ALGEBRAIC CRYPTANALYSIS OF SMS4

JEREMY ERICKSON, TAYLOR UNIVERSITY

Abstract. The SMS4 block cipher is part of the Chinese WAPI wireless standard. It is a 32-round block cipher with 128-bit blocks and 128-bit keys. This paper explores algebraic attacks on SMS4 using Gröbner basis attacks on equation systems over GF(2) and GF(2^8), as well as attacks using a SAT solver derived from the GF(2) model.

1. Introduction

Algebraic cryptanalysis is a relatively new field of cryptology. The basic idea is to model a cipher using a system of polynomial equations over a finite field. This approach has gained attention since Nicolas Courtois claimed that it could be used to attack AES, which has a simple algebraic structure [6]. This attack has also been attempted on other ciphers such as DES [5].

The SMS4 cipher was written by the Chinese government as part of their WAPI standard for wireless networks. The best currently known attacks on SMS4 are the 14-round rectangle attack and 16-round impossible differential attack discussed by Jiqiang Lu in [10]. However, as shown in [9], SMS4 itself has a simple algebraic structure, similar to AES. Thus, I have attempted to attack it with algebraic attacks over GF(2). Wen Ji and Lei Hu have also attempted to attack SMS4 with an algebraic attack over GF(2^8), as shown in [8], but their paper presents only a theoretical analysis, not an actual implementation. In this paper I present an attempt to attack SMS4 with algebraic attacks over GF(2), including several potential alterations. I also present preliminary results of attacks based on GF(2^8) as in [8].

Both types of attacks were implemented using the Magma computer algebra system ([3]), as well as using the MiniSAT boolean satisfiability solver in some cases.

2. Structure of SMS4

SMS4 is an unbalanced Feistel cipher with a block size of 128 bits and a key size of 128 bits. Each block is divided into four 32-bit blocks, referred to as "words", which in turn consists of four 8-bit bytes. The official specification document for SMS4 is available in Chinese at `http://www.oscca.gov.cn/UpFile/`

Much of this work was done in conjunction with Chris Christensen and Amber Rogers of Northern Kentucky University, Brian Nixon of the University of Michigan, and Jintai Ding, Crystal Clough, John Baena, Daniel Cabarcas, and Zachary Vance of the University of Cincinnati, as part of an NSF-sponsored Research Experience for Undergraduates, funded by the Department of Defense.

The Computational Algebra Group at the University of Sydney graciously provided Magma for this project.

200621016423197990.pdf, and an English translation is provided at [7]. There are several fundamental components to the cipher.

2.1. **The SMS4 S-Box.** The official definition of the SMS4 S-box is based on a table of values (See table 1.) However, as shown in [9], the S-box can also be

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	d6	90	e9	fe	cc	e1	3d	b7	16	b6	14	c2	28	fb	2c	05
1	2b	67	9a	76	2a	be	04	c3	aa	44	13	26	49	86	06	99
2	9c	42	50	f4	91	ef	98	7a	33	54	0b	43	ed	cf	ac	62
3	e4	b3	1c	a9	c9	08	e8	95	80	df	94	fa	75	8f	3f	a6
4	47	07	a7	fc	f3	73	17	ba	83	59	3c	19	e6	85	4f	a8
5	68	6b	81	b2	71	64	da	8b	f8	eb	0f	4b	70	56	9d	35
6	1e	24	0e	5e	63	58	d1	a2	25	22	7c	3b	01	21	78	87
7	d4	00	46	57	9f	d3	27	52	4c	36	02	e7	a0	c4	c8	9e
8	ea	bf	8a	d2	40	c7	38	b5	a3	f7	f2	ce	f9	61	15	a1
9	e0	ae	5d	a4	9b	34	1a	55	ad	93	32	30	f5	8c	b1	e3
a	1d	f6	e2	2e	82	66	ca	60	c0	29	23	ab	0d	53	4e	6f
b	d5	db	37	45	de	fd	8e	2f	03	ff	6a	72	6d	6c	5b	51
c	8d	1b	af	92	bb	dd	bc	7f	11	d9	5c	41	1f	10	5a	d8
d	0a	c1	31	88	a5	cd	7b	bd	2d	74	d0	12	b8	e5	b4	b0
e	89	69	97	4a	0c	96	77	7e	65	b9	f1	09	c5	6e	c6	84
f	18	f0	7d	ec	3a	dc	4d	20	79	ee	5f	3e	d7	cb	39	48

TABLE 1. The SMS4 S-box, with the first input nibble as the row index and the second as column index

represented as an affine transformation over GF(2), followed by an inversion over GF(2^8), followed by another affine transformation over GF(2). The system is thus written as

$$s(x) = I(x \cdot A + C) \cdot A + C \tag{1}$$

where I indicates inversion over GF(2^8) (with the inverse of 0 defined as 0) and the necessary field conversion. The values are given as

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$C = (1, 1, 0, 0, 1, 0, 1, 1)$$

To convert from GF(2) to GF(2^8), we use the irreducible polynomial

$$f(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$$

and let the first term represent the constant term in a polynomial of degree 7, the second term represent the x coefficient, etc.

However, calculation shows that these results do not match the table provided in the SMS4 specification without modification. If the input to and output from the S-box are each reversed, then the output is correct. Thus, this paper proposes the alternate model of the S-box

$$s(x) = A_2 \cdot I(A_1 \cdot x + C_1) + C_2 \tag{2}$$

with parameters

$$A_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$C_1 = (1, 1, 0, 0, 1, 0, 1, 1)^T$$

$$C_2 = (1, 1, 0, 1, 0, 0, 1, 1)^T$$

Compared to the original model, this model uses right multiplication, reverses the columns of A for A_1, reverses the rows of A for A_2, uses C^T for C_1 and reverses C_1 for C_2. This achieves reversing the input and output, compared to equation 1. Calculations reveal that this model matches the table.

When the S-box is used in SMS4, it is applied 4 times in parallel, to an entire word. Thus, $X \in \mathrm{GF}(2)^{32}$ is split into $(x_1, x_2, x_3, x_4) \in (\mathrm{GF}(2)^8)^4$, and

$$S(X) = s(x_1)s(x_2)s(x_3)s(x_4)$$

2.2. **The Linear Diffusion Transformation.** SMS4 uses two linear diffusion transformations, one for the round function and one for the key schedule. Here the notation $\lll$ represents circular left shifting. Each function operates on $x \in \mathrm{GF}(2)^{32}$. For the round function, we use

$$L(x) = x \oplus (x \lll 2) \oplus (x \lll 10) \oplus (x \lll 18) \oplus (x \lll 24) \tag{3}$$

For the key schedule, we use

$$L'(x) = x \oplus (x \lll 13) \oplus (x \lll 23) \tag{4}$$

2.3. **The SMS4 Key Schedule.** We define a vector $(Y_i, Y_{i+1}, Y_{i+2}, Y_{i+3}) \in (\mathrm{GF}(2)^{32})^4$ as the key schedule input to round i.

Denote the input key as (K_0, K_1, K_2, K_3). Then

$$Y_0 = K_0 \oplus 0xa3b1bac6$$
$$Y_1 = K_1 \oplus 0x56aa3350$$
$$Y_2 = K_2 \oplus 0x677d9197$$
$$Y_3 = K_3 \oplus 0xb27022dc$$

Also denote $CK_i = (ck_{i,0}, ck_{i,1}, ck_{i,2}, ck_{i,3}) \in (\mathbb{Z}_2^8)^4$ where $ck_{i,j} = 28i + 7j \mod 256$, represented in binary.

Then

$$RK_i = Y_{i+4} = Y_i \oplus L'(S(Y_{i+1} \oplus Y_{i+2} \oplus Y_{i+3} \oplus CK_i)) \tag{5}$$

2.4. **The SMS4 Round Function.** We define a vector $(X_i, X_{i+1}, X_{i+2}, X_{i+3}) \in (\mathrm{GF}(2)^32)^4$ as the input to round i, numbering the rounds from 0. Thus, (X_0, X_1, X_2, X_3) represents the plaintext. Then,

$$X_{i+4} = X_i \oplus L(S(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus RK_i)) \tag{6}$$

The output of the last four rounds is reversed (at the word level) to generate the ciphertext. Thus, the ciphertext is $(X_{35}, X_{34}, X_{33}, X_{32})$.

3. Simplified SMS4

To provide for some basic exploration of the behavior of algebraic attacks over a larger number of rounds, as well as to provide a form of SMS4 that can be worked out by hand, this paper proposes a simplified SMS4 algorithm, which will be referred to from here as S-SMS4.

The basic operations of S-SMS4 are identical to full SMS4, except that all operations on 128-bit blocks become operations on 32-bit blocks, operations on 32-bit words become operations on 8-bit "words", and operations on 8-bit bytes become operations on 4-bit nibbles.

3.1. **The S-SMS4 S-box.** The S-box of SMS4 was designed from the description of the full SMS4 S-box in [9]. The new S-box is designed to transform a 4-bit vector to another 4-bit vector, but otherwise follows equation 1 plus reversing input and output. Thus, a smaller cyclic matrix is the basic A with its bottom row as the row vector for C. Thus,

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$
$$C = (1, 1, 0, 1)$$

Accounting for the reversal and using the form of equation 2, however, we derive the following matrices in a similar manner:

$$A_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

	00	01	10	11
00	1001	0001	1011	0010
01	1111	0110	1000	1101
10	0111	1100	0011	1110
11	0100	0000	1010	0101

TABLE 2. S-SMS4 S-box, in the same form as table 1

$$A_2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$
$$C_1 = (1, 1, 0, 1)^T$$
$$C_2 = (1, 0, 1, 1)^T$$

The inversion step is similar to full SMS4, except that we use $\mathrm{GF}(2^4)$ with an irreducible polynomial of

$$f(x) = x^4 + x^3 + x^2 + x + 1$$

This S-box can also be written as a table, see table 2.

Because the S-SMS4 S-box only accepts one byte as input, which is two nibbles, we split $X \in (\mathrm{GF}(2))^{32}$ into $(x_1, x_2) \in (\mathrm{GF}(2)^4)^2$ and write:

$$S(X) = s(x_1)s(x_2)$$

These parameters were chosen in attempt to model the properties of the full SMS4 S-box, providing a small field size over which inversion remains non-trivial.

3.2. **The S-SMS4 Linear Diffusion Transformation.** S-SMS4, like full SMS4, uses two linear diffusion transformations. Each function operates on $x \in \mathrm{GF}(2)^8$. For the round function,

$$L(x) = x \oplus (x \lll 2) \oplus (x \lll 6) \tag{7}$$

For the key schedule,

$$L'(x) = x \oplus (x \lll 3) \oplus (x \lll 5) \tag{8}$$

3.3. **The S-SMS4 Key Schedule.** The key schedule for S-SMS4 is analogous to full SMS4, except that we must use shorter initial keys and CK values. The initial keys are as follows:

$$Y_0 = K_0 \oplus 0xa3$$
$$Y_1 = K_1 \oplus 0xb1$$
$$Y_2 = K_2 \oplus 0xba$$
$$Y_3 = K_3 \oplus 0xc6$$

Denote $CK_i = (ck_{i,0}, ck_{i,1}) \in (\mathbb{Z}_2^4)^2$ where $ck_{i,j} = 11i + 3j \mod 16$, represented in binary. Then equation 5 holds.

3.4. **The S-SMS4 Round Function.** With the notations in this section, equation 6 holds for simplified SMS4. We denote S-SMS4 to have eight rounds, thus the ciphertext is $(X_{11}, X_{10}, X_9, X_8)$.

4. Gröbner Basis and SAT Solver Attacks over GF(2)

The primary attempt to attack SMS4 in this paper is based on solving a system of equations over GF(2). The equations are divided into two groups, one representing the key schedule for the entire cipher, and one with a representation of the round function process for each plaintext/ciphertext pair. The Magma code is written in such a way that full and simplified SMS4 can easily be compared by changing which predefined functions are loaded. (Some code in the Magma implementation was written by Brian Nixon.)

4.1. **Modelling the Inversion Step.** The calculation of inversion in the S-box calculation can be done with several models. The simplest model is $XY - 1 = 0$ for $X, Y \in \mathrm{GF}(2^8)$, which is only of degree 2. However, this model will not yield a solution for $X = 0$, and thus is not always sufficient to model the cipher. An alternative model is $Y^2X - Y = 0$, which is still degree 2 when represented over GF(2) due to the property that $x^2 = x, x \in \mathrm{GF}(2)$. However, in this case, an extraneous solution $Y = 0$ is present for any input X, even when X is nonzero. Thus, the equation $X^2Y - X = 0$ must be added to the system to yield a correct solution. For this reason, most previous attacks (c.f.e. [8]) use $XY - 1 = 0$ as the model for the cipher.

To model the inversion over GF(2), we simply represent the coefficients on the polynomials for X and Y as variables in GF(2), perform the necessary operations, and mod by the appropriate irreducible polynomial. This was done using Magma for this project.

The question with the simplified $XY - 1 = 0$ model is how frequently it provides a correct result. One of 256 inputs (for SMS4) or 16 inputs (for S-SMS4) will require the inversion of zero. The round function for each round calls $S(x)$ once, and thus $s(x)$ four times for SMS4 and two times for S-SMS4. The same numbers occur for the key schedule, so a round of SMS4 will use inversion eight times and a round of S-SMS4 will use inversion four times. The simplest probability model is to assume that all inputs to the S-boxes are independent and random. Denote the number of rounds as r. Thus, for SMS4:

$$P_{zinv} = 1 - \left(\frac{255}{256}\right)^{8r} \tag{9}$$

and for S-SMS4:

$$P_{zinv} = 1 - \left(\frac{15}{16}\right)^{4r} \tag{10}$$

In particular, for the full 32 rounds of SMS4, $P_{zinv} = 1 - \left(\frac{255}{256}\right)^{256} \approx 1 - \frac{1}{e}$ or 63.28%, and for the full 8 rounds of S-SMS4, $P_{zinv} = 1 - \left(\frac{15}{16}\right)^{32} \approx 87.32\%$ To test this model, we can also examine the expected number of zeros, which is simply $\frac{8r}{256}$ for full SMS4 and $\frac{4r}{16}$ for S-SMS4.

To verify these numbers, I used Magma to perform 10,000 random encryptions (random key and plaintext) for each multiple of 4 rounds, recording the number of zeros encountered at inversion and the number of encryption processes which used one or more zeros. The results are in table 3.

We can see that the model closely matches the data. Thus, SMS4 does behave like a random system, and we are justified in making our zero inversion predictions

	SMS4				S-SMS4			
	P_{zinv}		# Zeros		P_{zinv}		# Zeros	
r	Model	Test	Model	Test	Model	Test	Model	Test
4	11.77%	11.82%	0.1250	0.1248	64.39%	64.08%	1.0000	0.9962
8	22.16%	21.58%	0.2500	0.2426	87.32%	87.00%	2.0000	1.9819
12	31.32%	31.57%	0.3750	0.3798	95.49%	95.40%	3.0000	3.0105
16	39.41%	39.16%	0.5000	0.4973	98.39%	98.37%	4.0000	4.0079
20	46.54%	46.24%	0.6250	0.6213	99.43%	99.39%	5.0000	5.0092
24	52.83%	53.21%	0.7500	0.7641	99.80%	99.81%	6.0000	5.9828
28	58.38%	58.38%	0.8750	0.8863	99.93%	99.96%	7.0000	7.0271
32	63.28%	63.25%	1.0000	0.9952	99.97%	99.98%	8.0000	8.0152

TABLE 3. Model vs. Tests for Zero Inversion Frequency

based on that assumption. In the case of breaking the full cipher, a zero inversion probability of 63.28% is low enough that either model would be reasonable, although a method that can handle zero inversions would be better if as fast.

4.2. **Modelling the Key Schedule.** The key schedule is modelled separately from the rounds, because it only needs to be modelled once for any key to be broken, even if there is more than one plaintext/ciphertext pair. The following system of equations is used for each round r (indexed starting from 0) and byte (full)/nibble (simplified) index i (also indexed from 0). The model of the S-box inversion is excluded, as the system was tested using several representations. The real system has a set of equations indicating that $ZK_{r,i}$ and $WK_{r,i}$ correspond to inverses in GF(2^8) or GF(2^4).

A single subscript (e.g. BK_r) indicates a word in GF(2)^{32} or GF(2)^8, and a double subscript (e.g. $BK_{r,i}$) indicates a byte or nibble within the word.

$$\begin{aligned} BK_r &= Y_{r+1} \oplus Y_{r+2} \oplus Y_{r+3} \oplus CK_r \\ ZK_{r,i} &= A_1 \cdot BK_{r,i} \oplus C_1 \\ DK_{r,i} &= A_2 \cdot WK_{r,i} \oplus C_2 \\ EK_r &= L'(DK_r) \\ Y_{r+4} &= Y_r + EK_r \end{aligned}$$

When this system of equations is actually implemented in Magma, each variable in GF(2) (with the exception of WK variables) has its own equation. Vectors are used here for simplicity of explanation.

4.3. **Modelling the Round Function.** The model for the round function is similar to the model of the key schedule. However, there is now a separate equation for each plaintext/ciphertext pair p. Thus, in this case, a double subscript indicates a word and a triple subscript indicates a nibble or byte. As in the case of the key schedule, the model also has a set of equations indicating that $Z_{p,r,i}$ and $W_{p,r,i}$ correspond to inverses in the appropriate extension field.

$$\begin{aligned} B_{p,r} &= X_{p,r+1} \oplus X_{p,r+2} \oplus X_{p,r+3} \oplus Y_{r+4} \\ Z_{p,r,i} &= A_1 \cdot B_{p,r,i} \oplus C_1 \end{aligned}$$

$$D_{p,r,i} = A_2 \cdot W_{p,r,i} \oplus C_2$$
$$E_{p,r} = L(D_{p,r})$$
$$X_{p,r+4} = X_{p,r} + E_{p,r}$$

As with the key schedule, when this system is actually implemented in Magma, each variable in GF(2) (with the exception of W variables) has its own equation.

4.4. **SAT Solver Attacks.** In addition to Gröbner Basis attacks, some attacks over GF(2) were also attempted using MiniSAT. To convert the polynomials generated by Magma into the proper input form, Amber Rogers and I wrote a Perl script using the method of [2]. Our conversion system did not attempt to reorder or rewrite the equations to change the random seed, and we used a cutting size of 6 as the paper suggested. The script allowed us to convert our equations to MiniSAT format, run the MiniSAT solver, and verify that the solution contained the correct key.

5. Results of GF(2)-Based Attacks

These attacks were tested on an Intel®Core™2 Quad CPU running at 2.40 GHz. The system had 16 GB of RAM and was running 64-bit Ubuntu 8.04.1, Magma version 2.14-15 and MiniSAT version 2.0.

5.1. **Field Equations.** The field equations over GF(2) are of the form $x^2 = x$ and hold $\forall x \in \text{GF}(2)$. These equations allow the degree of a polynomial to be lowered if it contains any power of a variable, as by induction this property extends to $x^n = x, \forall n \in \mathbb{N}$. Thus, when the equation system was implemented in Magma, I included the field equation for each of the variables as part or the system. However, with the field equations included, Magma ran out of memory rather than obtaining a solution, but without them it was able to find the solution. This result is surprising, as the field equations should only be able to help.

5.2. **Magma vs. SAT solver.** For small numbers of rounds, I tested Magma and MiniSAT both with the same systems of equations. The variance in solution time could be quite high for MiniSAT, so table 4 contains typical measurements for the two systems, both with SMS4 and S-SMS4. For each test, two plaintext/ciphertext pairs were used.

	Magma		MiniSAT	
Rounds	Time (s)	Mem (MB)	Time (s)	Mem (MB)
SMS4				
4	321.500	415.98	235.575	70.74
5	940.720	854.61	>6000	-
S-SMS4				
4	7.860	28.07	0.032002	16.28
5	20.420	59.27	0.100006	17.07
6	1188.470	1075.45	0.364022	18.36
7	71233.620	11733.60	6.044380	24.33

Table 4. Typical test results

The results for S-SMS4 seemed to indicate that the SAT solver could provide a more efficient solution, as is the case with only 4 rounds of SMS4. However, as verified in several tests, the SAT solver did not finish within 100 minutes with 5 rounds of SMS4. Thus, Magma holds more promise for real attacks on SMS4.

5.3. **Varied Number of Plaintext/Ciphertext Pairs.** One test I ran varied the number of plaintext/ciphertext pairs. The $X^2Y - X = 0$, $Y^2X - Y = 0$ representation of the inversion was used. Results are in table 5. In the cases of 6 and 7 rounds with only 1 pair, Magma ran out of memory for both cipher types. All other missing cases indicate that either Magma did run out of memory or the solution required more than 2 hours to complete. (The solver was killed after two hours.) It is possible that some of these missing cases would run out of memory if run to completion.

We can see from the data for 5 and 6 rounds that the optimal number of plaintext/ciphertext pairs should be two. Furthermore, as shown in figure 1, the time complexity appears quadratic, while the memory complexity appears approximately linear.

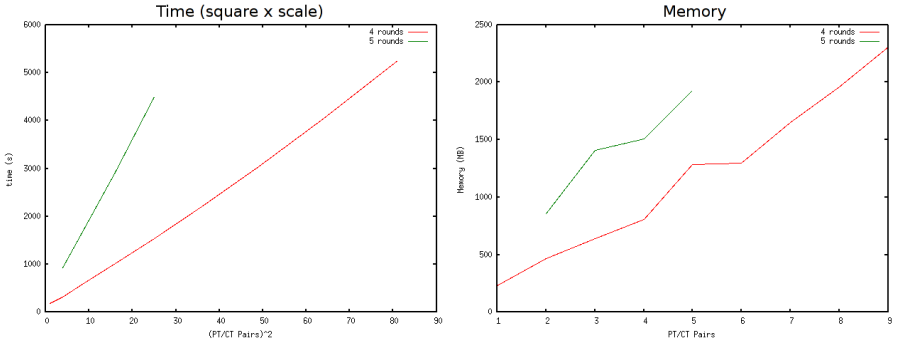

FIGURE 1. Time and memory complexity with varying pairs (full SMS4)

5.4. **Elminating the Key Schedule.** The previous section demonstrated that adding more redundant equations to solve for the same key may not result in a faster solution. The added variables increased the complexity of the system, which likely accounts for the slow down. From this observation, another simplificiation of the cipher that may be attempted is to ignore the key schedule rather than include it. In the case of one plaintext-ciphertext pair, this yields a unique solution only in the case of four rounds. Even in the case of five rounds, numerous solutions are typically returned. However, with more than one plaintext-ciphertext pair, the results are usually either unique or contain a small number of solutions. (The actual key can be obtained from the first four broken round keys by using a process similar to SMS4 decryption. Thus, modelling the key schedule is not actually necessary to determine the key.)

Test results for both SMS4 and S-SMS4 are included in table 6. The data which includes the key schedule is the same as in the section comparing numbers of sets, and it is included for comparison. For most cases, removing the key schedule seems to provide a significant improvement in solution time. However, the case of 6

		SMS4		S-SMS4	
Rounds	PT/CT Pairs	Time (s)	Mem (MB)	Time (s)	Mem (MB)
4	1	187.120	230.06	1.510	12.09
	2	325.910	467.04	3.009	19.16
	3	612.629	640.23	5.299	20.81
	4	1011.000	807.10	12.050	28.07
	5	1536.049	1286.81	18.519	39.14
	6	2204.170	1295.34	18.050	37.65
	7	3032.769	1652.99	25.170	76.85
	8	4043.800	1960.68	34.179	83.84
	9	5251.460	2306.81	45.079	131.16
5	1	2602.820	14457.93	171.680	225.17
	2	926.129	860.56	8.259	20.59
	3	1768.740	1408.27	12.359	40.31
	4	2920.429	1509.54	20.870	59.03
	5	4500.359	1929.46	32.859	78.38
	6	-	-	48.840	94.50
	7	-	-	69.459	111.22
	8	-	-	95.439	193.74
	9	-	-	127.620	210.05
6	1	-	-	-	-
	2	2222.860	12344.91	325.750	545.61
	3	-	-	622.990	734.40
	4	-	-	1194.339	1106.62
	5	-	-	2158.129	1327.62
	6	-	-	3703.400	1909.23
	7	-	-	6058.989	2572.89
	8	-	-	-	-
	9	-	-	-	-
7	1	-	-	-	-
	2	-	-	6082.100	5503.66
	3	-	-	6968.439	3395.06

TABLE 5. SMS4 and S-SMS4 tested with Magma

rounds (SMS4) seems to indicate that for that many rounds, the presence of the key schedule is necessary. This is not surprising, because with only four rounds, a relatively simple algebraic process can determine the key, due to the fact that all the values of X_i are known. Thus, the key schedule provides unnecessary extra information in the case of four rounds, and seemingly for five as well according to the data. In the case of S-SMS4, removing the key schedule only seems to help, probably due to its simpler algebraic structure as compared to the full cipher.

5.5. **Varied S-box Representations.** I also tested with both representations of the S-box described above, as well as the case of the incomplete model $X^2Y - X = 0$ for comparison. Surprisingly, Magma ran out of memory for any representation other than the fully correct model, even in cases where the system was correct as

		With Key Schedule		No Key Schedule	
Rounds	Pairs	Time (s)	Mem (MB)	Time (s)	Mem (MB)
SMS4					
4	2	325.910	467.04	26.359	108.95
	3	612.629	640.23	122.810	197.23
	4	1011.000	807.10	250.110	317.42
	5	1536.049	1286.81	479.250	617.46
	6	2204.170	1295.34	637.779	902.41
	7	3032.769	1652.99	1248.789	1212.59
	8	4043.800	1960.68	1499.650	1766.69
	9	5251.460	2306.81	2130.889	1368.74
5	2	926.129	860.56	104.579	191.09
	3	1768.740	1408.27	316.129	468.84
	4	2920.429	1509.54	730.720	886.63
	5	4500.359	1929.46	1383.690	1266.73
6	2	2222.860	12344.91	>7200	-
S-SMS4					
4	2	3.009	19.16	0.770	9.36
	3	5.299	20.81	1.459	12.76
	4	12.050	28.07	2.700	19.61
	5	18.519	39.14	4.639	22.44
	6	18.050	37.65	7.769	27.19
	7	25.170	76.85	12.089	36.93
	8	34.179	83.84	18.170	37.40
	9	45.079	131.16	25.339	71.13
5	2	8.259	20.59	1.480	11.64
	3	12.359	40.31	3.529	16.57
	4	20.870	59.03	7.669	28.48
	5	32.859	78.38	13.929	45.65
	6	48.840	94.50	23.629	65.23
	7	69.459	111.22	36.649	71.41
	8	95.439	193.74	53.859	88.65
	9	127.620	210.05	77.370	160.93
6	2	325.750	545.61	61.679	102.76
	3	622.990	734.40	127.390	176.76
	4	1194.339	1106.62	303.939	277.16
	5	2158.129	1328.62	723.850	445.93
	6	3703.400	1909.23	1452.930	843.32
	7	6058.989	2572.89	2619.429	1241.78
7	2	6082.100	5503.66	1968.109	2282.16
	3	6968.439	3395.06	2634.119	958.47

TABLE 6. Comparison to determine importance of key schedule

verified by using the SAT solver instead. However, I did discover that in the case of four rounds with no key schedule, Magma did seem to work. In the case of 5

rounds, Magma runs to completion, but finds an empty variety, even though the SAT solver finds the correct key given the same set of equations.

In the case of 4 rounds, using only $X^2Y - X = 0$ for two plaintext/ciphertext pairs finishes in 22.900 s, whereas the standard representation requires 26.359 seconds. The $XY - 1 = 0$ representation runs in 22.950 s. Thus, the three representations seem almost equivalent. With only one plaintext/ciphertext pair, $X^2Y - X = 0, Y^2X - Y = 0$ runs in 4.000 s, while $XY - 1 = 0$ runs in 3.630. These particular cases seem to indicate that even if the system did work as expected, the various S-box representations are mostly equivalent. Thus, only a small speedup can be obtained from $XY - 1 = 0$.

6. GF(2^8) Attacks

Using the method of [8], I wrote a Magma program to test the attack over GF(2^8). As discussed in [4], [11], and [1], XSL would not be expected to outperform a good Gröbner basis basis algorithm such as F4. Thus, Magma's builtin F4 algorithm is used, rather than XSL. The equation system from [8] is used almost unaltered. However, there is an off-by-one error in their indexing of all X variables (resulting in the use of X_{-1} in the first round), and their model of the S-box is taken from [9] unaltered and does not take reversal into account. Thus, I increased the index of the X variables by one and utilized my corrected S-box model from section 2.1. This resulted in a system that produced correct results for 4 rounds, with runtimes consistently between 240 and 250 seconds, and using 283.53MB of RAM each time. Because this attack used only one plaintext/ciphertext pair, and it has a time measurement between the values for one and two rounds with the similar GF(2) attack (187.120 and 325.910 s, respectively), with similar results for memory, the GF(2^8) attack seems to have similar efficiency to the GF(2) attack, at least for four rounds. However, upon testing with more rounds on my system, Magma runs out of memory, so I was unable to determine how the behavior varies with more rounds. Thus, I was not able to determine whether the speed claims in [8] are consistent with experiments, other than that their prediction of 2^{59} for the attack on four rounds is far too high. Further experimentation on a system with more available RAM is necessary.

7. Further Work

There are many possible improvements which I have not had time to explore. One area of exploration is mixing guessing with finding results algebraically. Either some of the key bits or some of the intermediate variables could be guessed. Experiments could be performed to determine the optimal number and choice of variables to guess to speed up the attack.

Another possible improvement would be to find the Gröbner basis of some part of the cipher, such as the S-box, and use that in place of the direct representation. Thus, some of the work of the solver would be done ahead of time. When we performed this experiment on DES during the summer REU portion of my research, it actually caused a slowdown. However, experimenting with the effects of different portions of precomputation on different portions of the cipher could yield fruitful results.

The ordering of the variables is significant in the case of the F4 algorithm used by Magma. Thus, experiments could be performed with a different ordering of

the variables, either when the system is written, or by using a different ordering algorithm (e.g. lexicographical, graded lexicographical, etc.) when running the algorithm.

Further work on the GF(2^8) attack is necessary. Experiments should be peformed with sufficient RAM to test several numbers of rounds, so that the increase in complexity can be measured. Many of the same alterations made with the GF(2) model, such as changing the S-box representation and varying the number of plaintext/ciphertext pairs could also be tested with the GF(2^8) model, to verify whether they have the same effect.

Finally, several of the unexpected results using Magma could be further explored. For example, why does adding the field equations slow down Magma and increase the memory usage, rather than leading to a more efficient solution? Also, why does the solver fail to find the correct solution with varied S-box representations even though the SAT solver reveals the system of equations is correct? Perhaps the same equations could be tested using another solver using F4 to see if the same results occur, to determine whether the issue is an implementation problem with Magma or some property of the algorithm.

References

[1] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between xl and gröbner basis algorithms. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 2004.

[2] Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over gf(2) via sat-solvers. Cryptology ePrint Archive, Report 2007/024, 2007. `http://eprint.iacr.org/`.

[3] Wieb Bosma, John J. Cannon, and Catherine Playoust. The magma algebra system i: The user language. *J. Symb. Comput.*, 24(3/4):235–265, 1997.

[4] Carlos Cid and Gaëtan Leurent. An analysis of the xsl algorithm. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005.

[5] Nicolas Courtois and Gregory V. Bard. Algebraic cryptanalysis of the data encryption standard. In Steven D. Galbraith, editor, *IMA Int. Conf.*, volume 4887 of *Lecture Notes in Computer Science*, pages 152–169. Springer, 2007.

[6] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yulian Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[7] Whitfield Diffie and George Ledin (translators). Sms4 encryption algorithm for wireless networks. Cryptology ePrint Archive, Report 2008/329, 2008. `http://eprint.iacr.org/`.

[8] Wen Ji and Lei Hu. New description of sms4 by an embedding over gf(2^8). In K. Srinathan, C. Pandu Rangan, and Moti Yung, editors, *INDOCRYPT*, volume 4859 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2007.

[9] Fen Liu, Wen Ji, Lei Hu, Jintai Ding, Shuwang Lv, Andrei Pyshkin, and Ralf-Philipp Weinmann. Analysis of the sms4 block cipher. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 158–170. Springer, 2007.

[10] Jiqiang Lu. Attacking reduced-round versions of the sms4 block cipher in the chinese wapi standard. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS*, volume 4861 of *Lecture Notes in Computer Science*, pages 306–318. Springer, 2007.

[11] Bo-Yin Yang, Jiun-Ming Chen, and Nicolas Courtois. On asymptotic security estimates in xl and gröbner bases-related algebraic cryptanalysis. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *ICICS*, volume 3269 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2004.