Chapter 2
The Block Cipher Keeloq and Algebraic Attacks

The purpose of this chapter is to supply a (relatively) new, feasible, and economically relevant example of algebraic cryptanalysis. The block cipher "Keeloq"[1] has been used in the remote keyless-entry system of many automobiles. It has a secret key consisting of 64 bits, takes a plaintext of 32 bits, and outputs a ciphertext of 32 bits. The cipher consists of 528 rounds. In this chapter, we define the cipher. We also show some "frontal assaults" that are not effective. In the next chapter, we describe a successful attack from the author's dissertation [31, Ch. 2]. Our attack is faster than brute force by a factor of around $2^{14.77}$ as shown in Section 3.5 on Page 24. A summary of the attack is given in Section 3.6 on Page 24.

Many other attacks on Keeloq are known, discovered since this attack of mine was first written back in January of 2007. In fact, it seems clear from the dates of publication of other attacks, that work on Keeloq was simultaneous among the several research teams involved (see Section 3.8.1 on Page 27). The purpose here is not to describe all or even some of the attacks on Keeloq but to give the reader a non-trivial but straight-forward example of algebraic cryptanalysis succeeding on a real-world cipher.

Notational Convention

For any ℓ-bit sequence, the least significant bit is numbered 0 and the most significant bit is numbered $\ell - 1$.

[1] This is to be pronounced "key lock." Some authors typeset it "KeeLoq" but we find the presence of a capital letter in the middle of a word to be offensive to the eye.

G.V. Bard, *Algebraic Cryptanalysis,* DOI: 10.1007/978-0-387-88757-9_2

2.1 What is Algebraic Cryptanalysis?

Given a particular cipher, algebraic cryptanalysis consists of two steps. First, one must convert the cipher and possibly some supplemental information (e.g. file formats) into a system of polynomial equations, usually over $\mathbb{GF}(2)$, but sometimes over other rings. Second, one must solve the system of equations and obtain from the solution the secret key of the cipher. This chapter deals with the first step only. The systems of equations were solved with SINGULAR [9], MAGMA [2], and with the SAT-solver techniques of Chapter 13, as well as ElimLin, an algorithm by Nicolas Courtois described in Section 12.5 on Page 219.

2.1.1 The CSP Model

In any constraint satisfaction problem, there are several constraints in several variables. A solution must satisfy all constraints, and there might be zero, one, or more than one solution. The constraints are models of a cipher's operation, representing known facts as equations. Most commonly, this includes μ plaintext-ciphertext pairs, $P_1,\ldots,P_\mu$ and $C_1,\ldots,C_\mu$, and the μ facts: $E(P_i) = C_i$ for all $i \in \{1,\ldots,\mu\}$. The key is represented by one unknown for each bit. Almost always there are additional constraints and variables besides these.

If no false assumptions are made, then because these messages were indeed sent, we know there must be some key that was used, and so at least one key satisfies all the constraints. And so it is either the case that there are one, or more than one solution. Generally, algebraic cryptanalysis consists of writing enough constraints to reduce the number of possible keys to one, and few enough constraints that the system is solvable in a reasonable amount of time. In particular, the entire process should be faster than brute force by some margin.

2.2 The Keeloq Specification

In Figure 2.1 on Page 11, the diagram for Keeloq is given. The top rectangle is a 32-bit shift-register. It initially is filled with the plaintext. At each round, it is shifted one bit to the right, and a new bit is introduced. The computation of this new bit is the heart of the cipher.

Five particular bits of the top shift-register are tapped and are interpreted as a 5-bit integer, between 0 and 31. Then a non-linear function is applied, which will be described shortly (denoted NLF).

Meanwhile the key is placed initially in a 64-bit shift-register, which is also shifted one bit to the right at each iteration. The new bit introduced at the left is the bit formerly at the right, and so the key is merely rotating.

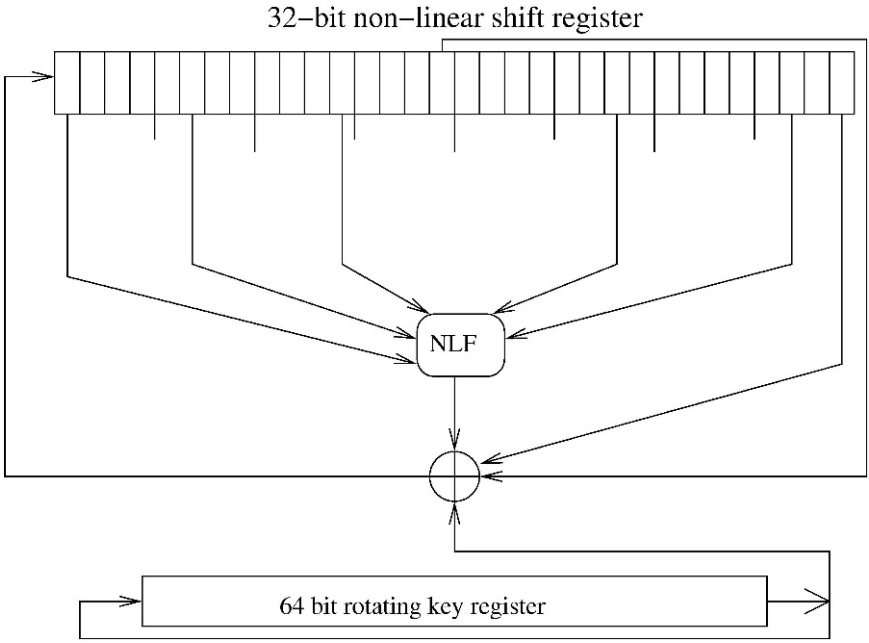

Fig. 2.1 The Keeloq Circuit Diagram

The least significant bit of the key, the output of the non-linear function, and two particular bits of the 32 bit shift-register are XORed together (added in $\mathbb{GF}(2)$). The 32-bit shift-register is shifted right and the sum is now the new bit to be inserted into the leftmost spot in the 32-bit shift-register.

After 528 rounds, the contents of the 32 bit shift-register form the ciphertext. Observe that only one bit of the key is used per round.

2.3 Modeling the Non-linear Function

The non-linear function $NLF(a,b,c,d,e)$ is denoted $NLF_{\texttt{3A5C742E}}$. This means that if (a,b,c,d,e) is viewed as an integer i between 0 and 31, i.e. as a 5-bit number, then the value of $NLF(a,b,c,d,e)$ is the ith bit of the 32-bit hexadecimal value `3A5C742E`. Recall, the least significant bit is numbered as 0.

The following formula is a cubic polynomial and gives equivalent output to the NLF for all input values, and was obtained by a Karnaugh map. In this case, the Karnaugh map is a grid with (for five dimensions) two variables in rows (i.e. 4 rows), and three variables in columns (i.e. 8 columns). The rows and columns are arranged via the Gray Code. This is a simple technique to rapidly arrive at the algebraic normal form (i.e. polynomial), listed below, by first trying to draw boxes around

regions of ones of size 32, 16, 8, 4, 2, and finally 1. See a text such as [38, Ch. 3] for details.

$$NLF(a,b,c,d,e) = d + e + ac + ae + bc + be + cd + de + ade + ace + abd + abc$$

2.3.1 I/O Relations and the NLF

Also note that while the degree of this function is 3, there is an I/O relation of degree 2, below. An I/O relation is a polynomial in the input variables and output variables of a function, such that no matter what values are given for input to the function, the I/O relation always evaluates to zero. Note y signifies the output of the non-linear function.

$$(e + b + a + y)(c + d + y) = 0$$

This can be thought of as a constraint that the function must always satisfy. If there are enough of these, then the function is uniquely defined. What makes them cryptanalyticly interesting is that the degree of the I/O relations can be much lower than the degree of the function itself. Since the degree dramatically impacts the difficulty of solving the polynomial system, this is very useful. The I/O degree of a function is the lowest degree of any of its I/O relations, other than the zero polynomial.

Generally, I/O-relations of low degree can be used for generating attacks but that is not the case here, because we have only one relation. Heuristically, relations that are valid with low probability for a random function and random input produce a more rapid narrowing of the keyspace in the sense of a Constraint Satisfaction Problem or CSP. We are unaware of any attack on Keeloq that uses this I/O-relation.

An example of the possibility of using I/O degree to cryptanalytic advantage is the attack from the author's joint paper on DES (the Data Encryption Standard), with Nicolas T. Courtois, where the S-Boxes have I/O degree 2 but their actual closed-form formulas are of higher degree [76].

2.4 Describing the Shift-Registers

The shift-registers that appear in Keeloq are an example of a very common, almost ubiquitous circuit design block in both stream ciphers (see Section 5.1.1.1 on Page 55) and block ciphers.

2.4.1 Disposing of the Secret Key Shift-Register

The 64-bit shift-register containing the secret key rotates by one bit per round. Only one bit per round (the rightmost) is used during the encryption process. Furthermore, the key is not modified as it rotates. Therefore, the key bit being used is the same in round $t, t+64, t+128, t+192, \ldots$

Therefore we can dispose of the key shift-register entirely. Denote $k_{63}, \ldots, k_0$ the original secret key. The key bit used during round t is merely $k_{t-1 \bmod 64}$.

2.4.2 Disposing of the Plaintext Shift-Register

Denote the initial condition of this shift-register as $L_{31}, \ldots, L_0$. This corresponds to the plaintext $P_{31}, \ldots, P_0$. Then in round 1, the values will move one place to the right, and a new value will enter in the first bit. Call this new bit L_{32}. Thus the bit generated in the ith round will be L_{31+i}, and in the 528th and thus last round will be L_{559}. The ciphertext is the final condition of this shift-register, which is $L_{559}, \ldots, L_{528} = C_{31}, \ldots, C_0$, likewise $L_{31}, \ldots, L_0 = P_{31}, \ldots, P_0$.

Change of Indexing

A change of indexing is useful here. The computation of L_i, for $32 \leq i \leq 559$, occurs during the round numbered $t = i - 31$. Thus the key bit used during the computation of L_i is $k_{i-32 \bmod 64}$.

2.5 The Polynomial System of Equations

This now gives rise to the following system of equations.

$$\begin{aligned} L_i &= P_i && \forall i \in [0,31] \\ L_i &= k_{i-32 \bmod 64} + L_{i-32} + L_{i-16} \\ &\quad + NLF(L_{i-1}, L_{i-6}, L_{i-12}, L_{i-23}, L_{i-30}) && \forall i \in [32,559] \\ C_{i-528} &= L_i && \forall i \in [528,559] \end{aligned}$$

Note, some descriptions of the cipher omit the $+L_{i-16}$. This should have no impact on the attack at all. Most research papers and the specification given by the company [94] includes the $+L_{i-16}$, as does our diagram; some early papers omitted it.

Since the NLF is actually a cubic function this is a cubic system of equations. Substituting, we obtain

$$
\begin{aligned}
L_i &= P_i && \forall i \in [0,31] \\
L_i &= k_{i-32 \bmod 64} + L_{i-32} + L_{i-16} + L_{i-23} + L_{i-30} \\
&\quad + L_{i-1}L_{i-12} + L_{i-1}L_{i-30} + L_{i-6}L_{i-12} + L_{i-6}L_{i-30} && \forall i \in [32,559] \\
&\quad + L_{i-12}L_{i-23} + L_{i-23}L_{i-30} + L_{i-1}L_{i-23}L_{i-30} \\
&\quad + L_{i-1}L_{i-12}L_{i-30} + L_{i-1}L_{i-6}L_{i-23} + L_{i-1}L_{i-6}L_{i-12} \\
C_{i-528} &= L_i && \forall i \in [528,559]
\end{aligned}
$$

In other words, the above equations are to be repeated for each i in the stated intervals, and for each of μ total plaintext-ciphertext message pairs. In practice, $\mu \geq 2$ was both necessary and sufficient.

2.6 Variable and Equation Count

Consider a plaintext-ciphertext pair $\mathbf{P}, \mathbf{C}$. There are 560 equations, one for each L_i, with $i \in [0,559]$, plus another 32 for the C_i, with $i \in [0,32]$. However, the first 32 of these are of the form $L_i = P_i$ for $i \in [0,32]$, and the last 32 of these are of the form $L_i = C_{i-528}$ for $i \in [528,559]$. Thus we can use string substitution and drop down to 528 equations. This is precisely one equation for each round, which defines the new bit introduced into the shift register.

The 64 bits of the key are unknown. Also, of the 560 L_i, the first and last 32 are known, but the inner 496 are not. This yields 560 unknowns. If there are μ plaintext-ciphertext message pairs, then there are 528μ equations. However, there are only $496\mu + 64$ variables, because the key does not change from pair to pair.

2.7 Dropping the Degree to Quadratic

The following is a specific application of the more general technique found in Section 11.4 on Page 192. We will change the system from cubic to quadratic by introducing a few variables. Instead of the previously derived

$$NLF(a,b,c,d,e) = d + e + ac + ae + bc + be + cd + de + ade + ace + abd + abc$$

one can write

$$
\begin{aligned}
NLF(a,b,c,d,e) &= d + e + ac + \beta + bc + be + cd + de + d\beta + c\beta + \alpha d + \alpha c \\
\alpha &= ab \\
\beta &= ae
\end{aligned}
$$

Since the non-linear function was the sole source of non-linear terms, this gives rise to a quadratic rather than cubic system of equations.

This introduces two new variables per original equation, and two new equations as well. Thus m equations and n variables becomes $3m$ equations and $n+2m$ variables. Therefore with μ plaintext-ciphertext message pairs, we have 1584μ equations and $1552\mu+64$ variables. Thus, it must be the case that $\mu > 1$ for the system to be expected to have at most one solution. As always with algebraic cryptanalysis, unless we make an assumption that is false, we always know the system of equations has at least one solution, because a message was indeed sent. And thus we very strongly expect to have a unique solution when $\mu > 1$.

$$
\begin{aligned}
L_i &= P_i && \forall i \in [0,31] \\
L_i &= k_{i-32 \bmod 64} + L_{i-32} + L_{i-16} + L_{i-23} + L_{i-30} \\
&\quad + L_{i-1}L_{i-12} + \beta_i + L_{i-6}L_{i-12} + L_{i-6}L_{i-30} + L_{i-12}L_{i-23} \\
&\quad + L_{i-23}L_{i-30} + \beta_i L_{i-23} + \beta_i L_{i-12} + \alpha_i L_{i-23} + \alpha_i L_{i-12} && \forall i \in [32,559] \\
\\
\alpha_i &= L_{i-1}L_{i-6} && \forall i \in [32,559] \\
\beta_i &= L_{i-1}L_{i-30} && \forall i \in [32,559] \\
C_{i-528} &= L_i && \forall i \in [528,559]
\end{aligned}
$$

Even with $\mu = 2$ this comes to 3168 equations and 3168 unknowns, well beyond the threshold of size for feasible polynomial system solving at the time this book was written, in late 2007.

2.8 Fixing or Guessing Bits in Advance

This is a specific application of the guess-and-determine methodology which is very common in many forms of cryptanalysis. For a general discussion, see Section 11.7 on Page 206, Section 12.4.4 on Page 217, or Section 11.7.2 on Page 207.

Sometimes in Gröbner basis algorithms or the XL algorithm, one fixes bits in advance [71, et al]. For example, in $\mathbb{GF}(2)$, there are only two possible values. Thus if one designates g particular variables, there are 2^g possible settings for them, but one needs to try $2^g/2$ on average (if we know exactly one solution exists). Naturally, fewer guesses would be needed if we merely search for one of many solutions. For each guess, one rewrites the system of equations either by substituting the guessed values, or if not, then by adding additional equations of the form: $k_1 = 1, k_2 = 0, \ldots$. If the resulting Gröbner Bases method or XL method running time is more than $2^g/2$ times faster as a result of this change, this is a profitable change.

In academic cryptanalysis however, one generates a key, encrypts μ messages, and writes equations based off of the plaintext-ciphertext pairs and various other constraints and facts. Therefore, one knows the key. Instead of guessing all 2^g possible values, we simply guess correctly. However, two additional steps must be required. First, we must adjust the final running time by a factor of 2^g in the worse-case, or $2^g/2$ in the average case. Second, we must ensure that the system identifies a wrong guess as fast, or faster, than solving the system in the event of a correct guess. See Section 11.7.1 on Page 206 for more details on this point.

2.9 The Failure of a Frontal Assault

First we tried a simple CSP. With μ plaintext messages under one key, for various values of μ we encrypted and obtained ciphertexts, and wrote equations as described already, in Section 2.7 on Page 15. We also used fewer rounds than 528, to see the impact of the number of rounds, as is standard. The experiments were an obvious failure, and so we began to look for a more efficient attack, presented in the next chapter. Note, the computer involved was a 1 GHz Intel PC with 1 gigabyte of RAM.

- With 64 rounds, and $\mu = 4$, and 10 key bits guessed, SINGULAR required 70 seconds, and ElimLin in 10 seconds.
- With 64 rounds, and $\mu = 2$ but the two plaintexts differing only in one bit (the least significant), SINGULAR required 5 seconds, and ElimLin 20 seconds. MINISAT[6] [104], using the techniques of Chapter 13, required 0.19 seconds. Note, it is natural that these attacks are faster, because many internal variables during the encryption will be identically-valued for the first and second message.
- With 96 rounds, $\mu = 4$, and 20 key bits guessed, MINISAT and the techniques of Chapter 13, required 0.3 seconds.
- With 128 rounds, and $\mu = 128$, with a random initial plaintext and each other plaintext being an increment of the previous, and 30 key bits guessed, ElimLin required 3 hours.
- With 128 rounds, and $\mu = 2$, with the plaintexts differing only in the least significant bit, and 30 key bits guessed, MINISAT requires 2 hours.

These results on 128 rounds are slower than brute-force. Therefore we did not try any larger number of rounds or finish trying each possible combination of software and trial parameters. Needless to say the 528 round versions did not terminate. Therefore, we needed a new attack, and this is the topic of the next chapter.

http://www.springer.com/978-0-387-88756-2

Algebraic Cryptanalysis
Bard, G.
2009, XXXIII, 356 p., Hardcover
ISBN: 978-0-387-88756-2