Designing Correlation Immune Boolean Functions With Minimal Hamming Weight Using Various Genetic Programming Methods

Jakub Husa
Brno University of Technology, Faculty of Information Technology,
IT4Innovations Centre of Excellence
Brno, Czech Republic
ihusa@fit.vutbr.cz

ABSTRACT

In this paper, we search for Boolean functions with properties useful in cryptography for preventing side-channel attacks. We use three genetic programming methods including linear genetic programming, which has not been used to design these functions before. Our results aim to provide a fair comparison by performing parameter optimization for each individual method, and deliver an insight into how well they cope with the demand for growing number of function inputs and higher degrees of correlation immunity.

CCS CONCEPTS

• **Computing methodologies → Genetic programming**; • **Security and privacy →** *Cryptanalysis and other attacks*;

KEYWORDS

Evolutionary algorithms, Genetic Programming, Cartesian Genetic Programming, Linear Genetic Programming, Cryptography, Boolean Functions, Hamming Weight, Correlation Immunity.

ACM Reference Format:
Jakub Husa. 2019. Designing Correlation Immune Boolean Functions With Minimal Hamming Weight Using Various Genetic Programming Methods. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3321925

1 INTRODUCTION

Side-channel attacks are attacks targeting the way a cipher is implemented in a device by probing for bits of the processed data, rather than attacking the cryptographic algorithm itself [3]. One way to prevent this is by *masking* the processed values with a suitable Boolean function.

A Boolean function is a function with n binary inputs and a single binary output. One way it can be represented is with a *truth table* of length 2^n, which defines a specific output for all possible inputs. To be suitable for masking a Boolean function needs to possess two properties, low (but non-zero) *Hamming weight* (HW), and high *correlation immunity* (CI) [2].

HW of a Boolean function is equal to the number of ones in its truth table. Low HW makes the function easy to implement and decreases the masking cost. CI of degree t means that the function's output is statistically independent of up to t of its inputs, and determines how many simultaneous probes will it be able to resists. For a formal definition of these properties and the known optimal values, we point interested readers to literature [1].

Because most analytical approaches for designing cryptographic Boolean functions only create balanced functions, which don't have a low HW [1], the task presents an interesting venue for use of heuristic approaches like *evolutionary algorithms* and specifically *genetic programming* (GP), which has already been shown to provide competitive results and discover new functions with better properties [5, 6].

2 OUR CONTRIBUTION

GP doesn't evolve candidate solutions directly, but as short executable structures (*individuals*, whose output is interpreted as a Boolean function. There exist multiple GP methods with different individual representations. *Tree-based* (TGP), which uses syntactic trees of limited depth, *Cartesian* (CGP), which uses an array of function nodes interconnected as an acyclic directed graph, and *Linear* (LGP), which uses a sequentially executed list of instructions operating over a finite set of registers. For more details on these methods, we refer interested readers to literature [4].

Because different GP methods usually use different population schemes, it is difficult to compare them fairly. For this reason, we have chosen two standard population schemes for all GP variants. *$(1 + \lambda)$ evolution strategy* (EST) creates new individuals by making mutated copies of the single best individual found so far. *Steady-state tournament* (SST) creates new individuals by randomly selecting three individuals for a tourney, and replacing the worst with a mutated offspring of the better two.

All GP methods use the same set of operands {AND, OR, XOR, XNOR}, and try to minimize *fitness* defined as infinity if the output is constant, and by equation 1 for all other Boolean functions:

$$Fitness_f = HW_f + min(0, t - CI_f) * 2^n \tag{1}$$

Where n is the number of function inputs and t the desired level of CI. For each GP method, population scheme, n, and t we have performed 100 independent runs. Each run was terminated either by reaching what is, to the best of our knowledge, the best solution ever found by GP [5] (and in most cases also the known optimal value [1]), or by reaching a set limit of 1 000 000 fitness function evaluations, in which case the run was considered unsuccessful.

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3321925

Table 2: Experimental results comparing selected GP methods utilizing two different population schemes.

EST	Experimental results of TGP					Experimental results of CGP					Experimental results of LGP				
	t = 1	t = 2	t = 3	t = 4	t = 5	t = 1	t = 2	t = 3	t = 4	t = 5	t = 1	t = 2	t = 3	t = 4	t = 5
n = 6	19744	784626	(14)			3406	35922	140169			8094	74746	356539		
n = 7	37151	(18)	(0)			6816	81721	278946			12569	165849	531192		
n = 8	20267	(27)	(0)	(0)		9184	27299	621912	972744		15126	87172	(39)	(34)	
n = 9	80859	(1)	(0)	(0)	(0)	12071	57672	182851	213011	(12)	17926	101449	493624	(40)	(4)
n = 10	102436	(0)	(0)	(0)	(0)	18764	121374	352436	868841	(9)	26634	151009	695616	(16)	(1)
SST	Experimental results of TGP					Experimental results of CGP					Experimental results of LGP				
	t = 1	t = 2	t = 3	t = 4	t = 5	t = 1	t = 2	t = 3	t = 4	t = 5	t = 1	t = 2	t = 3	t = 4	t = 5
n = 6	1005	21005	218505			6320	77160	135585			13730	314235	(32)		
n = 7	11005	30505	(47)			10955	210900	575650			17955	533445	(18)		
n = 8	12005	19505	(40)	(15)		17735	59330	953730	(24)		25065	204740	(2)	(9)	
n = 9	14005	23005	(32)	(6)	(0)	17330	88055	371415	482255	(13)	29490	225675	(27)	(20)	(1)
n = 10	15005	34005	(37)	(4)	(0)	22570	122535	467810	(40)	(7)	44325	641850	(8)	(0)	(0)

Table 1: Results of parameter optimization.

Optimized	Examined		Results for EST	
property	range	TGP	CGP	LGP
Population	5–1000	5	5	5
Chrom. length	50–1500	–	700	200
Tree depth	4–12	7	–	–
Free registers	5–100	–	–	20
Mutation rate	0.01–1.0	1.0	0.055	0.02
Examined	Examined		Results for SST	
Property	Range	TGP	CGP	LGP
Population	5–1000	1000	10	10
Chrom. length	50–1500	–	700	200
Tree depth	4–12	7	–	–
Free registers	5–100	–	–	20
Mutation rate	0.01–1.0	0.5	0.02	0.02

TGP used a sub-tree crossover and its mutation rate meant the probability that a random node would be replaced with a new, randomly generated subtree. CGP and LGP used a one-point crossover and their mutation rate meant the probability that each gene would be replaced with a random value. CGP also used an array with a single row and an unlimited L-back [4]. Each method had its other evolutionary parameters optimized one at a time using a simple hill climbing algorithm. The final values, as well as their examined ranges, are shown in table 1. Population sizes lower than 5 have not been examined to maintain a reasonable degree of parallelization.

Table 2 shows the results of our experiments. If more than half of the runs ended with success, we show the median number of fitness function evaluations required to find a solution. If fewer than half of the runs succeeded, and the median therefore cannot be determined, we show (in parentheses) how many of the 100 runs resulted in success. The cells where $t \geq \lceil \frac{2n-2}{3} \rceil$ are left empty because in their case constructing a Boolean function with optimal HW is trivial [1].

3 CONCLUSION

TGP provides the best results with SST and a very large population. It handles a growing number of function inputs extremely well but struggles to find functions with a high CI. TGP is affected by the choice of population scheme significantly more than the other two methods, likely because it uses a different mutation and crossover operators. CGP, notorious for the difficult use of crossover, performs the best with EST which only uses mutation. While CGP does not scale with an increasing number of inputs nearly as well as TGP, it is able to create large functions with a high CI, something that TGP fails to do.

LGP works well with shorter chromosomes and one free register per ten instructions. It scales in a manner similar to CGP, but even with the optimized settings requires approximately two times as many evaluations to obtain equivalent results. To the best of our knowledge, this is the first work that examines the use of LGP for the design of this type of Boolean functions.

ACKNOWLEDGMENTS

This work was supported by Czech Science Foundation project 19-10137S.

REFERENCES

[1] Claude Carlet and Xi Chen. 2018. Constructing Low-Weight d th-Order Correlation-Immune Boolean Functions Through the Fourier-Hadamard Transform. *IEEE Transactions on Information Theory* 64, 4 (2018), 2969–2978.

[2] Claude Carlet and Sylvain Guilley. 2013. Side-channel indistinguishability. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, 9.

[3] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. *Power analysis attacks: Revealing the secrets of smart cards*. Vol. 31. Springer Science & Business Media.

[4] Nicholas Freitag McPhee, Riccardo Poli, and William B Langdon. 2008. Field guide to genetic programming. (2008).

[5] Stjepan Picek, Claude Carlet, Sylvain Guilley, Julian F Miller, and Domagoj Jakobovic. 2016. Evolutionary algorithms for boolean functions in diverse domains of cryptography. *Evolutionary computation* 24, 4 (2016), 667–694.

[6] Stjepan Picek, Sylvain Guilley, Claude Carlet, Domagoj Jakobovic, and Julian F Miller. 2015. Evolutionary approach for finding correlation immune Boolean functions of order t with minimal Hamming weight. In *International Conference on Theory and Practice of Natural Computing*. Springer, 71–82.