Practical Contactless Attacks on Hitag2-Based Immobilizer and RKE systems

Hai-long LIU, Jing-shan MA, Sheng-yu ZHU, Zhao-jun LU and Zheng-lin LIU*

School of Optics and Electronic Information, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

*Corresponding author

Keywords: Hitag2, Immobilizer, Remote keyless entry (RKE), Vehicle security.

Abstract. Immobilizer and remote keyless entry (RKE) systems are widely used in the auto industry to improve the security and comfort. However, most of them have vulnerabilities which allow to recover the cryptographic key. In this paper, we present attacks to break the Hitag2-based immobilizer and RKE systems. These attacks require only several minutes of computation on a laptop after obtaining four to eight valid authentication traces, i.e., authentication messages between the immobilizer and the electronic key or rolling codes of the key. The attack on the immobilizer system can fully recover the cryptographic key which allows an adversary to authenticate to the vehicle and start the engine. The attack on the RKE system can determine the state of the last rolling code and predict the next rolling code. This allows an adversary to open the door quickly without leaving any physical traces. A combination of these two attacks provides the ability to open the door and start the engine of a Hitag2-based vehicle in minutes at a very low cost.

Introduction

Modern vehicles are equipped with a large number of different electronic control units (ECUs). An electronic immobilizer is one of them for anti-theft purpose. It provides a wireless authentication procedure before anyone tries to start the engine, which can prevent hot-wiring. It is mandatory in many countries to equip all sold vehicles with an immobilizer. Remote keyless entry (RKE) is another one which provides comfortable remote lock and unlock operations. These operations are generally based on an instruction protected by a unidirectional rolling code scheme. Both the immobilizer system and the RKE system use some cryptographic mechanisms, such as DST40 [1], Megamos [2], Keeloq [3], Hitag2 [4], Hitag3, and AES, for authentication or encryption. Unfortunately, these mechanisms are not invulnerable. There are many known attacks for some of these cipher schemes, such as [1-8].

The Hitag2 cipher scheme is known as one of the most widely used cryptographic mechanisms in modern vehicles. Many researches on its security have been published in recent years. Among these works the fast correlation attack on Hitag2 [7] is the most practical attack. This attack allows recovery of the cryptographic key with a minimum of four rolling codes and a few minutes computation. However, this attack requires to know the identifier of a car key (i.e., the UID in [7]), which is not contained in the transmitted message frame of the RKE transponder for some car models, such as BMW X1 (2014). An adversary can wirelessly pickpocket the identifier from the car key through the low-frequency (LF) interface using big antennas [4], [10]. However, this method needs to place the antennas close to the car key, and thus, makes the attack less practical and easier to be detected. Moreover, this attack will fail when the least significant bits of the initialization vector ($iv$) does not change in consecutive frames. The reason is described in detail in Section III.

In this paper, we analyze the Hitag2 cipher scheme and reveal some vulnerabilities which may cause serious security problems. Most attacks aim at recovering the cryptographic key. We provide another method to attack the Hitag2-based rolling code scheme through predicting the next rolling code. The main contributions of this paper are: 1) applying a practical attack on Hitag2-based immobilizer system to fully recover the cryptographic key which allows an adversary to counterfeit a car key to start the engine; 2) proposing a contactless attack on a Hitag2-based RKE system to lock or
unlock the door without the knowledge of the identifier. A combination of these two attacks provides the ability to open and start the engine of a Hitag2-based vehicle at a very low cost. Furthermore, we present some software countermeasures that can be deployed in order to minimize the risk of these attacks.

The remainder of this paper is structured as follows: Section II describes the Hitag2 stream cipher scheme and the Hitag2-based protocol used in immobilizer and RKE systems. Section III proposes the practical contactless attacks on immobilizer and RKE systems. The experiments and practical results are analyzed and the countermeasures to minimize the risk of these attack are present in section IV. Section V concludes this paper.

Hitag2 Stream Cipher

Hitag2 cipher scheme is described briefly in this section in order to keep the paper complete. The details can be found in [4], [7].

Some notation are introduced as follows. We use $s[x:y]$ to denote the substring of $s$ with index from $x$ to $y$ and $s[x]$ to denote the bit in string $s$ with index of $x$. $s[0]$ is the most significant bit of $s$. The symbol $\oplus$ denotes exclusive-or (XOR) and $\|$ denotes concatenation. We use $\overline{s}$ to denote bitwise complement of bit string $s$.

The Hitag2 cipher is a stream cipher with a 48-bit key. As shown in Figure 1, it consists of a 48-bit linear feedback shift register (LFSR) and a non-linear filter function. The feedback function $l$ and the filter function $f$ map the 48-bit value of the LFSR ($r[0:47]$) to single-bit values, $l(r[0:47])$ and $f(r[0:47])$ respectively:

$$l(r[0:47]) = r[0] \oplus r[2] \oplus r[3] \oplus r[6] \oplus r[7] \oplus r[8]$$
$$\oplus r[16] \oplus r[22] \oplus r[23] \oplus r[26] \oplus r[30]$$
$$\oplus r[41] \oplus r[42] \oplus r[43] \oplus r[46] \oplus r[47]$$

$$f(r[0:47]) = f_a(r[2], r[3], r[5], r[6]),$$
$$f_b(r[8], r[12], r[14], r[15]),$$
$$f_c(r[17], r[21], r[23], r[26]),$$
$$f_d(r[28], r[29], r[31], r[33]),$$
$$f_e(r[34], r[43], r[44], r[46])$$

where $f_a(i) = (0xA63C)[i]$, $f_b(i) = (0xA770)[i]$, and $f_e(i) = (0xD949CBB0)[i]$.

The encryption process of the Hitag2 cipher consists of two phases, i.e., a state initialization phase and a key stream generation phase. In the state initialization phase, the input values, i.e., an identifier of the car $id[0:31]$, an initialization vector $iv[0:31]$ and an encryption key $key[0:31]$, are used to initialize the internal state of the cipher. First, the LFSR is initialized as

$$r_0[0:47] = id[0:31] \| key[0:16]$$

Then, the LFSR is shifted left. The $i$-th ($i=1, 2...32$) shift is defined as
$r_i[0:47] = r_{i-1}[1:47] \parallel b_{i-1}$

(4)

Where the feedback bit $b_i$ is

$b_i = key[16+g1854+i] \oplus iv[i] \oplus f(r_i[0:47])$

(5)

By repeating the shift operation 32 times, the internal state is completely initialized as $r_{32}[0:47]$. In the key stream generation phase, the concatenation of outputs of the filter function constructs the $n$-bit key stream $ks$, i.e.,

$ks[i] = f(r_{32+i}[0:47]) \quad \forall i \in [0,n-1]$

(6)

After generating one bit of the keystream, the LFSR is shifted once to prepare for the next bit generation:

$r_{i+1}[0:47] = r_i[1:47] \parallel i(r_i[0:47])$

(7)

Note that the generated key stream is fully determined by the initial state $r_{32}[0:47]$.

**Im mobilizer System**

The mutual authentication protocol used in Hitag2-based immobilizer systems is shown in Figure 2.

![Figure 2. Authentication protocol of Hitag2-based immobilizer systems](image)

The authentication starts with a command sent by the reader. Then the transponder answers the command with its $id$. After verifying the received $id$, the reader generates a nonce, i.e., $iv$, and transmits it in plaintext. In order to prevent replay attacks, the reader generates a new $iv$ randomly for each authentication process. From this point onward, the reader and the transponder share the parameters, $id$ and $iv$, which are used in Hitag2 encryption process. Since the other parameter, $key$, is pre-shared, both the reader and the transponder can execute the encryption and generate the same 64-bit key stream.

The key stream is divided into two 32-bit parts, $a_R$ and $a_T$, for authenticating the reader and the transponder respectively. The $a_R$ is flipped bitwise and sent along with the $iv$ to prove knowledge of the pre-shared key. Once the transponder finishes the encryption, it can verify the received $a_R$ and responses with the generated $a_T$.

In the authentication protocol, the only secret shared between the reader and the valid transponder is a 48-bit encryption key. It is possible for an adversary to gather the information involved in the authentication and recover the key by cryptographic analyze.

**RKE System**

Unidirectional rolling code schemes are used in RKE systems for remote control authentication. The Hitag2 based rolling code scheme is briefly described as follows.

In the rolling code scheme, both the vehicle and the remote control maintain a counter $CNTR$ for synchronization and use the output key stream of the encryption for authentication. When a button on the remote control is pressed, it increments the counter, executes the encryption, and transmits a frame containing necessary information for authentication. After verification the vehicle open or close the door according to the command and synchronize its counter to the received value.
Different car models may have different structure of the transmitted frame. Figure 3 shows two typical structures of frame found in Trumpchi GS4 (2017) and BMW X1 (2014) respectively. The frame structure of Trumpchi GS4 is consistent with the description in [7], while the structure of BMW X1 is not found in the literature. The RKE frames mainly consist of 6 parts, i.e., a synchronization (SYNC), an identifier of the car key (UID or ID), a button identifier (BTN), a low part of the counter (CNTRL), a Hitag2 keystream (KS), and a checksum (CHK).

| 16-bit | 32-bit | 4-bit | 10-bit | 32-bit | 2-bit | 8-bit |
|------|------|------|------|------|------|------|

(a) Trumpchi GS4 (2017)

| 12-bit | 4-bit | 16-bit | 32-bit | 8-bit | 8-bit |
|------|------|-------|-------|------|------|

(b) BMW X1 (2014)

Figure 3. Structure of the frame sent by Hitag2-based RKE systems. There are two type of frames: (a) frame found in Trumpchi GS4 (2017), and some other car models described in [7], and (b) frame found in BMW X1 (2014).

For the RKE system in Trumpchi GS4, the $iv$ used in Hitag2 cipher consists with a 28-bit counter and a 4-bit button identifier, i.e., $iv[0:31]=\text{CNT}[0:27]\|\text{BTN}[0:3]$, and the 32-bit $id$ is directly sent out in the frame, i.e, $id[0:31]=\text{UID}[0:31]$.

For BMW X1, the $iv$ consists with a 32-bit counter, i.e., $iv[0:31]=\text{CNT}[0:31]$, and the button identifier is not involved in the encryption. The biggest difference between BMW X1 and Trumpchi GS4 is the identifier of the car key in the frame. For BMW, another associated identifier (ID), which is 16-bit and not used in the encryption, is transmit instead of the UID. When receiving an ID, the vehicle uses the corresponding UID to execute the encryption.

The vehicle can get enough information from the received frame and generate the 32-bit keystream to authenticate the remote control. This information can also be received by anyone else if he wants. Since the signal of RKE systems can be eavesdropped from a distance up to 100m, it is easier to gather the authentication information in RKE systems than in immobilizer systems. However, keeping the $id$ secret can make it much harder.

Analysis on Hitag2 Cipher Scheme

The Hitag2 cipher scheme uses a 48-bit key, which is not strong enough for modern computational capacity. Moreover, it has vulnerabilities which can be used to significantly reduce the computation complexity to recover the key or the internal state. This section presents analysis and practical attacks on Hitag2 systems in detail.

In Hitag2 scheme, the initial state is not obfuscated before generating the key stream. This vulnerability leaves a clue to trace the initial state from the key stream. An adversary can start with guessing part of the initial state and then repeatedly extend the guessing part by 1 bit until obtain the whole state. Since, in general, the correct guess has relatively greater probability to output the correct key stream, the probability can be used to filter out half of the guesses before extension to keep the guessing space from explosion.

However, it is hard to recover the key or the initial state only using this method. On the one hand, it is difficult to guarantee that the correct guess will pass all the filter processes. On the other hand, a 48-bit unknown key or state cannot be obtained from the knowledge of a 32-bit key stream. To solve these problems, several traces are required to provide correlation information.

The vulnerability in the state initialization phase of the Hitag2 scheme makes it possible to take advantage of the correlation between different traces. The internal state is initialized linearly so that the initial state can be partly determined by a partly-known key. Different traces have different initial states, however, they share the same key. Assuming that the other two parameters, $id$ and $iv$, are known, for each trace, part of the initial state can be calculated by guessing part of the key. Thus, a
combination of the probabilities to output the correct key streams can make the filter process perform better.

**Random IV**

A random IV is used in the Hitag2 scheme employed in immobilizer systems. According to the description in the previous section, an adversary can obtain the input parameters id, iv and the generated key stream which are sent in the authentication. These data are enough to conduct an attack to recover the key. The procedure is described as follows:

1. Guess part of the key. Set an $n$-bit window to cover part of the key, i.e., $key[0:n−1]$. Enumerate each state in the window as a guess. Thus, the size of the guessing space is $2^n$. The initial value of the window size, $n$, is adjustable to balance the computation time and success probability.

2. Initial part of the internal state for each guess of each trace. The adversary can calculate $r_{32}[0:n−1]$ for each guess since only the guessed part of the key, $key[0:n−1]$, is involved.

3. Compute the score of each guess on each trace and then compute the combination score of each guess.

The probability of a guess to correctly output the key stream bit $i$ of a trace $t$ is defined as the number of cases ($c_{g,t,i}$) in which the filter function can output the correct bit divided by the total number of possible cases ($2^{20−m}$):

$$p_{g,t,i} = \frac{c_{g,t,i}}{2^{20−m}}$$

where $m$ is the number of known bits input to the filter function.

After computing $p_{g,t,i}$, the guessing part of the state is shifted left to calculate $p_{g,t,i+1}$ until all guessing bits are shifted out of the state. Thus, the score of a guess $g$ on a trace $t$ is defined as:

$$s_{g,t} = \prod_{i=0}^{\min(n−1,31)} p_{g,t,i}$$

where $n$ is the size of the guessing window.

The combination (correlation) score of a guess $g$ on a set of traces is defined as:

$$s_g = \left( \prod_{t=0}^{T} s_{g,t} \right)^{1/T}$$

where $T$ is the number of the used traces.

4. Filter out about half of the guesses which have smaller combination score. The correct guess generally has larger combination score and will be remained after filter process with high probability. Make sure the number of the remaining guesses is close to but not greater than half of the total guesses.

5. Extend the remaining guesses by 1 bit and the window size increase by 1. A previous guess is divided into two guesses, i.e., the previous guess concatenated with bit 0 and 1 respectively. After extension, the number of the guesses approximately recover to the previous size. Then, go back to step 2 until all bits of the key are guessed.

6. The attack succeed if and only if there is only one guess with score greater than 0. This guess is the correct key.

If the attack failed, adjust the initial window size or the number of traces, and repeat the analyzing procedure above.

**Non-random IV**

A non-random IV is used in the rolling code scheme of RKE systems. As described in Section II, in RKE systems, $iv[0:31]=CNTR[0:27]∥BTN[0:3]$ or $iv[0:31]=CNTR[0:31]$. In a pretty long period of
normal operations, only the lower part of iv will change due to the increment of CNTR and the change of btn. Moreover, only part of CNTR is sent by the remote control.

The attack presented in the previous subsection can recover the key of the Hitag2 scheme with random iv, however, it will fail under the condition of non-random iv. First, in the state initialization phase, the lower 32 bits of the key, key[16:47], is XORed with iv to generate the feedback bits and this is the only process that key[16:47] involved in. We cannot fully obtain the key with the knowledge of only part of the iv. Second, since the lower few bits of iv is changed in consequent operations, only the corresponding part of the initial state will change. This lead to a result that the first bits of the key stream are less random than the latter bits. The good guess does not have any advantage and will be filtered out in the first several rounds since the correlation between different traces is too weak. Furthermore, for some car models, the id is not send by the remote control. There are methods present in [4], [10] to pickpocket the id from the car key. However, these methods make the attack less practical and easier to be detected.

Unfortunately, a non-random iv provides another option to directly guess the initial state. For a number of collected traces, their initial state can be divided into two parts, a constant part and a variable part. The constant part is guessed using a method similar with the case of random iv. The difference is that the guess starts from the middle of the state instead to keep the good guess from being filtered out. After finishing the guess process, there will be many candidates left. Then, the variable part can be used to find the correct constant part of the state which matches all the traces. The detail procedure is described as follows:

1. Determine the sizes of the constant part of the initial state, denoted as nc. Make sure the variable part of the state covers all the variable bits of iv in all used traces.

2. Guess part of the state. Set an n-bit window to cover part of the state, i.e., r32[nc − n:nc − 1]. Shift the window to the very left so that r32+n−n−n−1[0:n − 1] = r32[nc − n:nc − 1]. This is the point to output the key stream bit ks[nc − n].

3. Compute the score of each guess on each trace and then compute the combination score of each guess.

The score of a guess g on a trace t is defined as:

\[ s_{g,t} = \prod_{i=n_c-n}^{\min(n_c,31)} P_{g,t,i} \]  \hfill (11)

The combination score of a guess g on a set of traces is defined as:

\[ s_g = \left( \prod_{t=0}^{T} s_{g,t} \right)^{\frac{1}{T}} \]  \hfill (12)

where T is the number of the used traces.

4. Filter out about half of the guesses which have smaller combination score. Make sure the number of the remaining guesses is close to but not greater than half of the total guesses.

5. Extend the remaining guesses by 1 bit and the window size increase by 1. Note that the remaining guesses and the window are extended to the left. A previous guess is divided into two guesses, i.e., bit 0 and 1 concatenated with the previous guess respectively. Then, go back to step 2 until all bits of the constant part are guessed.

6. Find out the correct constant part of state. For each trace, enumerate all the possible cases of the variable part to find a case which can produce the correct key stream. The correct guess can produce key streams of all the traces. However, it is enough to find out the correct guess by checking three traces in practical. After finding out the correct constant part of state, the whole state of each key stream can be determined.

7. Guess the key stream of a target iv. First, select a known iv as a reference. We use the iv of the latest trace in practice. Then, compare the target iv with the selected iv to find the different bits. These different bits will determine the possible different bits of the state. Thus, the possible candidates of

510
the initial state can be obtained. The adversary can try each candidates, generate the target key stream and send command to open or close the door as a valid remote control.

**Practical Contactless Attacks and Countermeasures**

**Practical Results**

We have successfully employed the attacks on two vehicle brands, BMW X1 (2014) and Trumpchi GS4 (2017), for academic purpose. The rolling code frame of the tested two vehicles are different. Trumpchi GS4 sends out the identifier which is used for Hitag2 encryption, while BMW X1 transmits another 16-bit identifier instead.

The first step of a potential attack is to gather some traces. Car keys with a Hitag2 chip provide two wireless interfaces. One operates at a low frequency of 125 KHz for the immobilizer system, the other one operates at an ultra-high frequency (UHF) of 315 MHz or 433 MHz for the RKE system. Both interfaces can be eavesdropped to collect enough traces using low cost equipment such as Proxmark III and HackRF One.

A more challenging task is to recover the key or the initial state after trace collection. The algorithms described in the previous section are implemented on a standard laptop. After inputting 4 to 8 traces, it takes less than 30 seconds to recover the initial state of the latest trace and output the next possible rolling codes. These rolling codes are sent one by one to unlock the target vehicle. This attack on RKE system are automatic and could be accomplished within 1 minute. Taking 4 to 8 traces as input, it takes about 2 minutes to recover the key of the immobilizer system and start the engine. A combination of these two attacks provides the ability to open the door and start the engine of a Hitag2-based vehicle within a few minutes at a very low cost.

**Advantage**

This work has two main advantages. The first advantage is that it can break a Hitag2-based RKE system without any knowledge of the identifier which is used in the encryption. Although the identifier can be wirelessly pick pocketed from the car key, an adversary should be physically close to the target car key, and can only pickpocket the identifier one by one. The attack proposed in this work does not have these limitations. It can target all the cars which are equipped with a Hitag2-based RKE system and in the coverage of the UHF signal. One possible scenario is the park of a residential community in which most residents open and close the door of their cars one time a day. An adversary can place an eaves dropping equipment in his car or hide it somewhere else in the park for several days. After record and analysis of the RKE signal, he can open the door of almost all the Hitag2-based cars in the park.

The second advantage is that the two attack methods based on similar algorithms can break Hitag2-based immobilizer and RKE systems in minutes. The attacks in previous works show the ability to break an immobilizer system or a RKE system separately. This work analyzes the weaknesses of the Hitag2 cipher scheme and the risks of using it in the authentication scheme or the rolling code scheme. Since the weakness can be used to break Hitag2-based immobilizer and RKE systems at the same time, it has high risk to continue using these Hitag2-based systems. It is highly recommended to take some countermeasures to eliminate the risk.

**Countermeasures**

A long-term countermeasure is to use more secure cryptographic algorithms, such as AES. Here, we present some software countermeasures which can be applied to current Hitag2 chips to enhance the security.

1. Keep \( id \) secret.

   There are three input parameters in Hitag2 scheme: \( key \) is constant and must be kept secret; \( iv \) is variable and must be sent in authentication protocol; \( id \) is constant and is transmitted for authentication in immobilizer systems. However, \( id \) can be kept secret as another “key”. In the system enrollment phase, a pair of identifier \((id, aid)\) can be shared between the immobilizer and the car key.
The first item in the pair, \( id \), is used in Hitag2 encryption and kept secret. The second item, \( aid \), is transmitted in the authentication protocol. This countermeasure can be implemented in a low cost by updating the software, while it makes the proposed attack impossible to recover the key. Keeping \( id \) secret can break the step 2 of the attack on immobilizer systems. Without the knowledge of the corresponding \( id \), the adversary cannot initial part of the internal state by guessing part of the key.

2. Inverse \( iv \) before encryption.

Since \( id \) is not required in the proposed attack on RKE systems, keeping \( id \) secret is not enough. The attack is based on the feature that only a little part of the initial states changes for a number of neighbor traces. Inversing \( iv \) before encryption can introduce variance into the initial state earlier and make the states more different for neighbor traces. Since the initial state does not have constant part after applying this countermeasure, the proposed attack on RKE system will fail to determine the initial state.

Conclusion

In this paper, we analyzed the weaknesses of the Hitag2-based cipher scheme and its usage in immobilizer and RKE systems. Two practical methods were proposed to attack the Hitag2-based immobilizer and RKE systems. The algorithms of these methods are similar and share the same idea of guessing the target value from part to whole based on correlation information provided by several traces. The experimental results show that these attacks can break the Hitag2-based immobilizer and RKE systems within minutes at a very low cost. Moreover, for the proposed two attacks, we proposed two software countermeasures to eliminate the risk.

References

[1] S. C. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, and M. Szydlo, “Security Analysis of a Cryptographically-Enabled RFID device,” in 14th USENIX Security Symposium, 2005, pp 1–16.

[2] R. Verdult, F. D. Garcia, and B. Ege, “Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer,” in 22nd USENIX Security Symposium, 2013, pp. 703–718.

[3] A. Bogdanov, “Attacks on the KeeLoq Block Cipher and Authentication Systems,” in Workshop on RFID Security (RFID Sec’08), 2007.

[4] R. Verdult, F. D. Garcia, and J. Balasch, “Gone in 360 seconds: Hijacking with Hitag2,” in USENIX Security Symposium, 2012, pp. 237–252.

[5] R. Verdult, and F. D. Garcia, “Cryptanalysis of the Megamos Crypto automotive immobilizer,” in USENIX Security Symposium, 2015, pp. 17–22.

[6] W. Aerts et al, “A Practical Attack on KeeLoq,” J. Cryptol., vol. 25, no. 1, pp. 136–157, Jan. 2012.

[7] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlides, “Lock It and Still Lose It—On the (In) Security of Automotive Remote Keyless Entry Systems,” in 25th USENIX Security Symposium, 2016, pp 929–944.

[8] N. T. Courtois, S. O’Neil, and J. J. Quisquater, “Practical Algebraic Attacks on the Hitag2 stream cipher” in Information Security, International Conference, 2009, pp 167-176.

[9] R. Benadjila, M. Renard, J. Lopes-Esteves, and C. Kasmi, “One car two frames—Attacks on Hitag2 Remote Keyless Entry Systems Revisited,” in WOOT’17, 2017.

[10] K. Nohl, “Immobilizer Security,” in 8th International Conference on Embedded Security in Cars (ESCAR 2010), 2010.