Algebraic Analysis of the Simon Block Cipher Family

Håvard Raddum

Simula Research Laboratory, Norway

Abstract. This paper focuses on algebraic attacks on the Simon family of block ciphers. We construct equation systems using multiple plaintext/ciphertext pairs, and show that many variables in the cipher states coming from different plaintexts are linearly related. A simple solving algorithm exploiting these relations is developed and extensively tested on the different Simon variants, giving efficient algebraic attacks on up to 16 rounds of the largest Simon variants.
Key words: block cipher, algebraic attack, equation system, Simon

1 Introduction

The Simon and Speck families of block cipher were published by the National Security Agency in June 2013 [1]. Both families consist of lightweight block ciphers, where the Simon ciphers are optimized for hardware and the Speck ciphers are optimized for software. The ciphers use very little area and have high throughput.

In the publication from the NSA all ciphers were specified, but there was no security analysis. In the relatively short time since the Simon and Speck ciphers became known the cryptographic community has spent a lot of effort cryptanalyzing them. Some work focused on differential cryptanalysis has been published [2,3], as well as a paper on linear cryptanalysis of Simon [4]. Several other papers on cryptanalysis of Simon have also been posted to the IACR eprint archive [5,6,7,8,9,10,11].

The work in [10] is focused on cube attacks, but other than that we have not seen any published work on Simon's resistance to algebraic attacks. This paper investigates to what extent algebraic attacks may be applied to the Simon family of ciphers and helps to fill this gap in the cryptanalysis of Simon. The main contribution of this work is the analysis that finds linear relations between variables coming from different plaintext/ciphertext pairs. Using the linear relations we may reduce the number of variables a lot when constructing an equation system based on several chosen plaintexts. We use a simple and straight-forward solving algorithm to test how many rounds of different Simon variants that can be broken when the linear relations are exploited. For instance, for Simon with a 128-bit block we can solve a system representing 16 rounds of the cipher using 20 chosen plaintexts in three and a half minutes on a MacBook Air, even though the equation system initially contains 15488 variables.

The paper is organized as follows. In Section 2, we give a brief description of the Simon ciphers. In Section 3 we construct equation systems representing Simon, and in particular look at reusing variables across different plaintext/ciphertext pairs. In Section 4 we introduce a simple way to eliminate redundant variables, which turns into a solving algorithm when all variables get eliminated. Extensive experiments on solving equation systems representing five different Simon variants are also reported here. Section 5 concludes the paper.

2 Description of the Simon Family of Ciphers

The Simon family of block ciphers consists of 10 members. There are two parameters that can be varied: the block size and the key size. For each of the allowed choices of block and key sizes the number of rounds is given. The variants of Simon ciphers that have been defined are listed in Table 1.

Block size	Key size	Rounds
32	64	32
48	72	36
48	96	36
64	96	42
64	128	44
96	96	52
96	128	54
128	128	68
128	192	69
128	256	72

Table 1: Parameters for the Simon variants.

Each cipher is a traditional Feistel cipher with a simple round function, depicted in Figure 1. Two copies of the bitstring input to the round function are cyclically shifted by 1 and 8 positions to the left, respectively. These two words are joined together with bit-wise AND. Then the input word rotated by 2 positions to the left and the round key are XORed onto the output from the AND-operation and the result forms the output of the round function.

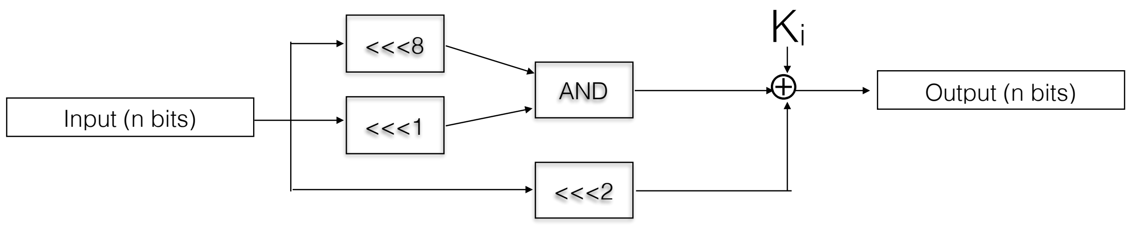


Fig. 1: Round function of the Simon ciphers.

Following the specification in [1], we let the left half of the cipher block be the input to the round function, and we include the swap also after the last round. The left and right halves of the plaintext are denoted by P_1 and P_0, respectively. The left and right halves of the ciphertext are similarly denoted C_1 and C_0. We start the numbering of the rounds at 0, so the first and last rounds in an r-round variant of Simon is round 0 and round $r-1$, respectively. The size of one half of the cipher block is denoted by n, and the key size is denoted by k. A member of the Simon family having block size $2n$ and key size k will be referred to as Simon$2n/k$.

The key schedule for Simon comes in three different variants, depending on which member of the Simon family is used. We omit the details here (they can be found in [1]), but note that all three variants are linear. The bits of each round key can be expressed as a linear combination of the user-selected key bits.

3 Constructing Simon Equation Systems Using Multiple Plaintext/Ciphertext Pairs

We proceed to create equation systems representing instances of a Simon cipher. First we create an equation system using a single plaintext and its corresponding ciphertext. In all systems we work with, all variables will represent the values of single bits, and all additions and multiplications will be over the binary field.

3.1 Equations Representing the Encryption of One Plaintext

The only non-linear part of Simon is the AND-operation in the round function. This operation works on individual bits in parallel, taking 2 input bits and producing 1 output bit. For two input variables x and y producing the output z the equation is simply $x \cdot y = z$.

To keep equations simple and quadratic, we introduce variables representing the cipher state at the output of the AND-operations. In addition, the user-selected key bits are also variables. As noted above, the bits of each round key can be expressed as linear combinations of the user-selected key bits. All other operations in Simon are linear, so every input and output bit to each AND-operation can be described as linear combinations of these variables.

Note that there is no key material introduced before the AND-operation in the round function, so in the first and last rounds we do not need to introduce any variables. The outputs of the AND-operations in these rounds are simply constants depending on the left half of the (known) plaintext and the right half of the ciphertext.

For an r-round version of Simon using block size $2n$ and key size of k bits we will get $n(r-2)$ basic equations in $n(r-2)+k$ variables. Each equation will have the form

$$l_a \cdot l_b = x_c, \tag{1}$$

where l_a and l_b are linear combinations of variables and x_c is a single variable in the output of the AND-operation.

The set of basic equations will represent an r-round encryption of some plaintext using a Simon cipher. The system of equations we construct following the description given so far is underdefined by k variables. The following proposition shows how to eliminate another $2n$ variables from the system.

Proposition 1. *Let $\mathcal{F}$ be an r-round Feistel cipher, where $(C_1, C_0) = \mathcal{F}(P_1, P_0)$. Denote the output of the round function in round j by u_j, for $0 \le j \le r-1$. Then both XOR sums*

$$\sum_{j=0}^{\lfloor \frac{r-1}{2} \rfloor} u_{2j} \text{ and } \sum_{j=0}^{\lfloor \frac{r-1}{2} \rfloor} u_{2j+1}$$

can be expressed as the XOR of one half of P and one half of C.

Proof: Denote the input to round j by i_j for $0 \le j \le r-1$, and assume $\mathcal{F}$ has a swap after the last round. Then $i_0 = P_1$ and $i_{r-1} = C_0$. Define $i_{-1} = P_0$ and $i_r = C_1$. In the Feistel cipher $\mathcal{F}$, i_j is given as $i_j = u_{j-1} + i_{j-2}$ for $1 \le j \le r$, or equivalently

$$u_j = i_{j-1} + i_{j+1}, \ 0 \le j \le r-1.$$

The sum of u's with even indices then becomes

$$\sum_{j=0}^{\lfloor \frac{r-1}{2} \rfloor} u_{2j} = (i_{-1} + i_1) + (i_1 + i_3) + (i_3 + i_5) + \ldots,$$

where the last term will be i_{r-1} when r is even and i_r when r is odd. All terms in the sum except for the first and last will cancel, so the sum just becomes the sum of P_0 and C_0 or C_1.

The sum $\sum u_{2j+1}$ similarly becomes a telescope sum where all terms except for the first and last will cancel. The first term will be P_1 and the last term will be C_0 when r is odd and C_1 when r is even. $\square$

All outputs from the round function in Simon can be expressed as linear combinations of the variables we have introduced, and using Proposition 1 we can derive $2n$ independent linear equations (assuming known plaintext) that can be used to eliminate $2n$ variables. After doing this the number of variables in the basic system becomes $n(r-4)+k$, and the system will be underdefined by $k-2n$ variables.

When the block and key sizes are equal one known plaintext is expected to be enough to uniquely determine the key in Simon, otherwise we will need at least two known plaintexts to have enough information to uniquely determine the user-selected key.

3.2 Equation System Representing Encryptions of Two Plaintexts

We call the set of equations and variables representing the encryption of plaintext P into ciphertext C for a (P, C)*-instance*.

Above we constructed the equations representing the encryption of the single plaintext P into C, introducing variables for the cipher state. If we repeat the construction for another plaintext/ciphertext pair P' and C' we must in general make new variables for the cipher state in the (P', C')-instance. However, we may construct P and P' in some particular way to try to make the cipher states in the first few rounds similar, or related. This will allow us to not introduce new variables for some of the cipher state bits in the (P', C')-instance, but rather re-use the variables from the (P, C)-instance.

The benefit will be a more overdefined total system, with fewer variables, which should be easier to solve. Two questions are: How many rounds of Simon must be executed before the cipher states of the two instances become unrelated, and how many new variables can be avoided in the (P', C')-instance?

Constructing Plaintexts The plaintexts P and P' may be constructed as follows. Let P_1 be a string of n 0-bits. Let the rightmost bit of P'_1 be 1, and the rest of the bits in P'_1 be 0. The output of the AND-operation for P_1 and P'_1 will be the all-zero word in both cases. After adding the input word shifted by 2 positions to the left, the difference in the output of the round function will be 4, hexadecimally. Choosing P_0 to be anything, and $P'_0 = P_0 + 4$, the differences will cancel in the addition at the end of the first round and the inputs to the second round will be the same for both texts. Hence the variables to be introduced in the output of the AND-operation in the second round of the (P', C')-instance are exactly the same as in the (P, C)-instance. The situation is depicted in Figure 2.

Tracing Relations Further The inputs to the third round will differ in the rightmost bit due to the XOR of P_1 and P'_1 onto the outputs from the second round. This difference means that two of the bits in the output of the AND-operation may be unequal in the two instances.

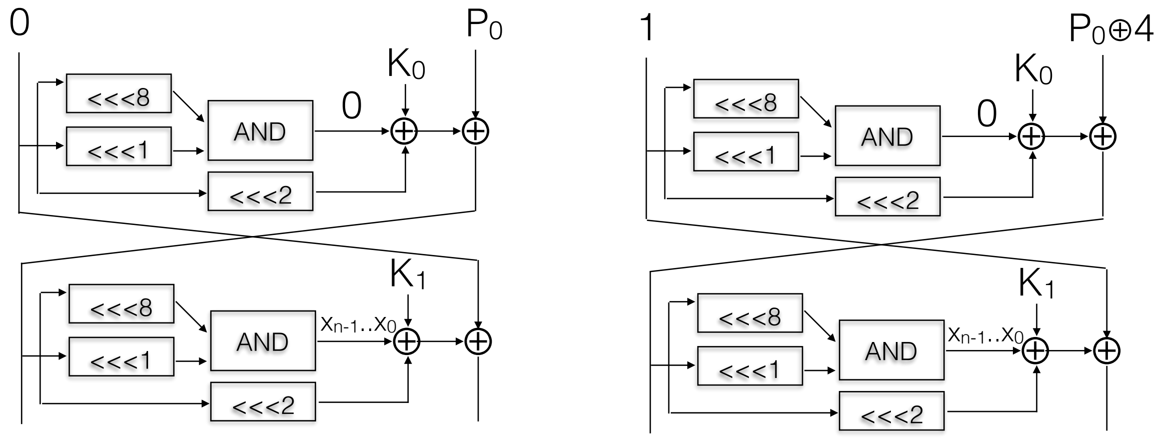


Fig. 2: Chosen plaintexts allowing reuse of all variables in second round.

The other $n-2$ bits will still be the same, so $n-2$ of the third-round variables in the (P, C)-instance carry over and can be used in the (P', C')-instance directly. However, we do not need to introduce new variables for the two bits that differ, either.

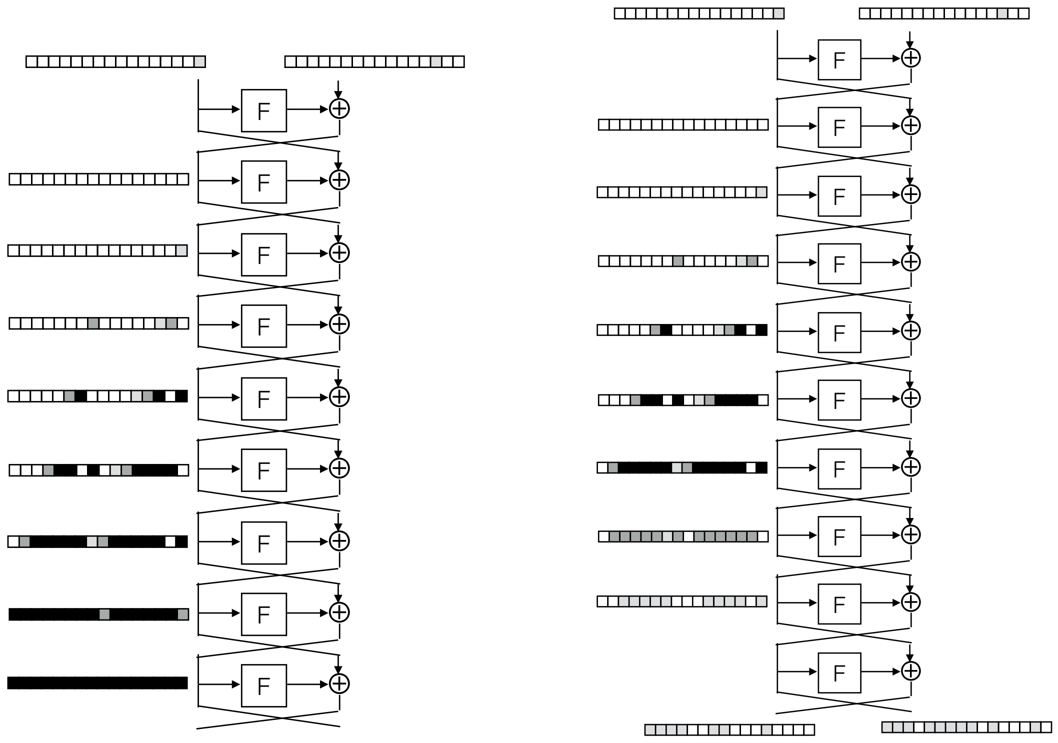
Let l_a be the linear combination representing the rightmost bit in the input of the third round of the (P, C)-instance. Then the same bit will be represented as $l_a + 1$ in the (P', C')-instance. One equation using l_a in the first instance will be $l_a \cdot l_b = x_c$ for some l_b and x_c. In the second instance the same equation will be $(l_a + 1) \cdot l_b = x_d$, where x_d is a (temporary) new variable in the second instance. Adding these two equations will cause the quadratic terms to cancel, and we get the relation $x_d = x_c + l_b$, which can be used to express x_d as a linear combination of variables only from the (P, C)-instance.

In general, when the two sets of input bits (represented as linear combinations) to the AND-operations only differ in the constant terms, the output bits are related through a linear equation. We can then reuse variables across the instances. Only when inputs in two instances differ in more than the constant term will we have to make new variables in the second instance.

Complete Mapping of Linear Relations We have made a computer program for tracing the relations between two (P, C)- and (P', C')-instances in Simon$2n/k$ for all different choices for n. Figure 3a shows how the individual bits in the input to the round functions in Simon32/64 will be related between two plaintexts prepared as explained above. The bits in the input to the round function can be either white, light grey, dark grey or black. A white bit indicates that the linear combinations for this bit are equal in the two instances. A light grey bit indicates that the linear combinations differ in the constant term only. A dark grey bit means that the linear combination in the second instance can be written exclusively with variables from the first instance. A black bit indicates that a new variable has been introduced, and that there is no linear relation between the bits in this position. As can be seen in Figure 3a, only in the ninth round input will all linear relations between the two instances have been completely wiped out.

Determining linear relations between cipher state bits can also be done from the ciphertext side. The attacker does not have control over the differences in the ciphertexts, so the linear relations between two instances are expected to be wiped out faster than from the chosen plaintexts. Figure 3b shows the situation in a 10-round version of Simon32/64 after tracing linearly related bits from both the plaintext and the ciphertext sides. In the figure, the colour of a bit has the lightest colour found when tracing from both sides.

In the example of Figure 3b there are only 21 bits of internal cipher state that must be represented by new variables in the (P', C')-instance. The number of new variables we have to introduce from a new (P', C')-instance will vary slightly according to how "lucky" we are with the ciphertexts. However, if we had created the second instance directly from the description of Section 3.1 we would have had to introduce 96 new variables in a 10-round version of Simon32/64.



(a) Tracing linearly related cipher state bits from chosen plaintexts only.

(b) Tracing linearly related cipher state bits from both plaintext and ciphertext sides.

Fig. 3: Tracing linearly related cipher state bits in Simon32/64.

We have traced and mapped the linearly related cipher state bits for all variants of Simon using plaintexts prepared as described above. The rotations in Simon shift the words with a constant number of positions, independent og the block size, so we can expect the number of rounds needed to wipe out all linear relations between two instances to increase in the larger variants. Figures for the larger Simon variants will not fit in the paper in a readable way, but in Table 2 we report on the number of white, light grey, dark grey and black bits in the input to each round for the different Simon variants.

3.3 Equation System Using Multiple (P, C)-Instances

We can make an equation system using as many (P, C)-instances as we like. For each new instance added, we can trace the linear relations the new instance has to all previous instances.

Input to Round	Simon32	Simon48	Simon64	Simon96	Simon128
0	(15,1,0,0)	(23,1,0,0)	(31,1,0,0)	(47,1,0,0)	(63,1,0,0)
1	(16,0,0,0)	(24,0,0,0)	(32,0,0,0)	(48,0,0,0)	(64,0,0,0)
2	(15,1,0,0)	(23,1,0,0)	(31,1,0,0)	(47,1,0,0)	(63,1,0,0)
3	(13,1,2,0)	(21,1,2,0))	(29,1,2,0)	(45,1,2,0)	(61,1,2,0)
4	(10,1,2,3)	(17,2,2,3)	(25,2,2,3)	(41,2,2,3)	(57,2,2,3)
5	(6,1,2,7)	(12,1,4,7)	(20,1,4,7)	(36,1,4,7)	(52,1,4,7)
6	(2,1,2,11)	(5,1,4,14)	(12,1,4,15)	(27,2,4,15)	(43,2,4,15)
7	(0,0,2,14)	(1,0,1,22)	(6,0,2,24)	(19,1,3,25)	(35,1,3,25)
8	(0,0,0,16)	(0,0,0,24)	(2,0,0,30)	(12,0,1,35)	(27,1,1,35
9	(0,0,0,16)	(0,0,0,24)	(0,0,0,32)	(6,0,0,42)	(20,0,1,43)
10	(0,0,0,16)	(0,0,0,24)	(0,0,0,32)	(2,0,0,46)	(12,0,1,51)
11	(0,0,0,16)	(0,0,0,24)	(0,0,0,32)	(0,0,0,48)	(6,0,0,58)
12	(0,0,0,16)	(0,0,0,24)	(0,0,0,32)	(0,0,0,48)	(2,0,0,62)
13	(0,0,0,16)	(0,0,0,24)	(0,0,0,32)	(0,0,0,48)	(0,0,0,64)

Table 2: Numbers and types of linearly related bits in Simon using two chosen plaintexts. In each 4-tuple the first entry is the number of equal linear combinations, the second entry is the number of linear combinations only differing in the constant term, the third entry is the number of linear combinations differing in more than only the constant term, and the fourth entry is the number of non-linearly related bits.

We expect to add fewer and fewer variables (up to some limit) for each new instance added. The number of equations we get from each new instance is initially $n(r-2)$, but we can expect many of the new equations to be expressible through equations from other instances and therefore redundant. However, the variables for the user-selected key remains the same, so the degree of overdefinedness in the total system will increase with the number of (P, C)-instances used. This is also confirmed in the experiments reported in Section 4.

When exploiting the fact that many of the cipher state bits from different (P, C)-instances are linearly related, the equations in the total system will be a lot more connected than if only the variables for the key were common. In the next section we develop a very simple solving algorithm that exploits the linear relations that have been shown to exist.

4 Simple Solving Algorithm and Experiments

By creating variables as described in Section 3.1 we can easily construct the basic equations in one (P, C)-instance. It is also straight-forward to construct a system from multiple (P, C)-instances by introducing new variables in each instance. Identifying and using all the linear relations we know exist in the first rounds of the Simon ciphers are a bit more complicated.

4.1 Identifying Linear Relations

Assume we have constructed all basic equations (1) as described in Section 3.1, including the use of Proposition 1, and say we used t chosen (P, C)-instances. Then we will get a system of $tn(r-2)$ equations in $tn(r-4) + k$ variables.

The form of each equation is $l_a \cdot l_b + x_c = 0$ for some linear combinations l_a and l_b. If we have two such equations

$$l_a \cdot l_b + x_c = 0$$
$$l_d \cdot l_e + x_f = 0$$

where either

$$\deg(l_a + l_d) = \deg(l_b + l_e) = 0 \text{ or } \deg(l_a + l_e) = \deg(l_b + l_d) = 0 \tag{2}$$

we know that adding the two equations together will yield a linear equation. This linear equation can be used to eliminate one variable. We could use diagrams like those in Figure 3 to find exactly which pairs of equations that will yield linear relations when added together. A simpler but more time consuming method is just to try adding all pairs of equations and see which additions that become linear.

To get a fast check of whether two equations will give a linear relation we do not actually multiply out the product $l_a \cdot l_b$ in (1), but rather just check if condition (2) holds.

4.2 Solving Algorithm

After finding all linear relations in a set of equations representing Simon, we can use them to eliminate variables. The whole process can then be repeated, to see if some other pairs of equations will yield new linear equations after the first substitution of variables. The pseudocode for the whole algorithm is given in Algorithm 1, and is nothing more than a simplified version of ElimLin [12].

Algorithm 1 Solving system of equations on the form $l_a \cdot l_b + l_c = 0$.

```
repeat
  for every pair of equations $f, g$ do
    if $f + g$ is linear then
      Eliminiate one variable from system
    end if
  end for
until No more linear equations found
```

The complexity of Algorithm 1 on a system with m equations in n variables can be estimated as follows. The inner for-loop iterates $\binom{m}{2} = \mathcal{O}(m^2)$ times. At least one variable has to be eliminated from the system each time the for-loop is run, otherwise the algorithm stops. As there are n variables in the system the for-loop can be run at most n times, so the worst-case complexity for the whole algorithm is $\mathcal{O}(m^2 n)$.

At the beginning of the research for this paper, Algorithm 1 was only considered to be a preprocessing step for the equation system before applying a more advanced solver. We were a bit surprised to find that the algorithm actually solved systems, by eliminating *all* variables. If we store all the linear equations used to eliminate variables we get a uniquely defined linear system of equations, which is easily solved to give the variables for the user-selected key. We have verified that the returned solution from the solver actually gives the correct key.

4.3 Experiments

We have investigated how often our algorithm is able to solve the various Simon systems, using different number of rounds r and different numbers t of (P, C)-instances. The results are reported in the tables below.

For each choice of r and t parameters we have generated 256 systems and tried to solve them using Algorithm 1. The number of times we succeeded is indicated in each cell. For

r \ t	4	5	6	8	10	12	16	20
7	0	0	0	0	37	42	44	198
8	0	68	109	164	250	251	252	254
9	26	213	240	253	256	255	256	256
10	0	2	152	255	255	255	255	256
11	0	0	0	0	0	3	35	50

Table 3: Number of times an equation system representing Simon32/64 was solved out of 256 tries.

r \ t	4	5	6	8	10	12	16	20
8	6	159	185	231	255	256	256	256
9	74	229	247	256	256	256	256	256
10	0	44	231	256	256	256	256	256
11	0	0	0	0	1	27	220	250

Table 4: Number of times an equation system representing Simon48/72 was solved out of 256 tries.

r \ t	4	5	6	8	10	12	16	20
8	0	7	20	54	211	226	242	256
9	1	115	177	229	255	256	256	256
10	5	135	242	256	256	256	256	256
11	0	0	38	253	256	256	256	256
12	0	0	0	0	5	137	255	256

Table 5: Number of times an equation system representing Simon64/96 was solved out of 256 tries.

r \ t	4	5	6	8	10	12	16	20
9	8	165	222	248	256	256	256	256
10	2	89	234	253	256	256	256	256
11	0	5	66	231	256	256	256	256
12	0	1	23	143	233	248	256	255
13	0	0	0	82	189	233	251	255
14	0	0	0	0	2	62	225	250

Table 6: Number of times an equation system representing Simon96/96 was solved out of 256 tries.

each system, the P_1-half of the plaintexts were chosen to be the bit string representing i, for $0 \le i \le t-1$. When t is less than 128 we know that the output of the AND-operation in the first round will be 0 for all P_1's. An initial value v was chosen at random for the other half of the plaintext, and the P_0-values were given as $v \oplus (i << 2)$ for each of the t plaintexts. The key was chosen at random for every system.

r \ t	4	5	6	8	10	12	16	20
10	1	63	226	247	256	256	256	256
11	0	7	71	236	255	256	256	256
12	0	0	15	153	227	250	256	256
13	0	0	12	77	166	220	250	254
14	0	0	1	35	104	164	220	243
15	0	0	0	15	51	106	172	214
16	0	0	0	0	0	13	106	182

Table 7: Number of times an equation system representing Simon128/128 was solved out of 256 tries.

Attempts to solve systems representing more rounds than reported in the tables were unsuccessful in the experiments.

When Algorithm 1 fails, we have tried to apply the CRHS method [13] for solving the remaining system. This has only been successful in the smaller cases of Simon32/64 and Simon48/72 in seven to nine round variants, using relatively few (i.e. three or four) plaintext/ciphertext pairs. When Algorithm 1 stops without returning a solution we are left with a non-linear equation systems containing several thousand variables in the larger cases. These systems were too big to handle for the CRHS method on an ordinary computer.

It is still possible that deeper insight into the particular equations for Simon systems may give a better solving algorithm for these special systems. This could be a topic for future work.

5 Conclusions

Simon's resistance against differential and linear cryptanalysis has been thoroughly investigated in [2,3,4]. In this paper we have looked at Simon's resistance against algebraic attacks, contributing to the cryptanalysis of the Simon family of ciphers.

We have traced how two (P, C)-instances of Simon, differing in a single bit in the plaintexts, remain related through the first rounds of the cipher. Two instances of Simon32/64 are still linearly connected in two bits in the input of the eighth round, and two instances of Simon128/128 are linearly related up to and including the thirteenth round. Using these relation mappings one can reduce the number of variables a lot when creating equation systems representing Simon.

We developed a simple solving algorithm that exploits the many linear relations found in Simon. Using several chosen plaintexts, we tested how many rounds of Simon that could be broken with this method. The experiments verified the findings of Section 3, in that we always could solve a few more rounds of Simon after the linear relations from the plaintext side had completely disappeared. It is unusual that one can solve an equation system representing 16 rounds of a real cipher with 128-bit block and key sizes. Even though the largest systems tested initially contained 15488 variables, and 4839 variables after the first iteration of finding linearly related cipher state bits, they could be solved in three and a half minutes on a MacBook Air 2013.

To compensate for its very lightweight structure, the Simon ciphers have rather many rounds. The attack reported here does not threaten the security of Simon, but it is nevertheless interesting to see that large non-linear equation systems representing some cipher can still be efficiently solved when the cipher has very simple operations.

References

1. R. Beaulieu, D.Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers. *The SIMON and SPECK Families of Lightweight Block Ciphers,* Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/2013/404.
2. A. Biryukov, A. Roy, V. Velichkov. *Differential Analysis of Block Ciphers SIMON and SPECK,* Fast Software Encryption 2014, LNCS 8540, pp. 546 – 570, Springer, 2014.
3. F. Abed, E. List, S. Lucks, J. Wenzel. *Differential Cryptanalysis of Round-Reduced Simon and Speck,* Fast Software Encryption 2014, LNCS 8540, pp. 525 – 545, Springer, 2014.
4. Q. Wang, Z. Liu, K. Varici, Y. Sasaki, V. Rijmen, Y. Todo. *Cryptanalysis of Reduced-round SIMON32 and SIMON48,* INDOCRYPT 2014, LNCS 8885, pp. 143 – 160, Springer, 2014.
5. H. A. Alkhzaimi, M. M. Lauridsen. *Cryptanalysis of the SIMON Family of Block Ciphers,* Cryptology ePrint Archive, Report 2013/543, 2013. http://eprint.iacr.org/2013/543.
6. J. Alizadeh, N. Bagheri, P. Gauravaram, A. Kumar, S. K. Sanadhya. *Linear Cryptanalysis of Round Reduced SIMON,* Cryptology ePrint Archive, Report 2013/663, 2013. http://eprint.iacr.org/2013/663.
7. N. Wang, X. Wang, K. Jia, J. Zhao. *Improved Differential Attacks on Reduced SIMON Versions,* Cryptology ePrint Archive, Report 2014/448, 2014. http://eprint.iacr.org/2014/448.
8. J. Alizadeh, H. A. Alkhazaimi, M. R. Aref, N. Bagheri, P. Gauravaram, M. M. Lauridsen. *Improved Linear Cryptanalysis of Round Reduced SIMON,* Cryptology ePrint Archive, Report 2014/681, 2014. http://eprint.iacr.org/2014/681.
9. D. Shi, L. Hu, S. Sun, L. Song, K. Qiao, X. Ma. *Improved Linear (hull) Cryptanalysis of Round-reduced Versions of SIMON*, Cryptology ePrint Archive, Report 2014/973, 2014. http://eprint.iacr.org/2014/973.

10. Z. Ahmadian, S. Rasoolzadeh, M. Salmasizadeh, M. R. Aref. *Automated Dynamic Cube Attack on Block Ciphers: Cryptanalysis of SIMON and KATAN,* Cryptology ePrint Archive, Report 2015/040, 2015. http://eprint.iacr.org/2015/040.
11. S. Kölbl, G. Leander, T. Tiessen. *Observations on the SIMON block cipher family,* Cryptology ePrint Archive, Report 2015/145, 2015. http://eprint.iacr.org/2015/145.
12. N. Courtois, G.V. Bard. *Algebraic Cryptanalysis of the Data Encryption Standard,* IMA International Conference, volume 4887, pp. 152 – 169, Springer, 2007.
13. T. E. Schilling, H. Raddum. *Solving Compressed Right Hand Side Equation Systems with Linear Absorption,* Sequences and Their Applications - SETA 2012, LNCS 7280, pp. 291 – 302, Springer, 2012.