

An Analysis of the RC4 Family of Stream Ciphers against Algebraic Attacks

Kenneth Koon-Ho Wong¹Gary Carter¹Ed Dawson¹

¹Information Security Institute
Queensland University of Technology
Brisbane, Australia

Email: {kk.wong,g.carter,e.dawson}@qut.edu.au

Abstract

To date, most applications of algebraic analysis and attacks on stream ciphers are on those based on linear feedback shift registers (LFSRs). In this paper, we extend algebraic analysis to non-LFSR based stream ciphers. Specifically, we perform an algebraic analysis on the RC4 family of stream ciphers, an example of stream ciphers based on dynamic tables, and investigate its implications to potential algebraic attacks on the cipher. This is, to our knowledge, the first paper that evaluates the security of RC4 against algebraic attacks through providing a full set of equations that describe the complex word manipulations in the system. For an arbitrary word size, we derive algebraic representations for the three main operations used in RC4, namely state extraction, word addition and state permutation. Equations relating the internal states and keystream of RC4 are then obtained from each component of the cipher based on these algebraic representations, and analysed in terms of their contributions to the security of RC4 against algebraic attacks. Interestingly, it is shown that each of the three main operations contained in the components has its own unique algebraic properties, and when their respective equations are combined, the resulting system becomes infeasible to solve. This results in a high level of security being achieved by RC4 against algebraic attacks. On the other hand, the removal of an operation from the cipher could compromise this security. Experiments on reduced versions of RC4 have been performed, which confirms the validity of our algebraic analysis and the conclusion that the full RC4 stream cipher seems to be immune to algebraic attacks at present.

1 Introduction

Algebraic attacks on stream ciphers, introduced by Courtois & Meier (2003) and Courtois (2004), are attacks in which the keystream is used to construct a system of multivariate polynomial equations with the keys or initial states of the stream ciphers as variables. Solving the system of equations amounts to recovering the keys or initial states. This method of attack was initially applied to block ciphers and public key cryptosystems (Courtois 2001, Courtois & Pieprzyk 2002). Algebraic analysis has been demonstrated at times to be a very useful tool for stream ciphers based on linear feedback shift registers (LFSRs). Several well known LFSR-based stream ciphers have fallen to

algebraic attacks (Al-Hinai et al. 2006, Armknecht & Krause 2003, Cho & Pieprzyk 2004, Courtois 2004, 2003, Courtois & Meier 2003, Wong et al. 2006). It is therefore appropriate to extend algebraic analysis to other well-known ciphers that are not based on LFSRs, in order to evaluate the possibility of successful algebraic attacks on them. This is the primary aim of this paper.

In this paper, we perform an algebraic analysis of the RC4 family of stream ciphers (Schneier 1996), which is a word-based stream cipher based on dynamic tables. We show how valid algebraic relations among the internal states of the cipher are obtained, in order to form a full system of equations describing the cipher. We then investigate the equations and evaluate the resistance of RC4 to algebraic attacks. To the best of our knowledge, this is the first paper on a full algebraic analysis of RC4. The types and number of equations generated from the cipher are discussed, and can be used as a guide for the level of security of RC4 against algebraic attacks, both at present and for future reference, as solution methods for large equation systems may become more efficient over time. The methods of analysis and results presented here could also be extended to RC4 variants, such as RC4A (Paul & Preneel 2004) and VMPC (Zoltak 2004).

To date, RC4 remains a widely used stream cipher in network and wireless applications, as well as in many commercial products. It is a word-based stream cipher, whose simple and elegant design by Rivest in 1987 had been kept secret until 1994. After the specification of the RC4 was revealed, the cipher became the target for cryptanalysis. The first published cryptanalysis of RC4 was by Golić (1997), followed by a number of interesting ones (Knudsen et al. 1998, Fluhrer & McGrew 2000, Mantin & Shamir 2001, Paul & Preneel 2003, 2004). Weaknesses identified in the RC4 cipher have motivated the proposal of several strengthened versions of RC4, such as RC4A (Paul & Preneel 2004). Other researchers were inspired by the design of the cipher and proposed stream ciphers based on the design of RC4 such as the 32 and 64-bit RC4 (Gong et al. 2005) and VMPC (Zoltak 2004). However, distinguishing attacks have since been shown to be effective on both the original and strengthened proposals of RC4 and on new RC4 variants (Maximov 2005, Tsunoo et al. 2005). Cryptanalysis of RC4 remains an active topic with recent developments in improved state and key recovery attacks (Biham & Carmeli 2008, Maximov & Khovratovich 2008, Basu et al. 2009).

In order to provide an algebraic analysis of the RC4 stream cipher, we first show how algebraic relationships can be obtained for the operations within the cipher. The three main operations used in RC4, namely word addition, state extraction and state permutation will be analysed. Algebraic representations

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Australasian Information Security Conference (AISC2010), Brisbane, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 103, Colin Boyd and Willy Susilo, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

for each of these three operations will be derived, and some of their properties will be discussed. Then, we convert these operations into valid expressions relating the internal states and keystream, and construct a system of polynomial equations from them. The solution of the system would amount to the recovery of the initial states. We analyse how each of the operations contribute to the number of equations generated, their respective degrees and form. We arrive at an observation that these three main operations contribute uniquely to the system of equations derived from the RC4 stream cipher, which give a high level of resistance to algebraic attacks only when combined into one cipher.

The paper is organised as follows. Section 2 provides a description of RC4. In section 3, we show how algebraic relations for the operations involved in RC4 can be obtained. In section 4, we construct equations that relate the initial states to the keystream for RC4. Section 5 provides a summary and analysis of the results, which are then used to determine the security of RC4 against algebraic attacks. Section 6 gives an account on actual attempts of algebraic attacks on the cipher using the methods presented. Section 7 concludes the paper.

2 Description of RC4

The RC4 family of stream ciphers is a word-based stream cipher, which has a very large internal state space compared to the key size. For a word size of n bits, it consists of a permutation table of 2^n words and two pointers i, j of one word each. The total internal state space of RC4 is therefore of size $\log_2(2^n!(2^n)^2)$ bits. For the common implementation with $n = 8$, this is approximately 1700 bits. Two algorithms govern the RC4 stream cipher, namely the key scheduling algorithm (KSA) and the pseudo-random generation algorithm (PRGA). In the KSA, a secret key k is used to load and mix the internal states S_i of the register S , resulting in S having some permutation of the 2^n possible n -bit words. The PRGA then proceeds to generate keystream using the states obtained from the KSA. The KSA and PRGA for RC4 are shown in Figure 1. The operations described in the pseudocode are wordwise and the keystream output is denoted by z .

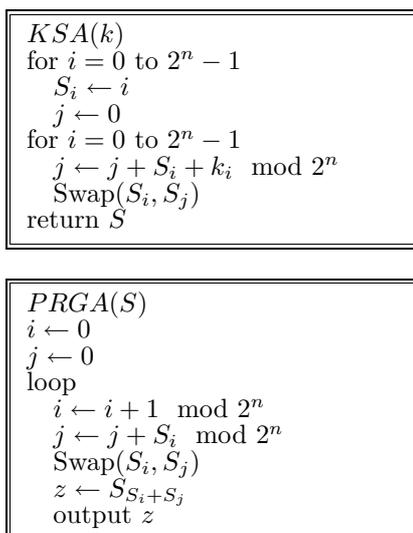


Figure 1: The KSA of RC4 (top), The PRNG of RC4 (bottom).

During the KSA, the identity permutation $(0, 1, \dots, 2^n - 1)$ is loaded into the register S . The

secret key k is then used to initialize S to a random permutation by shuffling the words in S according to the KSA. Once the KSA is complete, the cipher is ready for keystream generation. The PRGA is used to produce pseudo-random keystream words derived from the permutations in S . Each iteration of the PRGA loop produces one output word z , which constitutes n bits of keystream. In this paper, we consider the cipher from the start of the PRGA, to arrive at an initial state recovery algebraic analysis and attack, where the initial state is the permutation in S at that time.

3 Algebraic Analysis of RC4

The RC4 stream cipher of word size n uses one register S of length $2^n - 1$ with an n -bit word representing each state of S . Commonly, RC4 is used with $n = 8$. However, we will present an algebraic analysis that is applicable for arbitrary n . Before key initialisation, the states of S are set such that

$$S = (0, 1, \dots, 2^n - 1).$$

The cipher then initialises according to the KSA, which depends on the key k used. After the KSA, the register S arrives at its initial state S^0 , such that

$$S^0 = (x_0, x_1, \dots, x_{2^n-1}),$$

where x_i are n -bit words represented as elements in $\mathbb{Z}/2^n\mathbb{Z}$. Throughout this paper, we utilise the canonical isomorphism between the residue class ring $\mathbb{Z}/2^n\mathbb{Z}$ representing integers modulo 2^n and the product ring \mathbb{F}_2^n representing the bit strings of those integers, so that all equations describing RC4 are generated as polynomials with coefficients in \mathbb{F}_2 , with the word variables in $\mathbb{Z}/2^n\mathbb{Z}$ also split into bit variables in \mathbb{F}_2 . From here onwards, for $u \in \mathbb{Z}/2^n\mathbb{Z}$ and $0 \leq b \leq n-1$, let $u_{(b)} \in \mathbb{F}_2$ be the b -th least significant bit (LSB) of $u \in \mathbb{Z}/2^n\mathbb{Z}$, and $\mathbf{u} = (u_{(0)}, u_{(1)}, \dots, u_{(n-1)}) \in \mathbb{F}_2^n$ be the binary vector representing u . Additionally, to denote the b -th least significant bit of a state k of register S , we use the notation $S_{k,(b)}$. While the ring $\mathbb{Z}/2^n\mathbb{Z}$ or the extension field \mathbb{F}_{2^n} are possible candidates for this algebraic analysis, we have chosen not to use them due to their associated cumbersome representations and manipulations, compared to the much simpler arithmetic in \mathbb{F}_2 . In addition, there exists some technical difficulties for using these structures, as will be explained in the successive sections. We now derive the algebraic expressions of operations involved in RC4.

3.1 State Extraction

The value of one state S_i in the register S at position i is at times needed. As i is considered unknown, it is not possible to simply represent it as S_i in an algebraic analysis. Instead, we must derive an algebraic expression that extracts the correct word in S for any possible value of i . This state extraction operation is algebraically equivalent to evaluating the piecewise expression

$$S_i = \begin{cases} S_0, & i = 0 \\ S_1, & i = 1 \\ \vdots & \\ S_{2^n-1}, & i = 2^n - 1, \end{cases}$$

where S and i can be unknown. This expression can be split into n independent expressions, one for each

bit $S_{i,(b)}$ of S_i , where $0 \leq b \leq 1$. Each of these expressions is dependent on all bits $i_{(0)}, i_{(1)}, \dots, i_{(n-1)}$ of i , such that

$$S_{i,(b)} = \begin{cases} S_{0,(b)}, & \mathbf{i} = (0, 0, \dots, 0) \\ S_{1,(b)}, & \mathbf{i} = (0, 0, \dots, 1) \\ \vdots & \\ S_{2^n-1,(b)}, & \mathbf{i} = (1, 1, \dots, 1) \end{cases}, \quad 0 \leq b \leq n-1.$$

Since the variables are in \mathbb{F}_2 , the piecewise expression for each bit can then be arranged analogously as a boolean expression and written as

$$S_{i,(b)} = \sum_{u=0}^{2^n-1} \left(S_{u,(b)} \prod_{b=0}^{n-1} (i_{(b)} + u_{(b)} + 1) \right).$$

Here, the expression in the brackets is one when $i = u$ and zero otherwise, resulting in the correct bits of S_i being evaluated. If we instead use the residue class ring $\mathbb{Z}/2^n\mathbb{Z}$ to describe state extraction operation, it is not possible to convert the piecewise expression into a single algebraic expression, since we are not able to find a function that gives a nonzero value for the correct index i and give zeros for the other indices. Therefore, this operation has prevented us from using word-based algebraic analysis in $\mathbb{Z}/2^n\mathbb{Z}$. The above expression is ordered by S_u with polynomials in $i_{(b)}$ as coefficients. This can be rewritten to order by degrees of monomials in $i_{(b)}$ with S_k as coefficients as

$$S_{i,(b)} = \sum_{e=0}^{2^n-1} \left(\prod_{f=0}^{n-1} i_{(f)}^{e_{(f)}} \left(\sum_{k=1}^{2^n-1} S_{k,(b)} \left(\prod_{g=0}^{n-1} e_{(g)} (i_{(g)} + 1) + 1 \right) \right) \right).$$

Consider a bit position $S_{i,(b)}$ throughout the entire register. Due to the fact that S is a permutation of its initial states, the values of that position must contain an equal number of zeros and ones, which means that

$$\sum_{k=0}^{2^n-1} S_{k,(b)} = 0, \quad 0 \leq b \leq n-1.$$

Therefore, the expressions for state extraction can be reduced to

$$S_{i,(b)} = \sum_{e=0}^{2^n-2} \left(\prod_{f=0}^{n-1} i_{(f)}^{e_{(f)}} \left(\sum_{k=1}^{2^n-1} S_{k,(b)} \left(\prod_{g=0}^{n-1} e_{(g)} (i_{(g)} + 1) + 1 \right) \right) \right).$$

This removes the degree $n+1$ terms in the expression, and we are left with expressions of maximum degree n for the state extraction of S_i . For example, with $n = 2$ the expression is

$$\begin{aligned} S_{i,(b)} &= i_{(0)}i_{(1)}S_{0,(b)} + i_{(0)}i_{(1)}S_{1,(b)} \\ &\quad + i_{(0)}i_{(1)}S_{2,(b)} + i_{(0)}i_{(1)}S_{3,(b)} \\ &\quad + i_{(0)}S_{0,(b)} + i_{(0)}S_{3,(b)} \\ &\quad + i_{(1)}S_{1,(b)} + i_{(1)}S_{3,(b)} + S_{3,(b)} \\ &= i_{(0)}S_{0,(b)} + i_{(0)}S_{3,(b)} \\ &\quad + i_{(1)}S_{1,(b)} + i_{(1)}S_{3,(b)} + S_{3,(b)}, \end{aligned}$$

since $S_0 + S_1 + S_2 + S_3 = 0$. The expression for $S_{i,(b)}$ is therefore of degree 2.

3.2 Word Addition

The addition operation in RC4 is defined as word addition modulo 2^n , which we denote as $+_{2^n}$, as opposed to $+$, which is understood as addition modulo 2 throughout this paper, unless otherwise indicated. To obtain the equivalent operations using bit values in \mathbb{F}_2 , we use the additive group isomorphism between $\mathbb{Z}/2^n\mathbb{Z}$ and \mathbb{F}_2^n induced by addition on the binary digits of integers modulo 2^n . Let $u, v, w \in \mathbb{Z}/2^n\mathbb{Z}$ such that

$$u +_{2^n} v = w.$$

The equivalent addition over the binary digits in \mathbb{F}_2^n is defined as

$$\begin{aligned} \mathbf{u} + \mathbf{v} &= (u_{(0)}, u_{(1)}, \dots, u_{(n-1)}) \\ &\quad + (v_{(0)}, v_{(1)}, \dots, v_{(n-1)}) \\ &= (w_{(0)}, w_{(1)}, \dots, w_{(n-1)}) \\ &= \mathbf{w}, \end{aligned}$$

where $w_{(b)}$ satisfies

$$w_{(b)} = \sum_{k=0}^{b-1} \left(u_{(k)}v_{(k)} \prod_{l=k+1}^{b-1} (u_{(l)} + v_{(l)}) \right) + u_{(b)} + v_{(b)}, \quad 0 \leq b \leq n-1.$$

With this definition, we obtain have the additive group isomorphism

$$w_{(b)} = (u +_{2^n} v)_{(b)} = u_{(b)} + v_{(b)}.$$

This amounts to degree $n+1$ expressions in the bit variables for addition. It can be seen that these expressions are independent of n . If we were to obtain the algebraic expressions for word addition in the extension field \mathbb{F}_{2^n} , we would have to extract the individual carry bits using, for example, trace maps. This procedure is quite complex and will most likely yield high degree equations. Therefore, we have decided against using \mathbb{F}_{2^n} for this algebraic analysis. The first few expressions in increasing bit significance are as follows.

$$\begin{aligned} w_{(0)} &= u_{(0)} + v_{(0)}, \\ w_{(1)} &= u_{(0)}v_{(0)} + u_{(1)} + v_{(1)}, \\ w_{(2)} &= u_{(1)}u_{(0)}v_{(0)} + v_{(1)}u_{(0)}v_{(0)} \\ &\quad + u_{(1)}v_{(1)} + u_{(2)} + v_{(2)}, \\ w_{(3)} &= u_{(1)}u_{(2)}u_{(0)}v_{(0)} + u_{(1)}v_{(2)}u_{(0)}v_{(0)}, \\ &\quad + u_{(2)}v_{(1)}u_{(0)}v_{(0)} + v_{(1)}v_{(2)}u_{(0)}v_{(0)}, \\ &\quad + u_{(1)}u_{(2)}v_{(1)} + u_{(1)}v_{(1)}v_{(2)}, \\ &\quad + u_{(2)}v_{(2)} + u_{(3)} + v_{(3)}. \end{aligned}$$

3.3 State Permutation

Swapping states i, j in S can be algebraically described as the action of a permutation matrix \mathbf{M} on S . Given i, j , the entries $m_{r,s}$ of \mathbf{M} are constructed as follows.

- A diagonal entry $m_{r,r}$ is set if $i = j$ or both $i \neq r$ and $j \neq r$,
- An off-diagonal entry $m_{r,s}$ is set if $\{i, j\} = \{r, s\}$.

Using the above rules, the appropriate boolean function used by each entry $m_{r,s}$ of \mathbf{M} in the bits $i_{(b)}, j_{(b)}$

could be created, in a similar way as the state extraction operation. The diagonal entries can be expressed as

$$\begin{aligned}
 m_{r,r} = & \prod_{b=0}^{n-1} (i_{(b)} + j_{(b)} + 1) + \left(1 + \prod_{b=0}^{n-1} (i_{(b)} + r_{(b)} + 1) \right) \\
 & \times \left(1 + \prod_{b=0}^{n-1} (j_{(b)} + r_{(b)} + 1) \right) \\
 & + \prod_{b=0}^{n-1} (i_{(b)} + j_{(b)} + 1) \left(1 + \prod_{b=0}^{n-1} (i_{(b)} + r_{(b)} + 1) \right) \\
 & \times \left(1 + \prod_{b=0}^{n-1} (j_{(b)} + r_{(b)} + 1) \right).
 \end{aligned}$$

The off-diagonal entries can be expressed as

$$\begin{aligned}
 m_{r,s} = & \prod_{b=0}^{n-1} (i_{(b)} + r_{(b)} + 1) \prod_{b=0}^{n-1} (j_{(b)} + s_{(b)} + 1) \\
 & + \prod_{b=0}^{n-1} (i_{(b)} + s_{(b)} + 1) \prod_{b=0}^{n-1} (j_{(b)} + r_{(b)} + 1) = m_{s,r}.
 \end{aligned}$$

The resulting matrix \mathbf{M} is always symmetric. For example, the permutation matrix with $n = 2$ is

$$\mathbf{M} = \begin{pmatrix} m_{0,0} & m_{0,1} & m_{0,2} & m_{0,3} \\ m_{0,1} & m_{1,1} & m_{1,2} & m_{1,3} \\ m_{0,2} & m_{1,2} & m_{2,2} & m_{2,3} \\ m_{0,3} & m_{1,3} & m_{2,2} & m_{3,3} \end{pmatrix},$$

with entries

$$\begin{aligned}
 m_{0,0} = & i_{(0)}i_{(1)} + j_{(0)}j_{(1)} \\
 & + i_{(0)} + i_{(1)} + j_{(0)} + j_{(1)} + 1, \\
 m_{0,1} = & i_{(0)}i_{(1)}j_{(1)} + i_{(1)}j_{(0)}j_{(1)} + i_{(0)}i_{(1)} + i_{(0)}j_{(1)} \\
 & + i_{(1)}j_{(0)} + j_{(0)}j_{(1)} + i_{(0)} + j_{(0)}, \\
 m_{1,1} = & i_{(0)}i_{(1)} + j_{(0)}j_{(1)} + i_{(0)} + j_{(0)} + 1, \\
 m_{0,2} = & i_{(0)}i_{(1)}j_{(0)} + i_{(0)}j_{(0)}j_{(1)} + i_{(0)}i_{(1)} + i_{(0)}j_{(1)} \\
 & + i_{(1)}j_{(0)} + j_{(0)}j_{(1)} + i_{(1)} + j_{(1)}, \\
 m_{1,2} = & i_{(0)}i_{(1)}j_{(0)} + i_{(0)}i_{(1)}j_{(1)} + i_{(0)}j_{(0)}j_{(1)} \\
 & + i_{(1)}j_{(0)}j_{(1)} + i_{(0)}j_{(1)} + i_{(1)}j_{(0)}, \\
 m_{2,2} = & i_{(0)}i_{(1)} + j_{(0)}j_{(1)} + i_{(1)} + j_{(1)} + 1, \\
 m_{0,3} = & i_{(0)}i_{(1)}j_{(0)} + i_{(0)}i_{(1)}j_{(1)} + i_{(0)}j_{(0)}j_{(1)} \\
 & + i_{(1)}j_{(0)}j_{(1)} + i_{(0)}i_{(1)} + j_{(0)}j_{(1)}, \\
 m_{1,3} = & i_{(0)}i_{(1)}j_{(0)} + i_{(0)}j_{(0)}j_{(1)}, \\
 m_{2,3} = & i_{(0)}i_{(1)}j_{(1)} + i_{(1)}j_{(0)}j_{(1)}, \\
 m_{3,3} = & i_{(0)}i_{(1)} + j_{(0)}j_{(1)} + 1.
 \end{aligned}$$

The entries of \mathbf{M} have maximum degree $n + 1$. In the following section, we will show how these algebraic representations can be used to describe RC4 in an algebraic attack.

4 Equation Generation

In this section, we present techniques of equation generation for RC4. By introducing variables at each step of the algorithm, we can keep the equations generated to be of relatively low degrees, which could reduce solution time of the final system of equations. Where possible, we also show low degree multiples of the

equations generated, which could be used to simplify the system further (Courtois 2003). Let S^t, i^t, j^t be the values of S, i, j respectively at the end of clock t , where $t \geq 0$. We then have S^0, i^0, j^0 representing the initial states of S, i, j respectively. The relations among these internal states of RC4 and the keystream can then be expressed as follows.

$$\begin{aligned}
 i^t = & i^{t-1} +_{2^n} 1 && \text{(pointer increment),} \\
 j^t = & j^{t-1} +_{2^n} S_i^{t-1} && \text{(pointer addition),} \\
 S^t = & \mathbf{M}S^{t-1} && \text{(state permutation),} \\
 z^t = & S_{S_i^t +_{2^n} S_j^t} && \text{(keystream generation).}
 \end{aligned}$$

Each operation shown above will be algebraically analysed below.

4.1 Pointer Increment

In the first step, i is incremented by one. This addition is represented by

$$\begin{aligned}
 i_{(0)}^t = & i_{(0)}^{t-1} + 1, \\
 i_{(b)}^t = & i_{(b)}^{t-1} + \prod_{k=0}^{b-1} i_{(k)}^{t-1}, \quad 1 \leq b \leq n-1.
 \end{aligned}$$

Since it is known that $i^0 = 0$, the values of i^t are actually known for all $t \geq 0$. Therefore, no equations are needed to describe this step.

4.2 Pointer Addition

The contents of S_i are then extracted, which gives

$$S_{i,(b)}^t = \sum_{k=0}^{2^n-1} \left(S_{k,(b)}^t \prod_{l=0}^{n-1} (i_{(l)}^t + k_{(l)} + 1) \right), \quad 0 \leq b \leq n-1.$$

From the analysis in section 3.1, this can be expressed as

$$S_{i,(b)}^t = \sum_{e=0}^{2^n-1} \left(\prod_{f=0}^{n-1} i_{(f)}^{e_{(f)}} \left(\sum_{k=1}^{2^n-1} S_{i,(k)}^t \left(\prod_{g=0}^{n-1} e_{(g)}(i_{(g)} + 1) + 1 \right) \right) \right).$$

The addition for j is then given as follows.

$$\begin{aligned}
 j_{(0)}^t = & j_{(0)}^{t-1} + S_{i,(0)}^t, \\
 j_{(b)}^t = & \sum_{k=0}^{b-1} \left(j_{(k)}^{t-1} S_{i,(k)}^t \prod_{l=k+1}^{b-1} (j_{(l)}^{t-1} + S_{i,(l)}^t) \right) \\
 & + j_{(b)}^{t-1} + S_{i,(b)}^t, \quad 1 \leq b \leq n-1.
 \end{aligned}$$

This gives n equations of maximum degree n with n variables representing $j_{(0)}, j_{(1)}, \dots, j_{(n-1)}$ introduced at each clock. It is possible to move all terms to the left hand side and multiply the resulting expression by $(j_{(b-1)}^{t-1} + 1)(S_{i,(b-1)}^{t-1} + 1)$ to obtain

$$(j_{(b-1)}^{t-1} + 1)(S_{i,(b-1)}^{t-1} + 1)(j_{(b)}^{t-1} + S_{i,(b)}^t + j_{(b)}^t) = 0.$$

This would yield equations of maximum degree 3 for the this word addition operation.

4.3 State Permutation

The new pointers i^t, j^t are then used for state permutation in register S . Similar to the derivation before, the diagonal entries of the permutation matrix \mathbf{M} are given by

$$m_{r,r}^t = \prod_{b=0}^{n-1} (i_{(b)}^t + j_{(b)}^t + 1) \left(1 + \prod_{b=0}^{n-1} (i_{(b)}^t + r_{(b)}^t + 1) \right) \\ \times \left(1 + \prod_{b=0}^{n-1} (j_{(b)}^t + r_{(b)}^t + 1) \right) \\ + \prod_{b=0}^{n-1} (i_{(b)}^t + j_{(b)}^t + 1) \left(1 + \prod_{b=0}^{n-1} (i_{(b)}^t + r_{(b)}^t + 1) \right) \\ \times \left(1 + \prod_{b=0}^{n-1} (j_{(b)}^t + r_{(b)}^t + 1) \right).$$

The off-diagonal entries are given by

$$m_{r,s}^t = \prod_{b=0}^{n-1} (i_{(b)}^t + r_{(b)}^t + 1) \prod_{b=0}^{n-1} (j_{(b)}^t + s_{(b)}^t + 1) \\ + \prod_{b=0}^{n-1} (i_{(b)}^t + s_{(b)}^t + 1) \prod_{b=0}^{n-1} (j_{(b)}^t + r_{(b)}^t + 1) = m_{s,r}^t.$$

It can be observed that multiplying each entry $m_{r,s}$ of the permutation matrix \mathbf{M} by

$$\sigma_{r,s}^t = \left(\sum_{b=0}^{n-1} i_{(b)}^t + \sum_{b=0}^{n-1} r_{(b)}^t \right) \left(\sum_{b=0}^{n-1} j_{(b)}^t + \sum_{b=0}^{n-1} s_{(b)}^t \right).$$

gives a low degree multiple of the original expression of the entry, which is of maximum degree 3. In order to incorporate σ into our equations, we can relabel and multiply each entry of \mathbf{M} to obtain the degree 3 expressions $\sigma m_{r,s}$. The number of equations introduced as a result would be $2^{n-1}(2^n - 1)$, since \mathbf{M} is symmetric. An additional $2^n - 1$ linear expressions are required for the row sums of the matrix i.e. the new states of register S . This method would be quite uneconomical for an algebraic attack. Alternatively, let

$$\mathbf{M} = \begin{pmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,2^n-1} \\ m_{0,1} & m_{1,1} & \cdots & m_{1,2^n-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{0,2^n-1} & m_{1,2^n-1} & \cdots & m_{2^n-1,2^n-1} \end{pmatrix}, \\ \mathbf{S}^t = \begin{pmatrix} S_{0,(0)}^t & S_{0,(1)}^t & \cdots & S_{0,(n-1)}^t \\ S_{1,(0)}^t & S_{1,(1)}^t & \cdots & S_{1,(n-1)}^t \\ \vdots & \vdots & \ddots & \vdots \\ S_{2^n-1,(0)}^t & S_{2^n-1,(1)}^t & \cdots & S_{2^n-1,(n-1)}^t \end{pmatrix}.$$

The permutation action can then be described as the multiplication

$$\mathbf{S}^t = \mathbf{M}\mathbf{S}^{t-1}.$$

Hence, we have

$$S_{i,(b)}^t = \sum_{k=0}^{2^n-1} m_{b,k} S_{k,(b)}^{t-1}, \quad 0 \leq i \leq 2^n - 1,$$

where the $m_{u,v}$ are the matrix entries of \mathbf{M} . An examination into the equations generated reveals that the equations can be simplified. Since i^t is known, the

off-diagonal ones can only appear at rows i^t and column i^t in \mathbf{M} , which means that the other off-diagonal entries of \mathbf{M} are known to be zero. In particular, for each bit $0 \leq b \leq n - 1$, we have

$$S_{r,(b)}^t = \begin{cases} \sum_{k=0}^{2^n-1} m_{b,k} S_{k,(b)}^{t-1}, & r = i \\ m_{b,i} S_{i,(b)}^{t-1} + m_{b,r} S_{r,(b)}^{t-1}, & r \neq i. \end{cases}$$

These equations are of degree $n + 1$. When $r = i$, the expression $S_{r,(b)}^t$ is of the form

$$S_{r,(b)}^t = \prod_{b=0}^n j_{(b)}^t \sum_{k=0}^{2^n-1} S_{k,(b)}^t + a = a,$$

where a is of degree $n - 1$. As discussed in Section 3.1, the first term is zero, and we equations are then of degree $n - 1$. Overall, the state permutation operation results in $n2^n$ equations of maximum degree $n + 1$ with $n2^n$ variables representing values in the permutation matrices \mathbf{M} introduced at each clock. When $r \neq i$, it is possible to move all terms to the left hand side and obtain the equation

$$(S_{r,(b)}^t + m_{b,i} S_{i,(b)}^{t-1} + m_{b,r} S_{r,(b)}^{t-1})(S_{i,(b)}^{t-1} + S_{r,(b)}^{t-1} + 1) = 0.$$

This gives equations of degree 3. This alternative approach avoids relabelling of matrix entries, at the cost of having more high degree equations in the system.

4.4 Keystream Generation

Finally, state extraction is used twice to obtain a keystream word at each clock. Let r^t be the index of the state from which the keystream output is to be taken. Then,

$$r_{(b)}^t = S_{i,(b)}^t + S_{j,(b)}^t = \sum_{k=0}^{2^n-1} \left(S_{k,(b)} \prod_{b=0}^{n-1} (i_{(b)}^t + k_{(b)}) \right) \\ + \sum_{k=0}^{2^n-1} \left(S_{k,(b)} \prod_{b=0}^{n-1} (j_{(b)}^t + k_{(b)}) \right).$$

The keystream z^t is then given by extracting state r^t of register S . Thus,

$$z_{(b)}^t = S_{r^t,(b)} = \sum_{k=0}^{2^n-1} \left(S_{k,(b)} \prod_{b=0}^{n-1} (r_{(b)}^t + u_{(b)}) \right).$$

This amounts to $2n$ equations of degree n with n variables representing $r_{(0)}^t, r_{(1)}^t, \dots, r_{(n-1)}^t$ introduced at each clock, since z^t is assumed to be known.

4.5 Additional Equations

As discussed in Section 3.1, each bit position of the register S must sum to zero, that is,

$$\sum_{k=0}^{2^n-1} S_{k,(b)} = 0, \quad 0 \leq b \leq n - 1.$$

This provides n additional linear equations at each clock with no extra variables introduced.

Operation	e	v	d_1	d_2
Pointer Increment for i	0	0	0	0
Pointer Addition for j	n	n	3	n
State Permutation	$n2^n$	$n2^n$	3	$n + 1$
Keystream Generation	$2n$	$2n$	n	n
Additional Equations	n	0	1	1

Table 1: Summary of Equations Generated for RC4. For each clock, e is the number of equations generated, v is the number of variables introduced to the system, and d_1, d_2 are the maximum degrees of the equations with and without introducing low degree multiples, respectively.

5 Discussion

Based on the results from the previous sections, the number of equations generated at each clock for each operation is summarised in Table 1. From these results, we present an analysis of the RC4 cipher against algebraic attacks.

5.1 RC4 as an Algebraic Cipher

From the cipher description, one would normally expect the high nonlinearity of RC4 to arise from the state permutation operation. However, if low degree multiples are taken into account, it has been shown from the above analysis that the high nonlinearity is caused by state extraction, since there seems to be no low degree equations that can describe the operation in terms of the internal states. The state permutation, on the other hand, makes a primary contribution to the number of equations generated from the cipher. This is because it is the only operation that affects every state of the register, rather than just certain words or bits in the cipher. Not apparent from Table 1 is the important role of word addition with its effect of the carry bits. This operation relates all bits in each word of the internal states, and yields a system of equations that cannot be separated into smaller ones. If word addition is not present, the system could be split into n independent ones for each bit position, which can be solved independently. This can dramatically reduce the time complexity of an algebraic attack. It is quite interesting to see that each of the three main operations involved in the RC4 stream cipher has its own role in providing the overall security of the cipher when realised from an algebraic point of view, particularly since algebraic attacks had not yet appeared in their current form at the time when RC4 was designed. Together, the three operations in RC4 yield a strong algebraic system, and forms the basis of the resistance of the cipher against algebraic attacks.

5.2 Implications for RC4-Like Ciphers

Similar observations would be expected to arise if the same method of algebraic analysis is used on RC4 variants such as RC4A (Paul & Preneel 2004) and VMPC (Zoltak 2004), due to the similarities of their components. We also note that the algebraic properties of the three operations discussed above could form a sound set of design criteria for potential ciphers of this type. Specifically, in order to provide resistance to algebraic attacks, the cipher should contain components whose operations consist of some that translate to a large number of equations, some to equations of high degree, and some to equations that would make the whole system inseparable. An important point to note from the algebraic analysis of RC4 is that these algebraic properties need not come

from the same component. This is useful because components that satisfy more properties may contain operations that are more complicated and hence less efficient. Security against algebraic attacks need not be sacrificed for efficiency if careful tradeoffs are made between the number of components and their algebraic properties.

5.3 Algebraic Attacks on RC4

From the equation analysis, we obtain $n2^n + 3n$ equations in $n2^n + 3n$ variables at each clock of the cipher. With a register size of 2^n words, there are $n2^n$ additional initial state variables, but there are also $n(2^n + 1)$ additional equations. Since each clock gives n bits of output, we require keystream from at least 2^n clocks before a unique solution could be obtained. In fact, if no low degree multiples are used, we only require this amount of clocks to generate an overdefined system with a unique solution. In total, there would be $2^n(n2^n + 3n)$ equations in $2^n(n2^n + 3n + n(2^n + 1))$ variables. For the common 8-bit RC4 cipher, this gives a system of 534536 equations of maximum degree 9 in 532480 variables. These equations are very sparse, as each variable is only related to those at the immediately preceding, current, and immediately succeeding clocks. If low degree multiples are used, the numbers of equations and variables may rise if dependencies are found among the equations. More information and experimental results on solving equations without low degree multiples will be presented in Section 6.

The purpose of this paper is not to propose an algebraic attack, but to consider the impact of applying algebraic analysis techniques to non-algebraically oriented stream ciphers. As such, we do not provide a measure of the absolute or relative effectiveness of an algebraic attack against the RC4 stream cipher and its variants. Therefore, no comparisons are drawn against existing attacks, and we do not claim any advantages or disadvantages of this method over any existing ones. Sound complexity analyses on algebraic attacks are often difficult to reach due to their reliance on algorithms for solving large sparse multivariate systems of equations of varying forms, which in turn belongs to an area whose theory is yet to be fully developed and documented. Nevertheless, recent progress suggests that by implementing specialised routines to target equations generated from particular ciphers, one can improve the efficiency of equation solution greatly. These include the use of Gröbner basis (Courtois & Patarin 2003) and the Boolean Satisfiability (SAT) (Courtois & Bard 2007) algorithms. As the research on algebraic attacks progresses it is quite reasonable to believe that equation solution techniques will continue to improve in the foreseeable future. Therefore, it is important to discuss methods of generating systems of equations to describe ciphers, so that the feasibility of solution to these system and in turn the security of these respective ciphers can be constantly monitored into the future.

6 Experiments

Actual equation generation and solution attempts were made to verify the validity of the analysis and the feasibility of a successful attack on RC4. Table 2 shows the number of equations and variables that would be generated for the cipher with different word sizes. Our experiments were carried out using Magma 2.14 (Bosma et al. 1997) running on one 64-bit 1.6GHz Itanium 2 processor core on an SGI Altix 4700 supercomputer with 198 GB of shared memory.

n	Number of Variables	Number of Equations	Maximum Degree
2	64	74	3
3	288	315	4
4	1280	1348	5
5	5760	5925	6
6	26112	26502	7
7	118272	119175	8
8	532480	534536	9

Table 2: Summary of Equations Generated for RC4

The full sets of equations for $2 \leq n \leq 3$ have been successfully generated without low degree multiples, and their structures are in agreement with the analysis presented in Section 4. Using its Gröbner bases package in \mathbb{F}_2 , equations for $n = 2$ could be efficiently solved using 4 bits of keystream generated from a randomly chosen initial state. After more than 200 solution trials with keystream generated from random initial states, all of them returned a unique solution between 5.0 and 5.3 seconds. However, a few solution trials have been run with equations generated for $n = 3$, and were not successful after 48 hours of computation time each. Further investigation would be required to determine if it is infeasible to compute a solution using Gröbner basis techniques, or that the long computation time is caused by software restrictions. Nevertheless, based on the results of the experiment, we are quite confident in concluding that the full version of RC4 with $n = 8$ is most likely immune from algebraic attacks. This is based on the fact that methods for solving polynomial equations, such as Gröbner bases techniques, have time complexity exponential in the maximum degree of the equations (Becker & Weispfenning 1993). From our analysis in Section 4 and Table 2, it can be observed that the maximum degree of equations of RC4 rises linearly with the word size, so the time complexity would become infeasibly large very quickly.

7 Conclusion

This paper presents the first algebraic analysis of a non-LFSR based stream cipher, the RC4 family of stream ciphers. A method was shown for obtaining relationships between the internal states and the keystream of the word-based stream cipher. The state extraction, word addition, and state permutation operations were represented in terms of algebraic relations. These were used to form systems of equations describing the full keystream generation stage of the cipher. From these equations, we observed that having state extractions yields a system of high degree, having word addition makes the equation system inseparable, and state permutation is the main source of equations. Together, these operations constitute a strong systems of equations, in the sense that it would be infeasible to solve using currently known techniques. However, if any of these components are compromised, the strength of the system and in turn the security of RC4 against algebraic attacks would likely be reduced. It is interesting to arrive at this observation, given that the design of RC4 predates the introduction of algebraic attacks. Finally, our experimental results with reduced versions of RC4 suggest that the full RC4 is most likely immune from algebraic attacks at present. Further investigation into the cryptographic properties of RC4 is warranted for potential improvements on this first attempt at an algebraic analysis of the cipher. The findings of the algebraic properties of word-based operations in this paper could also be used as a reference for the design

of future ciphers that make use of similar components.

8 Acknowledgements

The first author wishes to thank Sultan Al-Hinai and Lynn Batten for the initial discussions on this work, and the High Performance Computing and Research Support at the Queensland University of Technology, Brisbane, Australia, for the assistance with the hardware and software used in our experiments.

References

- Al-Hinai, S., Batten, L., Colbert, B. & Wong, K. K. (2006), Algebraic attacks on clock-controlled stream ciphers, in L. M. Batten & R. Safavi-Naini, eds, '11th Australasian Conference on Information Security and Privacy — ACISP 2006', Vol. 4058 of *Lecture Notes in Computer Science*, Springer, pp. 1–16.
- Armknecht, F. & Krause, M. (2003), Algebraic attacks on combiners with memory, in D. Boneh, ed., 'Advances in Cryptology - Crypto 2003', Vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 162–175.
- Basu, R., Maitra, S., Paul, G. & Talukdar, T. (2009), On some sequences of the secret pseudo-random index j in RC4 key scheduling, in 'Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 18th International Symposium (AAECC-18)', Vol. 5527 of *Lecture Notes in Computer Science*, Springer, pp. 137–148.
- Becker, T. & Weispfenning, V. (1993), *Gröbner bases: A Computational Approach to Commutative Algebra*, Springer, New York, USA.
- Biham, E. & Carmeli, Y. (2008), Efficient reconstruction of RC4 keys from internal states, in 'Fast Software Encryption', Vol. 5086 of *Lecture Notes in Computer Science*, Springer, pp. 270–288.
- Bosma, W., Cannon, J. & Playoust, C. (1997), 'The MAGMA algebra system. I. The user language.', *Journal of Symbolic Computation* **24**(3-4), 235–265.
- Cho, J. Y. & Pieprzyk, J. (2004), Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering, in B. Roy & W. Meier, eds, 'Fast Software Encryption', Vol. 3017 of *Lecture Notes in Computer Science*, Springer, pp. 49–64.
- Courtois, N. (2001), The security of hidden field equations (HFE), in D. Naccache, ed., 'Topics in Cryptology - CT-RSA 2001', Vol. 2020 of *Lecture Notes in Computer Science*, Springer, pp. 266–281.
- Courtois, N. (2003), Fast algebraic attacks on stream ciphers with linear feedback, in D. Boneh, ed., 'Advances in Cryptology - Crypto 2003', Vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 176–194.
- Courtois, N. (2004), Algebraic attacks on combiners with memory and several outputs, in C. Park & S. Chee, eds, 'Information Security and Cryptology - ICISC 2004', Vol. 3506 of *Lecture Notes in Computer Science*, Springer.
- Courtois, N. & Bard, G. V. (2007), Algebraic cryptanalysis of the Data Encryption Standard, in 'Cryptography and Coding, 11th IMA International Conference', Vol. 4887 of *Lecture Notes in Computer Science*, pp. 152–169.

- Courtois, N. & Meier, W. (2003), Algebraic attacks on stream cipher with linear feedback, in E. Biham, ed., 'Advances in Cryptology - Eurocrypt 2003', Vol. 2656 of *Lecture Notes in Computer Science*, Springer.
- Courtois, N. & Patarin, J. (2003), About the XL algorithm over $GF(2)$, in 'Topics in Cryptology - CT-RSA 2003', Vol. 2612 of *Lecture Notes in Computer Science*, Springer, pp. 141–157.
- Courtois, N. & Pieprzyk, J. (2002), Cryptanalysis of block ciphers with overdefined systems of equations, in Y. Zheng, ed., 'Advances in Cryptology - Asiacrypt 2001', Vol. 2501 of *Lecture Notes in Computer Science*, Springer, pp. 267–287.
- Fluhrer, S. R. & McGrew, D. A. (2000), Statistical analysis of the alleged RC4 keystream generator, in B. Schneier, ed., 'Fast Software Encryption', Vol. 1978 of *Lecture Notes in Computer Science*, Springer, New York, USA, pp. 19–30.
- Golić, J. D. (1997), Linear statistical weakness of alleged RC4 keystream generator, in W. Fumy, ed., 'Advances in Cryptology - Eurocrypt '97', Vol. 1233 of *Lecture Notes in Computer Science*, Springer, pp. 226–238.
- Gong, G., Gupta, K. C., Hell, M. & Nawaz, Y. (2005), Towards a general RC4-like keystream generator, in D. Feng, D. Lin & M. Yung, eds, 'Information Security and Cryptology - CISC 2005', Vol. 3822 of *Lecture Notes in Computer Science*, Springer, pp. 162–174.
- Knudsen, L., Meier, W., Preneel, B., Rijmen, V. & Verdoolaege, S. (1998), Analysis methods for (alleged) RC4, in Ohta, ed., 'Advances in Cryptology - Asiacrypt '98', Vol. 1514, Springer, pp. 327–341.
- Mantin, I. & Shamir, A. (2001), A practical attack on broadcast RC4, in M. Matsui, ed., 'Fast Software Encryption', Vol. 2355 of *Lecture Notes in Computer Science*, Springer, pp. 152–164.
- Maximov, A. (2005), Two linear distinguishing attacks on VMPC and RC4A and weakness of RC4 family of stream ciphers, in H. Gilbert & H. Handschuh, eds, 'Fast Software Encryption', Vol. 3557 of *Lecture Notes in Computer Science*, Springer, Paris, France, pp. 342–359.
- Maximov, A. & Khovratovich, D. (2008), New state recovery attack on RC4, in 'Advances in Cryptology - Crypto 2008', Vol. 5157 of *Lecture Notes in Computer Science*, Springer, pp. 297–316.
- Paul, S. & Preneel, B. (2003), Analysis of non-fortuitous predicative states of the RC4 keystream generator, in T. Johansson & S. Maitra, eds, 'Progress in Cryptology - Indocrypt 2003', Vol. 2904 of *Lecture Notes in Computer Science*, Springer, pp. 52–67.
- Paul, S. & Preneel, B. (2004), A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher, in B. Roy & W. Meier, eds, 'Fast Software Encryption', Vol. 3017 of *Lecture Notes in Computer Science*, Springer, pp. 245–259.
- Schneier, B. (1996), *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd edn, John Wiley and Sons, New York, USA.
- Tsunoo, Y., Saito, T., Kubo, H., Shigeri, M., Suzaki, T. & Kawabata, T. (2005), The most efficient distinguishing attack on VMPC and RC4A, in 'Symmetric Key Encryption Workshop - SKEW 2005'.
- Wong, K. K., Colbert, B., Batten, L. & Al-Hinai, S. (2006), Algebraic attacks on clock-controlled cascade ciphers, in R. Barua & T. Lange, eds, 'Progress in Cryptology - Indocrypt 2006', Vol. 4329 of *Lecture Notes in Computer Science*, Springer, pp. 32–47.
- Zoltak, B. (2004), VMPC one-way function and stream cipher, in B. Roy & W. Meier, eds, 'Fast Software Encryption', Vol. 3017 of *Lecture Notes in Computer Science*, Springer, pp. 210–225.