Cryptanalysis of the KeeLoq block cipher

Andrey Bogdanov

Chair for Communication Security
Ruhr University Bochum, Germany

Abstract. KeeLoq is a block cipher used in numerous widespread passive entry and remote keyless entry systems as well as in various component identification applications. The KeeLoq algorithm has a 64-bit key and operates on 32-bit blocks. It is based on an NLFSR with a nonlinear feedback function of 5 variables.
In this paper a key recovery attack with complexity of about 2^{52} steps is proposed (one step is equivalent to a single KeeLoq encryption operation). In our attack we use the techniques of guess-and-determine, slide, and distinguishing attacks. Several real-world applications are vulnerable to the attack. To our best knowledge this is the first paper to describe and cryptanalyze the KeeLoq block cipher.

Key words: KeeLoq block cipher, cryptanalysis, slide attacks, guess-and-determine attacks, distinguishing attacks

1 Introduction

A large proportion of modern block ciphers are built upon Feistel networks. Such cryptographic algorithms as DES [22], Blowfish [26], KASUMI [6], GOST [32] or RC5 [25] are based on balanced Feistel networks. Other block ciphers use source-heavy or target-heavy unbalanced Feistel networks (e.g. REDOC III [27], Skipjack [23], etc.). The most extreme case of a source-heavy unbalanced Feistel network [28] is a nonlinear feedback shift register (NLFSR), which can be applied in both stream cipher and block cipher designs.

A number of NLFSR-based stream ciphers have been recently proposed (e.g. Achterbahn [8] and [7], Grain [9]) and successfully cryptanalyzed using linear [2] and nonlinear [11], [24] approximations of nonlinear feedback functions.

KeeLoq is a block cipher based on an NLFSR with a nonlinear boolean feedback function of 5 variables. The algorithm uses a 64-bit key and operates on 32-bit blocks. Its architecture consists of two registers (a 32-bit text register and a 64-bit key register), which are rotated in each of 528 encryption cycles, and of a nonlinear function (NLF) providing nonlinear feedback. One bit of the key is added to the output of the NLF modulo 2 in each cycle.

In this paper a key recovery attack on KeeLoq is proposed. It is based on the following weaknesses of the KeeLoq structure: First, the key schedule is self-similar, which allows us to mount a slide attack [3], [4]. It is supported by the relatively short block length which allows one to guess the first slid pair. Second,

the existence of an efficient linear approximation of the NLF is used to recover a part of the key. Then the rest of the key bits are obtained using other linear relations within KeeLoq.

The key recovery complexity of our attack is about 2^{52} computational steps. One computational step is equivalent to a single KeeLoq encryption. The attack requires all 2^{32} plaintext-ciphertext pairs and a memory of 2^{32} 32-bit words. Several computing devices can share the memory during the attack. All computations are perfectly parallelizable. The property inherited from the slide attacks [3], [4] is that the complexity of our attack is independent of the number of encryption cycles, which is as a rule not the case for linear or differential cryptanalysis, where the complexity often grows exponentially with the number of iterations.

The light-weight architecture of the KeeLoq cipher allows for an extremely low-cost and efficient hardware implementation. This contributed to the popularity of the KeeLoq cipher among designers of remote keyless entry systems, automotive and burglar alarm systems, automotive immobilizers, gate and garage door openers, identity tokens, component identification systems. For instance, the KeeLoq block cipher is used in the HomeLink wireless control systems to secure communication with some garage door openers [10]. The KeeLoq technology supplied by Microchip Technology Inc. includes the KeeLoq cipher and a number of authentication protocols. Our description of KeeLoq is based on the newly published article [31], [14] and a number of the manufacturer's documents [13], [15], [19], [20].

The paper is organized as follows. Section 2 describes the KeeLoq algorithm. In Section 3 a key recovery attack on KeeLoq is proposed. In Section 4 we discuss the impact of our attack with respect to real-word applications. We conclude in Section 5.

2 Description of the KeeLoq algorithm

KeeLoq is a block cipher with a 64-bit key which operates on 32-bit words [31], [14]. Its design is based on a nonlinear feedback shift register (NLFSR) of length 32 bits with a nonlinear feedback function of 5 variables. The feedback depends linearly on two other register bits and on the next key bit taken from the rotated key register of length 64 bits.

Let $V_n = \mathrm{GF}(2)^n$ be the set of all n-bit words and $Y^{(i)} = (y_{31}^{(i)}, \ldots, y_0^{(i)}) \in V_{32}$, $y_j^{(i)} \in \mathrm{GF}(2)$, describe the state of the text register in cycle i for $j = 0, \ldots, 31$ and $i = 0, 1, \ldots$ Let also $K^{(i)} = (k_{63}^{(i)}, \ldots, k_0^{(i)}) \in V_{64}$, $k_j^{(i)} \in \mathrm{GF}(2)$, denote the state of the key register in cycle i for $j = 0, \ldots, 63$ and $i = 0, 1, \ldots$ Then each cycle of encryption can be described using the following algorithm (see Figure 1):

Compute the feedback bit: $\varphi = NLF(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_9^{(i)}, y_1^{(i)}) \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus k_0^{(i)}$
Rotate text and insert feedback: $R^{(i+1)} = (\varphi, y_{31}^{(i)}, \ldots, y_1^{(i)})$
Rotate key: $K^{(i+1)} = (k_0^{(i)}, k_{63}^{(i)}, \ldots, k_1^{(i)})$.

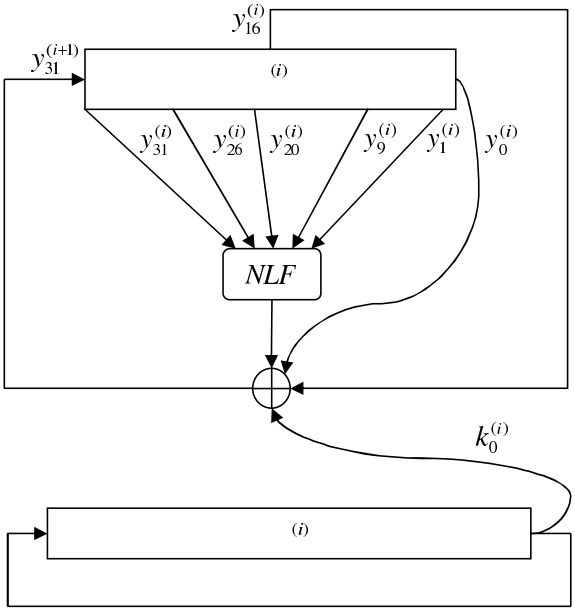

Fig. 1. The i-th KeeLoq encryption cycle

For encryption the key register is filled with the 64 key bits $K = (k_{63}, \dots k_0) \in V_{64}$, $k_j \in \mathrm{GF}(2)$, $j = 0, \dots, 63$, in the straightforward way: $K^{(0)} = K$. If $X = (x_{31}, \dots, x_0) \in V_{32}$, $x_j \in \mathrm{GF}(2)$, $j = 0, \dots, 31$, is a block of plaintext, the initial state of the text register is $Y^{(0)} = (x_{31}, \dots, x_0)$. The output of the algorithm is the ciphertext $Z = (z_{31}, \dots, z_0) = Y^{(528)} \in V_{32}$, $z_j \in \mathrm{GF}(2)$, $j = 0, \dots, 31$.

For decryption the key register is filled in the same way: $K^{(0)} = K = (k_{63}, \dots k_0) \in V_{64}$. But the decryption procedure complements the encryption. One decryption cycle can be defined by the following sequence of operations (see Figure 2):

Compute the feedback bit: $\varphi = NLF(y_{30}^{(i)}, y_{25}^{(i)}, y_{19}^{(i)}, y_{8}^{(i)}, y_{0}^{(i)}) \oplus y_{15}^{(i)} \oplus y_{31}^{(i)} \oplus k_{15}^{(i)}$
Rotate text and insert feedback: $R^{(i+1)} = (y_{30}^{(i)}, \dots, y_{0}^{(i)}, \varphi)$
Rotate key: $K^{(i+1)} = (k_{62}^{(i)}, \dots, k_{0}^{(i)}, k_{63}^{(i)})$.

The ciphertext and plaintext are input/output in a similar way: The ciphertext is input into the text register before decryption, $Y^{(0)} = Z$, and the plaintext can be read out after 528 decryption cycles, $Y^{(528)} = X$.

The NLF is a boolean function of 5 variables and is of degree 3. In the specification [14] the NLF is assigned using a table. The vector of outputs[1] is

[1] The least significant and most significant bits represent the value of $NLF(x_4, x_3, x_2, x_1, x_0)$ for $x_4 = x_3 = x_2 = x_1 = x_0 = 0$ and $x_4 = x_3 = x_2 = x_1 = x_0 = 1$, respectively.

`0x3A5C742E`. This corresponds to the following ANF:

$$\begin{aligned} NLF(x_4, x_3, x_2, x_1, x_0) = {} & x_0 \oplus x_1 \oplus \\ & x_0x_1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_0x_4 \oplus x_0x_3 \oplus x_2x_4 \oplus \\ & x_0x_1x_4 \oplus x_0x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4. \end{aligned} \tag{1}$$

The NLF is balanced and its correlation immunity order is 1, $\mathrm{cor}(NLF) = 1$ [29], [30]. This means that the NLF is 1-resilient [5], which is the maximum for a function of 5 variables with $\deg(NLF) = 3$ due to Siegenthaler's inequality [29]:

$$\deg(NLF) + \mathrm{cor}(NLF) \leq 4.$$

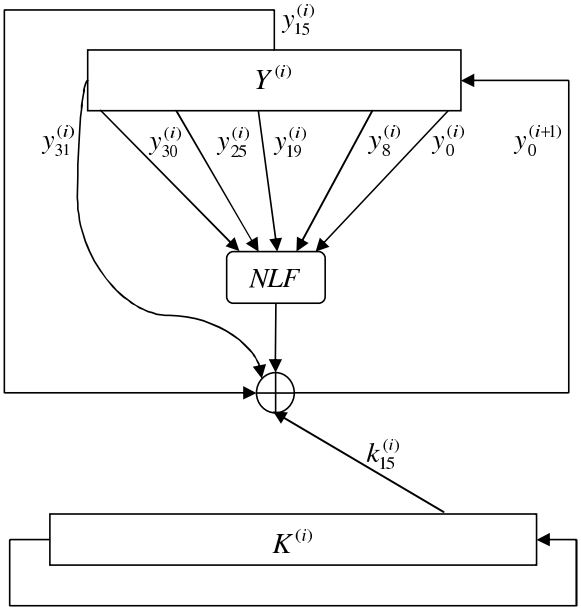

Fig. 2. The i-th KeeLoq decryption cycle

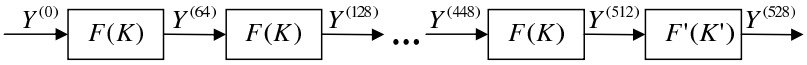

Fig. 3. Round structure of KeeLoq encryption

The KeeLoq algorithm has the following round structure. We define a KeeLoq round as the permutation $F(K) : V_{32} \to V_{32}$ depending on the key $K \in V_{64}$. A KeeLoq quarter round is defined as the permutation $F'(K') : V_{32} \to V_{32}$

depending on the subkey $K' = (k_{15}, \ldots, k_0) \in V_{16}$. Then the whole KeeLoq encryption mapping consists of successively computing 8 full round permutations $F(K)$ and consequently applying the last quarter round permutation $F'(K')$, see Figure 3. Note that the first 8 full rounds are identical. The decryption can be represented in a similar way using inverse permutations $F'(K')^{-1}$ and $F(K)^{-1}$.

The algorithm allows for an extremely simple hardware implementation comprised of a 32-bit shift register with taps on fixed positions, a 64-bit shift resister with a single tap and a 32-bit (5×1) look-up table (LUT) for the NLF. The LUT can be replaced with the corresponding logical elements according to (1).

3 Attack on the KeeLoq algorithm

Our attack is based on the following weaknesses of the algorithm:

- self-similar key schedule scheme,
- relatively short blocks of 32 bits,
- existence of an efficient linear approximation of the NLF.

The attack can be outlined in the following way. For each subkey $K' = (k_{15}, \ldots, k_0)$ and for a random 32-bit input $I_0 \in V_{32}$ guess the corresponding output $O_0 \in V_{32}$ after the 64 clock cycles which depends on the other 48 key bits $(k_{63}, \ldots, k_{16})$. Using the periodic structure of the KeeLoq key schedule generate several other pairs $(I_i, O_i) \in (V_{32})^2$, $i = 1, \ldots, N-1$ (*sliding step*). For a successful attack N has to be about 2^8. For each number of such pairs we mount a distinguishing attack to obtain linear relations on some unknown key bits with a high probability due to the fact that the KeeLoq NLF is not 2-resilient (*correlation step*). In this way it is possible to determine $(k_{47}, \ldots, k_{16})$ bit by bit. After this an input/output pair for 16 encryption cycles can be represented as a triangular system of linear equations with the remaining bits $(k_{63}, \ldots, k_{48})$ of K as variables. It can be solved using 16 simple computational operations (*linear step*).

3.1 Sliding step

Using a single input/output pair (I_0, O_0) for the full round of 64 cycles and knowing the first 16 key bits $K' = (k_{15}, \ldots, k_0)$ one can produce an arbitrary number of other input/output pairs for this round. This is possible due to the fact that (almost) all rounds in KeeLoq are identical permutations which is the property on which the slide attacks by Biryukov and Wagner are substantially based [3], [4]. Once a pair (I_i, O_i) is known, the next input/output pair is produced by encrypting I_i and O_i with the key to be recovered (it is a chosen plaintext attack) and obtaining (I'_{i+1}, O'_{i+1}) as ciphertext. Then I'_{i+1} and O'_{i+1} are decrypted using the guessed partial key $K' = (k_{15}, \ldots, k_0)$. The resulting plaintexts form the needed pair (I_{i+1}, O_{i+1}), see Figure 4.

As the sliding has to be performed for each guess of $K' = (k_{15}, \ldots, k_0)$ and each round output (2^{47} times on the average), its complexity is crucial for the efficiency of the whole attack. All input/output pairs for a given key can be generated in a pre-computation step using the chosen plaintext possibility and stored in memory of size 2^{32} 32-bit words.

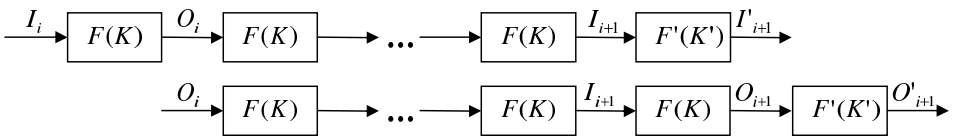

Fig. 4. Generating input/output pairs using sliding techniques

3.2 Correlation step

Once the needed number of pairs were found in the sliding step, the following weakness of the KeeLoq NLF with respect to correlation attacks is used due to the fact that the NLF is 1-resilient, but not 2-resilient.

Lemma 1. *For uniformly distributed $x_4, x_3, x_2 \in GF(2)$ the following holds:*

- $\Pr\{NLF(x_4, x_3, x_2, x_1, x_0) = 0 \mid x_0 \oplus x_1 = 0\} = \frac{5}{8}$,
- $\Pr\{NLF(x_4, x_3, x_2, x_1, x_0) = 1 \mid x_0 \oplus x_1 = 1\} = \frac{5}{8}$.

This means that the NLF can be efficiently approximated by $x_0 \oplus x_1$. So, if x_0, x_1 are known and x_4, x_3, x_2 are random and unknown, we can determine $f(K)$ by statistically filtering out the contribution of $NLF(x_4, x_3, x_2, x_1, x_0)$ to the equation

$$NLF(x_4, x_3, x_2, x_1, x_0) \oplus f(K) = 0$$

using a very limited number of such samples. $f(K)$ is a key-dependent boolean function remaining constant for all samples.

Here we show how to obtain k_{16} and k_{32} from I_i and O_i. The remaining key bits $(k_{47}, \ldots, k_{33})$ and $(k_{31}, \ldots, k_{17})$ can be obtain in the same way by using k_{32}, k_{16} and shifting input and output bits.

We denote $I_i = Y^{(0)}$ and $O_i = Y^{(64)}$ for each i. The idea is to make use of the correlation weakness of the dependency between the output bits $y_0^{(64)}$, $y_1^{(64)}$ and the input bits $Y^{(0)}$. One can compute $Y^{(16)}$ from $Y^{(0)}$, since $K' = (k_{15}, \ldots, k_0)$ is known. For the next bit $y_{16}^{(16)}$ which is the first key-dependent bit one has the following equation:

$$\begin{aligned} y_{16}^{(32)} = y_{31}^{(17)} &= NLF(y_{31}^{(16)}, y_{26}^{(16)}, y_{20}^{(16)}, y_9^{(16)}, y_1^{(16)}) \oplus y_0^{(16)} \oplus y_{16}^{(16)} \oplus k_{16} = \\ &= c_0 \oplus k_{16}, \end{aligned} \tag{2}$$

where $c_0 \in \mathrm{GF}(2)$ denotes the key-independent part of (2).

After 32 encryption cycles the following holds:

$$(y_{15}^{(32)}, y_{14}^{(32)}, \ldots, y_0^{(32)}) = (y_{31}^{(16)}, y_{30}^{(16)}, \ldots, y_{16}^{(16)}) \in V_{16}.$$

Thus, the least significant half of $Y^{(32)}$ is known. Then $y_0^{(64)}$ can be represented as:

$$\begin{aligned} y_0^{(64)} &= NLF(y_{31}^{(32)}, y_{26}^{(32)}, y_{20}^{(32)}, y_9^{(32)}, y_1^{(32)}) \oplus y_0^{(32)} \oplus y_{16}^{(32)} \oplus k_{32} = \\ &= NLF(y_{31}^{(32)}, y_{26}^{(32)}, y_{20}^{(32)}, y_9^{(32)}, y_1^{(32)}) \oplus y_0^{(32)} \oplus (c_0 \oplus k_{16}) \oplus k_{32}, \end{aligned} \tag{3}$$

where $y_0^{(64)}$, $y_0^{(32)}$, $y_1^{(32)}$, $y_9^{(32)}$, c_0 are known and $y_{31}^{(32)}$, $y_{26}^{(32)}$, $y_{20}^{(32)}$, k_{32}, k_{16} are unknown. As the first two inputs of the NLF are known, its contribution to (3) can be replaced with the random variate ε using Lemma 1:

$$NLF(y_{31}^{(32)}, y_{26}^{(32)}, y_{20}^{(32)}, y_9^{(32)}, y_1^{(32)}) \oplus y_9^{(32)} \oplus y_1^{(32)} = \varepsilon \tag{4}$$

with

$$\Pr\{\varepsilon = 0\} = \frac{5}{8}. \tag{5}$$

Then the following holds:

$$y_0^{(64)} \oplus y_0^{(32)} \oplus c_0 \oplus y_9^{(32)} \oplus y_1^{(32)} = \varepsilon \oplus k_{16} \oplus k_{32}. \tag{6}$$

In order to determine $k_{16} \oplus k_{32}$ one has to distinguish between the following two cases: $k_{16} \oplus k_{32} = 0$ and $k_{16} \oplus k_{32} = 1$. In the first case:

$$\Pr\{y_0^{(64)} \oplus y_0^{(32)} \oplus c_0 \oplus y_9^{(32)} \oplus y_1^{(32)} = 0\} = \frac{5}{8}.$$

Otherwise:

$$\Pr\{y_0^{(64)} \oplus y_0^{(32)} \oplus c_0 \oplus y_9^{(32)} \oplus y_1^{(32)} = 0\} = \frac{3}{8}.$$

Thus, the bias δ of the first random variable with respect to the second one is $\delta = \frac{1}{4}$. Our experiments show that about 2^7 equations (6) for different pairs (I_i, O_i), $i = 0, \ldots, 2^7 - 1$, are needed to recover $\alpha = k_{16} \oplus k_{32}$ with an acceptable error probability (for all 32 key-dependent linear combinations to be determined in this way), which agrees[2] with Theorem 6 of [1].

Next we consider $y_1^{(64)}$ and its dependencies from the input and key bits. Similar to (2) one has:

$$\begin{aligned} y_{16}^{(33)} &= NLF(y_{31}^{(17)}, y_{27}^{(16)}, y_{21}^{(16)}, y_{10}^{(16)}, y_2^{(16)}) \oplus y_1^{(16)} \oplus y_{17}^{(16)} \oplus k_{17} = \\ &= NLF(c_0 \oplus k_{16}, y_{27}^{(16)}, y_{21}^{(16)}, y_{10}^{(16)}, y_2^{(16)}) \oplus y_1^{(16)} \oplus y_{17}^{(16)} \oplus k_{17} = \\ &= c_1' \oplus c_2 k_{16} \oplus y_1^{(16)} \oplus y_{17}^{(16)} \oplus k_{17} = c_1 \oplus c_2 k_{16} \oplus k_{17}, \end{aligned} \tag{7}$$

where $c_1' \in \mathrm{GF}(2)$ is the free term of NLF, $c_2 \in \mathrm{GF}(2)$ is its linear term with respect to k_{16}, and $c_1 = c_1' \oplus y_1^{(16)} \oplus y_{17}^{(16)} \in \mathrm{GF}(2)$. Here c_1 and c_2 are known and depend on $Y^{(0)}$. Then the second output bit $y_1^{(64)}$ is represented as follows:

$$\begin{aligned} y_1^{(64)} &= NLF(y_{31}^{(33)}, y_{26}^{(33)}, y_{20}^{(33)}, y_9^{(33)}, y_1^{(33)}) \oplus y_0^{(33)} \oplus y_{16}^{(33)} \oplus k_{33} = \\ &= (\varepsilon \oplus y_9^{(33)} \oplus y_1^{(33)}) \oplus y_0^{(33)} \oplus (c_1 \oplus c_2 k_{16} \oplus k_{17}) \oplus k_{33}, \end{aligned} \tag{8}$$

where the random variate ε is assigned in a way similar to (4) and c_0, c_1, c_2, $y_0^{(33)}$, $y_9^{(33)}$, $y_1^{(33)}$ are known. To determine $k_{17} \oplus k_{33}$ pairs (I_i, O_i) with $c_2 = 0$ are

[2] Strictly speaking, the mentioned Theorem 6 cannot be applied here since Assumption 4 of [1] does not hold due to the fact that the mutual bias is relatively large in our case. But this suggests that our experimental estimations are correct.

selected[3]. Then ε in (8) is filtered out statistically, which recovers $\beta = k_{17} \oplus k_{33}$. After this the remaining pairs (I_i, O_i) (with $c_2 = 1$) are used to obtain $\gamma = k_{16} \oplus k_{17} \oplus k_{33}$ in the same way. Thus, $k_{16} = \beta \oplus \gamma$ and $k_{32} = \alpha \oplus k_{16}$.

Now k_{16}, k_{32} and $k_{17} \oplus k_{33}$ are known. In the next step we determine $k_{18} \oplus k_{34}$ and $k_{17} \oplus k_{18} \oplus k_{34}$ using the same plaintext/ciphertext pairs (I_i, O_i) and the same statistical recovery method. In this way all 32 key bits $(k_{47}, \ldots, k_{16})$ are obtained in only 16 rather simple computational steps.

3.3 Linear step and key verification

The remaining key bits $(k_{63}, \ldots, k_{48}) \in V_{32}$ can be recovered as follows. As $(k_{47}, \ldots, k_0)$ are known, $Y^{(48)}$ can be computed for each pair (I_i, O_i). $y_{16}^{(64)}$ can be expressed as:

$$y_{16}^{(64)} = NLF(y_{31}^{(48)}, y_{26}^{(48)}, y_{20}^{(48)}, y_{9}^{(48)}, y_{1}^{(48)}) \oplus y_{16}^{(48)} \oplus y_{0}^{(48)} \oplus k_{48}, \qquad (9)$$

which reveals k_{48} since the entire state $Y^{(48)}$ is known. Now $Y^{(49)}$ can be completely calculated which leads to the value of k_{49} using $y_{17}^{(64)}$, and so on. In this way the rest of the key is recovered.

At the end of the key recovery procedure we expect to obtain a number of key candidates. The upper bound for their average quantity is $2^{64-32} = 2^{32}$ due to the known plaintext unicity distance [12], since the block length is 32 bit and the key length is 64 bit. Thus, we need to verify each key candidate against max. $\lceil \frac{64+4}{32} \rceil = 3$ plaintext-ciphertext pairs for all 528 encryption cycles.

3.4 Attack complexity

The attack consists of the following stages:

- Precompute all plaintext-ciphertext pairs for the whole cipher;
- Guess the partial key K' and the output O_0 after one round for some input I_0;
- For each pair of guesses (K', O_0) do the following:
  - Obtain $2^8 - 1$ other pairs (I_i, O_i);
  - Determine $k_{16} \oplus k_{32}$ by evaluating c_0 for the first 2^7 pairs;
  - Determine $(k_{47}, \ldots, k_{16})$ by evaluating c_1 and c_2 2^8 times;
  - Determine $(k_{63}, \ldots, k_{48})$ by evaluating 2^4 nonlinear boolean functions;
  - Verify max. 2^{32} candidate keys using at most 3 plaintext-ciphertext pairs for the whole cipher and 3 full encryption operations.

If one step is equivalent to a single full KeeLoq encryption (528 encryption cycles), 2^{32} steps are needed for the precomputation. Each element has to be stored in a memory of 2^{32} 32-bit words.

For each guess of (I_0, O_0) and K' operations of the following complexity have to be be performed:

[3] Note that for random inputs I_i the probability of $c_2 = 0$ is 0.5. Therefore about $N/2$ out of N known pairs (I_i, O_i) will lead to $c_2 = 0$. This raises the required number of plaintext/ciphertext pairs to about 2^8.

- $2^9 - 2$ memory accesses for obtaining (I_i, O_i), $i = 1, \ldots, 2^8 - 1$. We assume a single memory access equivalent to 16 encryption cycles. This leads to approximately $2^9/32 = 2^4$ steps required to perform the memory accesses.
- 2^7 evaluations of c_0 and $k_{16} \oplus k_{32}$, each evaluation being equivalent to one encryption cycle. The complexity of this stage is then $2^7/528 \approx 2^{-2}$ steps.
- $16 \cdot 2^8 = 2^{12}$ evaluations of c_1 and c_2 for determining $(k_{47}, \ldots, k_{16})$. Each evaluation is computationally equivalent to 2 encryption cycles. This stage requires about $2^{12} \cdot 2/528 \approx 2^4$ steps.
- 2^4 evaluations of a boolean function to determine $(k_{63}, \ldots, k_{48})$. Each evaluation is equivalent to one encryption cycle which leads to a complexity of about $2^4 \cdot 2^{-9} = 2^{-5}$ steps.

Max. 2^{32} candidate keys have to be verified using at most 3 full encryptions which requires max. 2^{34} steps. Thus, the overall computational complexity of the attack is

$$2^{32} + \frac{2^{32} \cdot 2^{16}}{2} \cdot (2^4 + 2^{-2} + 2^4 + 2^{-5}) + 2^{34} \approx 2^{52} \text{ steps.}$$

The memory complexity is quite reasonable and is 2^{32} 32-bit words (16 GByte). This enables an attacker to place all precomputed values into RAM which substantially accelerates the implementation of the attack.

Most computations in our attack are perfectly parallelizable: The attacker can distribute the 2^{48} combinations of guesses between a number of computational machines (PCs, FPGAs, ASICs, etc.) sharing the 16 GBytes of memory containing the precomputed plaintext-ciphertext pairs.

Attacks of this type are potentially applicable to all NLFSR-based block cipher constructions of this kind. To avoid slide attacks some sort of round dependency should be introduced into the key schedule. Moreover, the correlation immunity order of the nonlinear feedback function has to be increased and the non-existence of efficient nonlinear approximations should be provided.

3.5 Experiments

We have implemented sliding, correlation and linear steps on a standard Pentium M PC with 1.73 GHz and 1 GB RAM for random keys. On the average we were able to recover the whole key in about 45000 tact cycles[4] for the correct guesses of (I_0, O_0) and K' as well as for the corresponding slid pairs (I_i, C_i). The implementation was performed in non-optimized C using Microsoft Visual C++ 7 compiler. Moreover, our experiments show that about 2^8 pairs (I_i, O_i) are required in average to recover the key correctly.

4 Attacks on KeeLoq-based systems in praxis

There are three major types of security protocols in which the KeeLoq block cipher is involved:

[4] A single encryption step on the same platform requires about 2200 tact cycles.

KeeLoq hopping codes: These are also known as rolling codes and provide authentication of an encoder to the decoder (the main system) by sending an encrypted counter value (unilateral communication), see [13]. The encoder and decoder share a 64-bit symmetric key and a 16-bit synchronized counter value. To authenticate itself the encoder encrypts the next counter value and sends it to the decoder which decrypts the message and verifies whether the received counter value is within the open window of length 16. A resynchronization mechanism exists to repair communication in case the counter value received exceeds the bounds of the open window. See also [15], [16], [18].

The KeeLoq hopping codes do not allow one to apply our attack since there are only 2^{16} possible ciphertexts, 16 bits of the plaintext being fixed for a given encoder.

KeeLoq IFF: The IFF (Identify Friend or Foe) systems provide authentication of a transponder to the main systems (decoder) using a simple challenge-response protocol (bilateral communication), see [19]. The transponder and decoder share a 64-bit symmetric key K. To require authentication the decoder sends a 32-bit random challenge to the transponder which should reply with the KeeLoq encrypted challenge using K. The decoder encrypts the genuine challenge using K and compares the message received as a reply with this value. See also [17].

The KeeLoq IFF systems are vulnerable to our attack since all input/output pairs are available to the attacker. Moreover, the key management schema of KeeLoq IFF [17] allows one to determine 4 bits of a 64-bit master key, which is the same for large series of encoders, by applying our attack to find a single individual key. This reduces the security of the IFF application to at least 2^{60}. Moreover, if some other KeeLoq key management schemata [13], [15], [16], [18] are used in KeeLoq IFF, 32 or 16 bits of the master key can be determined by finding a single individual key using our attack with complexity of 2^{52}. This would reduce the security of the rest of the encoders in the series to 2^{32} or 2^{48}, respectively.

PIC12F635/PIC16F636/639: Proprietary protocols based on PIC-controllers PIC12F635/PIC16F636/639 [21], [20] supplied by Microchip which are equipped with a hardware module implementing the KeeLoq block cipher.

The customized protocols using the KeeLoq block cipher with or without PIC12F635/PIC16F636/639 are potentially vulnerable to practical attacks on the algorithm and the underlying key management system, since the cipher only provides at most 52-bit security level due to our attack.

5 Conclusion

In this paper we proposed a practical key-recovery attack on the KeeLoq block cipher used in numerous automotive applications as well as in various property identification systems. To out best knowledge this is the first paper to describe and successfully cryptanalyze KeeLoq. We showed that several real-world

applications are vulnerable to our attack, including KeeLoq IFF systems and customized applications using the KeeLoq block cipher.

Our attack is a combination of guess-and-determine, sliding and distinguishing techniques. We first guess an input/output pair for one round and then use sliding to obtain more such pairs to be able to exploit some linearity of the nonlinear feedback function. The attack works with complexity of 2^{52} steps (while KeeLoq uses a 64-bit key), one step being equivalent to a single KeeLoq encryption. It requires all 2^{32} plaintexts and a memory of 2^{32} 32-bit words. The complexity does not depend on the number of encryption cycles (iterations). The attack is very well parallelizable.

References

1. T. Baigneres, P. Junod, and S. Vaudenay. How Far Can We Go Beyond Linear Cryptanalysis? In *Proc. of ASIACRYPT'04*, volume 3329 of *LNCS*. Springer-Verlag, 2004.
2. C. Berbain, H. Gilbert, and A. Maximov. Cryptanalysis of Grain. In *Proc. of FSE'06*, volume 4047 of *LNCS*. Springer-Verlag, 2006.
3. A. Biryukov and D. Wagner. Slide Attacks. In *Proc. of FSE'99*, volume 1636 of *LNCS*. Springer-Verlag, 1999.
4. A. Biryukov and D. Wagner. Advanced Slide Attacks. In *Proc. of EUROCRYPT'00*, volume 1807 of *LNCS*. Springer-Verlag, 2000.
5. B. Chor, O. Goldreich, J. Hastad, J. Fridman, S. Rudich, and R. Smolensky. The Bit Extraction Problem or t-Resilient Functions. In *26th Symposium on Foundations of Computer Science*, 1985.
6. ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms, Document 2: KASUMI Specification. Available from `http://portal.etsi.org/dvbandca/3GPPSPECIFICATIONS/3GTS35.202.pdf`, 1999.
7. B. Gammel, R. Goettfert, and O. Kniffler. Achterbahn-128/80. Available from `http://www.ecrypt.eu.org/stream/p2ciphers/achterbahn/achterbahn_p2.pdf`, June 2006.
8. B. M. Gammel, R. Goettfert, and O. Kniffler. The Achterbahn Stream Cipher. Available from `http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf`, April 2005.
9. M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. Available from `http://www.ecrypt.eu.org/stream/ciphers/grain/grain.pdf`, 2005.
10. HomeLink. Homelink and KeeLoq-based Rolling Code Garage Door Openers. Available from `http://www.homelink.com/home/keeloq.tml`, 2006.
11. T. Johansson, W. Meier, and F. Muller. Cryptanalysis of Achterbahn. In *Proc. of FSE'06*, volume 4047 of *LNCS*. Springer-Verlag, 2006.
12. A. Menezes, P. van Oorshot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
13. Microchip. An Introduction to KeeLoq Code Hopping. Available from `http://ww1.microchip.com/downloads/en/AppNotes/91002a.pdf`, 1996.
14. Microchip. Hopping Code Decoder using a PIC16C56, AN642. Available from `http://en.wikipedia.org/wiki/KeeLoq` and `http://www.keeloq.boom.ru/decryption.pdf`, 1998.

15. Microchip. HCS101 Fixed Code Encoder Data Sheet. Available from http://ww1.microchip.com/downloads/en/DeviceDoc/41115c.pdf, 2001.
16. Microchip. HCS301 KeeLoq Code Hopping Encoder Data Sheet. Available from http://ww1.microchip.com/downloads/en/devicedoc/21143b.pdf, 2001.
17. Microchip. HCS410 Keeloq Code Hopping Encoder and Transponder. Available from http://ww1.microchip.com/downloads/en/DeviceDoc/40158e.pdf, 2001.
18. Microchip. HCS301 KeeLoq Code Hopping Encoder Data Sheet. Available from http://ww1.microchip.com/downloads/en/devicedoc/21143b.pdf, 2002.
19. Microchip. Using KeeLoq to Validate Subsystem Compatibility, AN827. Available from http://ww1.microchip.com/downloads/en/AppNotes/00827a.pdf, 2002.
20. Microchip. PIC12F635/PIC16F636/PIC16F639 Cryptographic Module General Overview, TB086. Available from http://ww1.microchip.com/downloads/en/DeviceDoc/91086A.pdf, 2005.
21. Microchip. PIC12F635/PIC16F636/639 Data Sheet. Available from http://ww1.microchip.com/downloads/en/DeviceDoc/41232B.pdf, 2005.
22. NIST. Data Encryption Standard (DES). FIPS PUB 46-3.
23. NIST. Skipjack and kea algorithm specifcations, 1998. Version 2.0, 29 May 1998, Available from http://csrc.nist.gov/encryption/skipjack-kea.htm.
24. M. N. Plasencia. Cryptanalysis of Achterbahn-128/80. Available from http://www.ecrypt.eu.org/stream/papersdir/2006/055.pdf, November 2006.
25. R. L. Rivest. The RC5 Encryption Algorithm. In *Proc. of FSE'94*, volume 1073 of *LNCS*. Springer-Verlag, 1994.
26. B. Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher Blowfish. In *Proc. of FSE'94*, volume 809 of *LNCS*. Springer-Verlag, 1994.
27. B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2nd Ed. edition, 1996.
28. B. Schneier and J. Kelsey. Unbalanced Feistel Networks and Block-Cipher Design. In *Proc. of FSE'96*, volume 1039 of *LNCS*. Springer-Verlag, 1996.
29. T. Siegenthaler. Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Trans. on Inform. Theory*, IT-30(5), 1984.
30. T. Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Trans. on Computers*, 34(1), 1985.
31. Wikipedia. Keeloq algorithm. Available from http://en.wikipedia.org/wiki/KeeLoq, November 2006.
32. I. A. Zabotin, G.P. Glazkov, and V.B. Isaeva. Cryptographic Protection for Information Processing Systems, Cryptographic Transformation Algorithm. Government Standard of the USSR, GOST 28147-89, 1989. (Translated by A. Malchik, with editorial and typograpihc assistance of W. Diffie).