Attacking Bivium with MiniSat

Cameron McDonald, Chris Charnes and Josef Pieprzyk

Centre for Advanced Computing, Algorithms and Cryptography,
Department of Computing, Macquarie University
{cmcdonal,charnes,josef}@ics.mq.edu.au

Abstract. Trivium is a stream cipher candidate of the eStream project. It has successfully moved into phase 2 of the selection process and is currently in the focus group under the hardware category. As of yet there has been no attack on Trivium faster than exhaustive search.
Bivium is a class of simplified versions of Trivium that are built on the same design principles. Their design serves as a tool for investigating Trivium-like ciphers with a reduced complexity. This provides an insight into effective methods of attack that could be extended to Trivium. There have been successful attempts in the cryptanalysis of Bivium ciphers.
This paper focuses on a type of guess and determine attack that utilises the satisfiability solver MiniSat. Given a minimal amount of keystream MiniSat determines the remaining unknown state, leading to complete key recovery.

1 Introduction

The eStream project [5] was established with the aim of finding a cryptographic primitive for a stream cipher. There are two main categories, software encryption and hardware encryption. Of the criteria specified, the two main goals being that the cipher is secure and fast. There were 34 proposals, some of which have successfully passed the introductory phase 1 and have continued to phase 2. Several of the ciphers exhibited weaknesses in security and were dropped. Of the ciphers that passed to phase 2, a handful gained special recognition for their design and have been placed within a Focus group under their corresponding category. One of these ciphers is Trivium which was designed for hardware encryption. Trivium, designed by Cannière and Preneel [3], is both simple and elegant in its design. Although this design has attracted alot of interest from cryptanalysts, it remains unbroken.

It is well known that, breaking a good cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type", see Shannon [6]. The structure of Trivium directly establishes a system of sparse quadratic equations. However, solving a system of quadratic equations (known as the MQ problem) is generally NP-hard.

This paper converts the problem of solving a system of non-linear equations into a corresponding satisfiability problem. The method involves the application of multiple strategies to convert the system of equations in algebraic normal form

(ANF) to a system of clauses in conjunctive normal form (CNF). A subset of state variables are guessed to reduce the complexity of the system, which is then solved by the satisfiability solver MiniSat [8]. The solution returned by MiniSat is the remaining unknown state variables. Once the entire state is known, the cipher can be clocked backwards to eventually recover the private key. This attack requires minimal known keystream. The following analysis focuses on the cipher Bivium [4] to gain an understanding of the effectiveness of this method and how it may be applied to Trivium.

2 Previous results

Raddum introduces a new method of solving systems of sparse quadratic equations and applies it to Trivium, see [4]. The complexity arrising from this attack on Trivium is $O(2^{162})$, which is much worse than exhaustive key search. In his paper, Raddum introduces two simplified versions of Trivium, called Bivium-A and Bivium-B. The first version was broken "in about one day", the second required about 2^{56} seconds.

A.Maximov and A. Biryukov [2] use a different approach to solve the system of equations produced by Trivium by 'guessing' the value of state bits (or the product of state bits). In some cases this reduces the system of quadratic equations to a system of linear equations that can be solved (for example by Gaussian Elimination). The complexity of this attack is $O(c \cdot 2^{83.5})$ for Trivium and $O(c \cdot 2^{36.1})$ for Bivium, where c is the time taken to solve a sparse system of linear equations. They do not give a complexity estimate for the value of c, however from private correspondence they provide a rough estimate of $O(2^{16.2})$ for Bivium.

3 Description of Bivium

Let $x_i^t \in GF(2)$ denote the $i-$th variable of vector x at clock time t. Addition and multiplication, denoted by $+$ and $\cdot$ respectively, are done over $GF(2)$.

Bivium consists of a non-linear feedback shift register (NLFSR) coupled with a linear filter function (LF). The NLFSR operates on a 177-bit state, denoted by $(s_1^t, ..., s_{177}^t)$. The LF takes a linear combination of the state to produce the keystream. Each clock of the cipher involves updating two bits of the state and outputting one bit of keystream, denoted by z_i^t. The cipher continues to run until the required number of keystream bits is output. The following algorithm

provides a full description of the keystream generation:

for $i = 1$ to N **do**
$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$z_i \leftarrow t_1 + t_2$
$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{69}$
$(s_1, s_2, ..., s_{93}) \leftarrow (t_2, s_1, ..., s_{92})$
$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, ..., s_{176})$
end for

Like Trivium, Bivium incorporates a similar Key and IV setup where the cipher is clocked a number of times to initialise the state. Our analysis does not make use of this initialisation process and will not be discussed further.

4 Attacking Bivium

This attack method consists of two separate stages, a pre-attack computation stage and an attack stage. The pre-attack computation involves the following events:

1. Produce a system of equations that describe Bivium.
2. Convert the system of ANF equations into a system of CNF clauses.

Each of these events have their own strategy which will be discussed. The attack stage consists of the following repeated sequence of events:

1. Guess m state variables and substitute into the system of clauses.
2. Solve the resulting clauses.

Both these events also contribute several different strategies.

The time complexity of the attack is dependent on the number of state variables guessed and the time taken to solve the clauses. If the time taken to solve the resulting clauses is T_m clock-cycles for a m-bit guess, the complete state can be recovered in $T_m \cdot 2^{m-1}$ clock-cycles on the average.

5 Pre-Attack stage

5.1 System of Bivium equations

A system of keystream equations are composed such that each output keystream bit (z_i^t) is related to the same state (at clock time t). That is, equations of the following form:

$$z_i^t + A_i(s_1^t, \ldots, s_{177}^t) = 0,$$

where $i = 1 \ldots N$ (N is the number of equations required). A subset of the keystream equations is listed in the Appendix 10.

5.2 Converting ANF to CNF

The input to MiniSat is in the DIMACS graph format [7] which specifies that set of clauses are in conjunctive normal form (CNF).

The running time complexity of MiniSat depends on the number of variables and clauses in the system. Converting an equation in ANF with α monomials produces $2^{\alpha-1}$ CNF clauses; see [1].

Different strategies are needed to handle clause expansion introduced by this process:

1. The 'cutting' number - split the equations to reduce the number of monomials per equation (density).
2. Gaussian Elimination - unevenly distribute the frequency of the variables.
3. Replace common groups of monomials by new variables.

The cutting method introduced by Bard et al.[1] was a necessary step to reduce the clause production in the conversion of the equations to CNF. There is a tradeoff between the sparsity of the equations and the production of new variables since each cut introduces a new variable. Experiments determined that the 'optimal' cutting number for Bivium equations was five.

It is stated in [1] that a preprocessing scheme based on Gaussian elimination reduces the complexity of the system. However in our situation this method increased the density of the equations, thereby degrading the performance of MiniSat.

The last method searches for groups of monomials which are repeated in multiple equations. A new variable is introduced which represents the sum of the common group of monomials. This reduces the density of the equations while minimising the introduction of new variables.

For example, consider the following Bivium equations:

$$\begin{aligned} &s_{66} + s_{93} + s_{162} + s_{177} + z_1 = 0, \\ &s_{27} + s_{69} + s_{96} + s_{111} + s_{162} + s_{175} \cdot s_{176} + s_{177} + z_{67} = 0. \end{aligned}$$

The total number of clauses produced from these equations is $2^4 + 2^7 = 144$. However the Bivium feedback function is directly related to the output function and so $(s_{162} + s_{177})$ occurs in both equations.

Introducing a new variable $a_1 = s_{162} + s_{177}$ gives a new system of equations:

$$\begin{aligned} &s_{66} + s_{93} + a_1 + z_1 = 0, \\ &s_{27} + s_{69} + s_{96} + s_{111} + s_{175} \cdot s_{176} + a_1 + z_{67} = 0, \\ &s_{66} + s_{177} + a_1 = 0. \end{aligned}$$

Now the total number of clauses produced by this set of equations is $2^3+2^6+2^2 = 76$. This reduction in clauses is obtained at the cost of introducing a new variable. The limit on how many new variables are introduced and the total reduction of clauses is determined by experimentation.

6 Attack stage

The attack stage consists of guessing a subset of the state variables and using MiniSat to recover the remaining variables. MiniSat will return UNSATISFIABLE if the guess is incorrect, in that case another guess is tried. This process continues until MiniSat returns SATISFIABLE. At this stage the entire state vector has been recovered.

6.1 Guess state variables

Many strategies can be used in selecting which subset of state bits to guess. These include:

1. Select a subset of the variables which occur with the highest frequency in the set of equations - reduces the maximum number of monomials per variable guessed.
2. Select a subset of the variables that occur in the quadratic monomials - reduces the non-linearity of the system.
3. Select a subset of the quadratic monomials - (In this strategy we assign a guess to the product of two variables. The initial guess should be zero, since the probability of this being true is 3/4 (for independent variables))
4. Select a subset of alternate variables - all quadratic monomials are the products of two adjacent bits (see example below).

Consider the quadratic monomials in the following subset of Bivium equations:

$$
\begin{aligned}
s_{24} + s_{91} \cdot s_{92} + s_{93} + s_{108} + s_{159} + s_{171} + s_{172} \cdot s_{173} + s_{174} + z_{70} = 0,\\
s_{23} + s_{90} \cdot s_{91} + s_{92} + s_{107} + s_{158} + s_{170} + s_{171} \cdot s_{172} + s_{173} + z_{71} = 0,\\
s_{22} + s_{89} \cdot s_{90} + s_{91} + s_{106} + s_{157} + s_{169} + s_{170} \cdot s_{171} + s_{172} + z_{72} = 0,\\
s_{21} + s_{88} \cdot s_{89} + s_{90} + s_{105} + s_{156} + s_{168} + s_{159} \cdot s_{170} + s_{171} + z_{73} = 0.
\end{aligned}
$$

To reduce all the quadratic terms to linear terms we need only guess the variables $\{s_{89}, s_{91}, s_{170}, s_{172}\}$. Thus 8 quadratic terms become linear by guessing four of the variables.

6.2 Solving the equations

To solve the system of equations arsing from Bivium we used the satisfiability solver MiniSat [8]. As the complexity of the MiniSat algorithm is very difficult to analyse we have developed several solution strategies based on experiments. The different strategies provided a large spread of MiniSat computing times making it difficult to pinpoint the best strategy to use in the average case. (We hope to report more on this in the future.)

Hamming weights of solutions We observed that the Hamming weights of the solutions (state vectors) affected the performance of MiniSat. Figure 1 plots the average running time of MiniSat against the Hamming weight of random states when 40 bits of the state are guessed. In the graph we distinguish between times that MiniSat requires to return SAT or UNSAT.

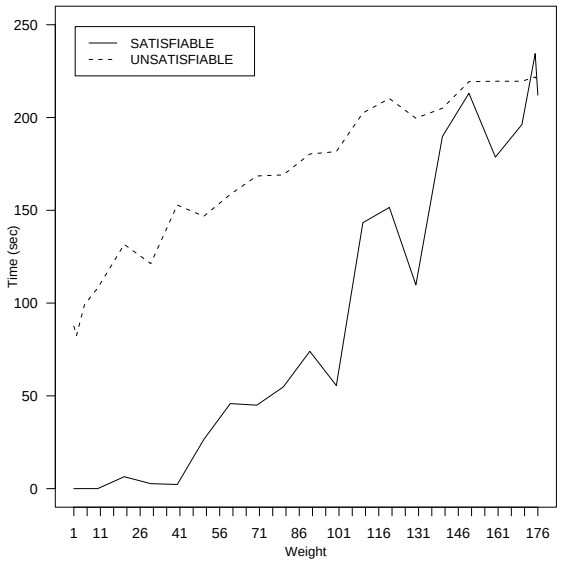

Fig. 1. MiniSat Time vs. State Weight

From this graph we derive two important observations which affect the average running time of MiniSat:

1. MiniSat returns faster when the guess is correct.
2. MiniSat running time increases with weight.

Therefore during the attack stage a limit should be set on the running time of MiniSat. If this limit is reached the guess is most likely incorrect and another guess should be tried.

Furthermore, in order to evaluate an average running time for MiniSat, we must observe how the probability distribution of the state varies with respect to the weight. The observed weights of random states over multiple clock times are given in Figure 2.

This graph indicates that for the majority of time clocks the state has weight of approximately half the state length. Since the bits being guessed during the attack stage are a subset of the state we have no reason to suspect that the

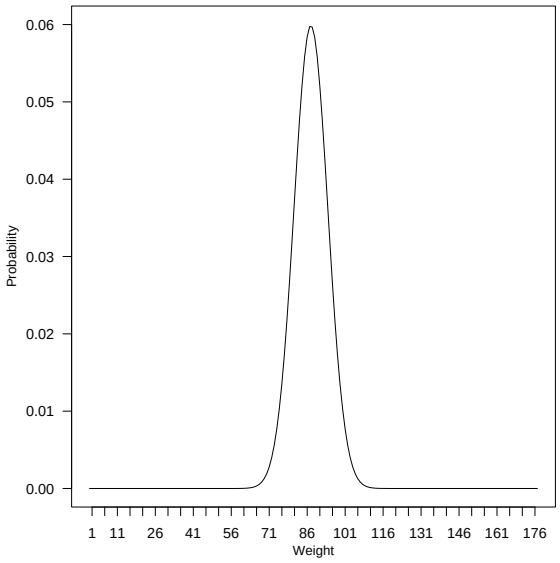

Fig. 2. Probability distribution on state weight

weight distribution of these subsets (correct guesses) is uniformly distributed. In fact the weight distribution is Hypergeometric.

For a state of length n with weight w_n, the probability of a subset of length m having weight (w_m) equal to k is:

$$Pr(w_m = k) = \frac{\binom{w_n}{k}\binom{n-w_n}{m-k}}{\binom{n}{m}}.$$

We can generalize this result by averaging over the probability distribution of Bivium states with respect to weight. We obtain the overall probability that a m-bit subset of any Bivium state has weight k:

$$Pr(w_m = k) = \sum_{i=0}^{n} Pr(w_n = i) \cdot \frac{\binom{i}{k}\binom{n-i}{m-k}}{\binom{n}{m}}.$$

For example, the probability distribution of 10-bit subsets of state with respect to weight is displayed in the Table 1.

On the average we expect to reach a correct guess when

$$\sum_{i=k}^{l} Pr(w_m = i) \geq 1/2, \tag{1}$$

k	0	1	2	3	4	5	6	7	8	9	10
$Pr(w_m = k)$	0.001	0.01	0.05	0.12	0.21	0.25	0.20	0.11	0.04	0.01	0.0009

Table 1. Probability distribution of 10-bit subsets

for some interval $[k, l]$. The total number of guesses required to achieve this is

$$G = \sum_{i=k}^{l} \binom{m}{i}.$$

By this method we will obtain an increase in efficiency when $G < 2^{m-1}$, since 2^{m-1} is the expected average when iterating over all consecutive guesses from 0 to $2^m - 1$.

7 Total attack time

Let T_m denote the time that MiniSat takes to return SAT or UNSAT given a m-bit guess. Then the total attack time is $T_m \cdot 2^{m-1}$.

Figure 3 gives the total attack times for different values of m which was observed experimentally.

We see that the optimal attack time on Bivium occurs when 34-bits of the state are guessed. The average MiniSat running time on this system is $440 \approx 2^{8.8}$ seconds. Using the Hamming weight guessing strategy (Section 6.2), the corresponding probabilities that a 34-bit guess has weight k are shown in Figure 4.

From this distribution it follows that the values of k and l satisfying (1) are 17 and 20 respectively. The corresponding number of guesses required to achieve this is $G = \sum_{i=17}^{20} \binom{34}{i} \approx 2^{32.8}$.

This strategy gives a slight increase in efficiency compared to a generic guessing strategy such as iterating all consecutive values from 0 to $2^m - 1$.

The overall average attack time on Bivium is $2^{32.8} \cdot 2^{8.8} = 2^{41.6}$ seconds. Raddum states that approximately $2^{13.3}$ keys can be searched exhaustively in 1 second, which gives 2^{55} as the approximate complexity of our attack.

7.1 Minimum running time of MiniSat

In our experiments we encountered many instances where MiniSat completed in a fraction of the average time. In 10% of the tests there was at least one case which was solved in only 5% of the average time. Hence if we run parallel MiniSat processes on a minimum of 10 different blocks of keystream we can expect to encounter a system that is solved in 5% of the average time. This strategy gives the best complexity of $2^{50.6}$ for a 34-bit guess.

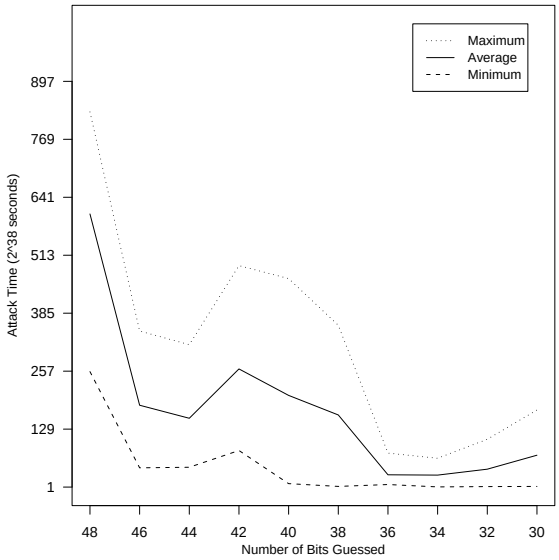

Fig. 3. Attack time vs. Number of bits guessed

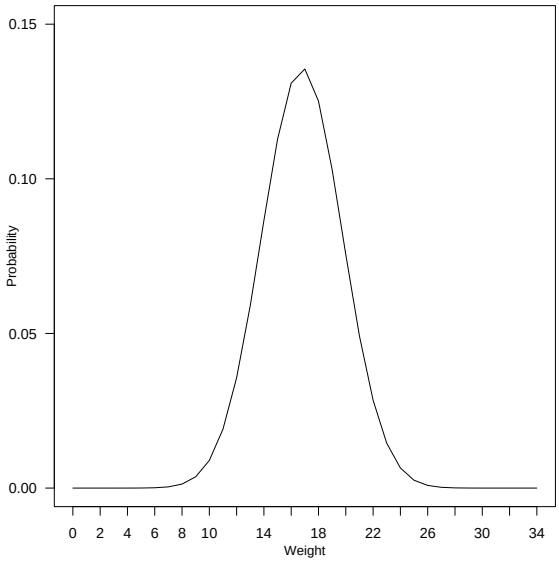

Fig. 4. Probability distribution of the weight of a 34-bit guess

8 Bivium-A

Bivium-A is a simplification of Trivium which was analysed by Raddum [4]. Bivium-A differs from Bivium (Bivium-B) in the linear output filter. The output is

$$z_i \leftarrow t_1,$$

a combination of only two state variables. This leads to keystream equations of the type:

$$z_i + s_j + s_k = 0.$$

Where $z_i \in \{0, 1\}$. Upon substitution of the observed keystream, these simple keystream relations directly connecting two state variables allow a large reduction in the number of variables in the system. In comparison to Bivium, this system of clauses has such a reduced complexity that it can be solved directly by MiniSat without the need for guessing state variables.

The system of equations is converted to CNF by cutting the equations into blocks of five. MiniSat was run on a sample of 100 random states and the results are plotted in Figure 5.

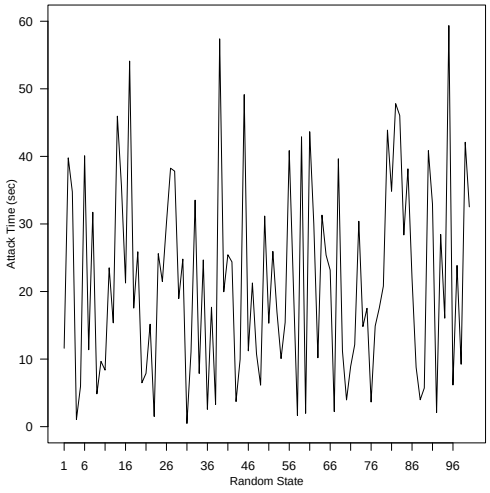

Fig. 5. Attack time for Bivium-A

The average running time to recover the state (and hence the key) of Bivium-A is 21 seconds. The minimum observed time was 0.5 seconds, while the maximum was 60 seconds. This can be compared to Raddum's algorithm which recovers the state of Bivium-A in 'about a day' [4].

9 Summary of our results

Due to the unpredictable behaviour and complexity of the MiniSat algorithm, the majority of results given in this paper are derived from experiments. We estimate the best complexity for the attack on Bivium to be $O(2^{51})$. This algebraic attack recovers the private key with observing minimal amounts of keystream (1780 bits). The simpler version Bivium-A is completely broken within 21 seconds on the average, where only 178 bits of keystream are required.

References

1. G. Bard, N. Courtois and C. Jefferson. *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers.* Cryptology ePrint Archive, Report 2007/024, 2007.

2. A. Maximov and A. Biryukov. *Two Trivial Attacks on Trivium.* Cryptology ePrint Archive, Report 2007/021, 2007.

3. C. De Cannière and B. Preneel. *TRIVIUM - a stream cipher construction inspired by block cipher design principles.* eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030, 2005. `http://www.ecrypt.eu.org/stream/trivium.html`

4. H. Raddum. *Cryptanalytic results on TRIVIUM.* eSTREAM, ECRYPT Stream Cipher Project, Report 2006/039, 2006. `http://www.ecrypt.eu.org/stream`

5. estream `http://www.ecrypt.eu.org/stream/`

6. C. Shannon. *Communication theory of secrecy systems.* Bell System Technical Journal 28, 1949.

7. DIMACS http://www.cs.ubc.ca/ hoos/SATLIB/Benchmarks/SAT/satformat.ps

8. MiniSat 2.0 http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/MiniSat.html

10 Appendix

$$
\begin{aligned}
&s_{66} + s_{93} + s_{162} + s_{177} + z_1 = 0,\\
&s_{65} + s_{92} + s_{161} + s_{176} + z_2 = 0,\\
&\ldots\\
&s_1 + s_{28} + s_{97} + s_{112} + z_{66} = 0,\\
&s_{27} + s_{69} + s_{96} + s_{111} + s_{162} + s_{175} \cdot s_{176} + s_{177} + z_{67} = 0,\\
&s_{26} + s_{68} + s_{95} + s_{110} + s_{161} + s_{174} \cdot s_{175} + s_{176} + z_{68} = 0,\\
&s_{25} + s_{67} + s_{94} + s_{109} + s_{160} + s_{173} \cdot s_{174} + s_{175} + z_{69} = 0,\\
&s_{24} + s_{91} \cdot s_{92} + s_{93} + s_{108} + s_{159} + s_{171} + s_{172} \cdot s_{173} + s_{174} + z_{70} = 0,\\
&s_{23} + s_{90} \cdot s_{91} + s_{92} + s_{107} + s_{158} + s_{170} + s_{171} \cdot s_{172} + s_{173} + z_{71} = 0,\\
&\ldots\\
&s_{10} + s_{77} \cdot s_{78} + s_{79} + s_{94} + s_{145} + s_{157} + s_{158} \cdot s_{159} + s_{160} + z_{84} = 0,\\
&s_9 + s_{66} + s_{76} \cdot s_{77} + s_{78} + s_{91} \cdot s_{92} + s_{93} + s_{144} + s_{156} + s_{157} \cdot s_{158} + s_{159} + s_{171} + z_{85} = 0,\\
&s_8 + s_{65} + s_{75} \cdot s_{76} + s_{77} + s_{90} \cdot s_{91} + s_{92} + s_{143} + s_{155} + s_{156} \cdot s_{157} + s_{158} + s_{170} + z_{86} = 0,\\
&\ldots\\
&s_1 + s_{58} + s_{68} \cdot s_{69} + s_{70} + s_{83} \cdot s_{84} + s_{85} + s_{136} + s_{148} + s_{149} \cdot s_{150} + s_{151} + s_{163} + z_{93} = 0,\\
&s_{57} + s_{67} \cdot s_{68} + s_{82} \cdot s_{83} + s_{84} + s_{135} + s_{147} + s_{148} \cdot s_{149} + s_{150} + s_{175} \cdot s_{176} + s_{177} + z_{94} = 0,\\
&\ldots\\
&s_{16} + s_{26} \cdot s_{27} + s_{41} \cdot s_{42} + s_{43} + s_{94} + s_{106} + s_{107} \cdot s_{108} + s_{109} + s_{134} \cdot s_{135} + s_{136} + z_{135} = 0,\\
&s_{15} + s_{25} \cdot s_{26} + s_{40} \cdot s_{41} + s_{42} + s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{105} + s_{106} \cdot s_{107} + s_{108} + s_{133} \cdot s_{134}\\
&\quad + s_{135} + s_{171} + z_{136} = 0,\\
&s_{14} + s_{24} \cdot s_{25} + s_{39} \cdot s_{40} + s_{41} + s_{65} + s_{90} \cdot s_{91} + s_{92} + s_{104} + s_{105} \cdot s_{106} + s_{107} + s_{132} \cdot s_{133}\\
&\quad + s_{134} + s_{170} + z_{137} = 0,\\
&\ldots\\
&s_4 + s_{14} \cdot s_{15} + s_{29} \cdot s_{30} + s_{31} + s_{55} + s_{80} \cdot s_{81} + s_{82} + s_{94} + s_{95} \cdot s_{96} + s_{97} + s_{122} \cdot s_{123}\\
&\quad + s_{124} + s_{160} + z_{147} = 0,\\
&s_3 + s_{13} \cdot s_{14} + s_{28} \cdot s_{29} + s_{30} + s_{54} + s_{66} + s_{79} \cdot s_{80} + s_{81} + s_{91} \cdot s_{92} + s_{93} + s_{94} \cdot s_{95} + s_{96}\\
&\quad + s_{121} \cdot s_{122} + s_{123} + s_{159} + s_{171} + z_{148} = 0
\end{aligned}
$$