

# One Car, Two Frames: Attacks on Hitag-2 Remote Keyless Entry Systems Revisited

Ryad Benadjila<sup>1</sup>, Mathieu Renard<sup>2</sup>, José Lopes-Esteves<sup>2</sup>, and Chaouki Kasmi<sup>2</sup>

<sup>1</sup>Thales Communications & Security, ryadbenadjila@gmail.com

<sup>2</sup>French Network and Information Security Agency – ANSSI, forename.name@ssi.gouv.fr

## Abstract

Since 2006, many papers were devoted to the analysis of the security of the Hitag-2 algorithm in the context of vehicles access control. While this algorithm was proven to be cryptographically broken, it is still in use in the car industry. Recently, new vulnerabilities regarding Hitag-2 based Remote Keyless Entry systems have been exposed, leading to the design of an attack allowing to unlock a vehicle and requiring the capture of four to eight radio packets.

However, in this study, it will be shown that specific implementations of Hitag-2 based RKE systems exist, which apply interesting countermeasures and thus are not vulnerable to the aforementioned attack. Furthermore, a detailed black box analysis of such system, from the physical layer up to the rolling code implementation will be proposed. Finally, a new cryptographic weakness will be exposed, which can be exploited to forge valid packets without retrieving the cryptographic key and to open the target vehicle, requiring the capture of only two radio packets.

## 1 Introduction

Nowadays, manufacturers tend to interconnect modern cars' embedded systems to various decentralized services such as multimedia, voice commands and speech recognition servers, localization and GPS, etc. They add more and more layers of complexity in a world that, until very recently, was motivated to fulfill very basic features and was mainly focused on safety issues. As many recent papers have shown [19], cyber security is becoming a serious concern, especially when critical Electronic Control Units (ECU) handling the car sensitive parts (breaks, engine, etc.) can be compromised via this interconnection.

Most of published studies assume a physical access to the targeted vehicles. In this context, preserving

the integrity of the interior of a car, i.e. ensuring that an adversary has no physical access to potential attack vectors (such as the multimedia radio system), becomes a critical issue. Hence, all related technologies involved in opening/closing the car doors should enforce adequate security functions.

In 2016, Garcia *et al.* [14] analyzed some of these systems called RKE (for Remote Keyless Entry): they exposed vulnerabilities allowing an adversary to open and close a car after capturing a few RF packets, by forging a rogue key. One of their main contributions is a new attack on the Hitag-2 algorithm.

Following a similar approach, this paper presents a new cryptanalysis of Hitag-2 in a RKE context. This work deals with the discovery of Hitag-2 based RKE systems implementing countermeasures that render the attack proposed in [14] hard to replicate. We will refer to such systems as “hardened RKE” in what follows. Exploiting a new flaw discovered in Hitag-2 allows us to attack such systems with only two captured RF packets. In addition to the new findings, the description of the experimental methodology that allowed us to understand how the hardened RKE systems work, without any ECU code reverse engineering or access to private documentation of the components, is proposed. The study was carried out in a black box context through the analysis of the RF packets sent to the victim car and the use of publicly available blank keys.

This paper is organized as follows: in Section 2, the related work published in the literature is summarized. In Section 3, a focus on the Hitag-2 algorithm is proposed and known cryptographic weaknesses and attacks are discussed. New findings related to a so-called hardened Hitag-2 RKE system are presented in Section 4. In Section 5, new attacks discovered and implemented during our experiments are presented. In supplement, some countermeasures are discussed.

## 2 Related work on RKE systems

The purpose of this section is to introduce the context of Remote Keyless Entry systems, and to provide a brief state of the art of the attacks against them.

### 2.1 Keys, immobilizers and RKE

Modern car keys are commonly used for two features:

- The **Immobilizers** aim at detecting the presence of a valid car key in order to unlock the ignition sequence. Near-field radiocommunications (RFID) are generally used for these systems, with a frequency modulated carrier usually around 125 kHz. The barrel of the start lock contains a close field loop (antenna) so that an authentication protocol between the key and the ECU can be performed when the key is inserted.
- The **Remote Keyless Entry (RKE)** systems control the remote open/close mechanism. Pushing the key buttons generates a one-way radio signal sent to a receiver embedded in the car. Usually, the communication channel uses ultra-high frequencies (UHF, 433 MHz or 868 MHz in Europe), with a range of few meters.

Both of these functions can be integrated inside the same electronic component called a **transponder**. The cryptography used for each of these features may differ depending on the manufacturers or OEMs. Combined attacks can be developed when both systems use the same encryption key, allowing to both enter and start the engine of the car [14].

Nowadays, RKE systems are gradually being replaced by Passive Keyless Entry (PKE) and Passive Keyless Entry and Start (PKES) systems on new generations of vehicles. A proximity detection mechanism controls the feature unlocking on the car side.

This article focuses on the analysis of classical RKE systems based on **Hitag-2**, in a context of attacks allowing to open and close the car without access to the genuine key. Garcia *et al.* have shown in [14] that many recent car models still use this kind of system.

## 2.2 Attacks on RKE

### 2.2.1 Generic attacks

As RKE systems rely on a unidirectional radio frequency transmission, they are prone to generic physical layer attacks such as jamming, eavesdropping, relay and replay attacks.

In [12], the authors demonstrate the feasibility of relay attacks on PKE systems. Although their study has been limited to PKE near-field low-frequency communications, the same principle applies to RKE UHF communications.

A jamming attack on RKE systems is proposed in [21]. This attack is based on the possible user inattention when she closes her vehicle. The idea is to jam the radio frequency transmission when the user presses the vehicle lock button, leaving the vehicle opened.

A trivial replay attack must be considered if an adversary has access to the key for a few seconds and the car is out of reach, since it is possible to record opening/closing commands and replay them later. This weakness is due to the fact that RKE systems are mono-directional: they solely rely on sending a single frame to the car. In order to limit the risk of replay attacks, RKE systems include counters or rolling codes in the radio frames (this will be detailed in the next sections). However, it has been shown in [16] that a combination of jamming and replay techniques allows to circumvent a RKE system implementing these countermeasures.

A main limitation of generic attacks on RKE systems is that they are conducted in a black box fashion: they require the adversary to record and replay genuine frames generated by the original key. This can be useful for a “one-shot” attack, but is somewhat limited if one wants to be able to interact with the car at any time. This is where targeted attacks prove to be more efficient: understanding how radio frames are built and analyzing the weaknesses of a specific RKE system could allow an adversary to forge a rogue key with unlimited access.

### 2.2.2 Cryptographic weaknesses

Modern RKE systems use cryptography in order to prevent valid frames forgery. A symmetric secret key is shared between the remote and the car ECU, and this key is used to produce a rolling code that an adversary cannot forge without knowing the key. This is usually done by providing a remote control unique identifier and a counter as inputs to a cipher producing an authentication code. The ECU is then able to check this code.

The security of the RKE scheme hence relies on the way the authentication code is produced, and on the strength of the underlying cryptography. Though the AES cipher becomes a standard in modern cars, proprietary and weak algorithms have been used for many years by the industry. Aside from the algorithm itself, other weaknesses can lead to serious attacks,

as it has been shown on Volkswagen brand vehicles which all share the same master key [14].

Since 2006, several keyless entry related algorithms have been affected by cryptographic weaknesses. Examples of such ciphers are KeeLoq [7] and Hitag-2 [28, 18, 10, 9, 26]. The security analysis of Hitag-2 remains a topic of interest since it has been observed to be still in use by the industry, especially in the context of vehicles access control. Recently, a new cryptanalysis targeting RKE systems based on this cipher has been presented in [14]. According to the authors, adversaries only need to record from four to eight radio frames to recover the encryption key, forge a rogue remote, and open/close the victim car at will.

### 3 Hitag-2

Hitag-2 is a stream cipher designed by Mikron, a company acquired by Philips Semiconductors. The algorithm emerged in the 1990s and has been kept proprietary until 2007 when Wiener reverse engineered it [28].

Since then, many public documents, leaked datasheets [28, 3] and academic papers [18, 10, 20, 28, 9, 24, 23, 15, 17] have been published on the subject. The purpose of the current section is to give a brief description of the algorithm, how it is used as a security function of some RKE systems, and finally focus on the state of the art of its weaknesses and attacks.

#### 3.1 Notations

The following notations will be used in the article:

- The binary finite field containing 0 and 1 is  $\mathbb{F}_2$ . The addition operation XOR on this field is  $\oplus$ . The multiplication operation (logical AND) is noted  $\&$ .  $\mathbb{F}_2^n$  is the Cartesian product:  

$$\mathbb{F}_2^n = \underbrace{\mathbb{F}_2 \times \cdots \times \mathbb{F}_2}_{n \text{ time}}$$
- A string  $x \in \mathbb{F}_2^n$  composed of  $n$  bits will have its  $i^{\text{th}}$  bit noted  $x_i$ , with  $0 \leq i < n$ . Hence  $x = x_0 \dots x_{n-1}$ . For example, in hexadecimal notation 0x01 (equivalent to 00000001) is an eight-bit string where  $x_0$  to  $x_6$  are zero, and  $x_7 = 1$ . The string with  $n$  zeros is  $0^n$ , and the string with  $n$  ones is  $1^n$ .
- Given the previous notations,  $(0x010000)_i$  denotes the  $i^{\text{th}}$  bit of the 24-bit string 0x010000. Thus  $(0x010000)_7 = 1$  and  $(0x010000)_i = 0$  for all  $0 \leq i \leq 23$  and  $i \neq 7$ .

- The  $\oplus$  operation between two bit strings  $x$  and  $y$  of the same size  $n$  is the string resulting from bit-wise addition:  $x \oplus y$ . Thus  $0x0011 \oplus 0x1111 = 0x1100$ .
- The complement  $\bar{x}$  of a string  $x$  is the bitwise negation. Thus  $\bar{0x01} = 0xFE$ .
- The concatenation of two strings  $x$  and  $y$ , possibly of different sizes, will be noted indifferently  $xy$  or  $x\|y$ . For example  $0x010\|0x1111 = 0x0101111$ .

#### 3.2 Description

A detailed description of Hitag-2 is given in [9, 14], and a summary is also provided in Appendix A.

Hitag-2 belongs to the family of lightweight stream ciphers designed to fit low power consumption when implemented in hardware. Many of its design principles are shared with the Mifare Classic Crypto1 algorithm [11].

Hitag-2 is based on a 48-bit secret key, a 48-bit internal state, a linear function acting as Linear Feedback Shift Register (LFSR) on the internal state and one non-linear filtering function  $f$  which takes 20 bits at input and produces one bit as output at each clock cycle. The algorithm can produce as many bits as necessary for its stream encryption using its keystream. However, only the first 32 bits (noted  $ks$ ) are used as an authentication code in RKE systems, as it will be described in Section 3.3.

#### 3.3 Hitag-2 used in RKE systems

##### 3.3.1 Radio packets dissection

During their study of Hitag-2 based protocols [14], Garcia *et al.* reverse engineered the RF part used by the PCF7946 [1] chip from Philips. This transponder is advertised to use Hitag-2 in RKE contexts, and an important contribution of their paper has been the dissection of radio packets transmitted from the key to the car, as well as how the fields in these packets are connected to the usage of the stream cipher. The packets transmitted by the PCF7946 transponder are composed of 7 fields listed in Table. 1.

Name	Size (bits)	Description
SYNC	16	Synchronisation (0x0001)
UID	32	Unique Identifier
BTN	4	Button identifier
CNTRL	10	Low part of RKE counter
KS	32	Hitag-2 keystream
CHK	8	Checksum

Table 1: Hitag-2 radio packet structure for PCF7946

The unique identifier UID has a length of 32 bits. The BTN button ID is encoded on 4 bits (which allows to index 16 buttons on one key). The counter CNTRL has a length of 10 bits, and the 32-bit keystream KS is the output of Hitag-2. In addition, a 16-bit synchronization sequence SYNC is sent in preamble to synchronize the clock on the receiver side. A checksum CHK is computed as an 8-bit sequence resulting from the XOR of all previous bytes in the packet (except bytes of SYNC). This checksum allows error detection. Finally, since the number of bits in the packet is not a multiple of 8 bits (102 bits), a padding of two bits '10' is added after KS, giving a total of 104 bits per radio packet.

The 10-bit counter CNTRL is in fact a low part of the 28-bit counter used by the Hitag-2 cipher (L is for *low*). The 18-bit high part of the counter CNTRH (H for *high*) is kept secret on both the remote control and the ECU sides. Consequently, the counter CNTR used by the cryptographic function is 28 bits while only 10 bits are sent over-the-air.

UID, BTN and CNTRL are sent with the most significant bit first. For example, if the lower part of the counter is equal to CNTRL = 1, then the 10-bits string is sent in the following order: 0000000001.

### 3.3.2 Hitag-2 cipher inputs and output

In order to produce the authenticator KS, the sender and the receiver will have to feed three inputs to the Hitag-2 cipher: a key  $k$ , an identifier  $id$  and an initialization vector  $iv$ . The relation between these inputs and the fields that compose the radio packets sent over-the-air is described in [14].

The secret key  $k$  is obviously unknown to an external adversary and remains sealed in the transponder and in the ECU. The 32-bit value of UID immediately maps to  $id$ . The initialization vector  $iv$  is composed of elements extracted from the button identifier and the counter. Let  $btn$  be the 4-bit string representing the value BTN. Let  $ctr$  be a 28-bit string forged by naturally concatenating the low and high parts of the counter CNTRH and CNTRL. The value of  $iv$  is then:

$$iv = iv_0 \dots iv_{31} = ctr \parallel btn = ctr_0 \dots ctr_{27} btn_0 \dots btn_3$$

The output of the Hitag-2 algorithm  $ks$  maps to KS with the least significant bit being sent first (in the order of bits production):  $KS = ks_0 \dots ks_{31}$ .

In the following sections, we will name **triplets** the inputs/output tuples of Hitag-2 without the unknown secret key:  $(id, iv, ks)$  is a triplet containing the 32-bit authenticator  $ks$  produced when feeding the cipher with the identifier  $id$ , the initialization vector  $iv$ , and an unknown key  $k$ .

## 3.4 Hitag-2: cryptographic weaknesses and attacks

The security of Hitag-2 is a subject of research since 2007. Many attacks exploiting weaknesses of the algorithm in various use cases have been published. These attacks are due to the 48 bits key length, the low degree of the filtering function  $f$  (small variations of the inputs lead to small variation of the output), and the fact that the internal state is the same during the first 48 cycles of each session of a given transponder.

We provide hereafter a brief overview of some attacks exploiting these weaknesses. The efficiency of these attacks strongly relies on the use case and the context where Hitag-2 is considered.

- **Exhaustive search:** a 48-bit key is below modern standards, making brute-force attacks practical. Two triplets  $(id, iv, ks)$  are required to find the key  $k$ . Since  $ks$  length is 32 bits, an average of  $2^{48-32} = 2^{16}$  keys produce the same triplet through collisions: the second triplet brings disambiguation with very high probability. Such exhaustive searches have been implemented on various platforms (CPU, FPGA and GPU): a summary is given in Table. 2.

Source	Platform	Time
[10] (2009)	CPU 2GHz	4 years
[23] (2011)	FPGA COPACOBANA Cluster (Spartan 3 - XC3S1000)	2 hours
[15] (2012)	GPU Nvidia Tesla C2050	11 hours

Table 2: Hitag-2 brute-force attacks

- **Algebraic attacks:** as for many stream ciphers, the relations between the output  $ks$  and the inputs  $k$ ,  $id$  and  $iv$  can be expressed as a system of multivariate quadratic boolean equations. Courtois *et al.* showed [10] that in the case of Hitag-2, these equations can be shaped to provide an efficient input to SAT solvers. The estimated time of resolution is 6 hours using 16 input/output sets with a chosen  $iv$ , or 45 hours using 4 input/output sets with random  $iv$  (on a standard PC). These algebraic attacks have been improved in [20] using CryptoSAT, a dedicated solver, and in [22]. However, a limitation of these attacks is that to be really efficient, they either require chosen  $iv$  or many bits of generated keystream, which does not suit very well with RKE contexts.
- **Cryptanalytic attacks:** the first practical cryptanalysis of Hitag-2 exploiting its particular weaknesses has been introduced in [25]. The

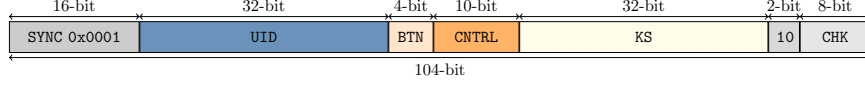


Figure 1: Structure of the data sent by the component PCF7946

authors attack the immobilizer system of several cars using 134 frames acquired by sniffing the near-field communication between the key and the ECU. The attack consists of filtering candidate keys and strongly reduces the  $2^{48}$  exhaustive search space: bad candidates are invalidated using relations between the keystream and the cipher inputs, and time-memory trade-off precomputed tables are used to optimize the process. The attack is practical when the adversary has a close access to the anti-start system: only 5 minutes on a standard laptop are needed to recover the key.

- **Correlation attack:** the cryptanalysis of Hitag-2 previously described is not practical in a RKE context since the adversary would have to get 134 radio packets. Waiting for the legitimate user to open or close her car so many times is hardly practicable. This is why the authors of [14] have introduced a new cryptanalysis where capturing only 4 to 8 frames is needed to extract the secret key from the transponder. For the sake of succinctness, we will not detail all the steps of the attack: we will only give an overview of its key concepts that will explain why this cryptanalysis operates poorly in some RKE contexts that we have discovered (see Section 4.3.4).

Following the same concepts introduced in [25], the main idea of the attack is to limit the exhaustive search by identifying good candidate keys with a high correlation score (this concept is explained hereafter). In the sequel, we suppose that the adversary has recorded several radio frames of the same transponder (possibly corresponding to different buttons), i.e. she has multiple triplets  $(id, iv, ks)$ , the value of  $id$  being the same for all these triplets. The score of  $n$  bit candidates ( $16 \leq n \leq 48$ ) is computed via a correlation relative to the observed output bits  $ks$  of the algorithm.

Let's take the example of the 16-bit length candidates: the adversary guesses the 16 lowest significant bits  $k_0 \dots k_{15}$  of the key  $k$ . These bits are used during the initialization phase of Hitag-2, and are located on the left part of the

internal state at the  $32^{th}$  clock cycle. Namely  $a_{32} \dots a_{32+15} = k_0 \dots k_{15}$  (for more details see Appendix A, and Fig. 6). The bit  $ks_0$  is then produced by using 8 bits among  $k_0 \dots k_{15}$ , selected by the filtering function  $f$  (bits 2, 3, 5, 6, 8, 12, 14 and 15 of the internal state). The other 12 bits used when computing  $ks_0$  are other bits of the internal state related to non-guessed key bits. Hence, it is possible to compute a correlation score by averaging over the  $2^{12}$  non-guessed bits: the result reflects the likelihood of realizing the observed bit  $ks_0$  given the guessed bits of the key. All the captured frames participate to making this score more accurate, since many bits  $ks_0$  are observed. The score on a single bit is very limited: the attack is extended to bits  $ks_1 \dots ks_{15}$  to refine it. The 16 bits of the guessed candidate act on these bits of keystream during clock cycles  $\leq 32 + 16$  (they are thrown out of the internal state after). Finally, the score of a 16-bit length candidate key is denoted  $\tilde{k}^{16} = \tilde{k}_0 \dots \tilde{k}_{15}$  and is computed as an average over all the bits of keystream that they produce, as well as over all the triplets available to the adversary. The same score computation is extended to the candidate keys of which  $n$  bits are guessed  $\tilde{k}^n = \tilde{k}_0 \dots \tilde{k}_{n-1}$ , moving  $n$  from 16 to 48. At each step, only the best candidates are kept using a ranking on a fixed size table.

The best known attack on Hitag-2 in a RKE context is the correlation based cryptanalysis. The authors of [14] advertise a result in 10 minutes on average on a standard laptop, with 4 to 8 captured frames and a table with a fixed size of 400,000 candidates in memory. They also discuss a major issue that must be overcome when dealing with the PCF7946 produced radio frames: the 18-bit high part of the counter CNTRH (used to produce a part of the  $iv$  inputs of Hitag-2) is not known to the adversary, since it is not sent over-the-air. The authors propose to get around this issue by observing that it takes 1024 key pushes on the transponder key to produce a carry to CNTRH. If CNTRH is set to 0 at manufacturing time, the value of CNTRH is presumably low and can be inferred using the age of the vehicle. Consequently, the adversary can repeat the cryptanalysis as many times as necessary by forging a new  $iv$  and ultimately

find the secret key after a reasonable time. They have validated their assumption on various vehicles. As we will see in the next sections, this assumption might become inadequate for some **Hitag-2** based RKE systems such as the one we have been experimenting with.

## 4 Hardened Hitag-2 RKE unveiled

This section will provide details on some interesting points that were discovered during our experiments with a **Hitag-2** RKE system at our disposal. As we will see, some discrepancies have been observed on our system. As a consequence it has shown to be immune to the correlation attack presented in [14].

First, we briefly present the radio framework that we have used to capture and dissect the packets. Then, we detail the discrepancies implemented in the hardened RKE system, and why the correlation attack fails.

### 4.1 From RF signal to bits

In the framework of wireless communication security analysis, it is necessary to set-up an environment to receive and analyse RF signals. Thus, the first task is to define adequate hardware and software resources fitting with the characteristics of the signal under test. One could list the following parameters to start with: central frequency, channel bandwidth and any details about the complexity of the physical layer. To get access to these parameters, it is possible to investigate the open literature about the technology, standards, FCC certification documents or any related document dealing with the integrated circuits used in the device. If no information is available about the targeted protocol, an empirical reverse engineering can be applied, starting with the detection of the central frequency.

The RKE system has been intensively studied, open source signal processing tools are available for receiving, demodulating and decoding packets (e.g. [6]). During the review of the existing literature, the characteristics of the physical and logical layers have been found. A summary of these parameters is given in Table. 3.

Parameter	Value
Working frequency	ISM 433 MHz
Modulation	ASK/FSK
Channel encoding	Manchester
Packet format	see Table. 1

Table 3: RKE physical and logical layers

In what follows, a white box methodology (in the

sense that most of the physical and logical layers are known) is proposed to check the parameters accuracy regarding the device under test (DUT).

#### 4.1.1 Demodulation

The working frequency of the DUT is 433.92 MHz. A preliminary step is the analysis of the spectrum (analysis in the frequency domain). As the RKE system is a narrow bandwidth technology a simple RTL-SDR [27] is used with the related GNU-Radio script. After setting the center frequency to 433.92 MHz and a bandwidth of 2 MHz, the central frequency of the system has been confirmed.

The next step is to confirm the modulation type. It is known that amplitude and frequency modulations can be used by common RKE systems. The choice is related the configuration of the RF IC by the manufacturer of the key. A *waterfall* representation (power spectral density in time-domain and frequency domain) has confirmed that an amplitude modulation is used by the DUT.

#### 4.1.2 Decoding

Once the demodulation has been applied (the signal obtained after demodulation is represented on Fig. 2 - time domain waveform), it is necessary to recover the symbol duration and the type of encoding.

It can be trivially observed that the demodulated frame contains a preamble for synchronization purposes and the same packet repeated twice. In each packet, a preamble is also present from which we can directly obtain the symbol duration (Fig. 3).

In order to assess the channel encoding scheme used by the DUT, the knowledge of the packet structure allows us to eliminate efficiently possible schemes. In particular, the synchronization preamble and the knowledge of the total size of the packets lead us to consider a Manchester encoding scheme. As it can be observed, a zero crossing transition appears at least once in a symbol duration, a pure NRZ scheme can be taken out of consideration. Indeed, due to the expected entropy of the data contained in the packets (counter, output of an encryption algorithm, checksums), there is a low probability of having so few symbol repetitions.

Finally, the presence of checksums allows for verifying the decoding accuracy.

### 4.2 Correlation attack failure

Once the radio frames are retrieved and decoded following the methodology presented in 4.1, the goal

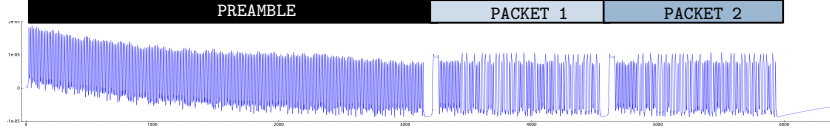


Figure 2: Demodulated radio frame

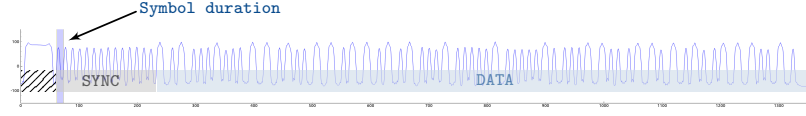


Figure 3: Demodulated packet. The symbol duration is highlighted.

is to extract the 48-bit key  $k$  from the target PCF7946 transponder.

We have implemented the correlation attack of [14] described in 3.4, but it failed at extracting the key even with hundreds of captured radio packets as input.

Software implementations are prone to errors, and in order to set aside such a cause of cryptanalysis failure we have validated our implementation against crafted frames following the specifications given in [14]. In our experimental test cases, the attack allowed to recover the correct key with less than 12 distinct packets with a probability of  $\frac{2}{3}$ , and this probability drastically increases with more packets. These expected results confirm the efficiency of the cryptanalysis in this setup, and points towards some divergence between the RKE system at our disposal and the specifications we have implemented.

### 4.3 Black box reverse engineering

In order to understand why our RKE system setup does not fit the expected specifications, a reverse engineering of the way the Hitag-2 output KS is produced given the elements sent over-the-air in the radio packets was necessary.

One way to achieve this would be to analyze the firmware embedded in the ECU or the one embedded in the PCF7946 transponder. However, extracting the ECU firmware implies tampering with the vehicle, which was not an option. Moreover, most of the PCF7946 logic is hardwired, and this transponder is hardened against physical tampering: extracting the key implies hardware reverse engineering, which is time and resource consuming.

This leaves us with a black box analysis of the system. In order to overcome the strong limitations of a blind investigation, we used publicly available blank remote keys allowing us to set up an arbitrary

cryptographic key in the transponder. This approach allowed us to indeed discover a hardened RKE system detailed in this section.

#### 4.3.1 Using blank keys

Many car manufacturers allow legitimate users to program new remote keys for backup purposes (in the case a key is lost for example). In the case of the PCF7946, the transponder can be switched in a programming mode where parts of the EEPROM can be written with user defined values: this is the case for the cryptographic key and some configuration flags. For obvious security reasons, this programming mode should be permanently disabled using commands that blow electronic fuses in the transponder after the sensitive elements have been written in EEPROM.

The protocol set up to communicate with the transponder in this mode uses the 125 kHz near-field frequency (see [17] for a complete description), and this is generally performed transparently between the ECU and the blank key after the ECU is put in a specific mode with a documented actions sequence.

It is possible to interact with blank keys using a dedicated NFC reader. As Fig. 4 reveals, we can read the UID and the secret key  $k$  embedded in the transponder: the key has the default manufacturing value `0x4f4e4d494b52`<sup>1</sup> documented in the datasheets [2] and in the reference code [28].

Unfortunately, these parts of the blank key's EEPROM do not reveal other critical elements involved in the Hitag-2 RKE protocol, namely the low and high fields of the counter CNTRL and CNTRH.

#### 4.3.2 Variations in the Hitag-2 inputs

In order to understand how the initialization vector *iv* input of Hitag-2 is constructed using the fields

<sup>1</sup>This key is in fact the transcription in ASCII of MIKRON, the name of the company that designed Hitag-2.



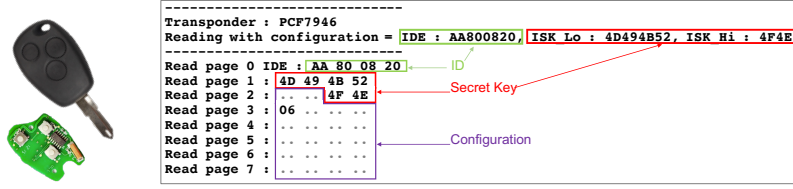


Figure 4: A blank key and its EEPROM data

of the RF packets, we have exploited the following property:

#### Property

The *iv* is only 32 bits long: it is possible to perform an exhaustive search over the  $2^{32}$  possible values in a reasonable time (seconds on a standard PC). This allows to find values realizing an observed keystream *ks*, given known and fixed identifier *id* and key *k*.

Due to the nature of the randomization phase of **Hitag-2**, different *iv* could produce the same keystream bits with fixed *id* and *k*. While these collisions add some “noise” when trying to infer how the *iv* depends on (BTN, CNTRL, CNTRH), repeating the exhaustive search on different packets with various values for the buttons and the low counter make a structured pattern emerge.

The structure of *iv* that we have discovered on our experimental RKE system shows two major differences when compared to the specifications given in [14], as presented on Fig. 5:

- The values of BTN, CNTRL and CNTRH are not used in the same order to form *iv*.
- A random mask MSK seems to be inserted in the least significant bits of *iv*, somehow mixed with or taking the role of CNTRH. Inferring that this is not related to a particularly high value of CNTRH (but rather done on purpose) has been confirmed with further investigations while interacting with the test vehicle. This will be detailed in 4.3.3. One should however notice that MSK is still a high part of the counter associated to the RKE, and is changed when CNTRL overflows through carry propagation (every 1024 packets since CNTRL is 10 bits long).

#### Notation

In the sequel of the article, since CNTRH and MSK can be conceptually assimilated, we will only use MSK as a notation for this 18-bit field of *iv*.

### 4.3.3 Randomization of the *iv*

During our tests on the target vehicle, it has been discovered that the random mask MSK that takes place in the least significant bits of *iv* seems to be in fact a countermeasure implemented in the ECU against replay attacks.

When the ECU detects a chronological inconsistency between the received RF packet and the stored internal counter, it programs a modification of the value of CNTRL and MSK in the transponder at the next start-up of the engine. This resynchronization of the ECU with the transponder is performed through commands using the near-field 125 KHz protocol. The rationale behind this is an out-of-band authentication through the immobilizer channel, confirming that the genuine key is indeed used, before updating the RKE parameters.

A chronological inconsistency is typically detected when a RF packet is legitimate in the sense that the authenticator *ks* is valid, but the field CNTRL is below the low part of the counter stored in the ECU.

At this point of the article, one can wonder how it was possible for us to detect this countermeasure. We have in fact used the improved attacks presented in 5. This resynchronization countermeasure has also practical impacts on the correlation attack as we will describe in 4.3.4.

### 4.3.4 Impacts on the correlation attack

The two discrepancies that were discovered explain why the correlation attack does not give proper results:

1. The difference in the bit order extracted from BTN and CNTRL implies that in the case of our RKE system, the low part (least significant bits) of *iv* has almost no variation in consecutive packets. The reason is that this part corresponds to MSK which changes every 1024 packets. Since the least significant bits of *iv* are the ones that participate the most to the production of the first bits of *ks*, they usually act as a catalyst when averaging the score on distinct RF packets: they emphasize a good candidate and make it



get higher scores in the first steps of the correlation attack. When there is not enough variation in these bits with the different RF packets, the good candidate can get evicted by the ranking algorithm with high probability in the early generations (i.e. when dealing with candidates beginning with size 16).

On the contrary, when the least significant bits of  $iv$  correspond to **CNTRL** as this is the case in the RKE systems presented in [14], consecutive packets yield in much better performance for the correlation attack.

2. The random 18-bit value **MSK** is unknown to the adversary (not sent over-the-air), and the solution proposed by the authors of [14] to guess it from the age of the car cannot be used. We will see in 5.2 that one can nonetheless get rid of this unknown part using the equivalent keys property of **Hitag-2**.

## 5 Attacking hardened **Hitag-2** RKE

For the reasons described in 4.3.4, we consider the RKE system under study as a hardened one. A new attack methodology must be designed to fulfill the goal of implementing a rogue remote key that generates packets containing legitimate authenticators to open and close a target vehicle.

The purpose of the current section is to describe new attacks on **Hitag-2** in the hardened RKE context. First, we describe the implementation of an optimized brute-force attack on **Hitag-2** that allows an adversary to recover the key using two triplets in a reasonable time. Then, we show how combining brute-force with an interesting property of **Hitag-2** allowed us to overcome the issues described in 4.3.4 with new attacks. Finally, we discuss the possible countermeasures against them.

### 5.1 Optimized brute-force attack

We have developed an optimized brute-force attack on **Hitag-2** in **OpenCL**. This exhaustive search is based on the particularly adapted bitslice way of operating the algorithm. Our implementation is inspired from the one presented in [15], with adaptations though. While the original code only focuses on one GPU (11 hours on a Nvidia Tesla C2050, see Table. 2), our version removes minor specific optimizations to gain in scalability: it uses as many CPU and GPU resources as available.

This results in a 18 hours search over the  $2^{48}$  possible keys on a single Nvidia GeForce GTX780 Ti,

improved to 45 minutes on an Amazon EC2 instance with 16 Tesla K80 and 64 CPU<sup>2</sup>, and 15 minutes on three of these instances in parallel. The cost of these 15 minutes is approximately 45 euros.

Consequently, exhaustive search now becomes a practical option for an adversary when it comes to break **Hitag-2** in a context where time matters (which can be the case for RKE where the target vehicle might not stay stationary for hours). All the adversary needs is a distant connexion to EC2 or any other High Performance Distributed Computing system (big bandwidth between the client and the computing cluster is not necessary since only a small amount of data is exchanged).

Finally, an interesting convenience of the brute-force attack on **Hitag-2** compared to other approaches is that it only requires two outputs and known inputs of the algorithm to extract the key.

### 5.2 Equivalent keys of **Hitag-2**

We have presented an optimized exhaustive search allowing to recover a **Hitag-2** key given two input triplets  $(id, iv, ks)$ .

Nevertheless, one major issue that arises when trying to apply this attack to the context of hardened RKE is that  $iv$  is not fully known to the adversary: only 14 bits are extracted from the RF packets, formed by the 4-bit button value **BTN** and the 10-bit low counter value **CNTRL**. The 18-bit **MSK** value is unknown and random, and performing  $2^{18}$  brute-force attacks over all the possible values of this mask is obviously not achievable: hundreds of thousands years of computational power would be needed.

In order to solve this problem and to be able to generate valid radio frames, we have exploited the following interesting property of the **Hitag-2** algorithm:

#### Property

Let  $M = M_0 \dots M_{31} \in \mathbb{F}_2^{32}$  be a 32-bit mask.  
Applying the **Hitag-2** algorithm with inputs:

$\Rightarrow$  identifier  $id$ , key  $k$ , initialization vector  $iv$   
produces the same keystream  $ks$  as with  
inputs :

$\Rightarrow$  identifier  $id$ , key  $k \oplus (0^{16} \| M)$ , initialization  
vector  $iv \oplus M$ .

It is trivial to see why this property holds using the equations of the cipher during the randomization step (see Appendix A).

<sup>2</sup>The instance is p2.16xlarge, see <https://aws.amazon.com/en/ec2/instance-types/>



Figure 5: Bits order difference in Hitag-2 *iv* – (a) Article [14] (b) Hardened RKE

We have  $\forall i \in [0, 31]$  (32 clock cycles):

$$\begin{aligned} a_{48+i} &= k_{16+i} \oplus iv_i \oplus f(a_i \dots a_{47+i}) \\ &= (k_{16+i} \oplus M_i) \oplus (iv_i \oplus M_i) \oplus f(a_i \dots a_{47+i}) \end{aligned}$$

Hence, the 32 most significant bits of the key  $k_{16} \dots k_{47}$  are **xored** with the initialization vector  $iv_0 \dots iv_{31}$ . Masking these two values with the same mask  $M$  will be offset during this operation.

Now, let's consider a set of radio packets that have been collected from hardened RKE using a PCF7946 transponder. This set is made of triplets  $(id, \hat{iv}, ks)$  where  $\hat{iv}$  are the values of the initialization vectors where the 18 least significant bits are zeroed (i.e.  $\hat{iv}$  are only composed of the 14 known bits extracted from the radio packet). We will name these elements equivalent triplets. Let  $iv$  be the real values of the initialization vectors including the unknown part MSK, meaning that  $\hat{iv} = iv \oplus (\text{MSK} \parallel 0^{14})$ .

The property of Hitag-2 previously exposed allows us to derive the following interesting result:

#### Result

A cryptanalysis performed on equivalent triplets  $(id, \hat{iv}, ks)$  for a transponder with a key  $k$  allows to find an equivalent key such that:

$$\hat{k} = k \oplus (0^{16} \parallel \text{MSK} \parallel 0^{14})$$

The keystream authenticators  $ks$  generated by the equivalent key  $\hat{k}$  are the same as the ones generated by  $k$  as long as MSK is not modified.

This means that given two radio packets, i.e. two equivalent triplets, an adversary can perform the brute-force attack presented in 5.1 and get an equivalent key  $\hat{k}$  in a few minutes. Thanks to the previous property, it is possible to forge valid radio packets using  $\hat{k}$  as long as the unknown value MSK is not modified. One should notice that it is important that the two packets used for the exhaustive search also share the same mask MSK, meaning that they must belong to the same 1024 low counter session (see the discussion below).

### 5.3 Wrapping up the new attacks

As we have already uncovered, MSK is modified when:

- The low counter CNTRL overflows and a carry is propagated to MSK (every 1024 button pushes).
- The ECU detects a replay attack and modifies MSK with a random value as described in 4.3.3.

When MSK is modified, capturing two new RF packets and performing a brute-force search again to find the new equivalent key can be a rather expensive option. However, using results from 5.2, we will present new attacks to efficiently deal with these situations. Let us assume that the adversary has previously captured two RF packets providing a first equivalent key  $\hat{k}$  after an exhaustive search. The adversary can then choose between two paths:

1. **Zero capture and guess:** the extraction of the equivalent key  $\hat{k}$  through brute-force provides a primitive to forge valid packets in a 1024 button push timeline when there is no ECU resynchronization. This means that if the low counter CNTRL has the value  $x$ , the adversary can generate  $(1024 - x)$  packets that the ECU will consider authentic. The first  $x$  packets can also be generated, but they are useless to the adversary since they belong to the “past”. Beyond these  $(1024 - x)$  packets, there is a carry and the adversary must adapt the equivalent key to generate legitimate packets for the new frame. Let  $\hat{k}'$  be the new equivalent key. Abusing the addition notation, we have  $\hat{k}' \approx k \oplus (0^{16} \parallel (\text{MSK} + 1) \parallel 0^{14})$ . Since an incrementation of a 18 bits value can produce at most 18 chained carries, it is possible to precompute the 18 possible equivalent keys  $\hat{k}'$  derived from  $\hat{k}$ . Then, it is easy to check which one of all these possible values indeed opens/closes the car, and keeps it as the new equivalent key.

It is also important to bring a precision at this point: when the car receives malformed frames, it does not activate its anti-replay countermeasure. Indeed, since the other 17 packets are derived from bad guesses for the  $iv$ , they will produce non-authentic packets and will simply be dropped by the ECU.

2. **Recapture and adapt:** in this scenario, we suppose that the adversary has an equivalent

key  $\widehat{k}$  associated to a mask  $\text{MSK}$ , but for some reason the 18-bit mask has changed to  $\text{MSK}'$  yielding in a new unknown equivalent key  $\widehat{k}'$ . In order to compute  $\widehat{k}'$ , a new radio packet is captured, providing a new equivalent triplet associated to this equivalent key that will be noted  $(id, \widehat{iv}', ks')$ . The adversary performs a fast exhaustive search<sup>3</sup> over the  $2^{18}$  elements to find a 18-bit value  $M$  such that the observed keystream  $ks'$  is realised by applying **Hitag-2** using  $\widehat{k}$  as a key,  $id$  and  $\widehat{iv}' \oplus (M \parallel 0^{14})$  as inputs. Then, the new equivalent key corresponds to  $\widehat{k}' = \widehat{k} \oplus (0^{16} \parallel M \parallel 0^{14})$ . It is easy to see that the value  $M$  found here is in fact the difference between the two masks:  $M = \text{MSK} \oplus \text{MSK}'$ .

Both of the scenarios previously described suppose a brute-force search using two packets as a preamble, which implies the access to a computing cluster if getting the key within minutes is needed (see 5.1). However, one should notice that any efficient cryptanalysis of **Hitag-2** can replace this step without modifying the rest of the attacks: the parts that use the equivalent keys concept still hold.

The “zero capture and guess” strategy better fits situations where the adversary ensures that no ECU resynchronization occurs (i.e. when the user does not open the car and start the engine after a rogue packet has been sent). This is the case when a one-shot open/close of the target is needed for example. This scenario uses only two captured packets (hence the title of the article).

The “recapture and adapt” strategy complies with situations where the adversary knows that the mask  $\text{MSK}$  has changed (either because of an incrementation overflow of  $\text{CNTRL}$ , or because of the ECU countermeasure): a new packet must be captured to adapt the equivalent key.

Interestingly, these attacks do not require the full recovery of the real secret key  $k$  embedded in the transponder, though information is learnt about the 18 unknown bits of  $k$  every 1024 packets (when  $\text{CNTRL}$  overflows). Nonetheless, it is possible to generate authentic packets using the equivalent keys, resulting in the implementation of a software rogue remote control.

This rogue key has been tested against the two attack scenarios previously described and validated with our target vehicle, using a YARD Stick One [13] and the RfCat framework [5] for the radio part.

<sup>3</sup>A few seconds on a standard PC.

## 5.4 Discussing countermeasures

Obviously, several misconceptions compose the root cause of the attack presented in this paper, starting with the use of the **Hitag-2** algorithm to produce authentication codes. This cipher has been known to be cryptographically broken for a decade: using obsolete or proprietary cryptography is widely discouraged in favour of public standards.

Independently of the cryptographic algorithm, the management of the secret elements, randomness sources and initialization vectors is also an important point. In particular, the key diversification strategy among devices has a direct importance regarding the impact of the compromise of one particular device, as recalled in [14].

Regarding the out-of-band update of secret parameters, the idea is quite interesting and would be even more efficient, in the context of the attack presented in this study, if this was performed every time the car is started. As a consequence, the adversary would face a very short time window to use the forged authentic frames against the RKE system. However, a successful attack against the near-field protocol could result in a ban of the legitimate key.

On the physical layer, several countermeasures could be implemented in order to increase the adversary profile. First, distance bounding protocols [8] seem to be a promising approach to mitigate relay attacks. Additionally, as suggested in [4], RF front-end fingerprinting can also be used for the identification of the legitimate transponder. Their deployment would however involve significant costs.

Finally, monitoring and logging systems could be both integrated in vehicles and made accessible to the end users. Indeed, with the integration of several information systems into modern cars, it seems relevant to provide users with means to supervise and monitor the activity of key functionalities, among which the last time the car was unlocked or the number of invalid unlock attempts would be of special interest in this case.

## 6 Conclusion

Modern transportation vehicles enclose an increasing amount of electronic devices in order to manage both safety and entertainment functionalities. These devices expose interfaces which can be accessed from inside the vehicle or wirelessly. Therefore, the mechanisms allowing to open vehicles become critical from an information security point of view.

In this study, an analysis of a **Hitag-2** based Remote Keyless Entry system has been proposed. A

black box methodology for performing this analysis has been detailed, with a focus on the physical layer characterisation and on the rolling code implementation. As an outcome, it has been shown that the tested system was not vulnerable to the most recent state of the art attacks. Furthermore, an interesting countermeasure was shown to be implemented, resulting in a near-field modification of some parameters of the RKE protocol when several inconsistent radio packets are received by the vehicle.

Interestingly, this analysis leads to the design of a new cryptanalysis based on an optimized exhaustive search and taking advantage of a specificity of the Hitag-2 algorithm. It has been demonstrated that it is possible, for an adversary having captured a limited number of packets (at least two), to forge valid opening radio frames without finding the secret key. Instead, it is possible to compute equivalent keys which allow to obtain the same Hitag-2 keystream than with the legitimate secret key. With this approach, the countermeasure could be trivially circumvented by capturing a single supplementary radio packet.

These attacks are mainly possible due to the persistent use of obsolete and proprietary ciphers. It is highly recommended to stick with standardized encryption algorithms with a proven robustness.

## References

- [1] Philips Semiconductors. PCF7946AT datasheet, 1999. Web: <http://www.datasheetq.com/datasheet-download/715109/0/Philips/PCF7946>.
- [2] NXP. PCF7936AS Securing Transponders (HITAG2) datasheet, 2010. Web: [http://www.mouser.com/catalog/specsheets/PCF7936AS\\_\\_3851\\_\\_C,1.pdf](http://www.mouser.com/catalog/specsheets/PCF7936AS__3851__C,1.pdf).
- [3] Philips Semiconductors. Hitag2 protocol datasheet, 1996, 2010.
- [4] ALINCOURT, E., RAY, C., RICORDEL, P.-M., DARÉ-EMZIVAT, D., AND BOUDRAA, A. Méthodologie d'extraction de signatures issues des signaux ais. SSTIC, 2016.
- [5] ATLAS. RfCat, 2014. Web: <https://bitbucket.org/atlas0fd00m/rfcat>.
- [6] BLOESSL, B. gr-keyfob. SDR Academy, 2015. Web: <https://github.com/bastibl/gr-keyfob>.
- [7] BOGDANOV, A. Cryptanalysis of the keeloq block cipher. Cryptology ePrint Archive, Report 2007/055, 2007. <http://eprint.iacr.org/2007/055>.
- [8] BRANDS, S., AND CHAUM, D. Distance-bounding protocols (extended abstract). In *EUROCRYPT'93, Lecture Notes in Computer Science 765* (1993), Springer-Verlag, pp. 344–359.
- [9] COURTOIS, N. T., AND O'NEIL, S. FSE Rump Session—Hitag 2 Cipher, 2011.
- [10] COURTOIS, N. T., O'NEIL, S., AND QUISQUATER, J.-J. Practical algebraic attacks on the hitag2 stream cipher. In *International Conference on Information Security* (2009), Springer, pp. 167–176.
- [11] DE KONING GANS, G., HOEPMAN, J., AND GARCIA, F. D. A Practical Attack on the MIFARE Classic. *CoRR abs/0803.2285* (2008). Web: <http://arxiv.org/abs/0803.2285>.
- [12] FRANCILLON, A., DANEV, B., AND CAPKUN, S. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. Cryptology ePrint Archive, Report 2010/332, 2010. <http://eprint.iacr.org/2010/332>.
- [13] GADGETS, G. S. YARD Stick One, 2015. Web: <http://greatscottgadgets.com/yardstickone/>.
- [14] GARCIA, F. D., OSWALD, D., KASPER, T., AND PAVLIDÈS, P. Lock it and still lose it —on the (in)security of automotive remote keyless entry systems. In *USENIX Security* (Austin, TX, 2016), USENIX Association.
- [15] IMMLER, V. Breaking hitag 2 revisited. In *Security, Privacy, and Applied Cryptography Engineering*. Springer, 2012, pp. 126–143.
- [16] KAMKAR, S. Drive It Like You Hacked It. Defcon 23, 2015. Web: <https://samy.pl/defcon2015/2015-defcon.pdf>.
- [17] KASPER, T. *Security Analysis of Pervasive Wireless Devices—Physical and Protocol Attacks in Practice*. PhD thesis, PhD thesis. 2011. Web: <https://wiki.cryptorub.de/Counter/get.php>, 2011.
- [18] LEMKE, K., SADEGHI, A.-R., AND STÜBLE, C. *Anti-theft Protection: Electronic Immobilizers*. Springer, Berlin, Heidelberg, 2006, pp. 51–67.
- [19] MILLER, C., AND VALASEK, C. Remote Exploitation of an Unaltered Passenger Vehicle. Web: <http://illmatics.com/Remote%20Car%20Hacking.pdf>.
- [20] PLÖTZ, H., AND NOHL, K. Breaking Hitag2, HAR2009, 2009, 2011.
- [21] SMITH, C. The Car Hacker's Handbook, 2014. Web: [http://opengarages.org/handbook/2014\\_car\\_hackers\\_handbook\\_compressed.pdf](http://opengarages.org/handbook/2014_car_hackers_handbook_compressed.pdf).
- [22] SOOS, M., NOHL, K., AND CASTELLUCCIA, C. *Extending SAT Solvers to Cryptographic Problems*. Springer, Berlin, Heidelberg, 2009, pp. 244–257.
- [23] ŠTEMBERA, P., AND NOVOTNY, M. Breaking Hitag2 with Reconfigurable Hardware. In *Digital System Design (DSD)* (2011), IEEE, pp. 558–563.
- [24] SUN, S., HU, L., XIE, Y., AND ZENG, X. Cube cryptanalysis of hitag2 stream cipher. In *International Conference on Cryptology and Network Security* (2011), Springer, pp. 15–25.
- [25] VERDULT, R., GARCIA, F. D., AND BALASCH, J. Gone in 360 seconds: Hijacking with hitag2. In *USENIX Security* (2012), pp. 237–252.
- [26] VERDULT, R., GARCIA, F. D., AND EGE, B. Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *USENIX Security* (2015), pp. 703–718.
- [27] WICKERT, M. A., AND LOVEJOY, M. R. Hands-on software defined radio experiments with the low-cost rtl-sdr dongle. In *Signal Processing and Signal Processing Education Workshop (SP/SPE)* (2015), IEEE, pp. 65–70.
- [28] WIENER, I. C. Software optimized 48-bit Philips/NXP Mifare Hitag2 PCF7936/46/47/52 stream cipher algorithm. Web: <http://cryptolib.com/ciphers/hitag2>.

## A Description of Hitag-2

**Internal state (48 bits)** at cycle  $i$  will be noted:

$$\alpha_i = a_i \dots a_{47+i} \in \mathbb{F}_2^{48}.$$

**Functions:** a linear function and a non-linear function.

- LFSR – linear function  $L : \mathbb{F}_2^{48} \rightarrow \mathbb{F}_2$  such that:

$$L(x) = x_0 \oplus x_2 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{16} \oplus x_{22} \oplus x_{23} \oplus x_{26} \oplus x_{30} \oplus x_{41} \oplus x_{42} \oplus x_{43} \oplus x_{46} \oplus x_{47}$$

- Non-linear function  $f : \mathbb{F}_2^{48} \rightarrow \mathbb{F}_2$  such that:

$$f(x) = f_c(f_a(x_2x_3x_5x_6), f_b(x_8x_{12}x_{14}x_{15}), f_b(x_{17}x_{21}x_{23}x_{26}), f_b(x_{28}x_{29}x_{31}x_{33}), f_a(x_{34}x_{43}x_{44}x_{46}))$$

Non linear functions  $f_a$ ,  $f_b$  et  $f_c$  are defined by:

- $f_a, f_b : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2$  with  $f_a(i) = (0xA63C)_i$  and  $f_b(i) = (0xA770)_i$
- $f_c : \mathbb{F}_2^5 \rightarrow \mathbb{F}_2$  with  $f_c(i) = (0xD949CBB0)_i$

**Inputs:** the three inputs of the algorithm are:

- a 32-bit initialization vector  $iv \in \mathbb{F}_2^{32}$
- a 48-bit key  $k \in \mathbb{F}_2^{48}$
- a 32-bit identifier  $id \in \mathbb{F}_2^{32}$

**Algorithm steps:** the algorithm is split in two main steps that change the internal state at each clock cycle.

1. The **initialization** and **randomization** phases: they make the internal state evolve during 80 clock cycles by using the three inputs. Only the non-linear function  $f$  is used.

$$\begin{aligned} a_i &= id_i, \forall i \in [0, 31] \text{ (32 cycles)} \\ a_{32+i} &= k_i, \forall i \in [0, 15] \text{ (16 cycles)} \\ a_{48+i} &= k_{16+i} \oplus iv_i \oplus f(a_i \dots a_{47+i}), \forall i \in [0, 31] \text{ (32 cycles)} \end{aligned}$$

2. The **nominal** phase: apply the LFSR on the internal state.

$$a_{80+i} = L(a_{32+i} \dots a_{79+i}), \forall i \in \mathbb{N}$$

**Outputs:** the output is a keystream producing one bit per clock cycle starting from the nominal phase. The output bits  $ks_i$  are computed from the internal state using the non-linear function:

$$ks_i = f(a_{32+i} \dots a_{79+i}), \forall i \in \mathbb{N}$$

The RKE system only use the 32 first bits of the keystream as an authentication code:  $ks = ks_0 \dots ks_{31}$ .

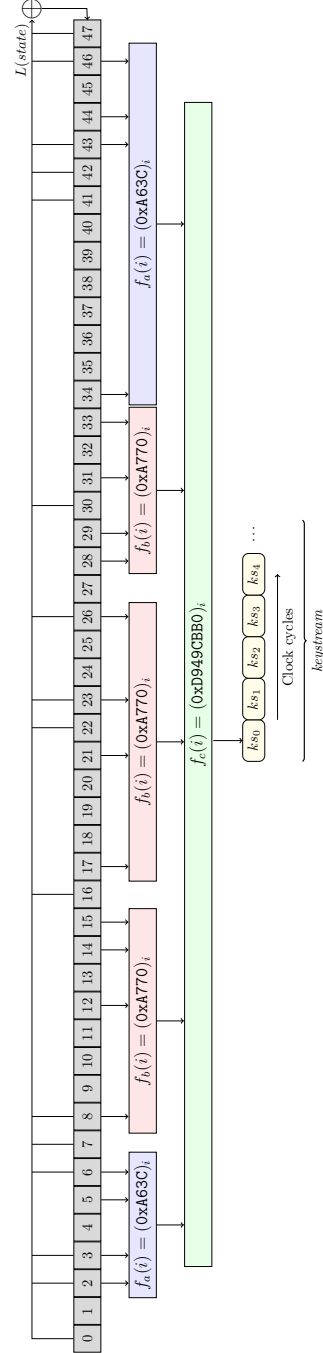


Figure 6: Hitag-2 – synthetic view of the algorithm