On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme

Thomas Eisenbarth[1], Timo Kasper[1], Amir Moradi[2,⋆], Christof Paar[1], Mahmoud Salmasizadeh[2], and Mohammad T. Manzuri Shalmani[2]

[1] Horst Görtz Institute for IT Security
Ruhr University Bochum, Germany
[2] Department of Computer Engineering and Electronic Research Center
Sharif University of Technology, Tehran, Iran
{eisenbarth,tkasper,moradi,cpaar}@crypto.rub.de
{salmasi,manzuri}@sharif.edu

Abstract. KeeLoq remote keyless entry systems are widely used for access control purposes such as garage openers or car door systems. We present the first successful differential power analysis attacks on numerous commercially available products employing KeeLoq code hopping. Our new techniques combine side-channel cryptanalysis with specific properties of the KeeLoq algorithm. They allow for efficiently revealing both the secret key of a remote transmitter and the manufacturer key stored in a receiver. As a result, a remote control can be cloned from only ten power traces, allowing for a practical key recovery in few minutes. After extracting the manufacturer key once, with similar techniques, we demonstrate how to recover the secret key of a remote control and replicate it from a distance, just by eavesdropping on at most two messages. This key-cloning without physical access to the device has serious real-world security implications, as the technically challenging part can be outsourced to specialists. Finally, we mount a denial of service attack on a KeeLoq access control system. All proposed attacks have been verified on several commercial KeeLoq products.

1 Motivation

The KeeLoq block cipher is widely used for security relevant applications, e.g., remote keyless entry (RKE) systems for car or building access, and passive radio frequency identification (RFID) transponders for car immobilizers [13]. In the course of the last year, the KeeLoq algorithm has moved into the focus of the international cryptographic research community. Shortly after the first cryptanalysis of the cipher [1], more analytical attacks were proposed [4, 5], revealing mathematical weaknesses of the cipher. The best known analytical attacks target the identify friend or foe (IFF) mode of KeeLoq and require at least 2^{16}

⋆ Amir Moradi performed most of the work described in this contribution as a visiting researcher at Ruhr University Bochum.

plaintext-ciphertext pairs from one transponder. This allows, after several days of computations, for a simple cloning of the transponder and, only in case of a weak key derivation method[1], for obtaining the manufacturer key that is required to generate keys for new valid transponders. Despite the impressive contribution to the cryptanalysis of the cipher, the real-world impacts of the previous attacks are somewhat limited, as described in Sect. 2.3.

Motivated by the ongoing research we investigate the vulnerability of actual KeeLoq implementations with respect to side-channel analysis (SCA), in order to evaluate the security of all KeeLoq modes (IFF and code hopping) and all key derivation schemes. As a result, we present three very practical key recovery attacks and a denial of service attack with severe implications for RKE systems that are currently used in the field. These new attacks — which combine differential power analysis (DPA) with the extend-and-prune strategy of [3] — can be applied to various implementations of KeeLoq. In particular, we have been able to successfully attack hardware realizations, i.e., the Microchip HCSXXX family of integrated circuits (ICs), as well as software implementations running on Microchip PIC microcontrollers. In contrast to the hitherto existing attacks, the techniques proposed by us are also applicable in case of more sophisticated key derivation schemes (cf. Sect. 2.2) and are suitable for breaking both the KeeLoq code hopping mode and the IFF mode.

Since the introduction of DPA in 1999 [6], it has become an established method for extracting cryptographic keys from security devices by exploiting power consumption traces. However, almost ten years later, there is a surprising discrepancy between the well established theory of power analysis (cf., e.g., the CHES workshop proceedings since 1999) and the very few, if any, confirmed DPA attacks on real-world security systems. The targets considered in the literature are often home-made or known implementations on platforms that are well-known to the attacker, and are typically examined in an ideal environment [16, 9, 14], for example with an artificially generated trigger signal. The practical relevance of such a white box cryptanalysis for real-world realizations of cryptography sometimes remains an open question. During our investigations, we were confronted with a known cipher, but with a black box implementation, i.e., no knowledge or information about the devices except for the characterization in the data sheet. This demanded for some extra efforts and reverse engineering of the unknown targets. Despite these obstructions, we were able to mount highly effective attacks with considerable implications on the security of commercial KeeLoq code hopping systems.

The remainder of this contribution is structured as follows. After an introduction to the KeeLoq cipher and its key derivation schemes in Sect. 2, we elaborate in Sect. 3 on how the secret key of a transmitter can be revealed using SCA with as few as ten power traces and only minutes of computation time. Similarly, the manufacturer key used in a receiver is obtained in less than one day. In Sect. 4, we describe several real-world attacks which follow from the

[1] If the key of the transmitter is derived from XORing a simple function of the device serial number with the manufacturer key, the latter can easily be obtained.

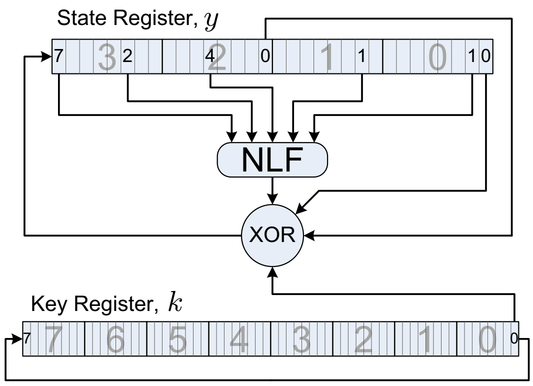


Fig. 1. Block diagram of the KeeLoq encryption

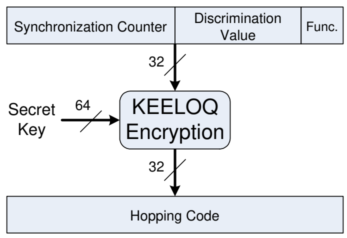


Fig. 2. Generation of KeeLoq hopping codes

key extraction. First, remotes which are in the possession of an attacker can be cloned. The most devastating attack allows to recover the secret key of a transmitter from a distance, just by eavesdropping on at most two hopping code messages. It is perceivable that a technically experienced person (with malicious intent) will develop a machine that allows for automatically spoofing KeeLoq code hopping systems. With such a machine, a completely unskilled attacker could gain access to objects that are protected by these systems without leaving any traces. Finally, we detail on putting an RKE system out of service which would prevent the legitimate owner to open a car door or to access a garage. All our attacks have been extensively tested and verified. We present various experimental results and provide figures for power analysis both based on electric current and the electromagnetic (EM) emanation of different KeeLoq devices.

2 Background

KeeLoq is a block cipher with a 64 bit key and a block size of 32 bits. As illustrated in Fig. 1, it can be viewed as a non-linear feedback shift register (NLFSR) where the feedback depends linearly on two register bits, one key bit, and a non-linear function (NLF). The NLF maps five other register bits to a single bit [1, 4, 5]. Prior to an encryption, the secret key and plaintext are loaded in the key register and the state register, respectively. In each clock cycle, the key register is rotated to the right and the state register is shifted to the right so that the fresh bit prepared by the XOR function becomes part of the state. After 528 clock cycles, the state register contains the ciphertext. The decryption process is similar to the encryption, except for the direction of the shifts and the taps for the NLF and the XOR function.

2.1 Code Hopping Protocol

In addition to KeeLoq IFF systems which provide authentication of a transmitter to the main system using a simple challenge-response protocol, KeeLoq is used in code hopping (or rolling code) applications [10]. In this mechanism, which is widely used, e.g., in car anti-theft systems and garage door openers, the transmitter is equipped with an encoder and the receiver with a decoder. Both share a secret key and a fixed discrimination value, *disc*, with 10 or 12 bits. In addition, they are synchronized with a 16 bit or 18 bit synchronization counter, *cnt*, which is incremented in the encoder each time a hopping code is transmitted. According to Fig. 2, the transmitter constructs a hopping code by encrypting a 32 bit message formed of *disc*, *cnt* and a 4 bit function information. The latter determines the task desired by a remote control, for instance, it enables to open or close more than one door in a garage opener system.

One message sent via the radio frequency (RF) interface consists of a hopping code followed by the serial number of the transmitter. The receiver decrypts the hopping code using the shared secret key to obtain *disc* and the current *cnt*. The transmitter is authenticated if *disc* is identical to the shared one and *cnt* fits in a window of valid values. Three windows are defined for the counter. If the difference between a received *cnt* and the last stored value is within the first window, i.e., 16 codes, the intended function will be executed after a single button press. Otherwise, the second window containing up to 2^{15} codes[2] is examined. In this so-called resynchronization window, the desired function is carried out only if two consecutive counter values are within it, i.e., after pressing the button twice. The third window contains the rest of the counter space. Any transmission with a *cnt* value within this window will be ignored, to exclude the repetition of a previous code and thus prevent replay attacks.

2.2 Key Derivation Schemes

There are two types of keys involved in a typical KeeLoq application. The device key is unique for each remote control and is shared by the transmitter and the receiver. It is established during a learning phase. The manufacturer key is mainly used for deriving device keys. It is to our knowledge identical for all receivers of a given manufacturer and hence enables producing transmitters that cannot be cloned by competitors. Since the manufacturer's key is critical for the security of the product, it is stored in a read protected memory of the receiver. The known key derivation schemes are reviewed in the following:

(a) According to Fig. 2.2.a, the device key is obtained by two KeeLoq decryptions. The two functions F_1 and F_2 (which are usually simple paddings) are applied to the serial number of the transmitter to form the plaintexts for the decryptions.

[2] These window sizes are recommended by Microchip, but they can be altered to fit the needs of a particular system.

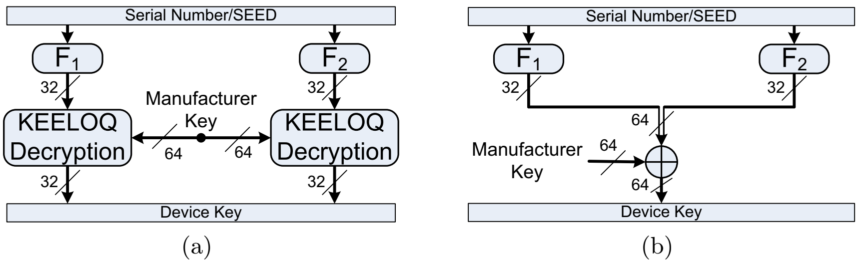


Fig. 3. Key derivation schemes

(b) The next key derivation scheme is similar to the previous one, except for a randomly generated seed value which is stored in the transmitter and is used instead of the serial number to generate the device key. During the learning phase, a transmitter can be forced to send its seed value.

(c) As presented in Fig. 2.2.b, sometimes the device key is generated from an XOR of a simple function of the serial number with the manufacturer key.

(d) The last scheme is similar to the third one. The device key is derived from an XOR of the manufacturer key and a simple function of the seed value of the transmitter.

Note that a manufacturer may develop a proprietary key derivation scheme not included in the above list.

2.3 Previous Work

The first two cryptanalytical attacks on the KeeLoq algorithm were published by Bogdanov [1]. One attack is based on slide and guess-and-determine techniques and needs about $2^{50.6}$ KeeLoq encryptions. The other one additionally uses a cycle structure analysis technique and requires 2^{37} encryptions. However, both attacks require the entire codebook, i.e., all 2^{32} plaintext-ciphertext pairs. Courtois *et al.* [4] proposed two attacks. One is a slide-algebraic attack demanding for $2^{51.4}$ KeeLoq encryptions and 2^{16} known plaintext-ciphertext pairs. The second slide attack can be carried out knowing almost the entire codebook. It reveals the secret key with a complexity of approximately 2^{27} KeeLoq encryptions. Recently, Indesteege *et al.* presented more practical attacks on the KeeLoq cipher [5]. All of them are based on slide and meet-in-the-middle attacks. The best one uses 2^{16} known plaintext-ciphertext pairs and has a complexity of $2^{44.5}$ KeeLoq encryptions. It can find the secret key in two days using 50 dual core computers.

The above attacks are applicable to KeeLoq IFF systems but they cannot be directly applied to the code hopping mode [10], which appears to be the dominant commercial application of KeeLoq. The required minimum of 2^{16} plaintext-ciphertext pairs cannot be obtained in case of a code hopping system, because an adversary has only access to the ciphertexts that are transmitted by a remote control, while the corresponding plaintexts are unknown. Although knowing a sequence of 2^{16} ciphertexts and the discrimination value of a code hopping encoder would be sufficient to perform the attack described in [5], the commercial products employing the KeeLoq code hopping protocol, i.e., HCS modules, do not allow an attacker to access this information.

3 DPA on KeeLoq

When we started to analyze the targets using KeeLoq, we were exposed to a "classical" situation for physical attacks: even though the algorithm was known, hardly anything was known about the implementation. We found that the transmitters usually employ HCSXXX modules of Microchip, featuring a hardware implementation of the cipher. The receivers we looked at are typically equipped with a read-protected PIC microcontroller on which a KeeLoq decryption routine is implemented in software. This section explains the details of DPA-attacking transmitters and receivers, starting with a general approach that is appropriate for both types of realizations.

Initial Cipher Analysis Before being able to actually perform a DPA on a particular implementation of a cipher, one needs to make certain assumptions about the leakage produced by it. Then, a DPA scheme for exploiting that leakage must be developed, which depends on the cipher structure as well as on the particularities of the given implementation.

Measurement The power traces are gathered by measuring the current via a shunt resistor connected to the ground pin of the target chip. In addition, we acquire the EM radiation of the device by means of near field probes. For convenience, we built a printed circuit board (PCB) that allows for emulating KeeLoq chips and for controlling a transmitter from a PC so that a measurement sequence can be executed automatically. The power traces were acquired using an Agilent Infiniium 54832D digital oscilloscope with a maximum sampling rate of 4 GS/s.

Data Pre-Processing and Alignment One problem of aligning the power traces of an unknown implementation is the absence of a suitable trigger signal. The solution for this is target-specific and detailed in Sect. 3.2 and Sect. 3.3 for transmitters and receivers, respectively. Another problem is that all of the target devices are clocked by a noisy internal oscillator. Hence we had to find a way to remove the clock jitter. We know that most of the data-dependent leakage occurs in the instant when the registers are clocked, producing peaks with varying amplitudes in each clock period. The amplitudes of these peaks directly correspond to the dynamic power consumption of the target circuit and thus hold most of the relevant information. Accordingly, we extract the peaks from the power consumption in software, and base our DPA attack solely on the amplitudes of the peaks. This peak extraction step has two advantages for the subsequent analysis: (*i*) the amount of data is greatly reduced, which facilitates the post-processing and the data storage, and (*ii*) more importantly, the peak extraction allows for an accurate alignment of the traces. Other methods for removing the clock jitter, such as Fourier transform, filtering, etc., turned out to be less effective and more complicated.

Developing and Performing the DPA After peak extraction and alignment steps, the traces can be processed by the DPA algorithm. For the transmitter modules we only knew the ciphertext and hence had to perform our attacks starting from the last round of the encryption. For the software implementation of the PICs we knew the plaintexts and started the attack of the first round of the decryption.

3.1 Building a Powerful DPA for KeeLoq

It is known that for successfully performing a DPA attack, some intermediate value of the cipher has to be identified that (*i*) depends on known data (like the plaintext or the ciphertext), (*ii*) depends on the key bits, and (*iii*) is easy to predict. Furthermore, it is advisable to choose a value that has a high degree of nonlinearity with respect to the key, to avoid so-called "ghost peaks" for "similar" keys [2]. For every DPA, a model for estimating the power consumption is needed. Compared to the two shift registers, the power consumption of the combinational part, i.e., a few XORs and the 5×1 non-linear function, is small and can be neglected. Note that the Hamming distance of the key register does not change, since the key is simply rotated. This leads to a theoretically constant power consumption of the key register in each clock cycle. Hence, we focus on the state register $\boldsymbol{y}$. We execute a correlation DPA attack (CPA) [2] based on the following hypothetical power model

$$\mathrm{P}_{Hyp}^{(i)} = \mathrm{HD}\left(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(i-1)}\right) = \mathrm{HW}\left(\boldsymbol{y}^{(i)} \oplus \boldsymbol{y}^{(i-1)}\right) \quad (1)$$

where $\mathrm{P}_{Hyp}^{(i)}$ denotes the hypothetical power consumption in the i^{th} round, HD and HW are Hamming distance and Hamming weight, respectively, $\boldsymbol{y}^{(i)}$ indicates the content of the state register in the i^{th} round, and $\oplus$ is a 32 bit XOR function. As mentioned before, the known ciphertext attack on the encryption is identical to the known plaintext attack on the decryption[3]. We describe the known ciphertext attack on the encryption. Starting from the 528^{th} round, 32 bits of the final state $\boldsymbol{y}^{(528)} = \left(y_0^{(528)}, \ldots, y_{31}^{(528)}\right)$, are known. Furthermore, 31 bits of $\boldsymbol{y}^{(527)}$, i.e., $\left(y_1^{(527)}, \ldots, y_{31}^{(527)}\right)$, are known because they are identical to

[3] Both attacks target state $\boldsymbol{y}^{(l)}$ of the decryption, which is the same as state $\boldsymbol{y}^{(528-l)}$ of the encryption.

$\left(y_0^{(528)}, \ldots, y_{30}^{(528)}\right)$. Therefore, just $y_0^{(527)}$ is unknown. According to Fig. 1, we can write

$$y_{31}^{(i+1)} = k_0^{(i)} \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus \mathrm{NLF}\left(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_9^{(i)}, y_1^{(i)}\right) \quad (2)$$

where $k_0^{(i)}$ is the rightmost bit of the key register in the i^{th} round. Knowing that $k_j^{(i)} = k_{(i+j) \bmod 64}$, we can rewrite Eq. (2) as

$$y_0^{(527)} = k_{15} \oplus y_{16}^{(527)} \oplus y_{31}^{(528)} \oplus \mathrm{NLF}\left(y_{31}^{(527)}, y_{26}^{(527)}, y_{20}^{(527)}, y_9^{(527)}, y_1^{(527)}\right) \quad (3)$$

Thus, recovering $y_0^{(527)}$ directly reveals one bit of the key register. This process is the same for recovering the LSB of the state register of the previous rounds, i.e., $y_0^{(i)},\ i = (526, 525, \ldots)$. However, Eq. (3), depends linearly on the key bit k_{15}. Above we stated that nonlinearity helps distinguishing correct key hypotheses from wrong ones. Hence, recovering the key bit-by-bit might not be the best choice[4]. Fortunately, according to Fig. 1, the LSB of the round state, $y_0^{(i)}$, enters the NLF leading to a nonlinear relation between the key bit k_{15} and the state $\boldsymbol{y}^{(526)}$. Accordingly, the nonlinearity for one key bit k_j increases in each round after it was clocked into the state.

Algorithm 1 A Scalable DPA for KeeLoq

Input: m : length of key guess, n: number of surviving key guesses, k: known previous key bits
Output: SurvivingKeys
1: KeyHyp $\leftarrow$ **all** $\{0,1\}^m$
2: **for all** KeyHyp$_i$; $0 \leq i < 2^m$ **do**
3: Perform CPA on round $(528 - m)$ using P_{Hyp} and k
4: **end for**
5: SurvivingKeys $\leftarrow$ n most probable partial keys of KeyHyp

Taking the increased nonlinearity in the successive rounds into account, we developed a scalable DPA, as described in Alg. 1, that allows for finding a subset n of surviving key candidates by guessing m bits of the key in an instant. Note that in step 3 of the algorithm the CPA is performed on round $(528 - m)$, hence taking advantage of a key bit passing the NLF m times. The significance of the known previous bits k will become clear below in the extended attack (Alg. 2), where Alg. 1 is executed repeatedly.

We performed simulations of the attack described in Alg. 1, assuming a Hamming distance leakage model. The simulated traces allow for testing our attacks

[4] Simulations show that an attack recovering the key bit by bit is much weaker than an attack that recovers several key bits at a time. Still, the key can also be recovered for single bit key guesses – in other words even a classical DPA on the LSB of the state register is feasible.

and also to evaluate how well an attack would work under “perfect” conditions. We generated a set of encryption traces with random plaintext input and com-

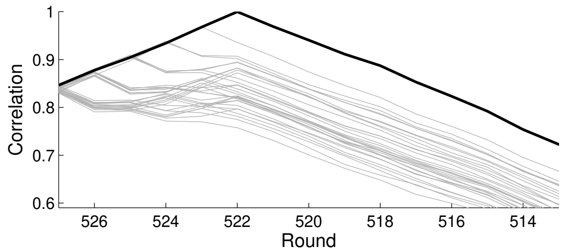


Fig. 4. Simulated correlation of key hypotheses as a function of KeeLoq rounds. Correct key guess (black solid line) vs. wrong key guesses (thin gray lines).

puted the Hamming distance of all registers for each round. We performed a correlation DPA where we predicted the Hamming distance of the state register of round 522, $\mathrm{P}_{Hyp} = \mathrm{HD}\left(\boldsymbol{y}^{(522)}\right)$. Fig. 4 shows the correlation for the $2^6 = 64$ key hypotheses over the first few rounds. Of course, the correlation is 1 for the right key (thick solid line) in round 522. Unfortunately, some of the wrong key guesses (thin gray lines) also yield a high correlation. This is due to the high linearity between both the state and the key guesses, and between the different states. Furthermore we get a high correlation in the rounds before and after the predicted round. This is because most of the bits of the shift register remain unchanged in the nearby rounds. The most probable wrong key guess is always the one that differs only in the LSB. This underlines our expectation that the linearity increases the error probability of guessing the less significant key bits.

Algorithm 2 Pruning for the Best Key Hypothesis

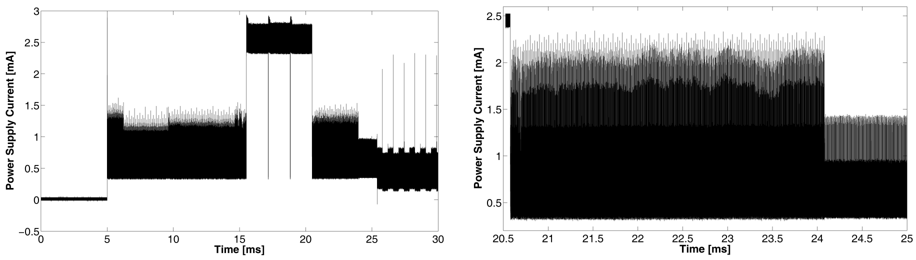
Input: m : length of key guess, n: number of surviving key guesses
Output: K: recovered key

```
K ← Algorithm 1(m, n, ∅)
for round = 1 to ⌈64/m⌉ do
    K′ ← ∅
    for all k_i ∈ K, 0 ≤ i < n do
        K′ ← K′ ∪ Algorithm 1(m, n, k_i)
    end for
    K ← n most probable keys of K′
end for
return K
```

To improve the strength of our attack and to take care of the misleading high correlations, we added another attack step. Alg. 1 can be repeated to guess all partial keys, one after the other. These iterations of the attack need to be done one after another, because we require the previous key bits and thus the state $\boldsymbol{y}$ as a known input for each execution of the algorithm. Since some of the bits of the previous key guess might be faulty, we keep a number n of the most probable partial key guesses as survivors. Wrong surviving candidates of the previous round will result in a misleading initial state $\boldsymbol{y}$ for the following attack round and hence strongly decrease the correlation of subsequent key guesses. This does not only allow for an assertion of the correct previous key guesses, but also for detecting faulty previous keys. Hence, the attack has an error-correcting property. If all key guesses of one round show a low correlation, we can go one step back and broaden the number of surviving key guesses n. Alg. 2 describes this procedure, which is similar to the "pruning process" described by Chari *et al.* in [3]. In the last round ($i = \lceil \frac{64}{m} \rceil$) the program verifies whether an error occurred and the key with the highest correlation coefficient is selected out of the n surviving keys. It will be shown in the following subsections that Alg. 2 results in a quite strong attack.

3.2 Details of the Hardware Attack

For attacking commercial KeeLoq code hopping encoders we first had to find the points in time in the power traces (Fig. 5.a) that correspond to the encryption function. We found that the encryption happens after writing to the EEPROM[5], i.e., in the time interval between 20.5 ms and 24 ms (Fig. 5.b). The power traces reveal that the frequency of the internal oscillators of the ICs is approximately 1.25 MHz.



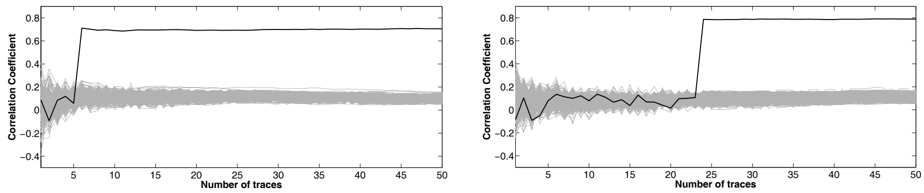
(a) From power up to start sending

(b) Encryption part

Fig. 5. Power consumption traces of a HCS module

We modified the attack described in Sect. 3.1 to correlate all known and predicted rounds to the corresponding power peaks. This is possible since we are

[5] The high amplitude periods of the power trace correspond to writing to the internal EEPROM.



(a) DIP (b) SOIC

Fig. 6. Correlation coefficients of key hypotheses of HCS201 ICs as a function of the number of measured traces.

able to locate the leakage of each round. The modified attack was performed on HCS200, HCS201, HCS300, HCS301, HCS361, HCS362, and HCS410 [11, 12] in both DIP and SOIC packages. In the best case we were able to recover the secret key of DIP package ICs from only six power traces when sampling at a rate of 200 MS/s. At most 30 power traces are sufficient to reveal the secret key of an HCS module in an SOIC package, which has a lower power consumption, resulting in a worse signal-to-noise ratio (SNR) of the measurements. Fig. 6 compares the correlation coefficients of the correct key of HCS201 chips in DIP and SOIC packages as a function of the number of traces. The sudden increase of the correlation is due to the error-correcting property of our attack, and also due to the fact that we repeated the attack for all 528 rounds of the algorithm in order to verify the revealed key.

We repeated the above experiments with an EM near field probe RF-U 5-2 [8] to directly monitor the electromagnetic emanation, instead of measuring the electric current via a shunt resistor. The probe was positioned close to the ground pin of the HCS201 IC in a DIP package, in order to acquire the peaks of the EM field that are caused by the change of electric current. Compared to inserting a resistor in series to the device, this differential electromagnetic analysis (DEMA) can be regarded as non-invasive, as no modification of the PCB is necessary. Contrary to our assumption that the SNR would suffer from environmental noise and thus much more traces would be required to recover the key, the results obtained and the number of traces needed are very comparable to the case of power traces acquired by means of a resistor (Fig. 5). In the best case, we succeeded with recovering the key after only 10 DEMA measurements.

To estimate the minimum technical requirements for the SCA, we performed experiments with varying sampling rates and evaluated the number of power traces required for recovering the correct key. Fig. 7 shows the results for attacking a HCS201 chip in a DIP package in the case of current measurements via a resistor. We conclude that our attack can be carried out effectively even with low-cost equipment, e.g., an oscilloscope with a maximum sample rate as low as 50 MS/s enables finding the secret key from only 60 power traces.

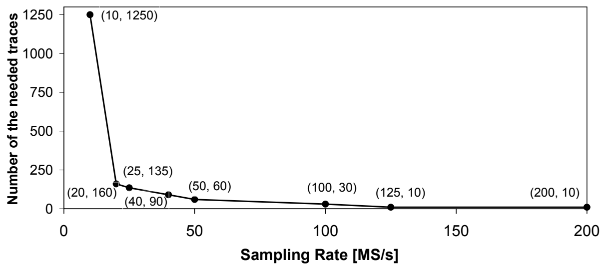


Fig. 7. Number of measurements required for revealing the secret key of a HCS201 IC in a DIP package as a function of the sampling rate. The numbers in parentheses give the exact coordinates of the points.

3.3 Details of the Software Attack

The next target of our attack is the code hopping decoder implemented in the receiver. We recall that the receiver contains the manufacturer key, which is an attractive target for a complete break of the system. A PIC microcontroller handles the key management, controls for instance the motor of the garage door or the locking system of the car, and performs the KeeLoq decryption in software.

Receivers usually offer a so-called "learning mode". In this learning mode the user can register new transmitters to cooperate with the receiver. We were able to identify the key derivation scheme of the target receiver as scheme (a) of Sect. 2.2. Hence we can recover the manufacturer key k_M by performing a DPA key recovery on the KeeLoq decryption that is performed during learning mode.

Before executing the DPA, we adapted the power model of the attack of Sect. 3.1 to a PIC software implementation. Typically, PIC microcontrollers leak the Hamming weight of the processed data [15]. Furthermore, one can assume that the state is stored in the 8 bit registers of the PIC microcontroller which are regularly accessed. Hence, instead of predicting the Hamming distance HD $\left(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(i-1)}\right)$ of the whole state – as was done for the hardware attack in Sect. 3.2 – we predict the Hamming weight of the least significant byte (LSB) of the KeeLoq state register:

$$P_{Hyp}^{(i)} = \mathrm{HW}\left(\boldsymbol{y}_{\mathrm{LSB}}^{(i)}\right) = \sum_{k=0}^{7} y_k^{(i)}$$

We performed the attack by putting the receiver into learning mode and sending hopping code messages with random serial numbers to the receiver[6]. Lacking any information in the power consumption of the PIC that could have been used as

[6] We emulated a remote control by connecting the RF interface of a transmitter to the parallel port of a PC.

trigger, we triggered the scope directly after transmitting the last bit via the RF interface. This results in our traces not being well-aligned, leading to a high number of power samples needed to perform a successful DPA attack.

While performing the attack we noticed that the correlation coefficient of the correct key become continuously worse with an increasing number of rounds. For the first few key bits, 1000 traces sampled at 125 MS/s are sufficient to find the key. Surprisingly, we need roughly ten times as many traces for recovering the full 64 bit key. This gradual decrease of the correlation is due to a misalignment that occurs during the execution of the KeeLoq algorithm. Hence, the problem is not a bad trigger condition, since the trigger affects all time instances in the same way. We assume that the program code is likely to have a data-dependent execution time for each round of KeeLoq, causing the increasing misalignment with an increasing number of rounds, and hence complicating the SCA.

4 Attacks and Implications

In the previous section we showed how the keys of hardware and software implementations of KeeLoq can be recovered by a skilled adversary using SCA. We will now evaluate the vulnerability of real-world systems to our attacks and illustrate the implications. We detail four different attack scenarios, which allow for breaking basically any system using KeeLoq with modest efforts. We focus on code hopping applications, since they are more commonly used and, due to the lack of known plaintexts, harder to cryptanalyze than IFF systems. Still, IFF systems are just as vulnerable to our DPA attacks as the code hopping devices. Some of the transmitters we analyzed even offer both operating modes. The success of some of our attacks depends on the knowledge about the particular key derivation scheme, as described in Sect. 2.2. However, they are appropriate for all key derivation schemes we are aware of.

Note that for all the attack scenarios described below it is very difficult, e.g., for an insurance or a prosecutor, to find evidence that a crime has been committed, as typically no traces are left when electronically circumventing an RKE system.

4.1 Cloning a Transmitter

For cloning a transmitter using power analysis, an adversary needs physical access to it to acquire at least 10 to 30 power traces. Hence, the button of the remote control has to be pressed several times, while measuring the power consumption and monitoring the transmitted hopping code messages. After recovering the device key k_{Dev} with the side-channel attack described in Sect. 3.2, the recorded messages can be decrypted, disclosing the discrimination and counter values of the original transmitter at the time of the attack. Now, the HCS module of a spare remote control can be programmed with the serial number, counter value and discrimination value of the master. Consequently, the freshly produced transmitter appears to be genuine to a receiver and allows for accessing the same target as the original.

Implications This attack applies to scenarios in which a transmitter is handed over to an untrustworthy person for some minutes, e.g., car rental or cleaning personnel. While possessing the transmitter, an attacker could clone it for future reuse. Depending on the time interval between recovering the key and using the reproduced remote control, the attacker will have to press the button of the transmitter several times, for catching up with the counter value in the receiver which might have been increased meanwhile by the legitimate operator.

Given that a technically demanding SCA has to be carried out in order to clone just one remote control, and physical access to it is required, it can be speculated that the impact of this attack is limited. The cost-benefit ratio is typically too low, except for the case that very high monetary values are involved, e.g., rental of expensive cars. In most cases it is easier, e.g., to smash a window, unless an attacker intends to remain unnoticed or to gain access to an object repeatedly. It is important to stress that in most modern cars the door access mechanism and the immobilizer are separate systems. Thus, even if a car can be opened with our attack, this does not imply that a criminal can equally easily drive away with it.

4.2 Recovering a Manufacturer Key

The key recovery of the manufacturer key k_M depends on the particular key derivation scheme. If scheme (c) or (d) of Sect. 2.2 is used, i.e., an XOR of a known input and the manufacturer key k_M, disclosing the latter is trivial. After a successful key recovery attack on one transmitter (see above) of the same brand, k_M is found by reversing the XOR function. The known input is either part of each hopping code message, in case of the serial number, or can be obtained from the remote control, in case of a seed. The derived k_M can be verified with a second transmitter.

An adversary targeting the manufacturer key k_M for scheme (a) or (b) of Sect. 2.2 requires physical access to one receiver of that manufacturer. The key of the KeeLoq decryption performed inside the receiver during the learning phase is recovered with an SCA requiring several thousand power traces, as described in Sect. 3.3. The k_M obtained from the DPA can be verified with a single hopping code message.

Knowing the manufacturer key k_M, valid device keys for producing transmitters with arbitrary serial numbers can be generated, just by applying the key derivation. The counterfeited remote controls will be recognized as genuine by all receivers of that manufacturer.

Implications The described key recovery requires access to a transmitter (key derivation by XOR) or a receiver (key derivation by KeeLoq) of the manufacturer to be attacked and, in case of a key derivation employing KeeLoq, a very skilled adversary for performing the SCA. The RKE devices can be simply purchased, e.g., from a hardware store, and even returned after extracting the key. The recovered k_M does not directly permit to unauthorizedly open a door secured by KEELOQ, because the newly produced transmitters need to be registered by the receiver first, implying physical access. Still, the described key recovery is highly relevant in the context of product piracy: the economic function of k_M is customer retention, e.g., a business model could comprise making the main profit on selling spare transmitters that operate with the receivers of that manufacturer. Without knowledge of the manufacturer key, valid transmitters cannot be produced to work with the receiver. However, a competitor possessing k_M could produce transmitters compatible to those receivers, or even produce whole RKE systems bearing the brand and being compatible to those of the original manufacturer, and hence severely affect the business of the latter. Due to the depicted high economic impact, it is very conceivable that this attack will be carried out by criminals sooner or later. In the worst case, this might result in publicly available lists of manufacturer keys on the web.

4.3 Cloning a Transmitter without Physical Access

Knowing the manufacturer key k_M, e.g., by performing the previous attack, and the key derivation method of a target device family, a remote control can be cloned by eavesdropping. The adversary has to intercept at most two hopping code messages, c_1 and c_2, sent by the target transmitter of the same brand. The process of finding the secret key of the eavesdropped transmitter depends on the key derivation schemes detailed in Sect. 2.2.

If the key is derived from the serial number of the transmitter (schemes (a) and (c)), finding its device key is straightforward, since the intercepted messages contain the serial number. The adversary can simply perform the key derivation using the known manufacturer key to obtain the device key. After decrypting one message c_i and thereby disclosing the current counter value, the adversary is able to generate valid hopping code messages for spoofing the receiver and gain access to a protected site. The computational complexity of this attack is a single KEELOQ decryption.

However, if a seed value is used during the key derivation (schemes (b) and (d)), recovering the secret key of the eavesdropped transmitter is more difficult. An exhaustive search needs to be performed to find the seed value. For recovering k_{Dev}, the adversary calculates $k_{Dev}^{seed} = \text{KeyDerivation}\,(k_M, seed)$ for each possible value of $seed$ and decrypts the two intercepted messages c_1 and c_2 using k_{Dev}^{seed}:

$$(cnt_1, disc_1) = \text{KEELOQ}^{-1}\left(c_1, k_{Dev}^{seed}\right) \qquad (cnt_2, disc_2) = \text{KEELOQ}^{-1}\left(c_2, k_{Dev}^{seed}\right)$$

Once both messages have the same discrimination value, i.e., $disc_1 = disc_2$, and similar counter values[7] cnt_1 and cnt_2, the correct device key is found[8].

[7] "Similar" counters means that the difference $cnt_2 - cnt_1$ is less than a small threshold, e.g., 16, depending on the period between the two eavesdrops.

[8] With a small probability we get false positives. These can easily be identified by sending one message to the target receiver.

There are three different seed sizes possible for KeeLoq systems, depending on the chip family. If a 32 bit seed value is used (e.g., HCS200), the adversary has to run on average 2^{32} KeeLoq decryptions to find the correct seed. According to our software implementations, this takes less than two hours on a 2.4 GHz Intel Core2 Quad PC. On a special-purpose computing machine such as COPACOBANA [7], the correct 32 bit seed value and hence the key can be recovered in just one second. In case of a 48 bit seed value (e.g., HCS360) it is not promising to recover the correct seed value using standard PCs. Still, it is possible to perform the 2^{48} required KeeLoq decryptions on average in about nine hours using COPACOBANA. However, chips like the HCS410, using a 60 bit seed, are not vulnerable to this attack. Running 2^{60} KeeLoq decryptions is not feasible in a reasonable time with currently existing equipment. We would like to mention that none of the real-world KeeLoq systems we analyzed used *any* seed. Moreover, if physical access to the transmitter is given, even 60 bit seed values are obtained by pressing one button.

Implications This attack has the most devastating impact and it scales very well. It affects all KeeLoq RKE systems of a manufacturer, as soon as the respective k_M is known. Extracting k_M, as described in Sect. 4.2, can be outsourced to criminal cryptographers who may construct (and sell) an easy-to-use machine that eavesdrops on two messages, automatically recovers the device key, and opens the target. Thus it enables a completely unskilled person to gain illicit access to objects secured with KeeLoq, at little cost and without leaving any traces. Building such an eavesdropping device is simple once the manufacturer key is available. It is sufficient to connect a legitimate receiver to a (laptop) PC and to monitor hopping codes from a range of up to several hundred meters, depending on the targeted RKE system.

4.4 Denial of Service

As detailed in Sect. 2.1 and in [10], the counters of a receiver and a transmitter are synchronized with every valid hopping code message received. This behavior can be exploited for putting an RKE system out of operation. We assume that the adversary has recovered the device key k_{Dev} of a target transmitter, e.g., by performing one of the attacks described above, and is thus able to generate valid hopping code messages. She sets the counter value to the maximum value inside the resynchronization window and sends two consecutive valid hopping codes. The receiver resynchronizes its counter to the new value. Consequently, the counter of the original transmitter is now considered to be outdated and the respective hopping codes are ignored by the receiver. Finally, the owner of the original transmitter needs to press the button very often, namely 2^{15} times, to increase the counter value back into the first window, where the transmitter produces valid hopping code messages.

Implications This attack allows for locking out a legitimate user, leaving the impression that the KeeLoq RKE device is out of service. It can be performed by an unskilled adversary in the following scenario. Similarly to the eavesdropping device mentioned in Sect. 4.3, a spare transmitter enables a standard PC to transmit self-generated hopping codes. The program code for generating valid hopping codes could be provided by a skilled criminal, e.g., via the internet. Hence, this attack can have dramatic consequences, especially for the automotive industry, where reliability is of paramount importance. Apart from compromising the corporate image, the additional costs for increased customer support, e.g., fixing the spoofed devices, have to be considered.

5 Conclusion

We presented the first successful DPA and DEMA attacks on KeeLoq implementations that can be applied to both IFF and code hopping devices. We demonstrated new techniques to reveal secret keys from commercial hardware and software implementations of KeeLoq. By applying these attacks, we illustrated how to conduct a complete break of the KeeLoq code hopping scheme.

We revealed a manufacturer key from a receiver using a few 1000 power traces, and recovered the device key of a remote control with as few as 10 traces. We found that the investigated hardware implementations show an almost perfect leakage and hence constitute an easier target for SCA than the software implementations, whose data-dependent time variations made the DPA more difficult.

Analyzing real-world applications of KeeLoq and taking into account several key derivation schemes, we developed an eavesdropping attack that allows for cloning a transmitter from a distance, i.e., by intercepting at most two hopping code messages. This attack could be prevented if a 60 bit seed value, with good random properties, would be used for the key derivation. In addition, we introduced a denial of service attack for the KeeLoq code hopping mode. Both attacks can be performed by a completely unskilled person, as the technically challenging part, including the key extraction by means of SCA, needs to be conducted only once for each manufacturer and can thus be outsourced to criminal cryptographers.

This contribution shows that widespread commercial applications, claiming to be highly secure, can be practically broken with modest cost and efforts using SCA attacks. Thus, physical attacks must not be considered to be only relevant to the smart card industry or to be a mere academic exercise. Rather, effective countermeasures against side-channel attacks need to be implemented not only in high-value systems such as smart cards, but also in other embedded security applications.

Acknowledgements

We would like to thank Andrey Bogdanov for helpful discussions regarding the KeeLoq SCA attack, and for bringing cryptanalysis of KeeLoq to our attention.

References

1. A. Bogdanov. Attacks on the KeeLoq Block Cipher and Authentication Systems. In *3rd Conference on RFID Security 2007 (RFIDSec 2007)*. `http://rfidsec07.etsit.uma.es/slides/papers/paper-22.pdf`.
2. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
3. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. *Cryptographic Hardware and Embedded Systems-Ches 2002: 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002: Revised Papers*, 2002.
4. N. T. Courtois, G. V. Bard, and D. Wagner. Algebraic and Slide Attacks on KeeLoq. In *Fast Software Encryption - FSE 2008*, Lecture Notes in Computer Science. Springer, 2008.
5. S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel. A Practical Attack on KeeLoq. In *Advances in Cryptology - EUROCRYPT 2008*, Lecture Notes in Computer Science. Springer, 2008.
6. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, London, UK, 1999. Springer-Verlag.
7. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 101–118. Springer, 2006.
8. Langer EMV-Technik. Details of Near Field Probe Set RF 2. `http://www.langer-emv.de/en/produkte/prod_rf2.htm`.
9. S. Mangard, N. Pramstaller, and E. Oswald. Successfully Attacking Masked AES Hardware Implementations. In *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.
10. Microchip. An Introduction to KeeLoq Code Hopping. `http://ww1.microchip.com/downloads/en/AppNotes/91002a.pdf`.
11. Microchip. HCS200, KeeLoq Code Hopping Encoder. `http://ww1.microchip.com/downloads/en/DeviceDoc/40138c.pdf`.
12. Microchip. HCS410, KeeLoq Code Hopping Encoder and Transponder. `http://ww1.microchip.com/downloads/en/DeviceDoc/40158e.pdf`.
13. Microchip. HCS410/WM, KeeLoq Crypto Read/Write Transponder Module. `http://ww1.microchip.com/downloads/en/DeviceDoc/41116b.pdf`.
14. S. B. Örs, E. Oswald, and B. Preneel. Power-Analysis Attacks on an FPGA - First Experimental Results. In *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 35–50. Springer, 2003.
15. E. Peeters, F. Standaert, and J. Quisquater. Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Integration, the VLSI Journal*, 40(1):52–60, 2007.
16. K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES: Combining Side Channel- and Differential-Attack. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 163–175. Springer, 2004.