# A cautionary note on the use of Gurobi for cryptanalysis

Muhammad ElSheikh and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Québec, Canada
{m_elshei,youssef}@ciise.concordia.ca

**Abstract.** Mixed Integer Linear Programming (MILP) is a powerful tool that helps to automate several cryptanalysis techniques for symmetric key primitives. Gurobi is one of the most popular solvers used by researchers to obtain useful results from the MILP models corresponding to these cryptanalysis techniques. In this report, we provide a cautionary note on the use of Gurobi in the context of bit-based division property integral attacks. In particular, we report four different examples in which Gurobi gives contradictory results when solving the same MILP model by just changing the number of used threads or reordering some constraints.

## 1 Introduction

Since Mouha *et al.* introduced the concept of using Mixed Integer Linear Programming (MILP) to count of the minimum number of active S-boxes in differential cryptanalysis [5], MILP has attracted attention of many cryptanalysis researchers. Cui *et al.* [1] proposed MILP models for both impossible differential attacks and zero-correlation attacks. Sasaki and Todo [6] developed a new search tool for impossible differential using MILP. Moreover, Xiang *et al.* [10] defined a systematic rules for constructing integral distinguisher using MILP. Later, Sun *et al.* complement this work by handling ARX-based ciphers (`modulo` operations) [7] and ciphers with non-bit-permutation linear layer[8]. Recently, Todo *et al.* utilize MILP model of the division property to improve the cube attacks [9].

In the context of differential and linear cryptanalysis, one might be able to manually verify the correctness of the differential/linear trail resulting from the MILP solver. On the other hand, the useful results in the context of impossible differential, zero-correlation, and division property based cryptanalysis, such as integral and cube attacks, rely on the infeasibility of the MILP model under consideration. Therefore, it is usually not possible to manually validate these results.

## 2 MILP modeling for the bit-based division property

Let $(a_0^0, a_1^0, \cdots, a_n^0) \rightarrow \cdots \rightarrow (a_0^r, a_1^r, \cdots, a_{n-1}^r)$ denote a division trail from the input division property to the output division property over $r$ rounds of an $n$-bit

block cipher. We consider the $i$-th bit of the output after $r$ rounds is balanced if there is no division trail from the input division property to the output division property $e_i$ (a unit vector whose $i$-th element is 1). Following [10], we can convert the propagation of the bit-based division property through $r$-round block cipher to a MILP model with an objective function $(Obj) : minimize\{a_0^r + a_1^r + \cdots + a_{n-1}^r\}$. For more details, we refer the reader to [10,8,2,3].

## 3 Examples for contradictory results from Gurobi

Throughout our experiments, we use `Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (linux64)` [4], to check if there are some balanced bits in four different models.

Algorithm 1 summarizes the steps we follow to test a given model. All the models in LP format and the actual code can be found at https://github.com/mhgharieb/GurobiOnCryptanalysis. We emphasize that the reported results are based on the above mentioned version of Gurobi and we have not tested other versions.

### 3.1 Example I

This example is for a 128-bit block cipher and its MILP model has the following statistics:

```
a model with 3661 rows, 41443 columns and 111491 nonzeros
Model fingerprint: 0x67963de8
Variable types: 0 continuous, 41443 integer (41443 binary)
Coefficient statistics:
        Matrix range     [1e+00, 6e+01]
        Objective range  [1e+00, 1e+00]
        Bounds range     [1e+00, 1e+00]
        RHS range        [1e+00, 5e+01]
```

We first set the parameter `LazyConstraints` to 1 for some reasons out of scope of this report. Then, we solved the model twice, using $numThreads = 48$ threads and $numThreads = 1$ thread. The result from 48-thread running is that the model is infeasible. This is an indication of balanced bits. In contrast, the single-thread running gives a feasible solution with $\mathcal{M}.ObjVal == 1$ which means there is an unbalanced bit. Clearly one of these two results must be wrong.

We then checked if the solution from the single-thread satisfies all the inequalities in the model using a simple bash/python script and indeed it passed the check. Thus, this is the correct result. Moreover, we have tried to figure out why Gurobi produces this behavior by checking any numerical issues in the model. Nonetheless, the coefficient statistics reported above informed us that the coefficient ranges are suitable. Therefore, we exclude the numerical issues.

Finally, we have reported this issue to Gurobi and received the following reply: *" ... was able to reproduce this behavior. ... It looks like Gurobi is generating an*

**Algorithm 1:** Pseudocode of our steps to check Gurobi

```
Input: Model.lp, numThreads// the model in LP format and some Gurobi
       parameter values
1  begin
       /* read the model from a file                                  */
2      M=read(Model.lp)
       /* Set some Gurobi parameters such as number of threads        */
3      M.setParam("Threads", numThreads)
       /* Solve the model                                             */
4      M.optimize()
       /* Check the model status after optimization                   */
5      if M.Status == 2 // M has a feasible solution
6      then
           /* if M.ObjVal == 1, there is a division trail from the
              input division property to e_i for an i bit             */
7          print("Find a solution")
           /* Export the solution to a file                           */
8          M.write("Solution.sol")
9      else if M.Status == 3 // M is infeasible and this is the
           indication of some blanaced bits
10     then
11         print("the model is infeasible")
12     else
13         print("Unkown Error")
14     end
15 end
```

*invalid cut. Our developers are currently working on fixing the issue. ... We will reach out to you if this issue is fixed in a future Gurobi release."*

### 3.2 Example II

This example is for a 288-bit permutation. Its MILP model has the following statistics:

```
a model with 10882 rows, 38880 columns and 175220 nonzeros
Model fingerprint: 0x87fb94b0
Variable types: 0 continuous, 38880 integer (38880 binary)
Coefficient statistics:
        Matrix range     [1e+00, 6e+01]
        Objective range  [1e+00, 1e+00]
        Bounds range     [1e+00, 1e+00]
        RHS range        [1e+00, 3e+02]
```

We first set the parameters `MIRCuts` and `CliqueCuts` to 2 for some reasons out of scope of this report. Like in the first example, we solved the model twice, using $numThreads = 80$ threads and $numThreads = 1$ thread. The result from

80-thread running is that the model is infeasible. In contrast, the single-thread running gives a feasible solution with $\mathcal{M}.ObjVal == 1$. We have checked that the single-thread solution satisfies all the inequalities in the model. Also, the coefficient statistics excludes any numerical issues.

### 3.3 Example III

This is the most important and serious example because, unlike the previous examples, we keep all the default values of Gurobi parameters without any change except the number of used threads. Nevertheless, Gurobi still gives contradictory results.

This example is for a 288-bit permutation. Its MILP model has the following statistics:

```
a model with 10711 rows, 38880 columns and 175049 nonzeros
Model fingerprint: 0x9f913e7e
Variable types: 0 continuous, 38880 integer (38880 binary)
Coefficient statistics:
        Matrix range     [1e+00, 6e+01]
        Objective range  [1e+00, 1e+00]
        Bounds range     [1e+00, 1e+00]
        RHS range        [1e+00, 3e+02]
```

Similar to the previous example, we solve the model twice, using $numThreads = 80$ threads and $numThreads = 1$ thread. The result from 80-thread running is that the model is infeasible. In contrast, the single-thread running gives a feasible solution with $\mathcal{M}.ObjVal == 1$. We have checked that the single-thread solution satisfies all the inequalities in the model. We have included 13 other models which follow this behavior at our github repository.

### 3.4 Example IV

This is another example that illustrates the odd behavior of Gurobi. In this example, we show that by just reordering some constraints in the MILP model, rerunning the solver using the same hardware, in terms of number of threads, and using the same default values of Gurobi parameters, the solver may still give contradictory results.

This example is for a 288-bit permutation. The original MILP model has the following statistics:

```
a model with 10969 rows, 38880 columns and 175307 nonzeros
Model fingerprint: 0x7ba61e9a
Variable types: 0 continuous, 38880 integer (38880 binary)
Coefficient statistics:
    Matrix range     [1e+00, 6e+01]
    Objective range  [1e+00, 1e+00]
    Bounds range     [1e+00, 1e+00]
    RHS range        [1e+00, 3e+02]
```

The reordered MILP model has the same statistics except:
`Model fingerprint: 0xa52a2fd8`. The mismatch between the two fingerprints is due to the reordering. One can easily verify that the two models are the same by sorting the constraints of each then compare digest hash of them. This can be done by a single line command on Linux:

```
md5sum <<< $(sort model.lp)
md5sum <<< $(sort model_reoredered.lp)
```

We solved the two models using $numThreads = 80$. The result of the original model is that the model is infeasible. On the other hand, the reordered model has a feasible solution with $\mathcal{M}.ObjVal == 1$. In the same manner as in the previous examples, we have verified that the the feasible solution satisfies all the constraints in both the original and the reordered model. We have included 5 other models which follow this behavior at our github repository.

## 4 Conclusion

Mixed Integer Linear Programming is a powerful technique in cryptanalysis and Gurobi is a widely utilized optimization solver that makes it applicable. However, at least for the time being, we should not take the result from automated solvers, including Gurobi, as it is because as we have shown in examples III and IV, just changing the used hardware in term of number of threads or reordering the constraints, the result may give false balanced bits which may lead to invalid attacks. In the case of an infeasible model. We have to determine why the model is infeasible. To this end, one can compute the Irreducible Infeasible Subsystem (IIS) which is a minimal subset of constraints and variable bounds that if any one of them is removed from the subsystem, the resulting subsystem is feasible.

## References

1. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations. IACR Cryptology ePrint Archive **2016**, 689 (2016)
2. ElSheikh, M., Tolba, M., Youssef, A.M.: Integral Attacks on Round-Reduced Bel-T-256. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography – SAC 2018. LNCS, vol. 11349, pp. 73–91. Springer International Publishing, Cham (2019)
3. ElSheikh, M., Youssef, A.M.: Integral Cryptanalysis of Reduced-Round Tweakable TWINE . Accepted, Cryptology and Network Security 2020 (CANS 2020)
4. Gurobi Optimization, L.: Gurobi Optimizer Reference Manual (2020), http://www.gurobi.com
5. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: International Conference on Information Security and Cryptology. pp. 57–76. Springer (2011)
6. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 185–215. Springer (2017)

7. Sun, L., Wang, W., Liu, R., Wang, M.: MILP-Aided Bit-Based Division Property for ARX-Based Block Cipher. Cryptology ePrint Archive, Report 2016/1101 (2016), https://eprint.iacr.org/2016/1101
8. Sun, L., Wang, W., Wang, M.: MILP-Aided Bit-Based Division Property for Primitives with Non-Bit-Permutation Linear Layers. Cryptology ePrint Archive, Report 2016/811 (2016), https://eprint.iacr.org/2016/811
9. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. IEEE Transactions on Computers **67**(12), 1720–1736 (2018)
10. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer (2016)