# Algebraic Cryptanalysis of SMS4: Gröbner Basis Attack and SAT Attack Compared

Jeremy Erickson[1], Jintai Ding[2,3], and Chris Christensen[4]

[1] The University of North Carolina at Chapel Hill, Chapel Hill, NC 27514
jerickso@cs.unc.edu[⋆]
[2] The University of Cincinnati, Cincinnati, OH 45221
jintai.ding@uc.edu
[3] South China University of Technology, Guangzhou, China
[4] Northern Kentucky University, Highland Heights, KY 41099
christensen@nku.edu

**Abstract.** The SMS4 block cipher is part of the Chinese WAPI wireless standard. This paper describes the specification and offers a specification for a toy version called simplified SMS4 (S-SMS4). We explore algebraic attacks on SMS4 and S-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. A comparison of SAT and Gröbner basis attacks is provided.

## 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 [1]. This attack has also been attempted on other ciphers such as DES [2]. Algebraic cryptanalysis has been shown very effective for families of stream ciphers.

The SMS4 cipher was designed 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 [3]. However, as shown in [4], SMS4 itself has a simple algebraic structure, similar to AES. Thus, we 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 both GF(2) and GF($2^8$), as shown in [5] and [6], but their papers present only theoretical results, not any experimental results. We have found our experimental results contradictory to their analysis.

In this paper we present an attempt to attack SMS4 with algebraic attacks over GF(2), including several potential alterations. We also present preliminary

[⋆] Much of this work was done while Jeremy Erickson was an undergraduate at Taylor University, Upland, IN 46989.

results of attacks based on GF($2^8$) as in [5]. Both types of attacks were implemented using the Magma computer algebra system ([7]), as well as using the MiniSAT boolean satisfiability solver in some cases. Preliminary results indicate that Magma is more effective than MiniSAT for the full cipher, while MiniSAT outperforms Magma in the simplified version of the cipher created for our experiments. No effective guess-and-determine attack strategy was found, but some preliminary results were determined. We found evidence that the results in [5] and [6] are very inaccurate. This casts serious doubt about the their theoretical analysis and the previous analysis they relied on.

This paper begins with a description of the SMS4 algorithm itself, including a description of the simplified variant. We also describe implementation details for the attack itself, as well as experimental results. At the end we discuss the implications of these results.

## 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. An English translation of the official specification is provided at [8]. We also describe the cipher here. 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 [4], the S-box can also be 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

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

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

| | 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 |

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, we propose 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 (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, we propose 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 [4]. The new S-box is designed to transform a 4-bit vector to another 4-bit vector, but otherwise follows (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 (2), however, we derive the following matrices in a similar manner:

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

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

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

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

The inversion step is similar to full SMS4, except that we use 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.

**Table 2.** S-SMS4 S-box, in the same form as Table 1

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

Because the S-SMS4 S-box only accepts one byte as input, which is two nibbles, we split $X \in (\mathrm{GF}(2))^8$ 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. The $A$ matrix is simply a cyclic matrix with the same first row as the beginning of the first row in full SMS4. This first row seemed as appropriate as any.

## 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,

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

The specific parameters in these equations are fairly arbitrary; the attack could be generalized to use different numbers. We attempted to attain the essence of the full SMS4 transformations. In $L(x)$, the parameters have a GCF of 2, as in full SMS4. In $L'(x)$, the factors are prime numbers, as in full SMS4. The longer length of $L(x)$ could not be effectively preserved, due to the smaller vectors.

## 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:

$$Y_0 = K_0 \oplus 0xa3,$$

$$Y_1 = K_1 \oplus 0xb1,$$

$$Y_2 = K_2 \oplus 0xba,$$

$$Y_3 = K_3 \oplus 0xc6.$$

This initialization vector is simply the beginning of the full SMS4 initialization vector; however, the specific numbers cannot alter the difficulty of breaking the cipher. We could always solve the equations with any initialization vector (including 0) and then adjust the results at the end by adding the appropriate constant.

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 (5) holds. Note that we could have written $ck_{i,j} = si + tj \mod 16$ with other values of $s$ and $t$; our choice was arbitrary. However, this choice will hopefully generate similar mixing. In the original cipher $s = 4t$, but both $s$ and $t$ are much smaller than 256, so no directly analogous relationship exists.

### 3.4 The S-SMS4 Round Function

With the notations in this section, (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.

### 4.1 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($2^4$). (These equations are derived from $XY - X = 0$ and $YX - Y = 0$, so that they work correctly when inverting zero).

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.

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

$$Y_{r+4} = Y_r \oplus EK_r.$$

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.

This system shows intermediate variables for every step of the operation. In practice, faster times were obtained by combining several linear steps (by substituting the expression from the previous result rather than variables.) The best times were obtained using intermediate variables for $DK$ and $Y$ variables, substituting in expressions in all other cases.

## 4.2 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.

$$B_{p,r} = X_{p,r+1} \oplus X_{p,r+2} \oplus X_{p,r+3} \oplus Y_{r+4},$$

$$Z_{p,r,i} = A_1 \cdot B_{p,r,i} \oplus C_1,$$

$$D_{p,r,i} = A_2 \cdot W_{p,r,i} \oplus C_2,$$

$$E_{p,r} = L(D_{p,r}),$$

$$X_{p,r+4} = X_{p,r} \oplus E_{p,r}.$$

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.

Also as with the key schedule, fewer intermediates are actually needed to solve the system. Only the $X$ variables were left as intermediates in the final representation, and inverses of the last three equations were used so that we could still use the implicit representation of inversion.

## 4.3 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, we wrote a Perl script using the method of [9]. 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®Xeon®dual-core CPU running at 2.00 GHz. The system had 32 GB of RAM and ran 64-bit CentOS 5. Magma 2.15-10 and MiniSAT 2.0 were used.

For small numbers of rounds, we tested Magma and MiniSAT both with the same systems of equations. We discovered that SAT solvers did not finish within several hours. However, we have previous test data from the system with more intermediate variables running on a Core$^{\text{TM}}$2 Duo system running at 2.40 GHz with 16 GB of RAM, running Ubuntu 8.0.4.1. These systems actually had fewer intermediate variables when converted into SAT form, due to the intermediate variables created converting long sums into SAT. All tests were done using two plaintext/ciphertext pairs, which experiments demonstrated to be clearly optimal for Magma. Results are in Table 3 and graphed in Fig. 1.

**Table 3.** Typical test results

| | Magma | | MiniSAT | |
|---|---|---|---|---|
| Rounds | Time (s) | Mem (MB) | Time (s) | Mem (MB) |
| SMS4 | | | | |
| 4 | 2.370 | 100.20 | 235.575 | 70.74 |
| 5 | 7.720 | 207.68 | >6000 | - |
| S-SMS4 | | | | |
| 4 | 0.030 | 8.86 | 0.032002 | 16.28 |
| 5 | 0.070 | 10.66 | 0.100006 | 17.07 |
| 6 | 11.650 | 128.15 | 0.364022 | 18.36 |
| 7 | 81.780 | 555.34 | 6.044380 | 24.33 |

### 5.1 Magma vs. SAT Solver

The results for S-SMS4 seemed to indicate that the SAT solver could provide a more efficient solution, but this was not true for full SMS4. Also, 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. We believe that the most likely explanation for this result is that while SAT works primarily through guessing, which is effective when there are a small number of variables, F4 works through structured algebraic methods, which have higher initial cost but are better amortized by larger, more complicated sets of equations.

The computational complexity of the operations on full SMS4 cannot reliably be determined from two data points. However, our hope is that the data from simplified SMS4 is representative of the asymptotic complexity. The complexity of the F4 attack seems to be exponential, with a large jump between 5 and 6 rounds of the cipher. For four rounds we expect the result to be obtained quickly, because the entire state vector $X$ is known. It seems that the gap in the state vector becomes large enough to be highly significant when we reach

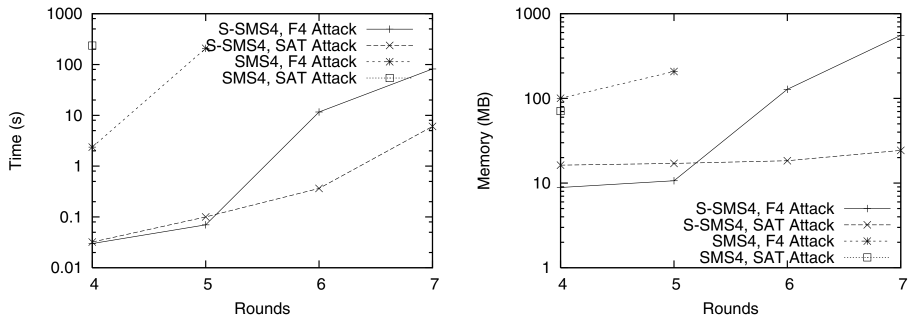

**Fig. 1.** Typical test results

6 rounds. This is likely the reason that full SMS4 could not be broken for 6 or more rounds on our system. The complexity of the SAT attack seems to be more directly exponential, without the significant gap.

## 6 Guess-and-Determine Attack

A common technique for algebraic cryptanalysis is to guess some of the variables, and then using algebraic cryptanalysis to solve others. If guessing $n$ variables reduces runtime by a factor greater than $2^n$, then the attack can save time, provided that an incorrect guess solves as quickly as a correct guess.

In order to test the guess-and-determine attack on SMS4, we decided to use 7 rounds of simplified SMS4. Using 6 or more rounds of full SMS4 would have been more ideal, but we were not able to break this many rounds in a reasonable amount of time for the tests. Table 4 in the appendix provides results from guess attacks.

As might be expected, no significant benefit was gained from guessing any single bit. Even testing guessing with multiple bits, no attempt at guessing $n$ bits gained an improvement factor of greater than $2^n$ in either time or memory. However, there were appreciable differences between guessing different values. It appears that guessing the first and/or last round keys achieve the most significant speedup.

From this data, no evidence indicates that guess-and-determine attacks can be effective against SMS4. However, the behavior may be different for significant numbers of rounds in the full cipher. For these systems, these tests indicate that guessing from the first and last round keys probably offers the most promise.

## 7 GF($2^8$) Attacks

Using the method of [4], which was repeated in [5], we wrote a Magma program to test the attack over GF($2^8$). As discussed in [10], [11], and [12], XSL would not be expected to outperform a good Gröbner basis algorithm such as F4. Thus, Magma's builtin F4 algorithm is used, rather than XSL. The equation system from [5] 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 [4] unaltered and does not take reversal into account. Thus, we increased the index of the $X$ variables by one and utilized our corrected S-box model from section 2.1. This resulted in a system that produced correct results for 4 rounds. Performance was consistent with a GF(2) system tested without the field equations, but with all intermediate variables, which is significantly slower than the GF(2) system with field equations or with fewer intermediate variables. Runtimes were 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 pairs with the similar GF(2) attack (187.120 and 325.910 s, respectively, when the field equations are removed), with similar results for memory, the GF($2^8$) attack seems to have similar efficiency to the GF(2) attack with all intermediate variables, at least for four rounds. However, upon testing with more rounds on our system, Magma runs out of memory. This likely indicates that the complexity increase from increasing the number of rounds greatly exceeds what is predicted in [5]. We also see that their complexity prediction of $2^{59}$ for the attack on four rounds is far too high, which also demonstrates a flaw in their model. Thus, experiments contradict their predictions. However, to determine the actual behavior of the system for larger numbers of rounds, further experimentation on a system with more RAM is necessary.

The authors of [5] have updated their analysis in [6], including adding an analysis of SMS4 over GF(2). Their complexity prediction of $2^{90}$ for the four round attack over GF($2^8$) is even higher, and their estimates for GF(2) attacks at 4 rounds ($2^{102}$) and 5 rounds ($2^{121}$) show drastically higher complexity, and higher increase in complexity, than we observed in our experiments. Also, unlike their claim, the growth does appear from our experiments to be exponential for simplified SMS4, so we suspect it is indeed exponential for full SMS4 over GF(2). Thus, the equations used to analyze the behavior of XL seem not to line up with the actual running time of the faster F4 algorithm at all.

## 8 Conclusion and Discussion

At this point, we have not demonstrated any practical attack on SMS4 which has great promise of efficiency for the full cipher. However, we still discovered useful results. It appears that F4 or a similar algorithm is likely to be more useful than a SAT solver for the full cipher. For the simplified algorithm, the SAT solver does appear to be more feasible. However, it could be that Magma's additional overhead to compute a Gröbner basis becomes less significant and could have better asymptotic complexity. Few intermediate variables are needed, but the round keys should be intermediate variables. Guess-and-determine attacks have not been shown to be effective, but may be effective if round key bits are guessed.

Also, breaking the system over GF($2^8$) does not appear to provide any benefit over solving over GF(2). Specifically, the claims of [5] and [6] are contradicted by our experimental results. This casts very serious doubts about the basis of their theoretical analysis.

There are many possible improvements which we have not had time to explore. There exist improved methods for converting equations into SAT solver input. In addition, not all methods and adjustments in [9] such as improving the optimal cutting number have been attempted. Thus, it is possible that the SAT solver attack could become more feasible with improved conversion.

Further work on the GF($2^8$) attack is possible. Experiments could be performed with sufficient RAM to test several numbers of rounds, so that the increase in complexity can be measured. This would allow a more substantial comparison of GF(2) and GF($2^8$) implementations.

Overall, we believe any existing direct algebraic analysis will not work well in attacking SMS4 and new methods that could fully utilize the hidden algebraic structures need to be developed to attack SMS4 more efficiently in terms of algebraic cryptanalysis.

## Acknowledgments

This work is a result of the 2008 and 2009 National Science Foundation-sponsored Research Experience for Undergraduates (REU) in Mathematical Cryptology jointly offered by Northern Kentucky University and the University of Cincinnati. Thanks to the United States Air Force Office of Scientific Research for funding the REUs and to the Computational Algebra Group within the School of Mathematics and Statistics of the University of Sidney for supplying Magma for the REUs and for this project. The first author was also supported by AT&T, IBM, and Sun Corps.; NSF grants CNS 0834270 and CNS 0834132; ARO grant W911NF-09-1-0535; and AFOSR grant FA9550-09-1-0549. The second author would also like to thank partial support of Grant-60973131 from NSF China and the support of Taft Foundation. We also worked with Amber Rogers of Northern Kentucky University on creating SAT solver conversion code, and portions of the code used in creating SMS4 equations were written by Brian Nixon of the University of Michigan.

## References

1. Courtois, N., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
2. Courtois, N., Bard, G.V.: Algebraic cryptanalysis of the data encryption standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 152–169. Springer, Heidelberg (2007)
3. Lu, J.: Attacking reduced-round versions of the sms4 block cipher in the chinese wapi standard. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 306–318. Springer, Heidelberg (2007)

4. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.P.: Analysis of the sms4 block cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007)
5. Ji, W., Hu, L.: New description of sms4 by an embedding over gf($2^8$). In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 238–251. Springer, Heidelberg (2007)
6. Ji, W., Hu, L., Ou, H.: Algebraic attack to sms4 and the comparison with aes. In: International Symposium on Information Assurance and Security, vol. 1, pp. 662–665 (2009)
7. Bosma, W., Cannon, J.J., Playoust, C.: The magma algebra system i: The user language. J. Symb. Comput. 24(3/4), 235–265 (1997)
8. Diffie, W., Ledin, G. (translators): Sms4 encryption algorithm for wireless networks. Cryptology ePrint Archive, Report 2008/329 (2008), `http://eprint.iacr.org/`
9. Bard, G.V., Courtois, N.T., Jefferson., C.: 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/`
10. Cid, C., Leurent, G.: An analysis of the xsl algorithm. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 333–352. Springer, Heidelberg (2005)
11. Yang, B.Y., Chen, J.M., Courtois, N.: On asymptotic security estimates in xl and gröbner bases-related algebraic cryptanalysis. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 401–413. Springer, Heidelberg (2004)
12. Ars, G., Faugère, J.C., Imai, H., Kawazoe, M., Sugita, M.: Comparison between XL and gröbner basis algorithms. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 354–371. Springer, Heidelberg (2004)

## Appendix: Guess-and-Determine Attack Results

Table 4 contains results for the guess-and-determine attack on 7 rounds of simplified SMS4.

**Table 4.** Results of guess-and-determine attacks

| Attack | # Bits | Time (s) | RAM (MB) | Speedup | RAM Reduc. |
|---|---|---|---|---|---|
| No guessing | 0 | 83.210 | 556.43 | 1.00 | 1.00 |
| Entire key | 32 | 0.350 | 45.92 | 237.74 | 12.12 |
| First key bit | 1 | 84.940 | 552.74 | 0.98 | 1.01 |
| Key bit 12 (0) | 1 | 81.790 | 553.85 | 1.02 | 1.00 |
| Key bit 13 (1) | 1 | 81.160 | 551.19 | 1.03 | 1.01 |
| Last key bit | 1 | 81.340 | 545.80 | 1.02 | 1.02 |
| First two key bits | 2 | 81.400 | 549.32 | 1.02 | 1.01 |
| First and 12 key bits | 2 | 81.080 | 549.45 | 1.03 | 1.01 |
| First and 13 key bits | 2 | 80.440 | 546.67 | 1.03 | 1.02 |
| First and last key bit | 2 | 80.050 | 540.07 | 1.04 | 1.03 |
| First three key bits | 3 | 80.410 | 540.99 | 1.03 | 1.03 |
| First four key bits | 4 | 79.650 | 537.08 | 1.04 | 1.04 |
| First eight key bits | 8 | 90.560 | 580.41 | 0.92 | 0.96 |
| First and last four | 8 | 78.820 | 511.29 | 1.06 | 1.09 |
| RK2 bit 1 | 1 | 81.090 | 554.57 | 1.03 | 1.00 |
| RK2 bits 1, 2 | 2 | 78.150 | 556.41 | 1.06 | 1.00 |
| RK2 bit 1, RK4 bit 1 | 2 | 77.820 | 532.19 | 1.07 | 1.05 |
| RK1 (all bits) | 8 | 19.980 | 152.00 | 4.16 | 3.66 |
| RK2 (all bits) | 8 | 57.290 | 429.42 | 1.45 | 1.30 |
| RK3 (all bits) | 8 | 57.210 | 384.42 | 1.45 | 1.45 |
| RK4 (all bits) | 8 | 46.890 | 304.93 | 1.77 | 1.82 |
| RK5 (all bits) | 8 | 50.440 | 333.27 | 1.65 | 1.67 |
| RK6 (all bits) | 8 | 56.590 | 364.54 | 1.47 | 1.53 |
| RK7 (all bits) | 8 | 21.940 | 173.77 | 3.79 | 3.20 |
| RK1, RK7 (all bits) | 16 | 0.270 | 45.92 | 308.19 | 12.12 |
| RK1 bit 1 | 1 | 69.110 | 531.79 | 1.20 | 1.05 |
| RK1 bits 1, 2 | 2 | 65.760 | 476.22 | 1.27 | 1.17 |
| RK1 bits 1, 8 | 2 | 64.019 | 471.69 | 1.30 | 1.18 |
| RK1 bit 1, RK7 bit 1 | 2 | 68.989 | 512.97 | 1.21 | 1.08 |
| RK1 bits 1-2, 7-8 | 4 | 46.920 | 365.62 | 1.77 | 1.52 |
| RK1 bits 1-4 | 4 | 22.460 | 156.08 | 3.70 | 3.57 |
| RK1 bits 5-8 | 4 | 26.710 | 234.14 | 3.12 | 2.38 |
| RK7 bits 6-8 | 3 | 27.940 | 181.75 | 2.98 | 3.06 |
| R1 intermediates | 8 | 72.140 | 495.74 | 1.15 | 1.12 |
| R7 intermediates | 8 | 39.140 | 294.49 | 2.13 | 1.89 |
| RK1-7 bit 1 | 7 | 50.729 | 400.43 | 1.64 | 1.39 |
| RK1-2, RK6-7 bit 1 | 4 | 64.510 | 483.45 | 1.29 | 1.15 |
| RK1, RK7 bits 1-2 | 4 | 50.909 | 385.66 | 1.63 | 1.44 |