The Journal of Open Source Software

sboxgates: A program for finding low gate count implementations of S-boxes

Marcus Dansarie[1, 2]

1 Swedish Defence University **2** University of Skövde, Sweden

DOI: 10.21105/joss.02946

Software

- Review
- Repository
- Archive

Editor: Viviane Pons

Reviewers:

- @gradvohl
- @HeshamAlsaadi

Submitted: 28 December 2020
Published: 16 June 2021

Summary

S-boxes are often the only nonlinear components in modern block ciphers. They are commonly selected to comply with very specific criteria in order to make a cipher secure against, for example, linear and differential attacks. An $M \times N$ S-box can be thought of as a lookup table that relates an M-bit input value to an N-bit output value, or as a set of N boolean functions of M variables (Schneier, 1996).

Although cipher specifications generally describe S-boxes using their lookup tables, they can also be described as boolean functions or logic gate circuits. `sboxgates`, which is presented here, finds equivalent logic gate circuits for S-boxes, given their lookup table specification. Generated circuits are output in a human-readable XML format. The software can convert the output files into C or CUDA (a parallel computing platform for Nvidia GPUs) source code. The generated circuits can also be converted to the DOT graph description language for visualization with Graphviz (Ellson et al., 2002).

Statement of need

Knowledge of a low gate count logic gate representation of an S-box can be of interest both when assessing the security of a cipher through cryptanalysis and when implementing it in hardware or software. When the design specification for an S-box is known, analytic approaches can sometimes be used to construct such a representation. The most notable case of this is the AES cipher where a very efficient representation of the S-box has been constructed in this manner (Canright, 2005). However, this is not possible in many cases, such as when the design specification is unknown or if the S-box is a randomly generated permutation.

While finding a large, inefficient, logic circuit representation is trivial, finding the representation with the fewest possible gates is an NP-complete problem (Knuth, 2015). The best known way to find a low gate count logic circuit representation for an S-box given its lookup table is to use Kwan's algorithm, which performs a heuristic search. Although not optimal, it has been shown to produce significantly better results than previous approaches (Kwan, 2000).

`sboxgates` implements Kwan's algorithm and supports generation of logic circuits for S-boxes with up to 8 input bits using any subset of the 16 possible two-input boolean functions. Additionally, the program can generate circuits that include three-bit lookup tables (LUTs). The LUT search function is parallelized using MPI (Walker & Dongarra, 1996).

The generated logic circuit representation of an S-box can be directly used in applications such as: creating bitslice implementations in software for CPUs and GPUs, creating small chip area or low gate count S-boxes for application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs), and compact satisfiability (SAT) problem generation.

A bitslice software implementation was first described by Biham (1997). It implements a cipher in a way that mimics a parallel hardware implementation. The number of parallel operations is equivalent to the platform register size, which can be up to 512 bits on modern machines. Some operations, such as bit permutations, have effectively no cost in bitslice implementations. For these reasons, bitslice implementations are generally many times faster than conventional implementations. The primary contributor to their time complexity is the size and efficiency of the S-box logic circuit (Biham, 1997). Modern Nvidia GPUs have an instruction (LOP3.LUT) that evaluates three-bit lookup tables. This makes them a very attractive target for bitslice cipher implementations that use LUTs.

In addition to bitslice implementations in software, which attempt to mimic hardware implementations, designs of actual hardware such as application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs) can also be made more efficient by small equivalent logic gate circuits for S-boxes.

In algebraic cryptanalysis, one attack method is to model a cipher along with its inputs and outputs as a SAT problem. This can be used to find, for example, weak keys in block ciphers or preimages in hash functions (Lafitte et al., 2014). SAT problems are typically expressed in conjunctive normal form (CNF) and logic circuits can quickly be converted into CNF using the Tseytin transform (Knuth, 2015). Thus, an efficient logic gate representation of an S-box can be transformed into an efficient CNF representation. CNF representations can in turn be transformed into a system of equations in GF(2) (Lafitte et al., 2014).

The only known software with similar functionality to `sboxgates` is `SBOXDiscovery` which is restricted to generating logic circuit representations of the DES S-boxes ("SBOXDiscovery," 2015). The software has been abandoned by its original author. Many of the optimizations of Kwan's algorithm made in `SBOXDiscovery` have been included in `sboxgates`.

References

Biham, E. (1997). A fast new DES implementation in software. In E. Biham (Ed.), *Fast Software Encryption. FSE 1997. Lecture Notes in Computer Science* (Vol. 1267, pp. 260–272). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0052352

Canright, D. (2005). A Very Compact S-Box for AES. In J. R. Rao & B. Sunar (Eds.), *Cryptographic Hardware and Embedded Systems. CHES 2005. Lecture Notes in Computer Science* (Vol. 3659, pp. 441–455). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11545262_32

Ellson, J., Gansner, E., Koutsofios, L., North, S. C., & Woodhull, G. (2002). Graphviz — Open Source Graph Drawing Tools. In P. Mutzel, M. Jünger, & S. Leipert (Eds.), *Graph Drawing. GD 2001. Lecture Notes in Computer Science* (Vol. 2265, pp. 483–484). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45848-4_57

Knuth, D. E. (2015). *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*. Pearson Education Inc., Boston, MA.

Kwan, M. (2000). *Reducing the Gate Count of Bitslice DES*. Cryptology ePrint Archive, Report 2000/051. https://eprint.iacr.org/2000/051

Lafitte, F., Nakahara, J., & Van Heule, D. (2014). Applications of SAT Solvers in Cryptanalysis: Finding Weak Keys and Preimages. *Journal on Satisfiability, Boolean Modeling and Computation*, *9*(1), 1–25. https://doi.org/10.3233/SAT190099

SBOXDiscovery. (2015). In *GitHub repository*. https://github.com/tripcode/SBOXDiscovery; GitHub.

Schneier, B. (1996). *Applied Cryptography : Protocols, Algorithms, and Source Code in C* (2. ed.). John Wiley & Sons, Inc.

Walker, D. W., & Dongarra, J. J. (1996). MPI: A Standard Message Passing Interface. *Supercomputer*, *12*(1), 56–68.