five minutes on an ordinary laptop.

Furthermore, besides looking into the security aspects of Hitag2 we also study how it is deployed and integrated in car immobilizer systems by different manufacturers. Our study reveals that in many vehicles the transponder is misconfigured by having readable or default keys, and predictable passwords, whereas the immobilizer unit employs weak pseudo-random number generators. All cars we tested use identifier white-listing as an additional security mechanism. This means that in order to use our third attack to hijack a car, an adversary first needs to eavesdrop, guess or wirelessly pickpocket a legitimate transponder *id*, see Section 7.5.

Following the principle of responsible disclosure, we have contacted the manufacturer NXP and informed them of our findings six months ahead of publication. We have also provided our assistance in compiling a document to inform their customers about these vulnerabilities. The communication with NXP has been friendly and constructive. NXP encourages the automotive industry for years to migrate to more secure products that incorporate strong and community-reviewed ciphers like AES [15]. It is surprising that the automotive industry is reluctant to migrate to secure products given the cost difference of a better chip (≤ 1 USD) in relation to the prices of high-end car models ($\geq 50,000$ USD).

2 Hardware setup

Before diving into details about Hitag2, this section introduces the experimental platform we have developed in order to carry out attacks in real-life deployments of car immobilizer systems. In particular, we have built a portable and highly flexible setup allowing us to i) eavesdrop communications between Hitag2 readers and transponders, ii) emulate a Hitag2 reader, and iii) emulate a Hitag2 transponder. Figure 5 depicts our setup in the setting of eavesdropping communications between a reader and a transponder.

The central element of our experimental platform is the Proxmark III board[3], originally developed by Jonathan Westhues[4], and designed to work with RFID

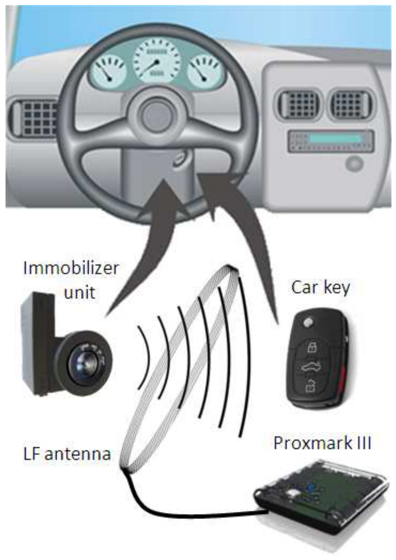


Figure 5: Experimental setup for eavesdropping

transponders ranging from low frequency (125 kHz) to high frequency (13.56 MHz). The Proxmark III board cost around 200 USD and comes equipped with a FPGA and an ARM microcontroller. Low-level RF operations such as modulation/demodulation are carried out by the FPGA, whereas high-level operations such as encoding/decoding of frames are performed in the microcontroller.

Hitag2 tags are low frequency transponders used in proximity area RFID applications [36]. Communication from reader to transponder is encoded using Binary Pulse Length Modulation (BPLM), whereas from transponder to reader it can be encoded using either Manchester or Biphase coding. In order to eavesdrop, generate, and read communications from reader to transponder, we added support for encoding/decoding BPLM signals, see Figure 6.

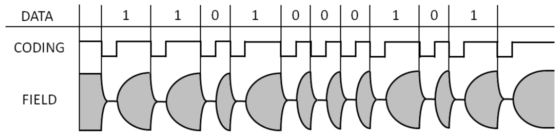


Figure 6: Reader modulation of a *read* command

For the transponder side, we have also added the functionalities to support the Manchester coding scheme as shown in Figure 7.

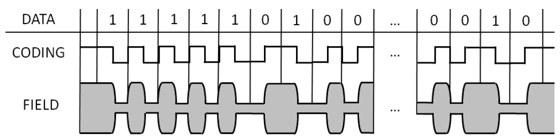


Figure 7: Communication from transponder to reader

[3] http://www.proxmark.org
[4] http://cq.cx/proxmark3.pl

3 Hitag2

This section describes Hitag2 in detail. Most of this information is in the public domain. We first describe the Hitag2 functionality, memory structure, and communication protocols, this comes mostly from the product data sheet [36]. Then we describe the cipher and the authentication protocol which was previously reverse engineered in [47]. In Section 3.7 we show that it is possible to run the cipher backwards which we use in our attacks.

We first need to introduce some notation. Let $\mathbb{F}_2 = \{\texttt{0},\texttt{1}\}$ the field of two elements (or the set of Booleans). The symbol $\oplus$ denotes exclusive-or (XOR) and 0^n denotes a bitstring of n zero-bits. Given two bitstrings x and y, xy denotes their concatenation. $\overline{x}$ denotes the bitwise complement of x. We write y_i to denote the i-th bit of y. For example, given the bitstring $y = \texttt{0x03}$, $y_0 = y_1 = 0$ and $y_6 = y_7 = 1$. We denote encryptions by $\{-\}$.

3.1 Functionality

Access to the Hitag2 memory contents is determined by pre-configured security policies. Hitag2 transponders offer up to three different modes of operation:

1. In *public mode* the contents of the user data pages are simply broadcast by the transponder once it is powered up.

2. In *password mode* reader and transponder authenticate each other by interchanging their passwords. Communication is carried out in the clear, therefore this authentication procedure is vulnerable to replay attacks.

3. In *crypto mode* the reader and the transponder perform a mutual authentication by means of a 48-bit shared key. Communication between reader and transponder is encrypted using a proprietary stream cipher. This mode is used in car immobilizer systems and will be the focus of this paper.

3.2 Memory

Hitag2 transponders have a total of 256 bits of non-volatile memory (EEPROM) organized in 8 blocks of 4 bytes each. Figure 8 illustrates the memory contents of a transponder configured in crypto mode. Block 0 stores the read-only transponder identifier; the secret key is stored in blocks 1 and 2; the password and configuration bits in block 3; blocks 4 till 7 store user defined memory. Access to any of the memory blocks in crypto mode is only granted to a reader after a successful mutual authentication.

Block	Contents
0	transponder identifier *id*
1	secret key low $k_0 \dots k_{31}$
2	secret key high $k_{32} \dots k_{47}$ — reserved
3	configuration — password
4 – 7	user defined memory

Figure 8: Hitag2 memory map in crypto mode [36]

3.3 Communication

The communication protocol between the reader and transponder is based on the master-slave principle. The reader sends a command to the transponder, which then responds after a predefined period of time. There are five different commands: *authenticate*, *read*, $\overline{\textit{read}}$, *write* and *halt*. As shown in Figure 9, the *authenticate* command has a fixed length of 5 bits, whereas the others have a length of at least 10 bits. Optionally, these 10 bits can be extended with a redundancy message of size multiple of 5 bits. A redundancy message is composed by the bit-complement of the last five bits of the command. According to the datasheet [36] this feature is introduced to *"achieve a higher confidence level"*.

In crypto mode the transponder starts in a halted state and is activated by the *authenticate* command. After a successful authentication, the transponder enters the active state in which it only accepts active commands which are encrypted. Every encrypted bit that is transferred consists of a plaintext bit XOR-ed with one bit of the keystream. The active commands have a 3-bit argument n which represents the offset (block number) in memory. From this point we address Hitag2 active commands by referring to *commands* and explicitly mention authentication otherwise.

Command	Bits	State
authenticate	`11000`	halted
read	$\texttt{11}n_0n_1n_2\texttt{00}\overline{n_0n_1n_2}\dots$	active
$\overline{\textit{read}}$	$\texttt{01}n_0n_1n_2\texttt{10}\overline{n_0n_1n_2}\dots$	active
write	$\texttt{10}n_0n_1n_2\texttt{01}\overline{n_0n_1n_2}\dots$	active
halt	$\texttt{00}n_0n_1n_2\texttt{11}\overline{n_0n_1n_2}\dots$	active

Figure 9: Hitag2 commands using block number n

Next we define the function *cmd* which constructs a bit string that represents a command c on block n with r redundancy messages.

Definition 3.1. *Let c be the first* 2*-bit command as defined in Figure 9, n be a* 3*-bit memory block number* *and r be the number of redundancy messages. Then, the function $cmd\colon \mathbb{F}_2^2 \times \mathbb{F}_2^3 \times \mathbb{N} \to \mathbb{F}_2^{(10+5r)}$ is defined by*

$$cmd(c,n,0) = cn\overline{cn}$$

$$cmd(c,n,r+1) = \begin{cases} cmd(c,n,r)cn, & r \text{ is odd;} \\ cmd(c,n,r)\overline{cn}, & \text{otherwise.} \end{cases}$$

For example, the command to read block 0 with two redundancy messages results in the following bit string.

$$cmd(11,0,2) = \texttt{11000 00111 11000 00111}$$

The encrypted messages between reader and transponder are transmitted without any parity bits. The transponder response always starts with a prefix of five ones, see Figure 10. In the remainder of this paper we will omit this prefix. A typical forward and backwards communication takes about 12 ms.

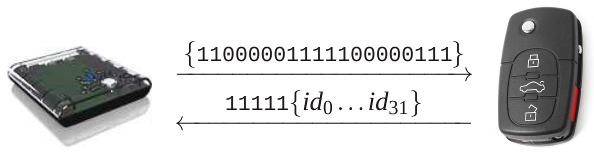


Figure 10: Message flow for reading memory block 0

3.4 Cipher

In crypto mode, the communication between transponder and reader (after a sucessful authentication) is encrypted with the Hitag2 stream cipher. This cipher has been reverse engineered in [47]. The cipher consists of a 48-bit linear feedback shift register (LFSR) and a non-linear filter function f. Each clock tick, twenty bits of the LFSR are put through the filter function, generating one bit of keystream. Then the LFSR shifts one bit to the left, using the generating polynomial to generate a new bit on the right. See Figure 11 for a schematic representation.

Definition 3.2. *The feedback function $L\colon \mathbb{F}_2^{48} \to \mathbb{F}_2$ is defined by $L(x_0 \dots x_{47}) := x_0 \oplus x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{16} \oplus x_{22} \oplus x_{23} \oplus x_{26} \oplus x_{30} \oplus x_{41} \oplus x_{42} \oplus x_{43} \oplus x_{46} \oplus x_{47}$.*

The filter function f consists of three different circuits f_a, f_b and f_c which output one bit each. The circuits f_a and f_b are employed more than once, using a total of twenty input bits from the LFSR. Their resulting bits are used as input for f_c. The circuits are represented by three boolean tables that contain the resulting bit for each input.

Definition 3.3 (Filter function)**.** *The filter function $f\colon \mathbb{F}_2^{48} \to \mathbb{F}_2$ is defined by*

$$\begin{aligned} f(x_0 \dots x_{47}) = f_c(&f_a(x_2x_3x_5x_6), f_b(x_8x_{12}x_{14}x_{15}), \\ &f_b(x_{17}x_{21}x_{23}x_{26}), f_b(x_{28}x_{29}x_{31}x_{33}), \\ &f_a(x_{34}x_{43}x_{44}x_{46})), \end{aligned}$$

where $f_a, f_b\colon \mathbb{F}_2^4 \to \mathbb{F}_2$ and $f_c\colon \mathbb{F}_2^5 \to \mathbb{F}_2$ are

$$f_a(i) = (\texttt{0xA63C})_i$$

$$f_b(i) = (\texttt{0xA770})_i$$

$$f_c(i) = (\texttt{0xD949CBB0})_i.$$

For future reference, note that each of the building blocks of f (and hence f itself) has the property that it outputs zero for half of the possible inputs (respectively one).

Remark 3.4 (Cipher schematic)**.** *Figure 11 is different from the schematic that was introduced by [47] and later used by [14, 19, 44, 45]. The input bits of the filter function in Figure 11 are shifted by one with respect to those of [47]. The filter function in the old schematic represents a keystream bit at the previous state $f(x_{i-1} \dots x_{i+46})$, while the one in Figure 11 represents a keystream bit of the current state $f(x_i \dots x_{i+47})$. Furthermore, we have adapted the boolean tables to be consistent with our notation.*

3.5 Authentication protocol

The authentication protocol used in Hitag2 in crypto mode, reversed engineered and published online in 2007 [47], is depicted in Figure 12. The reader starts the communication by sending an authenticate command, to which the transponder answers by sending its identifier id. From this point on, communication is encrypted, i.e., XOR-ed with the keystream. The reader responds with its encrypted challenge n_R and the answer $a_R = \texttt{0xFFFFFFFF}$ also encrypted to prove knowledge of the key; the transponder finishes with its encrypted answer a_T (corresponding to block 3 in Fig. 8) to the challenge of the reader.

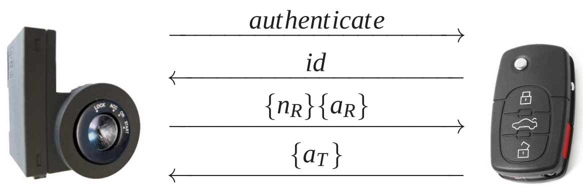


Figure 12: Hitag2 authentication protocol

During the authentication protocol, the internal state of the stream cipher is initialized. The initial state consists of the 32-bits identifier concatenated with the first 16 bits of the key. Then reader nonce n_R XORed with the last 32 bits of the key is shifted in. During initialization, the LFSR feedback is disabled. Since communication is encrypted from n_R onwards, the encryption of the later bits of n_R are influenced by its earlier bits. Authentication is achieved by reaching the same internal state of the cipher after shifting in n_R.

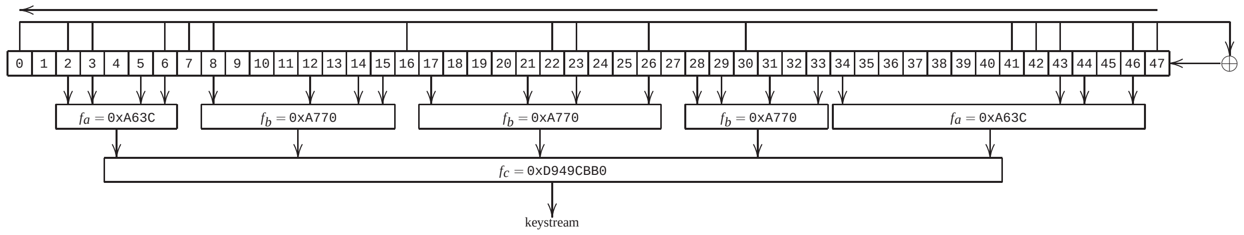


Figure 11: Structure of the Hitag2 stream cipher, based on [47]

3.6 Cipher Initialization

The following precisely defines the initialization of the cipher and the generation of the LFSR-stream $a_0a_1\dots$ and the keystream $b_0b_1\dots$.

Definition 3.5. *Given a key $k = k_0 \dots k_{47} \in \mathbb{F}_2^{48}$, an identifier $id = id_0 \dots id_{31} \in \mathbb{F}_2^{32}$, a reader nonce $n_R = n_{R_0} \dots n_{R_{31}} \in \mathbb{F}_2^{32}$, a reader answer $a_R = a_{R_0} \dots a_{R_{31}} \in \mathbb{F}_2^{32}$, and a transponder answer $a_T = a_{T_0} \dots a_{T_{31}} \in \mathbb{F}_2^{32}$, the internal state of the cipher at time i is $\alpha_i := a_i \dots a_{47+i} \in \mathbb{F}_2^{48}$. Here the $a_i \in \mathbb{F}_2$ are given by*

$$\begin{aligned} a_i &:= id_i & &\forall i \in [0,31]\\ a_{32+i} &:= k_i & &\forall i \in [0,15]\\ a_{48+i} &:= k_{16+i} \oplus n_{R_i} & &\forall i \in [0,31]\\ a_{80+i} &:= L(a_{32+i} \dots a_{79+i}) & &\forall i \in \mathbb{N}\,. \end{aligned}$$

Furthermore, we define the keystream bit $b_i \in \mathbb{F}_2$ at time i by

$$b_i := f(a_i \dots a_{47+i}) \qquad \forall i \in \mathbb{N}\,.$$

Define $\{n_R\}, \{a_R\}_i, \{a_T\}_i \in \mathbb{F}_2$ by

$$\begin{aligned} \{n_R\}_i &:= n_{R_i} \oplus b_i & &\forall i \in [0,31]\\ \{a_R\}_i &:= a_{R_i} \oplus b_{32+i} & &\forall i \in [0,31]\\ \{a_T\}_i &:= a_{T_i} \oplus b_{64+i} & &\forall i \in [0,31]. \end{aligned}$$

Note that the a_i, α_i, b_i, $\{n_R\}_i$, $\{a_R\}_i$, and $\{a_T\}_i$ are formally functions of k, id, and n_R. Instead of making this explicit by writing, e.g., $a_i(k, id, n_R)$, we just write a_i where k, id, and n_R are clear from the context.

3.7 Rollback

To recover the key it is sufficient to learn the internal state of the cipher α_i at any point i in time. Since an attacker knows id and $\{n_R\}$, the LFSR can then be rolled back to time zero.

Definition 3.6. *The rollback function $R\colon \mathbb{F}_2^{48} \to \mathbb{F}_2$ is defined by $R(x_1 \dots x_{48}) := x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{16} \oplus x_{22} \oplus x_{23} \oplus x_{26} \oplus x_{30} \oplus x_{41} \oplus x_{42} \oplus x_{43} \oplus x_{46} \oplus x_{47} \oplus x_{48}$.*

If one first shifts the LFSR left using L to generate a new bit on the right, then R recovers the bit that dropped out on the left, i.e.,

$$R(x_1 \dots x_{47}\, L(x_0 \dots x_{47})) = x_0\,. \tag{1}$$

Theorem 3.7. *In the situation from Definition 3.5, we have*

$$\begin{aligned} a_{32+i} &= R(a_{33+i} \dots a_{80+i}) & &\forall i \in \mathbb{N}\\ a_i &= id_i & &\forall i \in [0,31]\,. \end{aligned}$$

Proof. Straightforward, using Definition 3.5 and Equation (1). □

If an attacker manages to recover the internal state of the LFSR $\alpha_i = a_i a_{i+1} \dots a_{i+47}$ at some time i, then she can repeatedly apply Theorem 3.7 to recover $a_0a_1 \dots a_{79}$ and, consequently, the keystream $b_0b_1b_2\dots$. By having eavesdropped $\{n_R\}$ from the authentication protocol, the adversary can further calculate

$$n_{R_i} = \{n_R\}_i \oplus b_i \qquad \forall i \in [0,31]\,.$$

Finally, the adversary can compute the secret key as follows

$$\begin{aligned} k_i &= a_{32+i} & &\forall i \in [0,15]\\ k_{16+i} &= a_{48+i} \oplus n_{R_i} & &\forall i \in [0,31]\,. \end{aligned}$$

4 Hitag2 weaknesses

This section describes three weaknesses in the design of Hitag2. The first one is a protocol flaw while the last two concern the cipher's design. These weaknesses will later be exploited in Section 5.

4.1 Arbitrary length keystream oracle

This weakness describes that without knowledge of the secret key, but by having only one authentication attempt, it is possible to gather an arbitrary length of keystream bits from the transponder. Section 3.3 describes the reader commands that can modify or halt a Hitag2 transponder. As mentioned in Definition 3.1 it is possible to extend the length of such a command with a multiple of five bits. A 10-bit command can have an optional number of redundancy messages r so that the total bit count of the message is $10 + 5r$ bits. Due to power and memory constraints, Hitag2 seems to be designed to communicate without a send/receive buffer. Therefore, all cipher operations are performed directly at arrival or transmission of bits. Experiments show that a Hitag2 transponder successfully accepts encrypted commands from the reader which are sent with 1000 redundancy messages. The size of such a command consists of $10+5\times 1000 = 5010$ bits.

Since there is no challenge from the transponder it is possible to replay any valid $\{n_R\}\{a_R\}$ pair to the transponder to achieve a successful authentication. After receiving a_T, the internal state of the transponder is initialized and waits for an encrypted command from the reader as defined in Figure 9. Without knowledge of the keystream bits $b_{96}b_{97}\dots$ and onwards, all possible combinations need to be evaluated. A command consist of at least 10 bits, therefore there are 2^{10} possibilities. Each command requires a 3-bit parameter containing the block number. Both *read* and $\overline{\textit{read}}$ receive a 32-bit response, while the write and halt have a different response length. Hence, when searching for 10-bit encrypted commands that get a 32-bit response there are exactly 16 out of the 2^{10} values that match. On average the first *read* command is found after 32 attempts, the complement of this *read* and its parameters are a linear difference and therefore take only 15 attempts more.

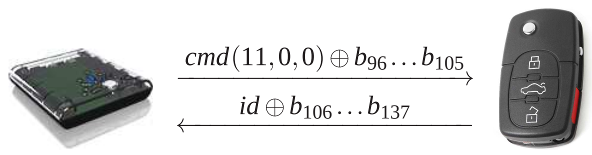

$cmd(11,0,0)\oplus b_{96}\dots b_{105}$
$id\oplus b_{106}\dots b_{137}$

Figure 13: Read *id* without redundancy messages

One of the 16 guesses represents the encrypted bits of the read command on the first memory block. This block contains the *id* which is known plaintext since it is transmitted in the clear during the authentication. Therefore, there is a guess such that the communicated bits are equal to the messages in Figure 13.

With the correct guess, 40 keystream bits can be recovered. This keystream is then used to encrypt a slightly modified read command on block 0 with six redundancy messages, as explained in Section 3.3. The transponder responds with the next 32-bit of keystream which are used to encrypt the identifier as shown in Figure 14. Hence the next 30 keystream bits were retrieved using previously recovered keystream and by extending the *read* command.

This operation can be repeated many times. For example, using the recovered keystream bits $b_{96}\dots b_{167}$ it is possible to construct a 70-bit *read* command with 12 redundancy messages etc. In practice it takes less than 30 seconds to recover 2048 bits of contiguous keystream.

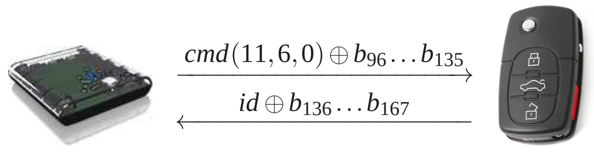

$cmd(11,6,0)\oplus b_{96}\dots b_{135}$
$id\oplus b_{136}\dots b_{167}$

Figure 14: Read *id* using 6 redundancy messages

4.2 Dependencies between sessions

Section 3.6 shows that at cipher state α_{79} the cipher is fully initialized and from there on the cipher only produces keystream. This shows that the 48-bit internal state of the cipher is randomized by a reader nonce n_R of only 32 bits. Consequently, at state α_{79}, only LFSR bits 16 to 47 are affected by the reader nonce. Therefore LFSR bits 0 to 15 remain constant throughout different session which gives a strong dependency between them. These 16 session persistent bits correspond to bits $k_0\dots k_{15}$ of the secret key.

4.3 Low degree determination of the filter function

The filter function $f\colon \mathbb{F}_2^{48}\to\mathbb{F}_2$ consists of three building blocks f_a, f_b and f_c arranged in a two layer structure, see Figure 11. Due to this particular structure, input bits $a_{34}\dots a_{47}$ only affect the rightmost input bit of f_c. Furthermore, simple inspection of f_c shows that in 8 out of 32 configurations of the input bits, the rightmost input bit has no influence on the output of f_c. In those cases the output of f_c is determined by its 4-leftmost input bits. Furthermore, this means that with probability $1/4$ the filter function f is determined by the 34-leftmost bits of the internal state. The following theorem states this precisely.

Theorem 4.1. *Let X be a uniformly distributed variable over $\mathbb{F}_2^{34}$. Then*

$$\mathbb{P}[\forall Y,Y'\in\mathbb{F}_2^{14} : f(XY)=f(XY')] = 1/4.$$

Proof. By inspection. □

Definition 4.2. *The function that checks for this property $P\colon \mathbb{F}_2^{48}\to\mathbb{F}_2$ is defined by*

$$P(x_0\dots x_{47}) = (\texttt{0x84D7})_i$$

where

$$i = f_a(x_2x_3x_5x_6)\,f_b(x_8x_{12}x_{14}x_{15})\,f_b(x_{17}x_{21}x_{23}x_{26})\,f_b(x_{28}x_{29}x_{31}x_{33}).$$

Because $P(x_0\dots x_{47})$ only depends on $x_0\dots x_{33}$ we shall overload notation and see $P(\cdot)$ as a function $\mathbb{F}_2^{34}\to\mathbb{F}_2$, writing $P(x_0\dots x_{47})$ as $P(x_0\dots x_{33}0^{14})$.

5 Attacks

This section describes three attacks against Hitag2. The first attack is straightforward and grants an adversary read and write access to the memory of the transponder. The cryptanalysis described in the second attack recovers the secret key after briefly communicating with the car and the transponder. This attack uses a general technique that can be applied to other LFSR-like stream ciphers. The third attack describes a custom cryptanalysis of the Hitag2 cipher. It only requires a few authentication attempts from the car and allows an adversary to recover the secret key with a computational complexity of 2^{35} operations. The last two attacks allow a trade-off between time/memory/data and time/traces respectively. For the sake of simplicity we describe these attacks with concrete values that are either optimal or what we consider 'sensible' in view of currently available hardware.

5.1 Malleability attack

This attack exploits the arbitrary length keystream oracle weakness described in Section 4.1, and the fact that during the authentication algorithm the transponder does not provide any challenge to the reader. This notorious weaknesses allow an adversary to first acquire keystream and then use it to read or write any block on the card with constant communication and computational complexity. After the recovery of the keystream bits $b_{96} \dots b_{137}$ as shown in Figure 13 an adversary can dump the complete memory of the transponder which includes its password. Recovery of the keystream and creating a memory dump from the transponder takes in total less than one second and requires only to be in proximity distance of the victim. This shows a similar scenario to [22] where Garcia et al. show how to wirelessly pickpocket a MIFARE Classic card from the victim.

The memory blocks where the cryptographic key is stored have an extra optional protection mechanism. There is a one time programable configuration bit which determines whether these blocks are readable or not. If the reader tries to read a protected block, then the transponder does not respond. In that case the adversary can still use the attacks presented in Section 5.2 and Section 5.3. If the transponder is not correctly configured, it enables an adversary to read all necessary data to start the car.

5.2 Time/memory tradeoff attack

This attack is very general and it can be applied to any LFSR-based stream cipher as long as enough contiguous keystream is available. This is in fact the case with Hitag2 due to the weakness described in Section 4.1. It extends the methods of similar time/memory tradeoffs articles published over the last decades [3, 6, 7, 11, 25, 38]. This attack requires communication with the reader and the transponder. The next proposition introduces a small trick that makes it possible to quickly perform n cipher steps at once. Intuitively, this proposition states that the linear difference between a state s and its n-th successor is a combination of the linear differences generated by each bit. This will be later used in the attack.

Proposition 5.1. *Let s be an LFSR state and $n \in \mathbb{N}$. Furthermore, let $d_i = \mathrm{suc}^n(2^i)$ i.e., the LFSR state that results from running the cipher n steps from the state 2^i. Then*

$$\mathrm{suc}^n(s) = \bigoplus_{i=0}^{47} (d_i \cdot s_i) .$$

To perform the attack the adversary A proceeds as follows:

1. Only once, A builds a table containing 2^{37} entries. Each entry in the table is of the form $\langle ks, s \rangle$ where $s \in \mathbb{F}_2^{48}$ is an LFSR state and $ks \in \mathbb{F}_2^{48}$ are 48 bits of keystream produced by the cipher when running from s. Starting from some state where $s \neq 0$, the adversary generates 48 bits of keystream and stores it. Then it uses Theorem 5.1 to quickly jump $n = 2^{11}$ cipher states to the next entry in the table. This reduces the computational complexity of building the table from 2^{48} to $48 \times 2^{37} = 2^{42.5}$ cipher ticks. Moreover, in order to improve lookup time the table is sorted on ks and divided into 2^{24} sub-tables encoded in the directory structure like `/ks_byte1/ks_byte2/ks_byte3.bin` where each `ks_byte3.bin` file has only 8 KB. The total size of this table amounts 1.2 TB.

2. A emulates a transponder and runs an authentication attempt with the target car. Following the authentication protocol, the car answers with a message $\{n_R\}\{a_R\}$.

3. Next, the attacker wirelessly replays this message to the legitimate transponder and uses the weakness described in Section 4.1 to obtain 256 bytes of keystream $ks_0 \dots ks_{2048}$. Note that this might be done while the key is in the victim's bag or pocket.

4. The adversary sets $i = 0$.

5. Then it looks up (in logarithmic time) the keystream $ks_i \dots ks_{i+47}$ in the table from step 1.

6. If the keystream is not in the table then it increments i and goes back to step 5. If there is a match, then the corresponding state is a candidate internal state. A uses the rest of the keystream to confirm is this is the internal state of the cipher.

7. Finally, the adversary uses Theorem 3.7 to rollback the cipher state and recover the secret key.

Complexity and time. In step 1 the adversary needs to pre-compute a 1.2 TB table which requires $2^{42.5}$ cipher ticks, which is equal to 2^{37} encryptions. During generation, each entry is stored directly in the corresponding `.bin` file as mentioned before. Each of these 8 KB files also needs to be sorted but it only takes a few minutes to sort them all. Computing and sorting the whole table takes less than one day on a standard laptop. Steps 2-3 take about 30 seconds to gather the 256 bytes of keystream from the transponder. Steps 4-6 require (in worst case) 2000 table lookups which take less than 30 seconds on a standard laptop. This adds to a total of one minute to execute the attack from begin to end.

5.3 Cryptanalytic attack

A combination of the weaknesses described in Section 4.2 and 4.3 enable an attacker to recover the secret key after gathering a few authentication attempts from a car. In case that identifier white-listing is used as a secondary security measure, which is in fact the case for all the cars we tested, the adversary first needs to obtain a valid transponder *id*, see Section 7.5.

The intuition behind the attack is simple. Suppose that an adversary has a guess for the first 34 bits of the key. One out of four traces is expected to have the property from Theorem 4.1 which enables the adversary to perform a test on the first bit of $\{a_R\}$. The dependencies between sessions described in Section 4.2 allow the attacker to perform this test many times decreasing drastically the amount of candidate (partial) keys. If an attacker gathers 136 traces this allows her (on average) to perform $136/4 = 34$ bit tests, i.e. just as much as key bits were guessed. For the small amount of candidate keys that pass these tests (typically 2 or 3), the adversary performs an exhaustive search for the remaining 14 bits of the key. A precise description of this attack follows.

1. The attacker uses a transponder emulator (like the Proxmark III) to initiate 136 authentication attempts with the car using a fixed transponder *id*. In this way the attacker gathers 136 traces of the form $\{n_R\}\{a_R\}$. Next the attacker starts searching for the secret key. For this we split the key k in three parts $k = \overleftarrow{k}\hat{k}\overrightarrow{k}$ where $\overleftarrow{k} = k_0 \ldots k_{15}$, $\hat{k} = k_{16} \ldots k_{33}$, and $\overrightarrow{k} = k_{34} \ldots k_{47}$.

2. for each $\overleftarrow{k} = k_0 \ldots k_{15} \in \mathbb{F}_2^{16}$ the attacker builds a table $T_{\overleftarrow{k}}$ containing entries
$$\langle y \oplus b_0 \ldots b_{17}, \overline{b_{32}}, \overleftarrow{k}y \rangle$$
for all $y \in \mathbb{F}_2^{18}$ such that $P(\overleftarrow{k}y0^{14}) = 1$. Note that the expected size of this table is $2^{18} \times 1/4 = 2^{16}$ which easily fits in memory.

3. For each $\hat{k} = k_{16} \ldots k_{33} \in \mathbb{F}_2^{18}$ and for each trace $\{n_R\}\{a_R\}$, the attacker sets $z := \hat{k} \oplus \{n_R\}_0 \ldots \{n_R\}_{17}$. If there is an entry in $T_{\overleftarrow{k}}$ for which $y \oplus b_0 \ldots b_{17}$ equals z but $\overline{b_{32}} \neq \{a_R\}_0$ then the attacker learns that $\hat{k}$ is a bad guess, so he tries the next one. Otherwise, if $\overline{b_{32}} = \{a_R\}_0$ then $\hat{k}$ is still a viable guess and therefore the adversary tries the next trace.

4. Each $\overleftarrow{k}\hat{k}$ that passed the test for all traces is a partial candidate key. For each such candidate (typically 2 or 3), the adversary performs an exhaustive search for the remaining key bits $\overrightarrow{k} = k_{34} \ldots k_{47}$. For each full candidate key, the adversary decrypts two traces and checks whether both $\{a_R\}$ decrypt to all ones as specified in the authentication protocol. If a candidate passes this test then it is the secret key. If none of them passes then the adversary goes back to Step 2 and tries the next $\overleftarrow{k}$.

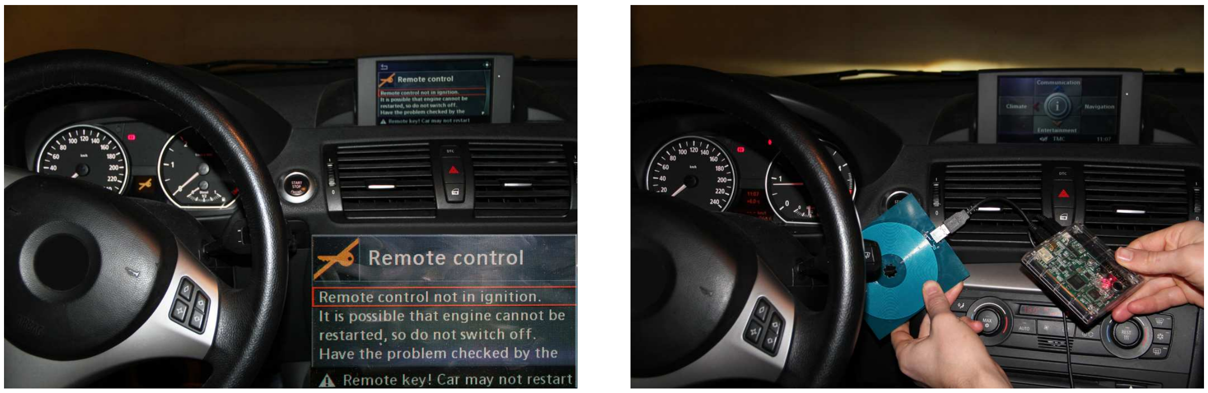
Complexity and time. In step 1, the adversary needs to gather 136 partial authentication traces. This can be done within 1 minute using the Proxmark III. In steps 2 and 3, the adversary needs to build 2^{16} tables. For each of these tables the adversary needs to compute 2^{18} encryptions plus 2^{18} table lookups. Step 4 has negligible complexity thus we ignore it. This adds to a total complexity of $2^{16} \times (2^{18} + 2^{18}) = 2^{35}$ encryptions/lookups. Note that it is straightforward to split up the search space of $\overleftarrow{k}$ in as many processes as you wish. On an standard quad-core laptop this computation takes less than five minutes. Therefore, the whole attack can be performed in less than 360 seconds which explains the title of the paper.

This attack is faster than other practical attacks proposed in [14, 45]. The following table shows a comparison between this attack and other attacks from the literature.

Attack	Description	Practical	Computation	Traces	Time
[45]	brute-force	**yes**	2 102 400 min	2	4 years
[14]	sat-solver	**yes**	2 880 min	4	2 days
[42]	sat-solver	no[1]	386 min	N/A	N/A
[44]	cube	no[2]	1 min	500	N/A
Our	cryptanalytic	**yes**	5 min	136	6 min

[1]Soos et al. require 50 bits of contiguous keystream.
[2]Sun et al. require control over the encrypted reader nonce $\{n_R\}$

Figure 15: Comparison of attack times and requirements



Remote control

Remote control not in ignition.
It is possible that engine cannot be restarted, so do not switch off.
Have the problem checked by the

Remote key! Car may not restart

Figure 16: Left: Authentication failure message
Right: Successful authentication using a Proxmark III

6 Starting a car

In order to elaborate on the practicality of our attacks, this section describes our experience with one concrete vehicle. For this we have chosen a German car, mainly due to the fact that it has keyless ignition. Instead of the typical mechanical key, this car has a hybrid remote control which contains a Hitag2 transponder. In the dashboard of the car there is a slot to insert the remote and a button to start the engine. When a piece of plastic of suitable size is inserted in this slot the car repeatedly attempts to authenticate the transponder (and fails). This car uses an identifier white-list as described in Section 7.5. The same section explains how to wirelessly pickpocket a valid identifier from the victim's remote. As soon as the car receives a valid identifier, the dashboard lights up and the LCD screen pops-up displaying the message shown in Figure 16-Left. Note also the sign on the dashboard. At this point we used the Proxmark to quickly gather enough traces and execute the attack from Section 5.3 to recover the secret key. This car is one of the few that we tested that does not have a predictable password so we wirelessly read it from the victim's remote. Then we use the Proxmark to emulate the transponder. Figure 16-Right shows that the car accepts the Proxmark as if it was the legitimate transponder. The same picture shows (by looking at the tachometer) that at this stage it is possible to start the engine.

7 Implementation weaknesses

To verify the practicality of our attacks, we have tested all three of them on at least 20 different car models from various makes. During our experiments we found that, besides the weaknesses in cipher and protocol, the transponder is often misconfigured and poorly integrated in the cars. Most of the cars we tested use a default or predictable transponder password. Some generate nonces with a very low entropy. Most car keys have vehicle-dependant information stored in the user defined memory of the transponder, but none of the tested cars actually check this data. Some cars use Hitag2 for keyless ignition systems, which are more vulnerable because they lack a physical key. This section summarizes some of the weaknesses we found during our practical experiments. Especially, Section 7.4 shows the implications of the attack described in Section 5.3 when the transponder uses a predictable password. Section 7.5 describes how to circumvent identifier white-listing. This is an additional security mechanism which is often used in vehicle immobilizers.

7.1 Weak random number generators

From the cars we tested, most pseudo-random number generators (PRNG) use the time as a seed. The time intervals do not have enough precision. Multiple authentication attempts within a time frame of one second get the same random number. Even worse, we came across two cars which have a PRNG with dangerously low entropy. The first one, a French car (A), produces nonces with only 8 bits of entropy, by setting 24 of the 32 bits always to zero as shown in Figure 17.

Origin	Message	Description
CAR	`18`	authenticate
TAG	`39 0F 20 10`	*id*
CAR	**`0A 00 00 00`** `23 71 90 14`	$\{\mathbf{n_R}\}\{a_R\}$
TAG	`27 23 F8 AF`	$\{a_T\}$
CAR	`18`	authenticate
TAG	`39 0F 20 10`	*id*
CAR	**`56 00 00 00`** `85 CA 95 BA`	$\{\mathbf{n_R}\}\{a_R\}$
TAG	`38 07 50 C5`	$\{a_T\}$

Figure 17: Random numbers generated by car A

Another French car (B), produced random looking nonces, but in fact, the last nibble of each byte was determined by the last nibble of the first byte. A subset of these nonces are shown shown in Figure 18.

$\{n_R\}$	$\{a_R\}$
20 D1 0B 08	56 36 F3 66
70 61 1B 58	1B 18 F3 38
B0 A1 5B 98	1E 94 62 3A
D0 41 FB B8	01 3B 54 10
25 1A 3C AD	15 88 5E 19
05 7A 9C 8D	F7 4D F7 70
C5 3A 5C 4D	30 B1 4A D4
E5 DA FC 6D	D8 BD 79 C3

Figure 18: Random numbers generated by car B

7.2 Low entropy keys

Some cars have repetitive patterns in their keys which makes them vulnerable to dictionary attacks. Recent models of a Korean car (C) use the key with the lowest entropy we came across. It tries to access the transponder in password mode as well as in crypto mode. For this it uses the default password MIKR and a key of the form 0xFFFF$******$FF as shown in Figure 19.

Origin	Message	Description
CAR	18	authenticate
TAG	E4 13 05 1A	*id*
CAR	**4D 49 4B 52**	password = MIKR
CAR	18	authenticate
TAG	E4 13 05 1A	*id*
CAR	DA 63 3D 24 A7 19 07 12	$\{n_R\}\{a_R\}$
TAG	EC 2A 4B 58	$\{a_T\}$

Figure 19: Car C authenticates using the default password and secret key 0xFFFF814632FF

7.3 Readable keys

Section 5.1 shows how to recover the memory dump of a Hitag2 transponder. Almost all makes protect the secret key against read operations by setting the bits of the configuration in such a way that block one and two are not readable. Although there are some exceptions. For example, experiments show that most cars from a French manufacturer have *not* set this protection bit. This enables an attacker to recover the secret key in an instant. Even more worrying, many of these cars have the optional feature to use a remote key-less entry system which have a much wider range and are therefore more vulnerable to wireless attacks. The combination of a transponder that is wirelessly accessible over a distance of several meters and a non protected readable key is most worrying.

7.4 Predictable transponder passwords

The transponder password is encrypted and sent in the transponder answer a_T of the authentication protocol. This is an additional security mechanism of the Hitag2 protocol apart from the cryptographic algorithm. Besides the fact that the transponder proves knowledge of the secret key, it sends its password encrypted. In general it is good to have some fall back scenario and countermeasure if the used cryptosystem gets broken. Section 5.3 demonstrates how to recover the secret key from a vehicle. But to start the engine, it is necessary to know the transponder password as well. Experiments show that at least half of the cars we tested on use default or predictable passwords.

7.5 Identifier pickpocketing

The first generation of vehicle immobilizers were not able to compute any cryptographic operations. These transponders were simply transmitting a constant (unique) identifier over the RF channel. Legitimate transponder identifiers were white-listed by the vehicle and only those transponders in the white-list would enable the engine to start. Most immobilizer units in cars still use such white-listing mechanism, which is actually encouraged by NXP. These cars would only attempt to authenticate transponders in their white-list. This is an extra obstacle for an attacker, namely recovering a genuine identifier from the victim before being able to execute any attack. There are (at least) two ways for an adversary to wirelessly pickpocket a Hitag2 identifier:

- One option is to use the low-frequency (LF) interface to wirelessly pickpocket the identifier from the victim's key. This can be done within proximity distance and takes only a few milliseconds. According to the Hitag2 datasheet [36], the communication range of a transponder is up to one meter. Although, Hitag2 transponders embedded into car keys are optimized for size and do not achieve such a communication distance. However, an adversary can use tuned equipment with big antennas that ignore radiation regulations (e.g., [17]) in order to reach a larger reading distance. Many examples in the literature show the simplicity and low-cost of such a setup [24, 30, 31, 43].

- Another option is to use the wide range ultra-high frequency (UHF) interface. For this an adversary needs to eavesdrop the transmission of a hybrid

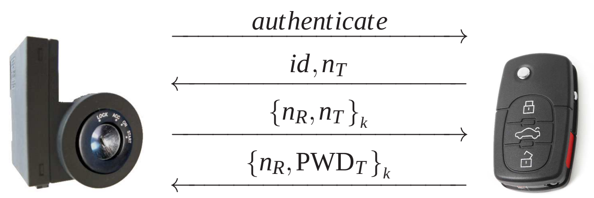
Hitag2 transponder [39] when the victim presses a button on the remote (e.g. to close the doors). Most keyless entry transponders broadcast their identifier in the clear on request (see for example [39]).

With respect to the LF interface, the UHF interface has a much wider transmission range. As shown in [18] it is not hard to eavesdrop such a transmission from a distance of 100 meters. From a security perspective, the first generation Hitag2 transponders have a physical advantage over the hybrid transponders since they only support the LF interface.

8 Mitigation

This section briefly discusses a simple but effective authentication protocol for car immobilizers and it also describes a number of mitigating measures for the attacks proposed in Section 5. For more details we refer the reader to [1, 9].

First of all we emphasize that it is important for the automotive industry to migrate from weak proprietary ciphers to a peer-reviewed one such as AES [15], used in cipher block chaining mode (CBC). A straightforward mutual authentication protocol is sketched in Figure 20. The random nonces n_R, n_T, secret key k and transponder password PWD_T should be at least 128 bits long. Comparable schemes are proposed in the literature [32, 33, 46, 48, 49].



$\xrightarrow{\textit{authenticate}}$
$\xleftarrow{id, n_T}$
$\xrightarrow{\{n_R, n_T\}_k}$
$\xleftarrow{\{n_R, \text{PWD}_T\}_k}$

Figure 20: Immobilizer authentication protocol using AES

There are already in the market immobilizer transponders which implement AES like the ATA5795[2] from Atmel and the Hitag AES / Pro[37] from NXP. It should be noted that, although they use a peer-reviewed encryption algorithm, their authentication protocol is still proprietary and therefore lacks public and academic scrutiny.

In order to reduce the applicability of our cryptographic attack, the automotive industry could consider the following measures. This attack is the most sensitive as it does not require access to the car key. These countermeasures should be interpreted as palliating (but not a solution) before migrating to a more secure and openly designed product.

- **Extend the transponder password**
  The transponder password is an important part of the authentication protocol but grievously it has only an entropy of 24 bits. Such a password is easy to find via exhaustive search. Furthermore, as we mentioned in Section 7.4, manufacturers often deployed their cars with predictable transponder passwords. As shown in Figure 8, there are four pages available of user defined memory in a Hitag2 transponder. These could be used to extend the transponder password with 128 bits of random data to increase its entropy. This implies that an adversary needs to get access to the transponder's memory before being able to steal a car.

- **Delay authentication after failure**
  The cryptographic car-only attack explained in Section 5.3 requires several authentication attempts to reduce the computational complexity. Extending the time an adversary needs to gather these traces increases the risk of being caught. To achieve this, the immobilizer introduces a pause before re-authenticating that grows incrementally or exponentially with the number of sequential incorrect authentications. An interesting technique to implement such a countermeasure is proposed in [40]. The robustness, availability and usability of the product is affected by this delay, but it increases the attack time considerably and therefore reduces the risk of car theft.

Besides these measures, it is important to improve the pseudo-random number generator in the vehicles which is used to generate reader nonces. Needless to say, the same applies to cryptographic keys and transponder passwords. NIST has proposed a statistical test suite which can be used to verify the quality of a pseudo-random number generator [41].

9 Conclusions

We have found many serious vulnerabilities in the Hitag2 and its usage in the automotive industry. In particular, Hitag2 allows replaying reader data to the transponder; provides an unlimited keystream oracle and uses only one low-entropy nonce to randomize a session. These weaknesses allow an adversary to recover the secret key within seconds when wireless access to the car and key is available. When only communication with the car is possible, the adversary needs less than six minutes to recover the secret key. The cars we tested use identifier white-listing. To circumvent this, the adversary first needs to obtain a valid transponder *id* by other means e.g., eavesdrop it when the victim locks the doors. This

UHF transmission can be intercepted from a distance of 100 meters [18]. We have executed all our attacks (from Section 5) in practice within the claimed attack times. We have experimented with more than 20 vehicles of various makes and models and found also several implementation weaknesses.

In line with the principle of responsible disclosure, we have notified the manufacturer NXP six months before disclosure. We have constructively collaborated with NXP, discussing mitigating measures and giving them feedback to help improve the security of their products.

10 Acknowledgments

The authors would like to thank Bart Jacobs for his firm support in the background. We are also thankful to E. Barendsen, L. van den Broek, J. de Bue, Y. van Dalen, E. Gouwens, R. Habraken, I. Haerkens, S. Hoppenbrouwers, K. Koster, S. Meeuwsen, J. Reule, J. Reule, I. Roggema, L. Spix, C. Terheggen, M. Vaal, S. Vernooij, U. Zeitler, B. Zwanenburg, and those who prefer to remain anonymous for (bravely) volunteering their cars for our experiments.

References

[1] Ross J. Anderson. *Security Engineering: A guide to building dependable distributed systems*. Wiley, 2010.

[2] Atmel. Embedded avr microcontroller including rf transmitter and immobilizer lf functionality for remote keyless entry - ATA5795, 2010.

[3] Steve Babbage. A space/time tradeoff in exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection*, volume 408 of *Conference Publications*, pages 161–166. IEEE Computer Society, 1995.

[4] Josep Balasch, Benedikt Gierlichs, Roel Verdult, Lejla Batina, and Ingrid Verbauwhede. Power analysis of Atmel CryptoMemory - recovering keys from secure EEPROMs. In *12th Cryptographers' Track at the RSA Conference (CT-RSA 2012)*, volume 7178 of *Lecture Notes in Computer Science*, pages 19–34. Springer-Verlag, 2012.

[5] Alex Biryukov, Ilya Kizhvatov, and Bin Zhang. Cryptanalysis of the Atmel cipher in SecureMemory, CryptoMemory and CryptoRF. In *9th Applied Cryptography and Network Security (ACNS 2011)*, pages 91–109. Springer-Verlag, 2011.

[6] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory trade-offs with multiple data. In *13th International Workshop on Selected Areas in Cryptography (SAC 2006)*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer-Verlag, 2006.

[7] Alex Biryukov and Adi Shamir. 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)*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2000.

[8] Andrey Bogdanov. Linear slide attacks on the KeeLoq block cipher. In *Information Security and Cryptology (INSCRYPT 2007)*, volume 4990 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2007.

[9] Andrey Bogdanov and Christof Paar. On the security and efficiency of real-world lightweight authentication protocols. In *1st Workshop on Secure Component and System Identification (SECSI 2008)*. ECRYPT, 2008.

[10] Stephen C. Bono, Matthew Green, Adam Stubblefield, Ari Juels, Aviel D. Rubin, and Michael Szydlo. Security analysis of a cryptographically-enabled RFID device. In *14th USENIX Security Symposium (USENIX Security 2005)*, pages 1–16. USENIX Association, 2005.

[11] Johan Borst, Bart Preneel, Joos Vandewalle, and Joos V. On the time-memory tradeoff between exhaustive key search and table precomputation. In *19th Symposium in Information Theory in the Benelux*, pages 111–118, 1998.

[12] Nicolas T. Courtois. The dark side of security by obscurity - and cloning MIFARE Classic rail and building passes, anywhere, anytime. In *4th International Conference on Security and Cryptography (SECRYPT 2009)*, pages 331–338. INSTICC Press, 2009.

[13] Nicolas T. Courtois, Gregory V. Bard, and David Wagner. Algebraic and slide attacks on KeeLoq. In *15th International Workshop on Fast Software Encryption (FSE 2000)*, volume 5086 of *Lecture Notes in Computer Science*, pages 97–115. Springer-Verlag, 2008.

[14] Nicolas T. Courtois, Sean O'Neil, and Jean-Jacques Quisquater. Practical algebraic attacks on the Hitag2 stream cipher. In *12th Information Security Conference (ISC 2009)*, volume 5735 of *Lecture Notes in Computer Science*, pages 167–176. Springer-Verlag, 2009.

[15] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.

[16] Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. A practical attack on the MIFARE Classic. In *8th Smart Card Research and Advanced Applications Conference (CARDIS 2008)*, volume 5189 of *Lecture Notes in Computer Science*, pages 267–282. Springer-Verlag, 2008.

[17] Federal Communications Commission FCC. Guidelines for evaluating the environmental effects of radio frequency radiation. Technical report, Federal Communications Commission FCC, April 2009.

[18] Aurélien Francillon, Boris Danev, and Srdjan Čapkun. Relay attacks on passive keyless entry and start systems in modern cars. In *18th Network and Distributed System Security Symposium (NDSS 2011)*. The Internet Society, 2011.

[19] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *13th European Symposium on Research in Computer Security (ESORICS 2008)*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer-Verlag, 2008.

[20] Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult. Exposing iClass key diversification. In *5th USENIX Workshop on Offensive Technologies (USENIX WOOT 2011)*, pages 128–136, San Francisco, CA, USA, 2011. USENIX Association.

[21] Flavio D. Garcia, Gerhard de Koning Gans, Roel Verdult, and Milosch Meriac. Dismantling iClass and iClass Elite. In *17th European Symposium on Research in Computer Security (ESORICS 2012)*, Lecture Notes in Computer Science. Springer-Verlag, 2012.

[22] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly pickpocketing a mifare classic card. In *30th IEEE Symposium on Security and Privacy (S&P 2009)*, pages 3–15. IEEE Computer Society, 2009.

[23] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Dismantling SecureMemory, CryptoMemory and CryptoRF. In *17th ACM Conference on Computer and Communications Security (CCS 2010)*, pages 250–259. ACM/SIGSAC, 2010.

[24] Gerhard P. Hancke. Practical attacks on proximity identification systems (short paper). In *27th IEEE Symposium on Security and Privacy (S&P 2006)*, pages 328–333. IEEE Computer Society, 2006.

[25] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[26] Motoki Hirano, Mikio Takeuchi, Takahisa Tomoda, and Kin-Ichiro Nakano. Keyless entry system with radio card transponder. *IEEE Transactions on Industrial Electronics*, 35:208–216, 1988.

[27] Sebastiaan Indesteege, Nathan Keller, Orr Dunkelmann, Eli Biham, and Bart Preneel. A practical attack on KeeLoq. In *27th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2008)*, volume 4965 of *Lecture Notes in Computer Science*, pages 1–8. Springer-Verlag, 2008.

[28] Markus Kasper, Timo Kasper, Amir Moradi, and Christof Paar. Breaking KeeLoq in a flash: on extracting keys at lightning speed. In *2nd International Conference on Cryptology in Africa, Progress in Cryptology (AFRICACRYPT 2009)*, volume 5580 of *Lecture Notes in Computer Science*, pages 403–420. Springer-Verlag, 2009.

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

[30] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 2005)*, pages 47–58. IEEE Computer Society, 2005.

[31] Ilan Kirschenbaum and Avishai Wool. How to build a low-cost, extended-range RFID skimmer. In *15th USENIX Security Symposium (USENIX Security 2006)*, pages 43–57. USENIX Association, 2006.

[32] Kerstin Lemke, Ahmad-Reza Sadeghi, and Christian Stble. An open approach for designing secure electronic immobilizers. In *Information Security Practice and Experience (ISPEC 2005)*, volume 3439 of *Lecture Notes in Computer Science*, pages 230–242. Springer-Verlag, 2005.

[33] Kerstin Lemke, Ahmad-Reza Sadeghi, and Christian Stüble. Anti-theft protection: Electronic immobilizers. *Embedded Security in Cars*, pages 51–67, 2006.

[34] Karsten Nohl. Immobilizer security. In *8th International Conference on Embedded Security in Cars (ESCAR 2010)*, 2010.

[35] Karsten Nohl, David Evans, Starbug, and Henryk Plötz. Reverse engineering a cryptographic RFID tag. In *17th USENIX Security Symposium (USENIX Security 2008)*, pages 185–193. USENIX Association, 2008.

[36] Transponder IC, Hitag2. Product Data Sheet, Nov 2010. NXP Semiconductors.

[37] Hitag pro. Product Data Sheet, 2011. NXP Semiconductors.

[38] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer-Verlag, 2003.

[39] Security transponder plus remote keyless entry – Hitag2 plus, PCF7946AT. Product Profile, Jun 1999. Philips Semiconductors.

[40] Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P. Burleson, and Kevin Fu. TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks. In *21st USENIX Security Symposium (USENIX Security 2012)*. USENIX Association, 2012.

[41] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications. *NIST Special Publication*, pages 800–822, 2001.

[42] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In *12th International Conference on Theory and Applications of Satisfiability Testing (SAT 2009)*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer-Verlag, 2009.

[43] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *7th International Workshop on Security Protocols (WSP 2000)*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–182. Springer-Verlag, 2000.

[44] Siwei Sun, Lei Hu, Yonghong Xie, and Xiangyong Zeng. Cube cryptanalysis of Hitag2 stream cipher. In *10th International Conference on Cryptology and Network Security (CANS 2011)*, volume 7092 of *Lecture Notes in Computer Science*, pages 15–25. Springer-Verlag, 2011.

[45] Petr Štembera and Martin Novotný. Breaking Hitag2 with reconfigurable hardware. In *14th Euromicro Conference on Digital System Design (DSD 2011)*, pages 558–563. IEEE Computer Society, 2011.

[46] Pang-Chieh Wang, Ting-Wei Hou, Jung-Hsuan Wu, and Bo-Chiuan Chen. A security module for car appliances. *International Journal of World Academy Of Science, Engineering and Technology*, 26:155–160, 2007.

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

[48] Marko Wolf, Andre Weimerskirch, and Thomas Wollinger. State of the art: Embedding security in vehicles. *EURASIP Journal on Embedded Systems*, 2007:074706, 2007.

[49] Jung-Hsuan Wu, Chien-Chuan Kung, Jhan-Hao Rao, Pang-Chieh Wang, Cheng-Liang Lin, and Ting-Wei Hou. Design of an in-vehicle anti-theft component. In *8th International Conference on Intelligent Systems Design and Applications (ISDA 2008)*, volume 1, pages 566–569. IEEE Computer Society, 2008.