Cryptanalysis of KeeLoq with COPACOBANA

Martin Novotný[1] and Timo Kasper[2]

[1] Faculty of Information Technology
Czech Technical University in Prague
Kolejní 550/2
160 00 Praha 6, Czech Republic
email: novotnym@fit.cvut.cz
[2] Embedded Security Group
Ruhr-University Bochum
Universitätsstrasse 150,
44801 Bochum, Germany
email: tkasper@crypto.rub.de

Abstract. Many real-world car door systems and garage openers are based on the KeeLoq cipher. Recently, the block cipher has been extensively studied. Several attacks have been published, including a complete break of a KeeLoq access control system. It is possible to instantly override the security of all KeeLoq code-hopping schemes in which the secret key of a remote-control is derived from its serial number. The latter can be intercepted from the communication between a receiver and a transmitter. In contrast, if a random SEED is used for the key derivation, the cryptanalysis demands for higher computation power and may become infeasible with a standard PC.
In this paper we develop a hardware architecture for the cryptanalysis of KeeLoq. Our brute-force attack, implemented on the Cost-Optimized Parallel Code-Breaker COPACOBANA, is able to reveal the secret key of a remote control in less than 0.5 seconds if a 32-bit seed is used and in less than 6 hours in case of a 48-bit seed. To obtain reasonable cryptographic strength against this type of attack, a 60-bit seed has to be used, for which COPACOBANA needs in the worst case about 1011 days for the key recovery. However, the attack is arbitrarily parallelizable and could thus be run on multiple COPACOBANAs to decrease the attack time.

Keywords: KeeLoq, COPACOBANA, cryptanalysis

1 Introduction

Electronic car or garage opening systems consist of remote controls, which replace traditional keys, and receivers which control the door.

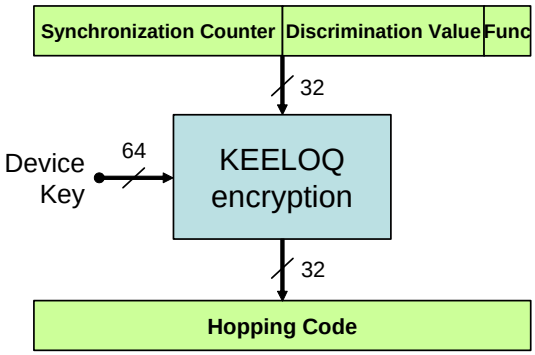


Fig. 1: KeeLoq encryption.

On having its button pressed a remote sends a hopping code to the receiver to open or close the door. A hopping code is generated by a KeeLoq encryption incorporating a 16-bit counter value, a 12-bit discrimination value and a 4-bit function value, as shown in Figure 1. While the counter is incremented in the remote each time a hopping code is generated, the discrimination and function values remain constant.

To obtain the device key on the side of the receiver, the serial number of the remote is either decrypted with a manufacturer key or xored with the manufacturer key, as shown in Figure 2 and in Figure 3a. Alternatively, a randomly generated seed value may by combined with the serial number for the key derivation. For the latter, Microchip proposes three scenarios: a) 28 bits of the serial number (N) are combined with 32 bits of the random seed (S) according to the pattern 0x0NNNNNNNSSSSSSSS (Scenario 2 in Figure 3b), b) 12 bits of the serial number are combined with 48 bits of the seed in the pattern 0x0NNNSSSSSSSSSSSS (Scenario 3 in Figure 3c), c) 60 bits of the seed in the pattern 0x0SSSSSSSSSSSSSSS (Scenario 4 in Figure 3d).

Since the KeeLoq cipher has been extensively studied [1], [2], [3], several different types of attack have been proposed. The attack described in [3] reveals the manufacturer key by means of power analysis. As the manufacturer key is shared by all devices of the same producer and since many commercial products derive the device keys from their serial numbers only (without using a seed), breaking the

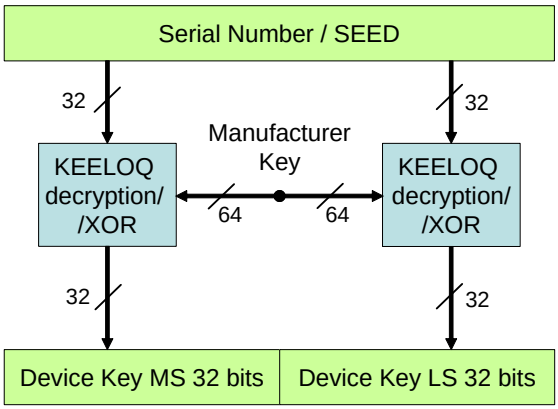


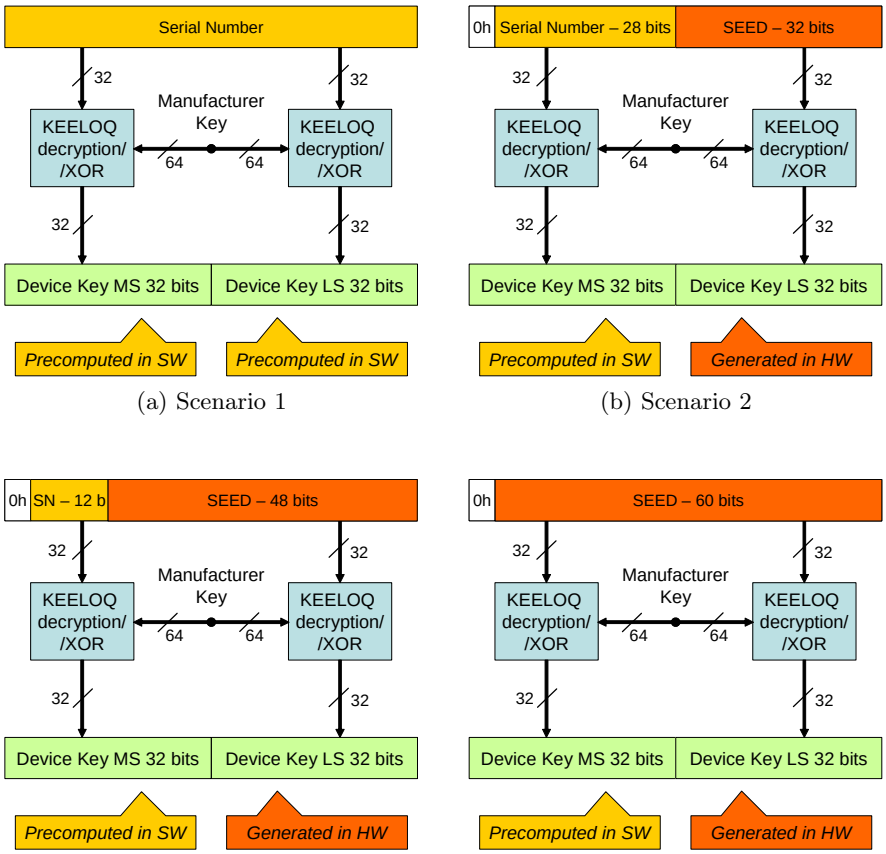
Fig. 2: Device key generation.

system is straightforward — the serial number is intercepted from the communication between the remote and the receiver, and the secret key of the remote is derived (Scenario 1 in Figure 3a).

The goal of this work is finding the correct Device Key when random seed is used for device key generation (Scenarios 2 through 4 in Figure 3). As illustrated in Figure 2, the 32 most significant bits (MSB) of the device key are derived from the higher 32 bits of the input value, while the lower 32 bits are generated from the lower 32 bits of the input. If a random seed is used, lower 32 bits of the device key are always random, while upper 32 bits may have either a fixed value (Scenario 2), or one of 2^{16} potential values (Scenario 3), or one of 2^{28} potential values (Scenario 4). Consequently, when implementing a brute-force attack, each combination of 32 MSBs of the device key may be precomputed in software and then combined with all 2^{32} combinations of 32 LSBs (generated in hardware by a counter), until the correct value of the device key is found.

2 KeeLoq Breaker

To break the cipher we need to intercept two hopping codes of the same device, generated from the same device key. Such hopping codes are generated from identical discrimination and function values, but from different counter values (see Figure 1). However, the difference



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Fig. 3: Scenarios for device key generation.

between the counter values will be small, if the two consecutive (or almost consecutive) hopping codes are intercepted.

We implemented a brute-force attack on KeeLoq on the parallel computation cluster COPACOBANA [4]. This cluster has been designed to support cryptanalytical calculations. The cluster is equipped with 120 low-cost Xilinx Spartan3-1000 FPGAs, which communicate with the host computer via the controller board. Note, that it is possible to employ several COPACOBANAs in order to further increase the performance.

The diagram of the circuit implemented in each FPGA is shown in Figure 4. A candidate for the device key is found by means of

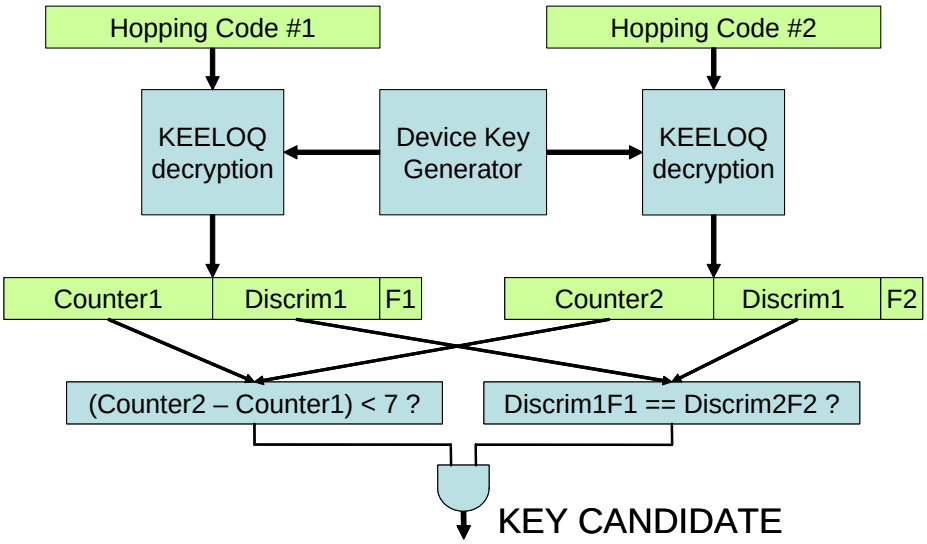


Fig. 4: KeeLoq breaker.

exhaustive key-search, if the decryptions of two intercepted hopping codes reveal identical discrimination values and moderately increased counter values.

The core of the implementation is a *Device Key Generator* consisting of a 32-bit register and a 32-bit counter. The register holds 32 MSBs of the device key (being precomputed in software and assigned by the host computer), while the counter is repeatedly increased to generate all possible values for the lower 32 bits of the device key. If all counter-values have been generated, and no key candidate has been found, the FPGA is assigned with the new value of upper 32 bits of the key.

A KeeLoq decryption is executed in 132 rounds. In our optimized implementation we unrolled both decryption units into a pipeline structure. Each path of the pipeline consists of 176 stages, i.e., each stage contains 4 rounds of the cipher (the number of stages was limited by available resources). The KeeLoq breaker occupies 6423 out of 7680 slices (83%) of the Xilinx Spartan 3-1000 FPGA. The maximum achievable clock frequency for the COPACOBANA was 110 MHz, i.e., each FPGA can test up to 110 million keys per second.

SEED length (bits)	1 FPGA (< 80 $)	1 COPACOBANA (< 10000 $)	100 COPACOBANAs (< 1000000 $)
32	39 secs	0.33 secs	3.3 msecs
48	29.6 days	5.9 hours	213 secs
60	332 years	1011 days	10.1 days

Table 1: Worst case times for the brute force attack on KeeLoq

3 Results and Conclusions

When a 32-bit seed is used, up to 2^{32} potential values of the device key need to be tested, in order to find the correct one. This takes $\frac{2^{32}}{120\times110\cdot10^6} \approx 0.33$ seconds on one COPACOBANA in the worst case. Finding the correct device key in case of a 48-bit seed takes up to $\frac{2^{48}}{120\times110\cdot10^6}$ seconds ≈ 5.9 hours on one COPACOBANA. For the 60-bit seed we need up to $\frac{2^{60}}{120\times110\cdot10^6}$ seconds ≈ 1011 days on one COPACOBANA. The attack is arbitrarily parallelizable and could thus be run on multiple COPACOBANAs to decrease the attack time. Worst case times for all possible seed lengths, and 1 FPGA, 1 COPACOBANA and 100 COPACOBANAs, respectively, are summarized in Table 1.

We conclude that using a 32-bit seed provides no security, since a key can be found in real-time. While a seed with 48 bits can be broken in less than 6 hours by one COPACOBANA, employing a 60-bit seed can provide reasonable security.

References

1. A. Bogdanov, "Attacks on the KeeLoq Block Cipher and Authentication Systems," in *3rd Conference on RFID Security 2007 (RFIDSec 2007)*, 2007. [Online]. Available: http://rfidsec07.etsit.uma.es/slides/papers/paper-22.pdf.
2. S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel, "A Practical Attack on KeeLoq," in *Advances in Cryptology - EUROCRYPT 2008*, 2008.
3. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani, "On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme." in *Advances in Cryptology - CRYPTO 2008*, 2008, pp. 203–220.
4. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, "Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker," in *Proceedings of CHES'06*, ser. LNCS, vol. 4249. Springer-Verlag, 2006, pp. 101–118.