On Optimality of $d+1$ TI Shared Functions of 8 Bits or Less

Dušan Božilov[1,2]

[1] NXP Semiconductors, Leuven, Belgium dusan.bozilov@nxp.com
[2] COSIC KU Leuven and imec, Leuven, Belgium

Abstract. We present a methodology for finding minimal number of output shares in $d+1$ TI by modeling the sharing as set covering problem and using different discrete optimization techniques to find solutions. We demonstrate the results of our technique by providing optimal or near-optimal sharings of several classes of Boolean functions of any degree up to 8 variables, for first and second order TI. These solutions present new lower bounds for the total number of shares for these types of functions

Keywords: Threshold Implementation · SCA · Masking

1 Introduction

Recently, side-channel community became more and more interested in low-latency masked implementations. The resulting effort produced a variety of AES [WMM20, GIB18, UHA17, SBHM20], KECCAK [ABP+18], PRINCE [MS16, BKN20] Threshold Implementations (TI) [NRR06], using different techniques to reduce the total cycle count. Trivial lower bound of the number of output shares of n variables is $(d+1)^t$ where t is the algebraic degree of the function. Construction presented in [BKN20] provides sharing with this minimal bound if $t = n-1$ for any security order d. [WMM20] results confirm these findings for $d = 1, 2, 3$ while providing minimal sharing for few cases when $t < n-1$. Additionally, construction method that is finds sharing that is multiples of $(d+1)^t$ is given, and applied to particular case of cubic 8-bit function to find first order sharing with 16 shares, double of what the lower bound is.

In this work we present a new searching methodology based on Set Covering Problem (SCP) for finding optimal and near-optimal shares using discrete optimization techniques to find quality solutions, and in some cases, prove their optimality with respect to the number of output shares, establishing new lower bounds for several cases.

2 Problem definition

Valid $d+1$ TI satisfies non-completeness and correctness properties to provide secure d-th order sharing. Non-completeness requires that output shares have to satisfy that for each variable one output share is allowed to use at most 1 share for any given variable. Correctness mandates that combining output shares together would provide same output result as in unshared function, for all inputs. Let us recall the table representation from [BKN20], for the ease of notation. Table columns represent different input variables, while rows represent different output shares. Entry (i, j) of i-th row and j-th column indicates which input share is associated to j-th variable in the i-th output share. Equation (1) gives an exampe of table representation of the first order sharing for function $xy + z$.

For any particular sharing table, non-completeness holds regardless of the function being shared, since for each output row, one column can have exactly one value, representing one input share.

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \begin{aligned} o_1 &= x_0y_0 + z_0 \\ o_2 &= x_0y_1 \\ o_3 &= x_1y_0 \\ o_4 &= x_1y_1 + z_1 \end{aligned} \tag{1}$$

The complexity of finding minimal sharing stems from the fact that sharing also has to be correct. Achieving correctness is tightly coupled with the algebraic degree, number of input bits and algebraic structure of the function. For a t-degree function with $n = t$ variables we require $(d+1)^t$ output shares, because all possible $(d+1)^t$ combinations with repetitions need to be present in output sharing. More interesting case is when $t < n$. The problem then becomes that t columns of the output sharing representing variables that form t degree term should contain all $(d+1)^t$ combinations with repetitions.

For given output sharing to satisfy correctness of a given specific function F, all of the terms in the ANF representation of F need to be correctly shared, i.e. for each term of t variables, columns representing output shares should contain all different $(d+1)^t$ combinations with repetitions. We evaluate the shares of l from 1 to $(d+1)^t$ in lexicographic order and say that output share S covers i-th share of term l if columns of S representing variables of l form the i-th share of t variables.

We first show how the problem of finding a correct sharing can be reduced to set covering problem (SCP). Each different output share among different $(d+1)^n$ shares will be considered as different set, and family of these sets will be referred to as $\mathcal{D}$. The universe $\mathcal{U}$ of all elements to be covered is created by going through ANF, and for each term l we add $(d+1)^t$ elements where t is degree of l, representing different shares of l. In other words, each share of each term is a separate element to be covered. Set S from $\mathcal{D}$ will contain element e from $\mathcal{U}$ if output share represented by S covers term share from F represented by e. Now given $\mathcal{D}$ and $\mathcal{U}$ we need to find subfamily $\mathcal{C} \subseteq \mathcal{D}$ with minimal cardinality such that union of sets from $\mathcal{C}$ is $\mathcal{U}$. With respect to elements e from $\mathcal{U}$, there exists at least one set S from $\mathcal{C}$ that contains e.

We can further represent SCP in terms of decision variables. For all possible output shares from $\mathcal{D}$ $1 \dots (d+1)^n$ we associate a $\{0,1\}$ variable x_S denoting if share S is chosen. Now our goal of finding correct and non-complete minimal sharing can be formulated as:

$$\text{minimize} \sum_{S \in \mathcal{D}} x_S \tag{2}$$

$$\text{subject to} \sum_{S : e \in S} x_S \geq 1, \ (\forall e, e \in \mathcal{U}) \tag{3}$$

Expression (2) is referred to as objective function. While inequalities given by (3) are called constraints.

Set covering problem is well known discrete optimization, that pops up in various application, for example logical minimizer, mobile network network base station placement, etc. Hence since we have reduced the problem of finding minimal sharing to set covering, we can try and use discrete optimization methods to find good solutions to it.

2.1 Optimization techniques

We have used 4 different techniques to solve the underlying set covering problem: Constraint Programming, Mixed Integer Programming, Randomized Greedy with restarts, and Simmulated Annealing with greedy heuristic.

Constraint Programming (CP) [RBW06] focuses on finding feasible solution given a number of constraints. Its original use is to determine if problem is satisfiable. But it can also be used in minimization optimization, by finding a feasible solution with objective cost N, then adding new constraint such that objective functions has to be $\leq N$, and restarting the process. The cycle is repeated until problem becomes unfeasible, and final objective value where feasible solution is found is the minimal one. We have used freely available MiniZinc [NSB+07] software to solve our set covering problem.

Mixed Integer Linear Programming (MILP) [Sch86] relaxes the problem such that decision variables becomes non-binary, but continuous real values, $x_S \in [0, 1]$, then tries to solve underlying Linear Programming Problem [Dan63] to establish lower bound of the objective function. Afterwards it tries to find a smallest solution such that all decision variables are integer, satisfying the original problem constraints. Like CP, MILP is able to prove optimality of the solution. We have used Gurobi 9.0 [GO20] solver for this part.

Randomized Greedy heuristic with restarts, or Iterated Greedy (IG) is a technique where solution is constructed in a greedy manner, and in each step we take set S that covers most uncovered elements so far. We stop when all elements are covered. All ties are broken randomly, i.e. if multiple sets cover equal number of still uncovered elements, algorithm randomly chooses one of them to add to the solution being built. We loop this approach multiple times, and in the end take the solution from the iteration that has smallest number of sets. Since ties are broken randomly, solutions will differ from run to run. Optimality of this technique cannot be proven, since greedy algorithm finds local minima, not a global one for SCP. However, in practice solutions it finds are often very close to optimal.

Simulated Annealing [KGV83] is meta-heuristic where a neighbor solution with a higher objective function cost is accepted with a probably that gradually decreases during execution time. Intuitively, accepting worse solutions allows us to explore more of the search space and escape local minima. Over time lowering of the acceptance probability guides the search more and more toward good solutions, while the earlier rounds are used to explore large search space in a more indiscriminate fashion. The probability parameter is called temperature. We utilized implementation approach given in [BJT99, Min08] where we separate execution into multiple rounds. After each round temperature is decreased by a constant factor *cool*. In each round a number I of neighbors is explored. Neighbor is constructed by removing some of the sets from the solution, then constructing new solution using remaining sets as a starting point, and adding new ones until all elements are covered again. We accept new solution if it is better than the previous one, or if not we still accept it with probability $exp(-\delta/temp)$ where δ is the difference in objective functions of new and current solution.

Here we reiterate implementation details from [Min08]. Parameters:

- **A** data structure providing relation information of which sets cover which element, normally given as a $\{0, 1\}$ matrix.
- **cool** Temperature reduction ratio between rounds $temp_{r+1} = temp_r \times cool$
- ρ Used to determine the percentage of shares to be dropped from current solution, during neighbor construction.
- **R** Number of rounds to run.
- **I** Number of neighbors examined in each round.
- **rand()** Function that returns uniformly randomly value in range $[0, 1]$.
- **temp** Temperature parameter used to determine the probability of accepting bad neighbor.

- **Construct()** Greedy heuristic method used to construct new covering after a percentage of shares has been removed from it.
- **Perturb()** Neighbor defining function. ρ factor of shares is stripped from the current solution. Shares removed have the least amount of uniquely covered elements
- **RemoveRedundant()** Executed after a new solution has been constructed. It can happen that some of the chosen shares are redundant since all of the elements covered by it are already fully covered by other shares in the candidate solution.

Simulated annealing algorithm for SCP is given by Algorithm 1.

Algorithm 1 Simulated Annealing algorithm for set cover problem.

Input: A, R, I, $temp$, $cool$, ρ
Result: Smallest found sharing $\mathcal{C}_{best}$
$\mathcal{C}^* := \{\}$
$\mathcal{C}^* := Construct(A, \mathcal{C}^*)$
$r := 0$
repeat
  $i := 0$
  repeat
    $\mathcal{C}' := Perturb(A, \mathcal{C}^*, \rho)$
    $\mathcal{C}' := Construct(A, \mathcal{C}')$
    $\mathcal{C}' := RemoveRedundant(A, \mathcal{C}')$
    $\delta := ObjectiveCost(\mathcal{C}') - ObjectiveCost(\mathcal{C}^*)$
    $rnd = rand()$
    if $(\delta \leq 0)$ *or* $(rnd \leq e^{-\delta/temp})$ **then**
      $\mathcal{C}^* := \mathcal{C}'$
      if $ObjectiveCost(\mathcal{C}' < ObjectiveCost(\mathcal{C}_{best})$ **then**
        $\mathcal{C}_{best} := \mathcal{C}'$
      end
    end
    $i := i + 1$
  until $i == I$
  $temp := temp \times cool$
  $r := r + 1$
until $r == R$
return $\mathcal{C}_{best}$

3 Sharing solutions

We have applied four discrete optimization techniques to all algebraic functions of up to 8 bits. Since optimal sharing has already been explored for case where degree $t \geq n-1$ with n number of bits, we have focused on algebraic functions where $t < n-1$. In order for the solution to be generic enough, we further assume the most extreme case. That is, if we have n variables and degree t we search for a sharing of a function that has all $\binom{n}{t}$ t-degree terms present in the ANF. Sharing of such function can be used for any n-bit function of degree t. However, a more efficient sharing for a given n-bit function F of degree t might exists, depending on the number of t-degree terms that are present in the ANF and their structure, and same discrete optimization methodology can be applied on F to find possibly better sharing.

First we have applied CP using Minizinc [NSB+07] with Chuffed 0.10.4 [CS14] and Google OR-tools [PF20] solvers. For $d = 1$, both options are able to find optimal sharing

Table 1: Number of shares found using CP solver. Values in bold mean that solver proved optimality.

	$d=1$					$d=2$				
t	2	3	4	5	6	2	3	4	5	6
$n=4$	**5**					**9**				
$n=5$	**6**	**10**				15	44			
$n=6$	**6**	**12**	**21**			–	–	–		
$n=7$	**6**	**12**	**24**	**42**		–	–	–	–	
$n=8$	**6**	**12**	**24**	52	**85**	–	–	–	–	–

Table 2: Number of shares found using MILP solver. Values in bold mean that solver proved optimality.

	$d=1$					$d=2$				
t	2	3	4	5	6	2	3	4	5	6
$n=4$	**5**					**9**				
$n=5$	**6**	**10**				**11**	**33**			
$n=6$	**6**	**12**	**21**			**12**	**33**	119		
$n=7$	**6**	**12**	**24**	**42**		**12**	45	153	440	
$n=8$	**6**	**12**	**24**	52	**85**	14	63	–	–	–

for all cases, except for 8 bit functions of degree 5, where a solution with 52 shares is found after few hours but without proof of optimality, even after running for the CP solver for several days on a regular PC. Optimality for other cases is proven within several tens of minutes. For second order case $d = 2$, CP solver is only able to prove optimality for the simplest of cases of 4 bits and degree 2. It is able to find some solutions when $n = 5$, but without proof of optimality. For $n > 5$, solver is unable to provide any solutions within few minutes, so we deem it not very suitable for those cases, due to the size of the search space. We did not see much difference in solution times between Chuffed and OR-tools solvers, although Chuffed seems to be able to finish the search a bit faster. The resulting number of shares using CP solvers is given in Table 1.

Next we have tried to use MILP solver, in particular Gurobi 9.0 solver [GO20]. For $d = 1$ the solver was able to find same solutions and prove optimality in less time. However, for the case of 8 bits and degree 5, solution of 52 was found again, but without proof of optimality. When $d = 2$, MILP solver was more successful than CP one, finding optimal solutions for degree 2 functions for all $n = 4, 5, 6, 7$, and degree three functions of 5 and 6 bits. However, more complex case seem to quickly become difficult for the solver. The resulting number of shares using MILP solver is given in Table 2.

It becomes apparent that CP and MILP solvers struggle with $d = 2$ which is no surprise due to the exponential increase in the number of decision variables. Hence for the more difficult cases heuristics are the only possible way to find good solutions. Iterated Greedy approach was done by using 100000 runs and choosing the best solution among these runs. It finishes is just a matter of seconds even for harder cases. The program is sometimes able to find optimal solutions of CP and MILP approaches, but even when it does not the solutions it finds are within 30 percent of minimal, where minimal solution is known. IG run results are given in Table 3.

As a way to improve on the results of IG heurstic, we have also tested the simulated annealing technique. First of all, IG technique was used with 20000 runs to provide initial solution, and then SA was run with $R = 150$ rounds and $I = 100$ iterations in each round.

Table 3: Number of shares found using IG heuristic.

	$d=1$					$d=2$				
t	2	3	4	5	6	2	3	4	5	6
$n=4$	5					9				
$n=5$	6	12				11	36			
$n=6$	6	12	22			13	40	128		
$n=7$	6	12	24	47		14	45	138	409	
$n=8$	6	12	30	56	96	15	45	135	405	1387

Table 4: Number of shares found using SA heuristic.

	$d=1$					$d=2$				
t	2	3	4	5	6	2	3	4	5	6
$n=4$	5					9				
$n=5$	6	10				11	33			
$n=6$	6	12	21			12	33	115		
$n=7$	6	12	24	42		12	40	130	379	
$n=8$	6	12	24	52	85	14	45	135	405	1234

Program ran extremely fast on a regular PC, and it was the slowest for 8 bit functions of degree 6 where it finished in about 5 minutes. SA program was able to find same results for $d=1$ as CP and MILP solver, finding optimal solutions in much faster times. For $d=2$ it was able to improve on some of the instances compared to IG approach, but solution quality was at most 10 percent better in instances where CP and MILP were unable to provide solutions in a reasonable amount of time. In some instances with 8 bits and degrees 3,4 and 5, it was unable to improve upon the solution provided by the IG approach. Modifying SA annealing parameters had limited impact on the quality of the solution. We determine that cooling factor of 0.91, starting temperature of 100, and removal coeficient ρ of 0.2 to be working good in almost all cases. ρ had most impact on the improvements, and we notice that smaller values are more beneficial for larger t values, about 0.1, while values of 0.3 work better on smaller t values. SA run results are given in Table 4.

Finally, we can put together the solutions of all four approaches to collect the best ones. Aggregate results are presented in Table 5.

Examining the best found solutions we can determine that the optimal sharing does not give a large increase in the number of output shares compared to the trivial bound of $(d+1)^t$. For $d=1$ increase is up to 50 percent, except in the hardest case with 8 variables

Table 5: Best sharing using all four approaches. Values in bold mean that solver proved optimality.

	$d=1$					$d=2$				
t	2	3	4	5	6	2	3	4	5	6
$n=4$	**5**					**9**				
$n=5$	**6**	**10**				**11**	**33**			
$n=6$	**6**	**12**	**21**			**12**	**33**	115		
$n=7$	**6**	**12**	**24**	**42**		**12**	40	130	379	
$n=8$	**6**	**12**	**24**	52	**85**	14	45	135	405	1234

Table 6: Comparison to previously known results.

	$d=1$					$d=2$				
t	2	3	4	5	6	2	3	4	5	6
Proposed methodology										
$n=4$	**5**					**9**				
$n=5$	**6**	**10**				**11**	**33**			
$n=6$	**6**	**12**	**21**			**12**	**33**	115		
$n=7$	**6**	**12**	**24**	**42**		**12**	40	130	379	
$n=8$	**6**	**12**	**24**	52	**85**	14	45	135	405	1234
Construction from [BKN20]										
$n=4$	8					27				
$n=5$	16	16				81	81			
$n=6$	32	32	32			243	243	243		
$n=7$	64	64	64	64		729	729	729	729	
$n=8$	128	128	128	128	128	2187	2187	2187	2187	2187
Construction from [WMM20]										
$n=4$	≥ 8					**9**				
$n=5$	≥ 8	≥ 16				≥ 18	≥ 54			
$n=6$	≥ 8	≥ 16	≥ 32			≥ 18	≥ 54	≥ 162		
$n=7$	≥ 8	≥ 16	≥ 32	≥ 64		≥ 18	≥ 54	≥ 162	≥ 486	
$n=8$	≥ 8	≥ 16	≥ 32	≥ 64	≥ 128	≥ 18	≥ 54	≥ 162	≥ 486	≥ 1458

and degree 5 functions where, we have 52 shares compared to the bound 32 shares, an increase of a little over than 60 percent. Similar situation happens with $d=2$ where found solutions are within 70 percent increase. Due to the particular case of SCP for sharing being somewhat pathological, where all shares cover equal amount of elements, it is very hard for solver to find optimal solutions with increasing number of total shares, while good solutions are still relatively easily discovered using greedy heuristic.

Comparison of our result with the recent ones presented in [BKN20] and [WMM20] is present in Table 6. Obviously, the method presented in [BKN20], although achieving optimality when $t=n-1$ is very ineffective in cases where $t<n-1$. Greedy heuristic given in [WMM20] finds solutions that are multiples of $(d+1)^t$. However, the authors only presented the solution for a very specific case of 8-bit degree 3 function, and for $(n=4,t=2,d=2)$, $(n=4,t=2,d=3)$, $(n=5,t=2,d=4)$, $(n=5,t=3,d=3)$, $(n=6,t=3,d=3)$ cases optimal solution is found while not providing more information. Hence in Table 6 we indicated the smallest number of output shares algorithm presented in [WMM20] could potentially find. Our method provides better results in all cases for first and second security order.

Concrete sharings are given by Tables 7, 8, 9 and 10 in the Appendix. If we examine the found solutions for $d=1$ we can see that many of the solutions have symmetric order. One would assume that optimal solutions will always have sharings with such structure, but apparently this is not the case. For example, if we provide this additional constraint to the CP solver, it will no longer find optimal sharing of 7 bit functions of degree 2 to be 12, but 14. Symmetric structure is obtained from the CP solver, probably based on the heuristic it was using to parse the search space.

4 Conclusion

Low-latency requirements for side-channel implementations impose a challenge for design of TI circuits due to the difficult and possibly NP-hard problem of finding the smallest

possible output sharing. While some special cases have been addressed previously, and some upper bounds on the number of output shares have been established, the gap for the quality of optimal or even good sharing still quite large. Work presented here provided optimal or near optimal sharings for many types of functions that were previously unknown.

We have also presented a methodology of how to find optimal sharing by solving the underlying set covering problem using different discrete optimization techniques. And it is easily applicable to any Boolean function. However, if higher security order is desired, only feasible option is to use heuristic methods, due to the size of the search space.

Nonlinear Boolean functions of up to 8 bits are used in almost all cryptographic constructs, hence the results shown here are applicable to practically all currently in use symmetric designs.

Finally, we determine that generic problem of finding optimal $d+1$ sharing of any function and any security order d remains an open problem. While discrete optimization techniques can be used to find good solutions in most practical applications, further advances that utilize the structure of underlying set covering problem are required to find optimal/good solutions for higher security orders.

Acknowledgement

The author would like to thank Ventzislav Nikov for fruitful discussions and helpful advice on various technical and editorial aspects of this paper.

References

[ABP+18] Victor Arribas, Begül Bilgin, George Petrides, Svetla Nikova, and Vincent Rijmen. Rhythmic keccak: Sca security and low latency in hw. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):269–290, Feb. 2018.

[BJT99] M.J. Brusco, L.W. Jacobs, and G.M. Thompson. A morphing procedure to supplement a simulated annealing heuristic for cost- andcoverage-correlated set-covering problems. *Annals of Operations Research*, 86(0):611–627, 1999.

[BKN20] Dušan Božilov, Miroslav Knežević, and Ventzislav Nikov. Optimized threshold implementations: Minimizing the latency of secure cryptographic accelerators. In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications*, pages 20–39, Cham, 2020. Springer International Publishing.

[CS14] Geoffrey Chu and Peter J. Stuckey. Chuffed solver description, 2014. https://github.com/chuffed/chuffed.

[Dan63] George B. Dantzig. *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963.

[GIB18] Hannes Gross, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2):1–21, May 2018.

[GO20] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020. http://www.gurobi.com.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.

[Min08] Dev Minotra. A study of heuristic-algorithms for set-covering problems, 06 2008.

[MS16] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 517–547, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS 2006*, pages 529–545. Springer LNCS, 2006.

[NSB+07] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In Christian Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, pages 529–543, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[PF20] Laurent Perron and Vincent Furnon. OR-Tools, 2020. https://developers.google.com/optimization/.

[RBW06] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., USA, 2006.

[SBHM20] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-latency hardware masking with application to aes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):300–326, Mar. 2020.

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Inc., USA, 1986.

[UHA17] Rei Ueno, Naofumi Homma, and Takafumi Aoki. Toward More Efficient DPA-Resistant AES Hardware Architecture Based on Threshold Implementation. In *Constructive Side-Channel Analysis and Secure Design - COSADE 2017*, pages 50–64, 2017.

[WMM20] Felix Wegener, Lauren Meyer, and Amir Moradi. Spin me right round rotational symmetry for fpga-specific aes: Extended version. *Journal of Cryptology*, 01 2020.

A Exact Sharings

Here we give a quick reference for the found sharings for the cases examined in section 3. To make the notation as succinct as possible, we will only enumerate the chosen shares in their lexicographical order, first first share having index 0. For example if we had a sharing with $d = 2, n = 4, t = 2$ given as $[2, 12, 25, 31, 44, 45, 60, 64, 77]$, it means that the actual nine shares are

$$(0,0,0,2)\ (0,1,1,0)\ (0,2,2,1)\ (1,0,1,1)\ (1,1,2,2)\ (1,2,0,0)\ (2,0,2,0)\ (2,1,0,1)\ (2,2,1,2)$$

Table 7: Sharing indices of best shares for security order $d = 1$.

	t	Sharing indices
$n = 4$		
	2	$(1, 6, 8, 11, 13)$
$n = 5$		
	2	$(3, 12, 20, 24, 29, 30)$
	3	$(2, 5, 8, 11, 14, 17, 20, 23, 26, 29)$
$n = 6$		
	2	$(2, 21, 30, 35, 45, 56)$
	3	$(3, 9, 10, 15, 20, 27, 36, 43, 48, 53, 54, 60)$
	4	$(1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61)$
$n = 7$		
	2	$(24, 43, 54, 66, 85, 109)$
	3	$(6, 25, 37, 43, 50, 60, 67, 76, 85, 90, 96, 127)$
	4	$(0, 9, 14, 21, 23, 26, 35, 36, 47, 50, 57, 60, 67,$ $70, 77, 80, 91, 92, 101, 104, 106, 113, 118, 127)$
	5	$(1, 6, 10, 12, 15, 16, 19, 21, 25, 30, 32, 35, 37, 41, 46, 50, 52, 55, 56, 59, 61, 66, 68,$ $71, 72, 75, 77, 81, 86, 90, 92, 95, 97, 102, 106, 108, 111, 112, 115, 117, 121, 126)$
$n = 8$		
	2	$(15, 64, 119, 154, 177, 236)$
	3	$(12, 27, 33, 54, 85, 106, 130, 189, 207, 216, 228, 243)$
	4	$(9, 20, 31, 36, 42, 51, 66, 71, 88, 109, 113, 126, 129, 142, 146,$ $167, 184, 189, 204, 213, 219, 224, 235, 246)$
	5	$(1, 6, 8, 15, 19, 28, 36, 43, 50, 53, 57, 62, 69, 74, 80, 87, 89, 94, 96, 99, 109, 110,$ $118, 122, 124, 127, 128, 131, 133, 137, 145, 148, 154, 159, 161, 162, 173, 174,$ $183, 184, 193, 198, 203, 204, 210, 221, 231, 232, 241, 244, 251, 254)$
	6	$(3, 5, 6, 9, 10, 12, 17, 18, 20, 24, 31, 33, 34, 36, 40, 47, 48, 55, 59, 61, 62, 65, 66,$ $68, 72, 79, 80, 87, 91, 93, 94, 96, 103, 107, 109, 110, 115, 117, 118, 121, 122,$ $124, 129, 130, 132, 136, 143, 144, 151, 155, 157, 158, 160, 167, 171, 173, 174,$ $179, 181, 182, 185, 186, 188, 192, 199, 203, 205, 206, 211, 213, 214, 217, 218,$ $220, 227, 229, 230, 233, 234, 236, 241, 242, 244, 248, 255)$

Table 8: Sharing indices of best shares for security order $d = 2$, part 1.

	t	Sharing indices
$n = 4$		
	2	$(2, 12, 25, 31, 44, 45, 60, 64, 77)$
$n = 5$		
	2	$(0, 49, 71, 93, 106, 110, 139, 185, 195, 199, 234)$
	3	$(0, 5, 17, 19, 31, 38, 51, 61, 64, 66, 77, 87, 94, 101, 109, 116, 117, 131, 138, 152, 153, 160, 169, 171, 183, 188, 192, 203, 205, 208, 218, 231, 238)$
$n = 6$		
	2	$(0, 41, 143, 238, 295, 369, 408, 470, 555, 580, 674, 676)$
	3	$(0, 52, 68, 104, 112, 145, 159, 178, 201, 209, 224, 269, 275, 280, 312, 331, 336, 369, 380, 416, 438, 481, 499, 534, 547, 560, 586, 611, 624, 653, 672, 676, 711)$
	4	$(7, 11, 23, 24, 32, 34, 39, 44, 46, 54, 67, 75, 80, 82, 86, 96, 103, 108, 121, 128, 132, 143, 146, 147, 158, 160, 165, 176, 178, 180, 188, 191, 199, 204, 210, 220, 222, 225, 233, 235, 246, 251, 256, 261, 268, 271, 276, 281, 292, 302, 307, 312, 317, 324, 331, 338, 344, 359, 360, 367, 372, 382, 395, 397, 402, 407, 415, 420, 427, 436, 446, 450, 458, 459, 466, 471, 482, 488, 490, 498, 503, 505, 513, 529, 536, 537, 547, 549, 554, 562, 573, 577, 588, 593, 598, 608, 609, 613, 623, 624, 637, 639, 655, 657, 671, 678, 683, 686, 700, 703, 707, 715, 719, 722, 726)$
$n = 7$		
	2	$(0, 483, 632, 679, 872, 995, 1144, 1257, 1525, 1667, 1812, 2050)$
	3	$(1, 131, 173, 222, 288, 349, 380, 445, 521, 552, 570, 611, 721, 753, 873, 877, 920, 977, 1009, 1043, 1086, 1209, 1264, 1316, 1393, 1419, 1435, 1488, 1501, 1532, 1547, 1642, 1762, 1794, 1863, 1916, 1953, 2053, 2103, 2174)$
	4	$(0, 23, 52, 65, 66, 97, 111, 118, 149, 157, 166, 186, 198, 209, 224, 255, 268, 287, 302, 315, 330, 365, 371, 379, 389, 404, 425, 439, 453, 472, 496, 515, 517, 519, 565, 572, 585, 601, 636, 665, 687, 694, 702, 725, 737, 748, 769, 807, 822, 852, 857, 871, 873, 887, 905, 919, 944, 948, 976, 999, 1022, 1043, 1064, 1069, 1088, 1099, 1110, 1152, 1176, 1190, 1200, 1213, 1224, 1258, 1272, 1285, 1289, 1297, 1322, 1334, 1344, 1363, 1383, 1399, 1409, 1441, 1473, 1490, 1503, 1513, 1541, 1564, 1583, 1614, 1629, 1633, 1653, 1669, 1690, 1694, 1703, 1738, 1740, 1761, 1777, 1805, 1813, 1833, 1845, 1866, 1870, 1880, 1882, 1901, 1931, 1951, 1955, 1965, 1997, 2012, 2038, 2040, 2052, 2087, 2098, 2108, 2145, 2149, 2164, 2184)$

Table 9: Sharing indices of best shares for security order $d = 2$, part 2.

	t	Sharing indices
$n = 7$		
	5	(0, 8, 14, 19, 24, 30, 36, 43, 47, 49, 56, 58, 69, 77, 83, 90, 95, 97, 103, 109, 113, 123, 134, 135, 143, 148, 154, 156, 163, 165, 179, 184, 185, 196, 199, 201, 206, 207, 221, 222, 227, 241, 245, 250, 256, 258, 266, 276, 282, 289, 296, 300, 307, 314, 315, 325, 327, 341, 342, 349, 353, 365, 367, 372, 382, 385, 389, 393, 401, 410, 414, 421, 425, 429, 432, 436, 443, 457, 460, 467, 472, 478, 480, 481, 485, 490, 497, 502, 507, 514, 521, 527, 531, 545, 546, 552, 560, 565, 572, 573, 577, 579, 593, 597, 601, 603, 614, 616, 622, 635, 637, 645, 648, 655, 662, 667, 677, 688, 690, 698, 699, 705, 711, 719, 724, 731, 736, 741, 746, 751, 764, 767, 769, 779, 780, 784, 786, 799, 801, 808, 809, 814, 816, 820, 830, 837, 841, 849, 853, 856, 860, 871, 872, 873, 878, 888, 896, 900, 910, 915, 920, 925, 933, 940, 945, 958, 962, 965, 966, 972, 977, 982, 992, 997, 1000, 1016, 1020, 1032, 1035, 1040, 1048, 1061, 1064, 1066, 1068, 1074, 1087, 1089, 1100, 1102, 1104, 1107, 1112, 1123, 1126, 1128, 1133, 1135, 1146, 1160, 1164, 1169, 1171, 1175, 1179, 1190, 1192, 1203, 1211, 1213, 1216, 1224, 1238, 1239, 1242, 1246, 1257, 1262, 1267, 1277, 1280, 1281, 1282, 1288, 1297, 1310, 1314, 1321, 1328, 1333, 1340, 1344, 1352, 1353, 1358, 1365, 1372, 1376, 1380, 1385, 1388, 1393, 1399, 1405, 1410, 1416, 1430, 1438, 1441, 1445, 1449, 1461, 1468, 1484, 1487, 1492, 1500, 1504, 1506, 1517, 1518, 1521, 1526, 1534, 1540, 1544, 1554, 1560, 1565, 1572, 1577, 1579, 1584, 1591, 1596, 1603, 1610, 1613, 1628, 1631, 1633, 1638, 1645, 1647, 1651, 1661, 1673, 1675, 1686, 1690, 1697, 1698, 1705, 1708, 1713, 1718, 1719, 1722, 1728, 1733, 1739, 1741, 1753, 1756, 1763, 1769, 1771, 1775, 1779, 1789, 1791, 1801, 1805, 1806, 1812, 1819, 1826, 1832, 1838, 1842, 1848, 1850, 1858, 1865, 1869, 1873, 1877, 1885, 1889, 1897, 1905, 1910, 1911, 1920, 1926, 1934, 1936, 1946, 1950, 1958, 1961, 1963, 1974, 1981, 1993, 1997, 1998, 2005, 2013, 2019, 2021, 2025, 2036, 2038, 2041, 2048, 2053, 2060, 2064, 2076, 2083, 2090, 2097, 2098, 2104, 2110, 2121, 2126, 2127, 2131, 2138, 2142, 2144, 2149, 2152, 2162, 2166, 2173, 2186)
$n = 8$		
	2	(0, 1255, 1923, 2045, 2347, 3100, 3210, 3599, 3844, 4729, 4796, 4926, 5490, 5909)
	3	(33, 424, 512, 635, 740, 919, 1191, 1348, 1534, 1608, 1706, 1830, 1907, 2025, 2170, 2261, 2277, 2354, 2589, 2626, 2710, 2784, 3148, 3227, 3269, 3423, 3852, 3982, 4193, 4318, 4510, 4667, 4846, 5037, 5223, 5287, 5353, 5445, 5633, 5730, 5825, 5861, 5939, 6138, 6511)
	4	(5, 33, 92, 151, 188, 207, 238, 268, 318, 378, 446, 495, 613, 644, 682, 807, 839, 904, 954, 984, 1015, 1037, 1102, 1133, 1140, 1162, 1273, 1310, 1338, 1397, 1495, 1560, 1651, 1682, 1703, 1731, 1799, 1936, 1997, 2016, 2026, 2057, 2085, 2122, 2172, 2261, 2268, 2299, 2364, 2497, 2562, 2594, 2653, 2674, 2705, 2733, 2792, 2820, 2857, 2888, 2955, 2986, 3023, 3042, 3073, 3083, 3133, 3194, 3250, 3368, 3396, 3426, 3448, 3486, 3635, 3652, 3770, 3789, 3857, 3906, 3937, 3975, 4028, 4062, 4093, 4115, 4145, 4195, 4291, 4320, 4387, 4418, 4474, 4593, 4645, 4676, 4710, 4741, 4763, 4797, 4859, 4908, 4949, 5064, 5086, 5103, 5165, 5252, 5317, 5369, 5457, 5488, 5544, 5605, 5627, 5692, 5742, 5752, 5783, 5811, 5858, 5877, 5908, 5946, 5968, 6005, 6033, 6144, 6181, 6203, 6241, 6374, 6439, 6504, 6557)
	5	(6, 9, 21, 35, 38, 61, 76, 100, 123, 140, 152, 188, 202, 214, 216, 245, 271, 295, 309, 330, 333, 374, 388, 409, 424, 447, 450, 464, 476, 479, 502, 516, 545, 560, 569, 581, 595, 607, 619, 633, 645, 657, 669, 683, 724, 734, 775, 798, 801, 810, 834, 839, 851, 863, 865, 877, 898, 901, 939, 956, 979, 982, 994, 1005, 1020, 1037, 1049, 1070, 1084, 1110, 1125, 1146, 1163, 1175, 1213, 1239, 1256, 1268, 1270, 1303, 1318, 1332, 1358, 1394, 1397, 1408, 1420, 1434, 1446, 1449, 1471, 1485, 1497, 1538, 1550, 1562, 1573, 1614, 1623, 1635, 1652, 1667, 1678, 1690, 1693, 1719, 1745, 1759, 1771, 1795, 1807, 1809, 1821, 1833, 1838, 1862, 1871, 1886, 1900, 1923, 1952, 1955, 1967, 1981, 1993, 2004, 2007, 2028, 2072, 2095, 2098, 2107, 2133, 2157, 2174, 2204, 2207, 2218, 2259, 2280, 2297, 2323, 2347, 2356, 2359, 2382, 2385, 2397, 2414, 2426, 2452, 2466, 2478, 2492, 2516, 2528, 2531, 2542, 2554, 2568, 2583, 2616, 2633, 2647, 2659, 2673, 2697, 2714, 2726, 2728, 2740, 2752, 2761, 2787, 2802, 2819, 2840, 2852, 2881, 2904, 2929, 2941, 2943, 2955, 2972, 2984, 2996, 3020, 3034, 3046, 3057, 3081, 3096, 3122, 3136, 3162, 3174, 3206, 3229, 3241, 3291, 3308, 3320, 3329, 3344, 3355, 3358, 3370, 3381, 3384, 3413, 3436, 3439, 3477, 3486, 3498, 3501, 3515, 3530, 3553, 3556, 3577, 3591, 3620, 3644, 3651, 3680, 3695, 3709, 3730, 3742, 3745, 3768, 3771, 3785, 3797, 3809, 3859, 3873, 3885, 3890, 3902, 3916, 3940, 3942, 3954, 3966, 3978, 4004, 4007, 4045, 4066, 4069, 4080, 4124, 4135, 4150, 4173, 4190, 4238, 4252, 4266, 4278, 4302, 4314, 4328, 4331, 4343, 4354, 4369, 4375, 4387, 4425, 4442, 4463, 4478, 4492, 4504, 4515, 4518, 4530, 4539, 4568, 4583, 4606, 4620, 4632, 4649, 4661, 4664, 4687, 4690, 4711, 4723, 4725, 4778, 4790, 4813, 4839, 4854, 4883, 4897, 4920, 4923, 4959, 4985, 4988, 4999, 5023, 5035, 5047, 5049, 5061, 5078, 5102, 5112, 5138, 5164, 5179, 5200, 5214, 5226, 5258, 5267, 5279, 5291, 5293, 5317, 5331, 5343, 5348, 5372, 5386, 5400, 5436, 5448, 5462, 5465, 5488, 5491, 5503, 5527, 5550, 5553, 5567, 5593, 5608, 5619, 5631, 5634, 5648, 5660, 5672, 5684, 5722, 5736, 5757, 5801, 5815, 5827, 5849, 5852, 5863, 5875, 5878, 5889, 5904, 5913, 5937, 5954, 5968, 6016, 6030, 6056, 6059, 6082, 6123, 6140, 6152, 6161, 6187, 6202, 6225, 6237, 6249, 6261, 6278, 6290, 6292, 6304, 6330, 6342, 6347, 6397, 6406, 6409, 6432, 6447, 6461, 6464, 6476, 6497, 6511, 6523, 6552)

Table 10: Sharing indices of best shares for security order $d = 2$, $n = 8$, $t = 6$.

Shares used
(1, 12, 20, 24, 33, 38, 41, 43, 50, 54, 58, 62, 68, 75, 79, 84, 87, 90, 97, 103, 107, 108, 113, 115, 116, 119, 125, 127, 129, 137, 138, 145, 148, 159, 162, 170, 173, 175, 185, 193, 201, 207, 209, 214, 223, 226, 230, 231, 236, 245, 246, 260, 265, 277, 279, 283, 288, 289, 296, 302, 307, 321, 328, 332, 335, 336, 344, 353, 357, 361, 374, 379, 392, 393, 396, 403, 410, 414, 421, 424, 429, 433, 437, 449, 450, 453, 459, 462, 465, 470, 471, 481, 485, 494, 496, 500, 504, 511, 515, 516, 528, 535, 544, 549, 552, 556, 560, 568, 572, 582, 583, 588, 598, 605, 606, 618, 621, 628, 638, 640, 644, 651, 655, 657, 665, 668, 670, 681, 684, 689, 691, 694, 701, 704, 715, 726, 732, 738, 743, 745, 751, 758, 760, 768, 774, 782, 790, 794, 798, 802, 806, 815, 821, 825, 831, 835, 840, 847, 850, 860, 861, 865, 872, 876, 882, 892, 897, 903, 909, 917, 925, 927, 935, 938, 940, 945, 950, 958, 969, 973, 980, 987, 990, 995, 1004, 1005, 1010, 1024, 1028, 1029, 1035, 1039, 1043, 1044, 1053, 1060, 1063, 1070, 1074, 1078, 1081, 1089, 1096, 1100, 1101, 1112, 1113, 1118, 1119, 1129, 1133, 1136, 1138, 1140, 1148, 1159, 1162, 1164, 1169, 1173, 1183, 1185, 1195, 1197, 1204, 1207, 1211, 1215, 1222, 1227, 1235, 1239, 1243, 1245, 1255, 1259, 1261, 1263, 1274, 1275, 1279, 1287, 1295, 1302, 1309, 1310, 1315, 1322, 1331, 1337, 1341, 1348, 1352, 1354, 1359, 1366, 1371, 1382, 1387, 1392, 1399, 1404, 1415, 1417, 1419, 1427, 1432, 1439, 1445, 1451, 1452, 1456, 1458, 1466, 1471, 1476, 1481, 1483, 1490, 1495, 1500, 1505, 1507, 1513, 1515, 1525, 1529, 1536, 1541, 1543, 1549, 1553, 1554, 1558, 1574, 1578, 1584, 1591, 1599, 1602, 1607, 1609, 1613, 1615, 1623, 1627, 1629, 1637, 1642, 1648, 1652, 1653, 1658, 1660, 1663, 1666, 1676, 1686, 1693, 1695, 1700, 1706, 1707, 1712, 1713, 1717, 1720, 1727, 1728, 1732, 1745, 1749, 1762, 1764, 1769, 1775, 1777, 1780, 1783, 1791, 1805, 1806, 1812, 1816, 1822, 1823, 1826, 1828, 1835, 1838, 1846, 1851, 1857, 1871, 1876, 1883, 1884, 1892, 1900, 1905, 1913, 1915, 1917, 1921, 1925, 1934, 1935, 1941, 1945, 1948, 1959, 1965, 1967, 1979, 1980, 1985, 1987, 1990, 1994, 1995, 2000, 2004, 2009, 2016, 2020, 2028, 2033, 2036, 2043, 2050, 2052, 2062, 2067, 2072, 2074, 2080, 2084, 2092, 2094, 2105, 2108, 2110, 2116, 2120, 2128, 2130, 2138, 2140, 2145, 2150, 2151, 2163, 2169, 2176, 2183, 2187, 2192, 2194, 2200, 2204, 2206, 2208, 2217, 2223, 2230, 2234, 2243, 2251, 2264, 2265, 2270, 2276, 2278, 2286, 2291, 2293, 2296, 2304, 2309, 2310, 2321, 2326, 2328, 2334, 2339, 2341, 2353, 2363, 2364, 2368, 2375, 2378, 2383, 2387, 2398, 2400, 2406, 2411, 2412, 2419, 2426, 2434, 2436, 2439, 2442, 2450, 2455, 2458, 2465, 2471, 2478, 2484, 2495, 2499, 2500, 2506, 2510, 2512, 2524, 2528, 2535, 2541, 2549, 2554, 2556, 2560, 2570, 2572, 2574, 2585, 2586, 2594, 2595, 2599, 2602, 2615, 2619, 2632, 2634, 2638, 2645, 2647, 2660, 2666, 2670, 2678, 2684, 2688, 2689, 2695, 2699, 2700, 2704, 2712, 2717, 2724, 2725, 2728, 2730, 2733, 2741, 2745, 2757, 2761, 2763, 2768, 2779, 2781, 2789, 2794, 2800, 2804, 2810, 2818, 2821, 2823, 2829, 2834, 2836, 2843, 2847, 2853, 2867, 2868, 2872, 2882, 2883, 2889, 2893, 2906, 2914, 2924, 2926, 2934, 2939, 2941, 2944, 2957, 2965, 2967, 2974, 2976, 2982, 2987, 2990, 3001, 3003, 3014, 3017, 3024, 3026, 3031, 3037, 3043, 3045, 3056, 3060, 3067, 3073, 3077, 3080, 3090, 3094, 3100, 3108, 3113, 3115, 3120, 3123, 3128, 3133, 3143, 3153, 3157, 3160, 3162, 3170, 3185, 3186, 3191, 3196, 3201, 3208, 3220, 3227, 3232, 3237, 3248, 3252, 3256, 3258, 3262, 3266, 3269, 3271, 3273, 3284, 3292, 3294, 3304, 3309, 3313, 3317, 3318, 3326, 3330, 3340, 3345, 3355, 3362, 3368, 3369, 3377, 3378, 3388, 3392, 3393, 3400, 3406, 3416, 3417, 3421, 3423, 3432, 3437, 3441, 3445, 3449, 3458, 3465, 3469, 3478, 3481, 3483, 3485, 3493, 3506, 3507, 3511, 3515, 3521, 3525, 3527, 3532, 3536, 3540, 3544, 3545, 3551, 3555, 3557, 3564, 3567, 3571, 3575, 3577, 3590, 3593, 3595, 3600, 3615, 3616, 3624, 3630, 3634, 3637, 3641, 3648, 3656, 3660, 3667, 3674, 3679, 3689, 3693, 3698, 3699, 3707, 3713, 3715, 3718, 3733, 3735, 3739, 3747, 3752, 3753, 3758, 3759, 3763, 3770, 3773, 3775, 3781, 3783, 3791, 3804, 3808, 3812, 3813, 3816, 3827, 3831, 3834, 3838, 3846, 3851, 3859, 3868, 3874, 3876, 3879, 3884, 3890, 3895, 3898, 3905, 3906, 3911, 3921, 3924, 3928, 3935, 3940, 3943, 3947, 3957, 3963, 3974, 3975, 3980, 3984, 3988, 3997, 4008, 4019, 4020, 4027, 4031, 4037, 4039, 4041, 4050, 4054, 4064, 4067, 4071, 4075, 4080, 4085, 4088, 4093, 4095, 4106, 4110, 4114, 4116, 4126, 4130, 4131, 4139, 4144, 4155, 4161, 4165, 4169, 4176, 4177, 4181, 4189, 4195, 4197, 4202, 4205, 4209, 4213, 4216, 4224, 4229, 4232, 4241, 4255, 4260, 4272, 4275, 4289, 4291, 4299, 4302, 4309, 4312, 4316, 4321, 4334, 4335, 4342, 4346, 4348, 4352, 4358, 4368, 4373, 4377, 4383, 4396, 4400, 4402, 4409, 4414, 4419, 4426, 4435, 4442, 4443, 4448, 4460, 4462, 4467, 4472, 4474, 4486, 4492, 4502, 4506, 4509, 4520, 4525, 4532, 4534, 4538, 4542, 4546, 4557, 4561, 4566, 4573, 4578, 4580, 4586, 4591, 4595, 4602, 4603, 4608, 4612, 4616, 4618, 4622, 4633, 4638, 4647, 4655, 4660, 4666, 4668, 4675, 4677, 4680, 4683, 4688, 4690, 4694, 4698, 4709, 4713, 4721, 4723, 4727, 4732, 4737, 4739, 4744, 4751, 4755, 4760, 4762, 4765, 4776, 4780, 4787, 4793, 4797, 4801, 4805, 4810, 4812, 4815, 4826, 4831, 4835, 4849, 4854, 4862, 4864, 4866, 4871, 4872, 4879, 4883, 4892, 4894, 4897, 4902, 4907, 4914, 4922, 4927, 4935, 4939, 4941, 4954, 4955, 4959, 4961, 4967, 4968, 4985, 4989, 4993, 4996, 5000, 5001, 5004, 5015, 5017, 5027, 5031, 5038, 5046, 5049, 5050, 5057, 5062, 5067, 5079, 5083, 5087, 5091, 5095, 5099, 5105, 5110, 5116, 5118, 5124, 5136, 5141, 5146, 5153, 5157, 5162, 5167, 5169, 5175, 5179, 5183, 5185, 5193, 5197, 5210, 5216, 5223, 5228, 5229, 5236, 5240, 5244, 5245, 5252, 5259, 5265, 5269, 5276, 5279, 5284, 5289, 5293, 5305, 5313, 5318, 5322, 5327, 5328, 5335, 5339, 5352, 5355, 5359, 5366, 5371, 5377, 5385, 5390, 5391, 5392, 5399, 5401, 5408, 5416, 5421, 5423, 5429, 5430, 5441, 5446, 5448, 5451, 5454, 5462, 5467, 5469, 5477, 5485, 5490, 5501, 5506, 5515, 5518, 5520, 5525, 5531, 5540, 5546, 5551, 5553, 5562, 5576, 5577, 5584, 5588, 5594, 5599, 5606, 5607, 5618, 5623, 5625, 5630, 5638, 5640, 5646, 5650, 5654, 5658, 5666, 5671, 5675, 5676, 5682, 5686, 5692, 5700, 5704, 5707, 5717, 5724, 5735, 5737, 5741, 5743, 5748, 5754, 5759, 5760, 5771, 5772, 5776, 5784, 5790, 5795, 5797, 5801, 5807, 5809, 5815, 5823, 5831, 5836, 5838, 5842, 5846, 5852, 5859, 5863, 5871, 5876, 5878, 5888, 5893, 5895, 5909, 5910, 5913, 5924, 5935, 5937, 5941, 5943, 5948, 5956, 5963, 5972, 5980, 5982, 5986, 5988, 5993, 6002, 6006, 6010, 6012, 6017, 6023, 6028, 6030, 6035, 6041, 6043, 6045, 6051, 6052, 6054, 6058, 6065, 6073, 6075, 6083, 6089, 6097, 6099, 6104, 6109, 6112, 6117, 6125, 6132, 6140, 6142, 6147, 6155, 6160, 6164, 6168, 6172, 6176, 6188, 6189, 6194, 6201, 6205, 6210, 6211, 6227, 6233, 6235, 6240, 6248, 6252, 6256, 6265, 6277, 6285, 6290, 6296, 6298, 6300, 6310, 6315, 6321, 6327, 6335, 6338, 6343, 6346, 6351, 6358, 6366, 6371, 6377, 6388, 6391, 6396, 6401, 6406, 6409, 6414, 6422, 6430, 6432, 6434, 6435, 6440, 6445, 6450, 6460, 6464, 6465, 6471, 6481, 6485, 6491, 6493, 6501, 6506, 6510, 6518, 6523, 6532, 6534, 6542, 6548, 6549, 6554, 6556)