MODELLING THE RC4 CIPHER AS AN INTEGER QUADRATIC PROGRAMMING PROBLEM

Ng Wen Xi[1], Teo Woo Hng Reanne[2], Esther Keziah Lim Rui Qi[3], Lim Zhan Feng[4], Ruth Ng Ii-Yung [4]
[1] Nanyang Girls' High School, 2 Linden Dr, Singapore 288683
[2] Eunoia Junior College, 2 Sin Ming Pl, Singapore 573838
[3] Methodist Girls' School, 11 Blackmore Dr, Singapore 599986
[4] DSO National Laboratories, 12 Science Park Dr, Singapore 118225

Abstract

RC4 is a cryptographic algorithm used in securing communications. In our work, we analyse the security of its Key Scheduling Algorithm (KSA), a component that initialises the internal state of the cipher using a secret key, by taking on the role of an attacker and reverse-engineering this KSA. Our approach is to model the KSA as an Integer Quadratic Programming (IQP) problem, to leverage known solvers for IQP to recover the secret key from the output of the KSA. Our main contributions include (1) expressing the KSA in mathematical expressions, and (2) linearising non-linear operators such as AND (see Section 5.1.1), OR (see Section 5.1.2), Modulo (see Section 5.2), multiplication of one binary and one non-binary variable (see Section 5.4), which are integral when used with the linearised piecewise function (see Section 5.3) to take after the 'swapping function' (Lines 3b-3d in Section 2.2.2) of KSA. By achieving these two goals, we successfully modelled KSA as an IQP problem, which allows for future deployment of efficient solvers such as CVXPY to solve complicated KSA instances. This work provides a structured method for exploring potential weaknesses in RC4's KSA and advancing the understanding of cipher vulnerabilities.

1 Introduction

Cryptography relies on ciphers to encrypt and decrypt data, protecting private communications from prying eyes. Stream ciphers achieve this by producing a pseudo-random sequence of bits or words using a fixed length secret key. Rivest Cipher 4 (RC4) is a popular stream cipher which follows the design principle of extracting pseudo-random bytes from pseudo-random permutations [2]. RC4 was designed by Ron Rivest for RSA Data Security as a trade secret in 1987. Compared to block ciphers, stream ciphers like RC4 process data byte by byte, thus having lower memory, computational load and processing power requirements. This makes implementation of RC4 rather simple and efficient, as seen from a wide application in network protocols such as Secure Sockets Layer, Transport Layer Security, Wired Equivalent Privacy, Wi-Fi Protected Access and in Microsoft Windows [3].

While there used to be a wide commercial usage of RC4, there have been other attacks developed against it [4], showing the statistical weaknesses that can be exploited to computationally distinguish the keystream of RC4 from a truly random sequence of bytes with a considerable probability of success [2]. However, these attacks stop short of retrieving the key which would be much more valuable to an attacker. Further, RC4 is still commonly used in applications such as WEP and Microsoft Kerberos.

Thus, we believe it is still relevant to demonstrate RC4's insecurity via new forms of mathematical analysis. In our work, we perform a 'known plaintext attack' of the RC4 cipher using Integer Quadratic Programming (IQP). In particular, we model the Key-Scheduling Algorithm (KSA), one of the two algorithms making up RC4, using Integer Quadratic Programming (IQP) to derive the key. IQP problems are optimisation problems where all variables take on integer values, with a quadratic objective function restricted by a list of linear constraints. The quadratic objective function is to be maximised or minimised depending on the problem.

Our contributions can be distilled to the following:
We compiled and devised linearisation techniques for five KSA operators. We believe that these linearisation techniques are applicable beyond the security analysis of RC4 since these operators are commonplace in other cryptographic algorithms. The operators addressed are: AND (see Section 5.1.1), OR (see Section 5.1.2), Modulo (see Section 5.2), a class of piecewise functions (see Section 5.3) and multiplication of binary and non-binary variables (see Section 5.4). In doing this, we built on and extended known techniques for the former three operators with novel insights and developed the latter two completely from scratch using subtle tricks to eliminate product terms.
We consolidate these techniques to model RC4's KSA. We then show that these five techniques are sufficient to model the entire KSA as an IQP problem. This is significant because efficient solvers such as CVXPY can be deployed to solve convex Quadratic Programming problems. We successfully modelled KSA into a convex IQP problem, and as such, solvers can be deployed in the future to solve complicated KSA problems.

2 Background

2.1 Integer Quadratic Programming (IQP)

Integer Quadratic Programming problems are optimisation problems where all variables take on integer values, with a quadratic objective function restricted by a list of linear constraints. The quadratic objective function is minimised or maximised depending on the nature of the problem. In our work, we first derived a mathematical expression for the KSA, before linearising all non-linear operators such that KSA is modelled into a convex IQP problem which can be solved using existing efficient Quadratic Programming solvers such as CVXPY.
We define the IQP problem to have the following inputs and output:

2.1.1 Inputs

(A) Objective function
The objective function takes the following form: $f(x_0, \ldots, x_{n-1}) = \sum_{i,j=0}^{n-1} a_{i,j} x_i x_j + b_i x_i + c$

$$n \text{ is the number of variables}$$
$$a_{i,j} \text{ is the coefficient of } x_i x_j$$
$$b_i \text{ is the coefficient of } x_i$$

$$c \; is \; a \; constant$$
$$x_0, x_1, x_2 \ldots, x_{n-1} \; are \; variables \; which \; take \; on \; integer \; values$$

The objective function is being optimised (i.e. maximised or minimised, depending on the nature of the problem).

(B) Constraints

All constraints take the following forms:$\sum_{i,j=0}^{n-1} d_{i,k} x_i + e \leq 0$

$$d_{i,k} \; is \; a \; constant \; coefficient \; of \, x_i$$
$$\text{e} \; is \; a \; constant$$

2.1.2 Output

The output takes the form $(t_0, \ldots, t_{n-1})$, where $(t_0, \ldots, t_{n-1})$ are the values of the variables $(x_0, \ldots, x_{n-1})$ when the objective function is optimised and all constraints are satisfied.

2.2 Key-Scheduling Algorithm (KSA)

2.2.1 Notational Convention

In Section 2.2.2, the pseudocode of KSA is presented and these two notations will be used throughout.

1. All variables and constants introduced in the FOR loop in KSA will take the following form:

$$element_{placing \; of \; element \; in \; tuple\text{-}for \; each \; loop \; number}^{loop \; number \; where \; term \; is \; first \; introduced}$$

2. All tuples in the KSA will have the elements <*n*>, where <*n*> is the 8-bit big endian representation of *n*. It is a string.

2.2.2 Pseudocode

<u>Key Scheduling Algorithm(*n*,***k***)</u>

```
1) $j^0 \leftarrow 0$
   Here, $j^0$ is assigned the value of 0
```

Let state, **s** *be a tuple of bytes of length $n^2 + n$, where n is an integer between 1 and 256*

```
2) (<$s_0^0$>,<$s_1^0$>, ..., <$s_{n-1}^0$>)←(<0>,<1>,...,<$n$-1>)
   Here, $s_0^0 \leftarrow 0$, $s_1^0 \leftarrow 1$,..., $s_{n-1}^0 \leftarrow n-1$
```

Let key, **k** *be a tuple of bytes of length n. (***k** *is made up of a key material tuple of length l repeated to match the length of the state, $l \leq n$)*

```
3) For $i = 0, ..., n-1$
   Let the loop number for the For Loop in the Key Scheduling Algorithm be i+1
   a) $j^{i+1} \leftarrow (j^i + s_i^i + k^i) \bmod n$
   b) $s_0^{i+1}, s_1^{i+1}, s_2^{i+1}, \ldots, s_{n-1}^{i+1} \leftarrow s_0^i, s_1^i, s_2^i, \ldots, s_{n-1}^i$
   c) $s_i^{i+1} \leftarrow s_{j^{i+1}}^i$
   d) $s_{j^{i+1}}^{i+1} \leftarrow s_i^i$
```

In lines 3b-3d, each ***s*** variable introduced in the new loop is assigned to the variable introduced in the previous loop with the same subscript. Exception occurs where subscript is *i or* j^{i+1}.When the subscript of the variable introduced in the new loop is *i*, it will be assigned the value of the variable introduced in the previous loop with subscript j^{i+1} and vice versa.

4) Return ***s***

3 Overview of Expressing KSA as a IQP problem

3.1 Intuition

To prove the insecurity of the RC4 cipher, we use IQP to perform a "known plaintext attack" to recover the key. In this case, we are deriving the key used when the initial and final state of KSA is known. The initial state is standard while the final state is given. We will call the output of KSA derived by the solver the derived final state, ***s***. The objective function is being set such that it is minimized when each byte of the derived final state, ***s***, is equal to the corresponding byte of the final state, ***fs***, given to the solver. The solver would then be able to output the accurate key when the objective function is minimised, and all constraints are fulfilled.

3.2 Inputs

(A) Objective Function: $\sum_{m=0}^{n-1}(fs_m - s_m^n)^2 = (fs_0 - s_0^n)^2 + (fs_1 - s_1^n)^2 + \cdots + (fs_{n-1} - s_{n-1}^n)^2$

The objective function is minimised.

Since the objective function is made up of square terms only, it is minimised when it is equal to 0. That occurs when each of the square terms are 0 which is when the difference between each byte of ***fs*** and the corresponding byte of ***s*** is 0.

(B) Constraints:

The KSA. The challenge lies in that solvers for IQP problems do not accept KSA pseudocode as constraints. This is because the Pseudocode are not valid mathematical equations, and they are nonlinear. Hence, we find methods to convert the KSA pseudocode into a set of linear constraints. To get the linear constraints, we would first express KSA as a set of equations (Section 4). Then, we find techniques to replace nonlinear mathematical operators used in the equations with linear constraints. (Section 5). Lastly, we apply techniques to the set of equations. (Section 10.3).

3.3 Output

Secret key, ***k*** = ($<k^0>$,$<k^1>$, …., $<k^{n-1}>$)

4 Expressing KSA as a set of equations.

Line of Pseudocode	Equations
1	$j^0 = 0$
2	($<s_0^0>$,$<s_1^0>$, ..., $<s_{n-1}^0>$) = (<0>,<1>,...,<n-1>)
3a	For i = 0, …, n-1

	j^{i+1} =(j^i+ s_i^i +k^i) mod n mod is nonlinear. To linearise, see section 5.2
3b,3c,3d For lines 3b-3d, we cannot simply replace '←' with '=' as 1. The solver is unable to solve when there are variables in the subscript of the terms. For example, $s_i^{i+1} = s^i_{j^{i+1}}$ $= s^i_{(j^i + s_i^i + k^i) \text{ mod n}}$ Here, the variable k^i is in the subscript. 2.The mathematical expressions would be non-valid. We cannot assign two different values to the same variable. For example, If $i = 0$ and $j^{i+1} = 1$, $s_0^1 = s_0^0$ and $s_0^1 = s_1^0$	$For\ \beta = 0, \dots, n-1$ $s_\beta^{i+1} = \sum_{\alpha=0}^{n-1} a_{\alpha,\beta}^{i+1} s_\alpha^i = a_{0,\beta}^{i+1} s_0^i + a_{1,\beta}^{i+1} s_1^i + a_{2,\beta}^{i+1} s_2^i + \dots + a_{n-1,\beta}^{i+1} s_{n-1}^i$ $where\ a_{\alpha,\beta}^{i+1}\ are\ decision\ variables$ $a_{\alpha,\beta}^{i+1} s_\alpha^i$ is nonlinear. To linearise, see section 5.4 If any one of the following criteria are met, $a_{\alpha,\beta}^{i+1}$ =1, else $a_{\alpha,\beta}^{i+1}$ =0 1.When $\alpha = i\ and\ \beta = j^{i+1}$ 2.When $\alpha = j^{i+1}\ and\ \beta = i$ 3.When $\alpha = \beta, \alpha \neq i\ and\ \alpha \neq j^{i+1}$ Let $u_{\alpha,\beta,1}^{i+1}$indicate if criteria 1 is being fulfilled. If criteria 1 is being fulfilled, $u_{\alpha,\beta,1}^{i+1} = 1$ else, $u_{\alpha,\beta,1}^{i+1} = 0$ Variables $u_{\alpha,\beta,2}^{i+1}$ and $u_{\alpha,\beta,3}^{i+1}$ work the same way for criteria 2 and 3. Thus, we can express $a_{\alpha,\beta}^{i+1}$ as: $a_{\alpha,\beta}^{i+1} = u_{\alpha,\beta,1}^{i+1} \vee u_{\alpha,\beta,2}^{i+1} \vee u_{\alpha,\beta,3}^{i+1}$ $\vee$ is nonlinear. To linearise, see section 5.1.2 From our criteria above, we can form the following sub criteria: 1. $\alpha = i$ 2. $\beta = j^{i+1}$ 3. $\alpha = j^{i+1}$ 4. $\beta = i$ 5. $\alpha = \beta$ 6. $\alpha \neq i$ 7. $\alpha \neq j^{i+1}$ Let $v_{\alpha,\beta,1}^{i+1}$ indicate if sub criteria 1 is fulfilled. $v_{\alpha,\beta,1}^{i+1} = \begin{cases} 1\ when\ \alpha = i, \\ 0\ when\ \alpha \neq i \end{cases}$ Variables $u_{\alpha,\beta,2}^{i+1}$, $u_{\alpha,\beta,3}^{i+1}$, $v_{\alpha,\beta,4}^{i+1}$ and $v_{\alpha,\beta,5}^{i+1}$ work the same way for criteria 2-5. Piecewise functions of this form are nonlinear. To linearise, see Section 5.3 Thus, $u_{\alpha,\beta,0}^{i+1}$, $u_{\alpha,\beta,1}^{i+1}$, $u_{\alpha,\beta,2}^{i+1}$ can be expressed as $u_{\alpha,\beta,1}^{i+1} = v_{\alpha,\beta,1}^{i+1} \wedge v_{\alpha,\beta,2}^{i+1}$ $u_{\alpha,\beta,2}^{i+1} = v_{\alpha,\beta,3}^{i+1} \wedge v_{\alpha,\beta,4}^{i+1}$ $u_{\alpha,\beta,3}^{i+1} = v_{\alpha,\beta,5}^{i+1} \wedge (1 - v_{\alpha,\beta,1}^{i+1}) \wedge (1 - v_{\alpha,\beta,3}^{i+1})$ $\wedge$ is nonlinear. To linearise, see section 5.1.1

Figure 1: Table showing each line of KSA pseudo code expressed as equations.

A full expression of KSA using a set of equations with all piecewise functions written out and with a more detailed explanation can be found in Section 10.3.

5 Linearising Operators Using Constraints

We will now be attempting to express Bitwise Operators AND and OR, Modulo, RC4 Piecewise Function and the multiplication of a binary and non-binary variable using linear constraints. We do this by defining the general form of how each mathematical operator is used before deriving the linear constraints based on this general form. The derivation of linear constraints is written out in detail for Bitwise And, Modulo and the RC4 Piecewise Function. The derivation of linear constraints for Bitwise OR and the multiplication of a binary and nonbinary variable uses a similar method to Bitwise AND. Thus, it can be found in the annex instead.

5.1 Bitwise Operators

Section 5.1.1 and 5.1.2 shows how we express bitwise operators AND and OR as linear constraints. As can been seen from Section 4, Bitwise exclusive OR is not needed in KSA equations. As such, the linear constraints used to express bitwise exclusive OR can be found in the annex instead.

5.1.1 Bitwise AND (∧)

The mathematical expression for bitwise operator AND where $z = x \wedge y$, given $x, y = 0 \; or \; 1,$ is $z = xy$ which is nonlinear. We can use the following method to find a set of linear constraints to force z to be equal to $x \wedge y$, given that $x, y = 0 \; or \; 1$.

First, we bound the values of z to include only binary values with the constraints $0 \leq z \leq 1$ and $z \in \mathbb{Z}$. There are now 8 possible sets of values x, y and z can take, which are represented as points A, B, C, D, E, F, G and H on the graph shown below. However, given that $x \wedge y = z$, there are only 4 possible sets of values x, y and z that satisfy x, y, z. They are points A(0,0,0), B(0,1,0), C(1,0,0) and D(1,1,1).

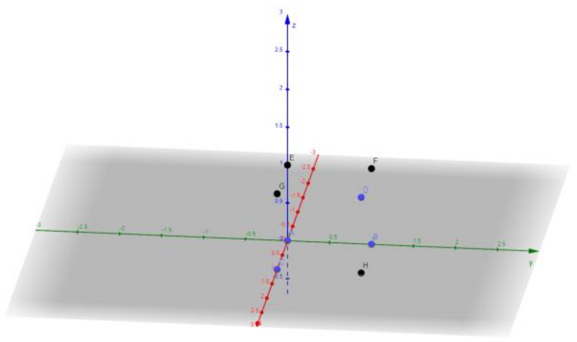


Figure 2.1 3D Graph showing points A, B, C, D, E, F, G and H

We now need to find linear constraints that can bound points A, B, C and D but not the remaining points: E(0,0,1), F(0,1,1), G(1,0,1) and H(1,1,0). To avoid points E (0,0,1), F(0,1,1) and G(1,0,1), we can use the planes $z = x$ and $z = y$ as upper bounds. Hence, we set the constraints:

$$z \leq y$$
$$z \leq x$$

Then, to avoid point H (1,1,0) we can use the plane $z = x + y - 1$ as a lower bound using the constraint:

$$z \geq x + y - 1$$

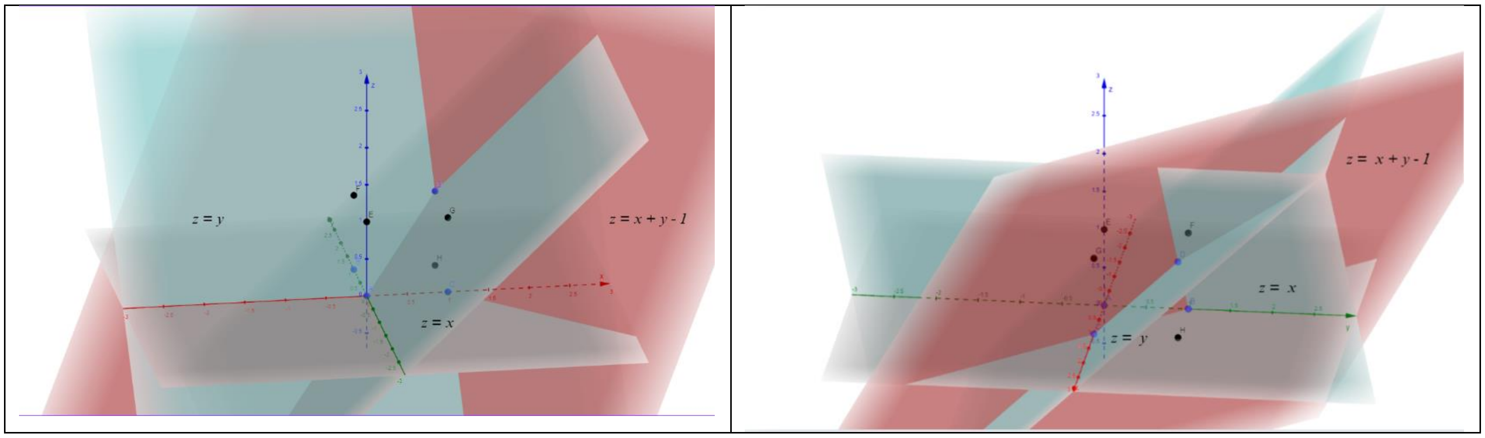


Figure 2.2 Different angles of the 3D Graph showing points A, B, C, D, E, F, G and H as well as planes $z = x$, $z = y$ and $z = x + y - 1$

As such, we can see how the following set of constraints forces z to be equal to $x \wedge y$.

$$z \leq x$$
$$z \leq y$$
$$z \geq x + y - 1$$
$$0 \leq z \leq 1$$
$$z \in \mathbb{Z}$$

5.1.2 Bitwise OR (∨)

Mathematically, given that $x, y = 0 \ or \ 1$ we can express $z = x \vee y$ as $z = x + y - xy$.
We can apply a similar method to derive the following set of constraints to force the value of z to be equal to $x \vee y$, given $x, y = 0 \ or \ 1$.

$$z \geq x$$
$$z \geq y$$
$$z \leq x + y$$
$$0 \leq z \leq 1$$
$$z \in \mathbb{Z}$$

The full derivation of the above equations can be found in Section 10.1.1

5.2 Modulo

The modulo function, where $a = b \bmod n$, a is the remainder of the division $\frac{b}{n}$. Another way to derive this remainder would be to subtract the highest possible multiple of n. This method can be expressed using the following linear equation and inequality:

$$a = b - xn$$
$$0 \leq a \leq n - 1$$
$$x \in \mathbb{Z}$$

5.3 RC4 Piecewise Function

The specific general form of piecewise function we use to determine value of variable $v_{\alpha,\beta,\gamma}^{i+1}$ is following:

$$v = \begin{cases} 1 \; when \; x = y, \\ 0 \; when \; x \neq y \end{cases}$$

$$where \; 0 \leq x \leq g, 0 \leq y \leq g \; and \; x, y \in \mathbb{Z}$$

Note: In our KSA equations, $g = n - 1$ as $\alpha, \beta, i \; and \; j^{i+1}$ are integers between 0 and $n - 1$.

Since $0 \leq x$ and $0 \leq y$, calculating if $x = y$ is the same as calculating if $x - y = 0$. Since $0 \leq x \leq g \; and \; 0 \leq y \leq g$, the greatest possible difference between x and y is g. Hence, we can then say that $-g \leq x - y \leq \; g$. Thus, we can rewrite the general form of the piecewise function as:

$$v = \begin{cases} 1 \; when \; x - y = 0 \\ 0 \; when \; x - y \neq 0 \end{cases}$$

$$where \; -g \leq x - y \leq g \; and \; x, y \in \mathbb{Z}$$

To find the linear constraints. Let us plot a point for all values of a for each value of $x - y$ where $g = 10$.

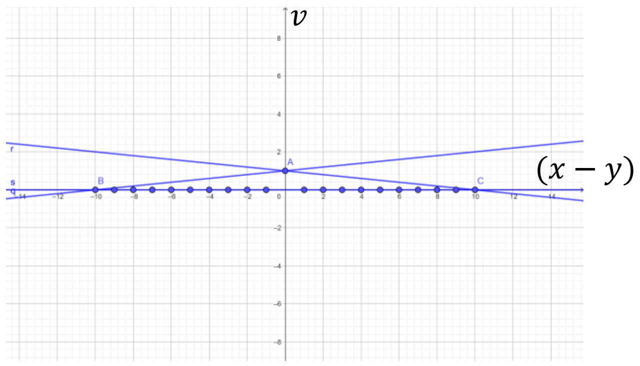


Figure 3.1 Graph of v *against* $(x - y)$

All points plotted can be bounded by lines from AB, BC, and CA.

The following points A, B and C expressed in terms of g are A (0,1), B($-g$,0) and C(g,0).

We can use three constraints to force a to fulfill the piecewise function above. The first inequality ensures that a falls under line AB, second under line AC and the third above line BC. The inequalities are as follows:

$$v \leq 1 - \frac{x - y}{g}$$

$$v \leq 1 + \frac{x + y}{g}$$

$$v \geq 0$$

$$v \in \mathbb{Z}$$

To test our constraints, let $x - y = 3$.

Using our piecewise functions, since $x \neq y$, $v = 0$

Using our constraints,

$$v \leq 1 - \frac{3}{10}$$

$$v \leq 1 + \frac{3}{10}$$
$$v \geq 0$$
$$v \in \mathbb{Z}$$

Thus, $v = 0$

Subbing the values of $(x - y)$ into the piecewise functions and linear constraints give the same values of v for all possible values of $(x - y)$ except when $(x - y) = 0$

When $(x - y) = 0$, v is bounded by the following linear constraints:

$$v \leq 1 - \frac{0}{10} = 1$$
$$v \leq 1 + \frac{0}{10} = 1$$
$$v \geq 0$$
$$v \in \mathbb{Z}$$

In this case, v can take on the value of 0 or 1.

For our report, we will refer to any case where a variable can take on more than one variable as an ambiguous case. When writing our KSA as a set of linear constraints, to force the variables $v_{\alpha,\beta,\gamma}^{i+1}$ to equal 1 when there is an ambiguous case, we add in the following constraint: $\sum_{\alpha=0}^{n-1} a_{\alpha,\beta}^{i+1} = 1$

This implies that only one $a_{\alpha,\beta}^{i+1}$ in the equation:$s_{\beta}^{i+1} = \sum_{\alpha=0}^{n-1} a_{\alpha,\beta}^{i+1} s_{\alpha}^{i} = a_{0,\beta}^{i+1} s_0^i + a_{1,\beta}^{i+1} s_1^i + a_{2,\beta}^{i+1} s_2^i + \ldots + a_{n-1,\beta}^{i+1} s_{n-1}^i$ will be 1 and the rest will be 0.

Before adding the constraint, when at least one of the following variables : $v_{\alpha,\beta,1}^{i+1}, v_{\alpha,\beta,2}^{i+1}, v_{\alpha,\beta,3}^{i+1}, v_{\alpha,\beta,4}^{i+1}$ *and* $v_{\alpha,\beta,5}^{i+1}$ are ambiguous, then, at least one of the following variables: $u_{\alpha,\beta,1}^{i+1}, u_{\alpha,\beta,2}^{i+1}$ *and* $u_{\alpha,\beta,3}^{i+1}$will be ambiguous too. Hence, $a_{\alpha,\beta}^{i+1} = 0 \; or \; 1$. However, with the constraint above, $a_{\alpha,\beta}^{i+1}$ is forced to be 1. This is because for the same β, there is only one set of the following variables: $v_{\alpha,\beta,1}^{i+1}, v_{\alpha,\beta,2}^{i+1}, v_{\alpha,\beta,3}^{i+1}, v_{\alpha,\beta,4}^{i+1}$ *and* $v_{\alpha,\beta,5}^{i+1}$ that has at least 1 ambiguous variables.

5.4 Multiplication of a binary and a non-binary variable

$$Let\ x\ be\ a\ binary\ variable$$
$$Let\ y\ be\ a\ non\ binary\ variable\ where\ y\ is\ an\ integer\ and\ 0 \leq y \leq c$$

Note: In our KSA equations, $c = n - 1$ as s_{α}^{i} is an integer between 0 and $n - 1$.

Using a similar graphing method shown in Section 5.1.1, we can express $z = xy$ using the following linear constraints:

$$z \leq y$$
$$z \leq cx$$
$$z \geq -c + y + cx$$
$$z \geq 0$$
$$z \in \mathbb{Z}$$

The full derivation of the above equations can be found in Section 10.2

6 Expressing KSA as a set of linear constraints

We have expressed all nonlinear mathematical operators as linear constraints. We can implement them in our set of equations that is used to express KSA. (Section 4) constraints that restrict the variables to be integers, e.g. $z \in \mathbb{Z}$, do not take the form of a linear constraint defined in Section 2.1.1. However, all variables can only be integers in an Integer Quadratic Problem. Thus, we can omit those constraints. After all nonlinear mathematical operators are replaced, we now have the general form of KSA fully represented by linear constraints. This can be found in Section 10.3. In general form and omitting constraints that restrict the variables to be integers, the nonlinear line 3 are replaced by 3 linear constraints. The nonlinear lines 3b, 3c and 3d are replaced by 45 constraints in total.

7 Conclusion and Future Work

In this work, we managed to successfully model KSA of the RC4 cipher into a convex Integer Quadratic Programming problem, by expressing the algorithm as mathematical expressions before linearising the constituting operators. The main challenge was to express the–lines 3b-3d of pseudocode where the arrow could not be replaced by an equality sign as well as to linearise the mathematical operators, particularly for the RC4 piecewise function and multiplication of a binary and non-binary variable.

As such, we propose for the following scope for future work:

1. Deploying IQP solvers to verify the feasibility of solving KSA problems using the IQP model devised in the paper.
2. Model the Pseudo-Random Generated Algorithm (the other Algorithm used in the RC4 Cipher) using IQP and similar linearisation techniques to obtain a full model for RC4 cipher.
3. The piecewise function is tailored to the usage in the KSA of the RC4 cipher and has to be implemented together with the other constraints in order to accurately model after the KSA. We aim to find methods such that we can express the piecewise function as linear constraints outside the RC4 cipher.

8 Acknowledgements

Firstly, we would like to express gratitude towards DSO National Laboratory for offering the Young Defence Scientists Programme, consisting of the World of Science programme and Research@YDSP. We are truly grateful to be given the opportunity to participate in WoS, and eventually being taken in as interns as part of R@YDSP under our mentors, Ruth and Zhan Feng. WoS and R@YDSP internship has made us developed a keen interest towards Cryptography and the application of Mathematics in information technologies and defence. We would also like to specially mention our fellow interns and the co-mentors from INFO-IS at SP20, for their support and company throughout this research.

9 References

[1] Erwin Kalvelagen. 2018. Linearizing multiple XOR operations. https://yetanothermathprogrammingconsultant.blogspot.com/2018/12/linearizing-multiple-xor-operations.html

[2] Jindal, P., & Singh, B. 2015. RC4 Encryption-A Literature Survey. Procedia Computer Science 46, 697–705. https://www.sciencedirect.com/science/article/pii/S1877050915001933

[3] Sarkar, S., Sen Gupta, S., Paul, G., & Maitra, S. 2014. Proving TLS-attack related open biases of RC4. Designs, Codes and Cryptography, 77(1), 231–253. https://dl.acm.org/doi/10.1007/s10623-014-0003-0

[4] Sen Gupta, S., Maitra, S., Paul, G. *et al.* 2014. (Non-)Random Sequences from (Non-) Random Permutations—Analysis of RC4 Stream Cipher. Journal of Cryptology 27, 67–108. https://link.springer.com/article/10.1007/s00145-012-9138-1

10 Annex

10.1 BITWISE OPERATORS

10.1.1 Bitwise OR

Mathematically, given that $x, y = 0\ or\ 1$ we can express $z = x \vee y$ as $z = x + y - xy$.
We can apply a similar method to derive the following set of constraints to force the value of z to be equal to $x \vee y$, given $x, y = 0\ or\ 1$.
Similar to the derivation of Bitwise AND in Section 5.1.1, we bound the values of z to include only binary values with the constraints $0 \leq z \leq 1$ and $z \in \mathbb{Z}$. There are now 8 possible sets of values x, y and z can take, which are represented as points A, B, C, D, E, F, G and H on the graph shown below. However, given that $x \wedge y = z$, there are only 4 possible sets of values x, y and z that satisfy x, y, z. They are points A(0,0,0), B(0,1,1), C(1,0,1) and D(1,1,1).

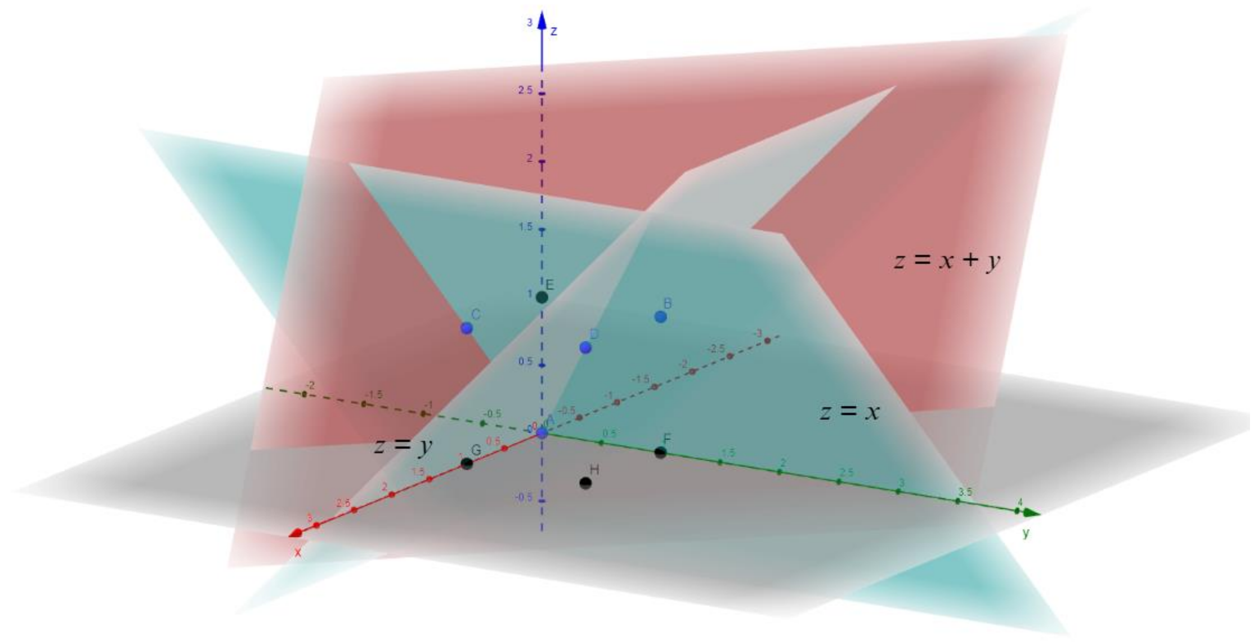


Figure 4: 3D graph showing the points A,B,C,D,E,F,G,H and planes $z = x$, $z = y$, $z = x + y$

From the above projection, we can see how the planes $z = x$ and $z = y$ instead act as lower bounds, to exclude undesired points F(0,1,0), G(1,0,0) and H(1,1,0). As such we can derive the following inequalities:

$$z \geq y$$
$$z \geq x$$

To exclude E(0,0,1), we can then use $z = x + y$ as an upper bound, deriving the inequality:

$$z \leq x + y$$

As such we can derive the following set of constraints such that $z = x \vee y$, given $x, y = 0 \text{ } or \text{ } 1$.

$$z \geq x$$
$$z \geq y$$
$$z \leq x + y$$
$$0 \leq x, y, z \leq 1$$
$$z \in \mathbb{Z}$$

10.1.2 Bitwise Exclusive OR

This derivation of linear constraints for Bitwise Exclusive OR is referenced from [1].
Mathematically, given that $x, y = 0 \text{ } or \text{ } 1$, we can express $z = x \oplus y$ as $z = x + y - 2xy$.
We can observe that:

x	y	$x + y$	z

0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	0

Case 1

If $x + y$ is 0 or 1, then z should be forced to be $x + y$ by constraints

Case 2

If $x + y$ is 2, then z should be forced to be $x + y - 2$ by constraints

As such we can derive the following set of constraints such that $z = x \oplus y$, given $x, y = 0 \; or \; 1$.

$$z = x + y - 2q$$
$$x - y - 1 \leq q \ \leq x + y$$
$$0 \leq z, q \leq 1$$
$$z, q \in \mathbb{Z}$$

10.2 Linearising the multiplication of one binary and one non-binary variable

$$Let\ x\ be\ a\ binary\ variable$$
$$Let\ y\ be\ a\ non\ binary\ variable\ where\ y\ is\ an\ integer\ and\ 0 \leq y \leq c$$

We can use the following method to force z to be to be equal to xy.

First, we plot a simple graph. To do so, let us take $c = 3$

The sets of values that x, y and z can hold is represented by points A, B, C, D, E, F, G and H on the graph below.

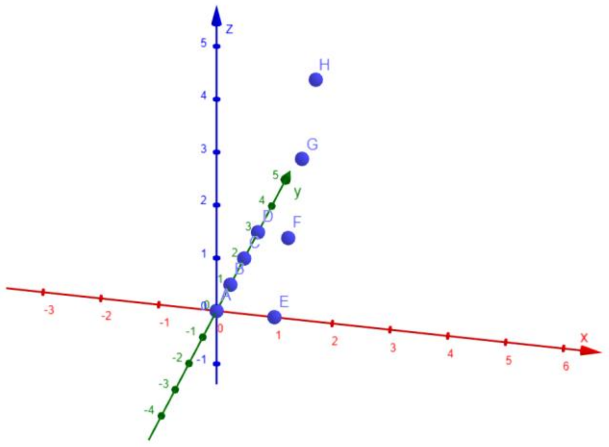


Figure 5: 3D graph with points A,B,C,D,E,F,G,H

We can effectively bound all points using these 4 planes.

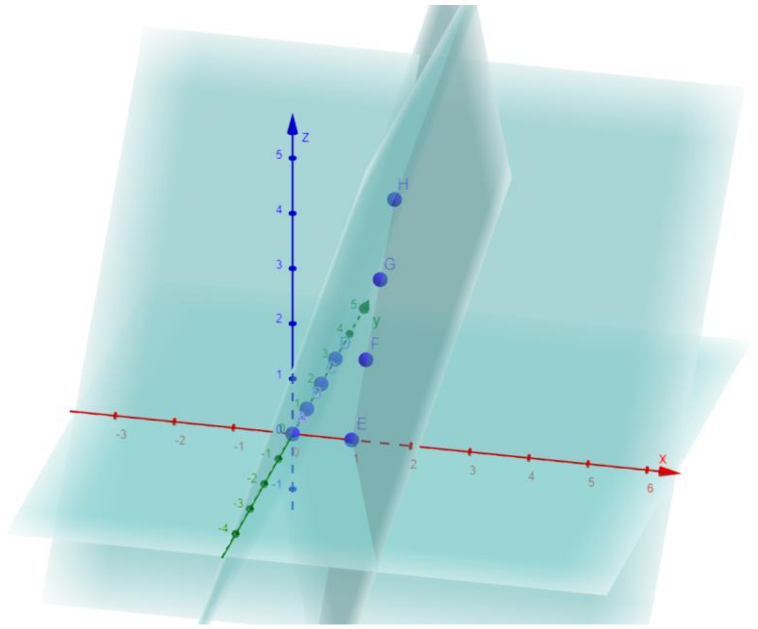


Figure 6: 3D graph with points A, B, C, D, E, F, G, H and 4 planes

The table below shows how these 4 planes are derived.

Plane	Analysis	Constraint derived
Equation of the plane 1: $z = 0$	Points wanted lie above or on the plane.	$z \geq 0$

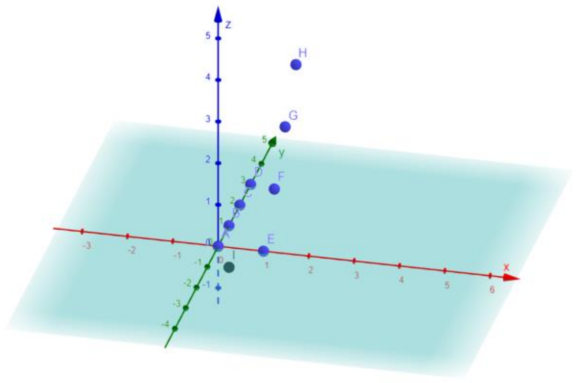
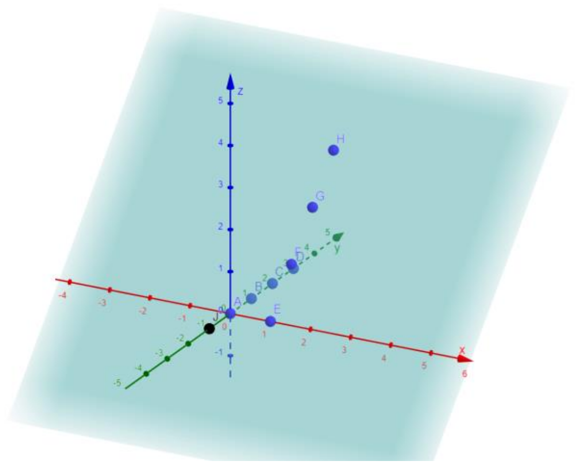
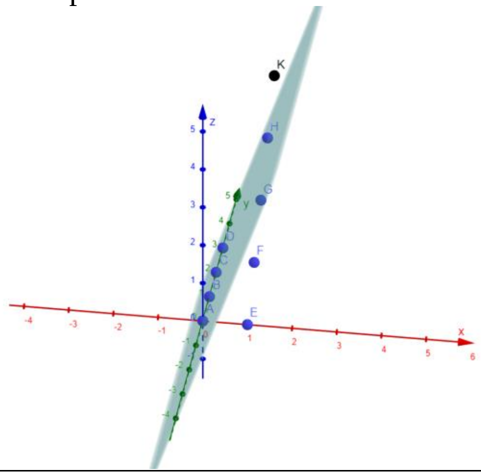
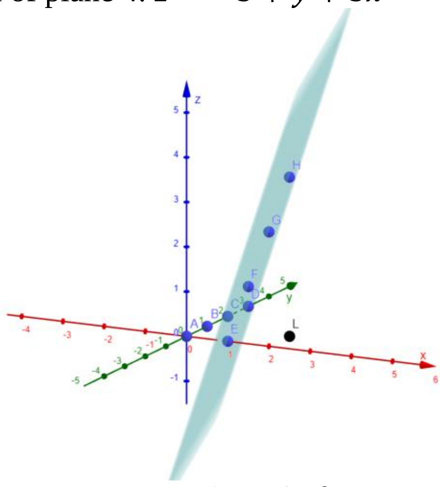
 *Figure 7: Plane 1*	Points below the plane (e.g. Point $I(0,1,-1)$) are eliminated.	
Equation of plane 2: $z = y$  *Figure 8: Plane 2*	Points wanted lie under or on the plane. Points above the plane (e.g. Point J$(0,-1,0)$) are eliminated.	$z \leq y$
Equation of plane 3: $z = 3x$ 	Points wanted lie under or on the plane. Points above the plane (e.g. Point K(1,4,4)) are eliminated.	$z \leq 3x$

Figure 9: Plane 3		
Equation of plane 4: $z = -3 + y + 3x$  Figure 10: Plane 4 Note: Vectors used to calculate equation of the plane are $\begin{pmatrix}1\\0\\0\end{pmatrix}$ and $\begin{pmatrix}-1\\c=3\\0\end{pmatrix}$	Points wanted lie above or on the plane. Points under the plane (e.g. Point L(2,1,0)) are eliminated.	$z \geq -3 + y + 3x$

We can now express $z = xy$ using the following linear constraints:

$$z \leq y$$
$$z \leq cx$$
$$z \geq -c + y + cx$$
$$z \geq 0$$
$$z \in \mathbb{Z}$$

10.3 KSA represented by linear constraints

The table shows how we linearise all operators in the nonlinear equations from Section 4, using the methods discussed Section 5 of the report. All variables are integers.

Notation:

$$\widetilde{variable}\ refers\ to\ the\ variable\ before\ the\ modulus\ is\ being\ taken$$

Non-Linear Equations	Linear Equations
$j^{i+1} = (j^i + s_i^i + k^i) \bmod n$	For $i = 0, \dots, 3$ $\widetilde{j^{i+1}} = j^i + s_i^i + k^i$ $j^{i+1} = \widetilde{j^{i+1}} - nb^{i+1}$ $0 \leq j^{i+1} \leq n-1$
$For\ \beta = o, \dots, n-1$	Let $r_{\alpha,\beta}^{i+1} = a_{\alpha,\beta}^{i+1} s_\alpha^i$ So,

$s_{\beta}^{i+1} = \sum_{\alpha=0}^{n-1} a_{\alpha,\beta}^{i+1} s_{\alpha}^{i}$ $= a_{0,\beta}^{i+1} s_0^i + a_{1,\beta}^{i+1} s_1^i + a_{2,\beta}^{i+1} s_2^i$ $+ \ldots + a_{n-1,\beta}^{i+1} s_{n-1}^{i}$ $where\ a_{\alpha,\beta}^{i+1}\ are\ decision\ variables$	$s_{\beta}^{i+1} = \sum_{\alpha=0}^{n-1} r_{\alpha,\beta}^{i+1} = r_{0,\beta}^{i+1} + r_{1,\beta}^{i+1} + \cdots + r_{n-1,\beta}^{i+1}$ $r_{\alpha,\beta}^{i+1} \geq 0$ $r_{\alpha,\beta}^{i+1} \leq (n-1) s_{\alpha}^{i}$ $r_{\alpha,\beta}^{i+1} \leq s_{\alpha}^{i}$ $r_{\alpha,\beta}^{i+1} \geq -(n-1) + s_{\alpha}^{i} + (n-1) a_{\alpha,\beta}^{i+1}$
If any one of the following criteria are met, $a_{\alpha,\beta}^{i+1}$ =1, else $a_{\alpha,\beta}^{i+1}$ =0 1.When $\alpha = i\ and\ \beta = j^{i+1}$ 2.When $\alpha = j^{i+1}\ and\ \beta = i$ 3.When $\alpha = \beta, \alpha \neq i\ and\ \alpha \neq j^{i+1}$ Let $u_{\alpha,\beta,1}^{i+1}$, $u_{\alpha,\beta,2}^{i+1}\ and\ u_{\alpha,\beta,3}^{i+1}$ be variables that indicate if each of the criteria are being fulfilled: If criteria 1 is being fulfilled, $u_{\alpha,\beta,1}^{i+1} = 1$ else, $u_{\alpha,\beta,1}^{i+1} = 0$ If criteria 2 is being fulfilled, $u_{\alpha,\beta,2}^{i+1} = 1$ else, $u_{\alpha,\beta,2}^{i+1} = 0$ If criteria 3 is being fulfilled, $u_{\alpha,\beta,3}^{i+1} = 1$ else, $u_{\alpha,\beta,3}^{i+1} = 0$	
Thus, we can express $a_{\alpha,\beta}^{i+1}$ as:	
$a_{\alpha,\beta}^{i+1} = u_{\alpha,\beta,1}^{i+1} \vee u_{\alpha,\beta,2}^{i+1} \vee u_{\alpha,\beta,3}^{i+1}$	$Let\ z_{\alpha,\beta}^{i+1} = u_{\alpha,\beta,1}^{i+1} \vee u_{\alpha,\beta,2}^{i+1}$ Constraints to force $z_{\alpha,\beta}^{i+1} = u_{\alpha,\beta,1}^{i+1} \vee u_{\alpha,\beta,2}^{i+1}$ $z_{\alpha,\beta}^{i+1} \geq u_{\alpha,\beta,1}^{i+1}$ $z_{\alpha,\beta}^{i+1} \geq u_{\alpha,\beta,2}^{i+1}$ $z_{\alpha,\beta}^{i+1} \leq u_{\alpha,\beta,1}^{i+1} + u_{\alpha,\beta,2}^{i+1}$ $0 \leq z_{\alpha,\beta}^{i+1} \leq 1$ Constraints to force $a_{\alpha,\beta}^{i+1} = z_{\alpha,\beta}^{i+1} \vee u_{\alpha,\beta,3}^{i+1}$ $a_{\alpha,\beta}^{i+1} \geq z_{\alpha,\beta}^{i+1}$ $a_{\alpha,\beta}^{i+1} \geq u_{\alpha,\beta,3}^{i+1}$ $a_{\alpha,\beta}^{i+1} \leq z_{\alpha,\beta}^{i+1} + u_{\alpha,\beta,3}^{i+1}$ $0 \leq a_{\alpha,\beta}^{i+1} \leq 1$
From our criteria above, we can form the following sub criteria: 1. $\alpha = i$ 2. $\beta = j^{i+1}$ 3. $\alpha = j^{i+1}$ 4. $\beta = i$ 5. $\alpha = \beta$ 6. $\alpha \neq i$ 7. $\alpha \neq j^{i+1}$	

Let $v^{i+1}_{\alpha,\beta,1}$, $v^{i+1}_{\alpha,\beta,2}$, $v^{i+1}_{\alpha,\beta,3}$, $v^{i+1}_{\alpha,\beta,4}$, $v^{i+1}_{\alpha,\beta,5}$ be indicator variables that check if each of the sub criteria 1-5 are fulfilled.	
$v^{i+1}_{\alpha,\beta,1} = \begin{cases} 1\ when\ \alpha = i, \\ 0\ when\ \alpha \neq i \end{cases}$	$v^{i+1}_{\alpha,\beta,1} \leq \frac{1}{(\mathrm{n}-1)}(\alpha - i) + 1$ $v^{i+1}_{\alpha,\beta,1} \leq -\frac{1}{(\mathrm{n}-1)}(\alpha - i) + 1$ $0 \leq v^{i+1}_{\alpha,\beta,1}$
$v^{i+1}_{\alpha,\beta,2} = \begin{cases} 1\ when\ \beta = j^{i+1} \\ 0\ when\ \beta \neq j^{i+1} \end{cases}$	$v^{i+1}_{\alpha,\beta,2} \leq \frac{1}{(\mathrm{n}-1)}(\beta - j^{i+1}) + 1$ $v^{i+1}_{\alpha,\beta,2} \leq -\frac{1}{(\mathrm{n}-1)}(\beta - j^{i+1}) + 1$ $0 \leq v^{i+1}_{\alpha,\beta,2}$
$v^{i+1}_{\alpha,\beta,3} = \begin{cases} 1\ when\ \alpha = j^{i+1} \\ 0\ when\ \alpha \neq j^{i+1} \end{cases}$	$v^{i+1}_{\alpha,\beta,3} \leq \frac{1}{(\mathrm{n}-1)}(\beta - i) + 1$ $v^{i+1}_{\alpha,\beta,3} \leq -\frac{1}{(\mathrm{n}-1)}(\beta - i) + 1$ $0 \leq v^{i+1}_{\alpha,\beta,3}$
$v^{i+1}_{\alpha,\beta,4} = \begin{cases} 1\ when\ \beta = i \\ 0\ when\ \beta \neq i \end{cases}$	$v^{i+1}_{\alpha,\beta,4} \leq \frac{1}{(\mathrm{n}-1)}(\alpha - j^{i+1}) + 1$ $v^{i+1}_{\alpha,\beta,4} \leq -\frac{1}{4(\mathrm{n}-1)}(\alpha - j^{i+1}) + 1$ $0 \leq v^{i+1}_{\alpha,\beta,4}$
$v^{i+1}_{\alpha,\beta,5} = \begin{cases} 1\ when\ \alpha = \beta \\ 0\ when\ \alpha \neq \beta \end{cases}$	$v^{i+1}_{\alpha,\beta,5} \leq \frac{1}{(\mathrm{n}-1)}(\alpha - \beta) + 1$ $v^{i+1}_{\alpha,\beta,5} \leq -\frac{1}{(\mathrm{n}-1)}(\alpha - \beta) + 1$ $0 \leq v^{i+1}_{\alpha,\beta,5}$
To check if sub criteria 6 and 7 are fulfilled, we can use bitwise NOT $v^{i+1}_{\alpha,\beta,0}$ and bitwise NOT $v^{i+1}_{\alpha,\beta,3}$respectively. $1 - v^{i+1}_{\alpha,\beta,1} = \begin{cases} 1\ when\ \alpha \neq i, \\ 0\ when\ \alpha = i \end{cases}$ $1 - v^{i+1}_{\alpha,\beta,3} = \begin{cases} 1\ when\ \alpha \neq j^{i+1} \\ 0\ when\ \alpha = j^{i+1} \end{cases}$	
-	To force the variables $v^{i+1}_{\alpha,\beta,\gamma}$ to equal 1 when there is an ambiguous case, we add in the following constraint: $\sum_{\alpha=0}^{n-1} a^{i+1}_{\alpha,\beta} = 1$
Thus, $u^{i+1}_{\alpha,\beta,1}$, $u^{i+1}_{\alpha,\beta,2}$, $u^{i+1}_{\alpha,\beta,3}$ can be expressed as	

$u^{i+1}_{\alpha,\beta,1} = v^{i+1}_{\alpha,\beta,1} \wedge v^{i+1}_{\alpha,\beta,2}$	$u^{i+1}_{\alpha,\beta,1} \le v^{i+1}_{\alpha,\beta,1}$ $u^{i+1}_{\alpha,\beta,1} \le v^{i+1}_{\alpha,\beta,2}$ $u^{i+1}_{\alpha,\beta,1} \ge v^{i+1}_{\alpha,\beta,1} + v^{i+1}_{\alpha,\beta,2} - 1$ $0 \le u^{i+1}_{\alpha,\beta,1} \le 1$
$u^{i+1}_{\alpha,\beta,2} = v^{i+1}_{\alpha,\beta,3} \wedge v^{i+1}_{\alpha,\beta,4}$	$u^{i+1}_{\alpha,\beta,2} \le v^{i+1}_{\alpha,\beta,3}$ $u^{i+1}_{\alpha,\beta,2} \le v^{i+1}_{\alpha,\beta,4}$ $u^{i+1}_{\alpha,\beta,2} \ge v^{i+1}_{\alpha,\beta,3} + v^{i+1}_{\alpha,\beta,4} - 1$ $0 \le u^{i+1}_{\alpha,\beta,2} \le 1$
$u^{i+1}_{\alpha,\beta,3} = v^{i+1}_{\alpha,\beta,5} \wedge (1 - v^{i+1}_{\alpha,\beta,1}) \wedge (1 - v^{i+1}_{\alpha,\beta,3})$	Let $w^{i+1}_{\alpha,\beta} = (1 - v^{i+1}_{\alpha,\beta,1}) \wedge (1 - v^{i+1}_{\alpha,\beta,3})$ Constraints to force $w^{i+1}_{\alpha,\beta} = (1 - v^{i+1}_{\alpha,\beta,1}) \wedge (1 - v^{i+1}_{\alpha,\beta,3})$ $w^{i+1}_{\alpha,\beta} \le (1 - v^{i+1}_{\alpha,\beta,1})$ $w^{i+1}_{\alpha,\beta} \le (1 - v^{i+1}_{\alpha,\beta,3})$ $w^{i+1}_{\alpha,\beta} \ge (1 - v^{i+1}_{\alpha,\beta,1}) + (1 - v^{i+1}_{\alpha,\beta,3}) - 1$ $0 \le w^{i+1}_{\alpha,\beta} \le 1$ Constraints to force $u^{i+1}_{\alpha,\beta,3} = v^{i+1}_{\alpha,\beta,5} \wedge w^{i+1}_{\alpha,\beta}$ $u^{i+1}_{\alpha,\beta,3} \le v^{i+1}_{\alpha,\beta,5}$ $u^{i+1}_{\alpha,\beta,3} \le w^{i+1}_{\alpha,\beta}$ $u^{i+1}_{\alpha,\beta,3} \ge v^{i+1}_{\alpha,\beta,5} + w^{i+1}_{\alpha,\beta} - 1$ $0 \le u^{i+1}_{\alpha,\beta,3} \le 1$

Figure 11: KSA represented by linear constraints