BoolSPLG: A library with parallel algorithms for Boolean functions and S-boxes for GPU

Dusan Bikov[1] **& Iliya Bouyukliev**[2]

[1]Faculty of Computer Science, Goce Delchev University, Shtip, Macedonia

[2]Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria

[1]dusan.bikov@ugd.edu.mk [2]iliyab@math.bas.bg

Abstract

In this work, we present a software package with parallel functions for computing some of the most important cryptographic properties of Boolean and Vector Boolean Function. These functions can be used for development of effective research algorithms. Moreover we present a console application building with some of these functions named BSbox-tools.

This library is implemented using CUDA parallel programming model for recent NVIDIA GPU architectures.

Introduction

The main objects which we consider are Boolean and Vector Boolean Function (or S-boxes) with good cryptographic properties. Construction of such kind of objects is very important but in most of the cases computationally expensive problem. Cryptographic properties which we investigate are nonlinearity, algebraic degree, autocorrelation, differential uniformity. Computing of this parameters is connected to Fourier-related transformation like Walsh-Hadamard, Reed-Muller (Möbius) transforms and Inverse Walsh-Hadamard [4]. These transformation (butterfly) algorithms are very effective and is suitable for parallel implementation.

For our research we develop package with CUDA C/C++ functions. All this function can be use for developing of algorithms (projects) from anyone why knows programming language C/C++, but is not so familiar with CUDA C. We develop console application BSbox-tools. This application is example for using of the function from the library. With help of this library we can investigate big invertible S-boxes (20x20) that are obtain from quasi-cyclic code [3].

There are many CPU libraries and mathematical software for computing cryptographic properties of Boolean Function and S-boxes, as examples we point out Sage, Matlab, VBF Library or SET. This software are good for purpose of education and basic calculations. For now, we do not know specialized parallel software for computing cryptographic properties of Boolean Function and S-boxes.

Main definitions

A Boolean function f of n variables is a mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$, where $\mathbb{F}_2 = \{0, 1\}$ is a field with 2 elements. Two natural representations of f of n variables are its Truth Table (TT(f)) and its Algebraic Normal Form $(ANF(f))$ [1]. The degree of $ANF(f)$ is the algebraic degree of the Boolean function denoted by $\deg(f)$.

An Walsh coefficient is defined by (Walsh Transform): $f^W(a) = \sum_{x\in\mathbb{F}_2^n}(-1)^{f(x)\oplus f_a(x)}$, where $a = (a_1, \ldots, a_n) \in \mathbb{F}_2^n$, $f_a(x) = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n$ and $f(x) \oplus \langle a, x\rangle = f(x_1, x_2, \ldots, x_n) \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n = f(x) \oplus f_a(x)$[1].

Linearity and nonlinearity of f: $Lin(f) = \max\{|f^W(a)| \mid a \in \mathbb{F}_2^n\}$, $nl(f) = 2^{n-1} - \frac{1}{2}Lin(f)$[1].

Autocorrelation of f: $AC(f) = \max\{|r_f(w)| \mid w \in \mathbb{F}_2^n\}$ and autocorrelation coefficient is defined $r_f(w) = \sum_{x\in\mathbb{F}_2^n}(-1)^{f(x)\oplus f(x\oplus w)}$, where $w \in \mathbb{F}_2^n$.

A vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ (also called (n, m) S-box or shortly S-box) can be represented by the vector $(f_1, f_2, \ldots, f_m)$, where f_i are Boolean functions in n variables, $i = 1, 2, \ldots, m$[2]. The functions f_i are called the coordinate functions of the S-boxes.

In order to study the cryptographic properties of S related to the linearity, algebraic degree and autocorrelation, we need to consider all non-zero linear combinations of the coordinate functions, denoted by $S_b = b \cdot S = b_1f_1 \oplus \cdots \oplus b_mf_m$, where $b = (b_1, \ldots, b_m) \in \mathbb{F}_2^m$.

The linearity and nonlinearity of S are defined as

$$Lin(S) = \max_{b\in\mathbb{F}_2^m\setminus\{0\}} Lin(b \cdot S), \quad nl(S) = \min_{b\in\mathbb{F}_2^m\setminus\{0\}} nl(b \cdot S).$$

Algebraic degree of S: $deg(S) = \min\{\deg(b \cdot S), b \in \mathbb{F}_2^m\setminus\{0\}\}$[2].

Autocorrelation of S: $AC(S) = \max_{b\in\mathbb{F}_2^m\setminus\{0\}} AC(b \cdot S)$.

Differential uniformity of S $(n \geq m)$: $\delta = \max_{\alpha\in\mathbb{F}_2^n\setminus\{0\},\beta\in\mathbb{F}_2^m} |\{x \in \mathbb{F}_2^n | S(x) \oplus S(x \oplus \alpha) = \beta\}|$.

Parallel implementation strategies

For fast transformation of given type, we combine and use different parallel programming strategies and techniques. Depending from the function we imply different level of parallelism. Thread can calculate value for fix number of (one, two or four depend on the radix) cell (coordinate). In Möbius transform we use bitwise presentation and computation. We prefer to use fast memory (register, local or shared) for function computing operation and proper data organization. Memory pattern is use for reordering and manipulation with data. Using of instructions (warp shuffles) are necessary for efficient computation.

Boolean S-box Library for GPUs (BoolSLG)

BoolSPLG (Boolean S-box Properties Library for GPUs) is a CUDA C/C++ library consisting procedures for analysis and compute cryptographic properties of Boolean and Vector Boolean function.

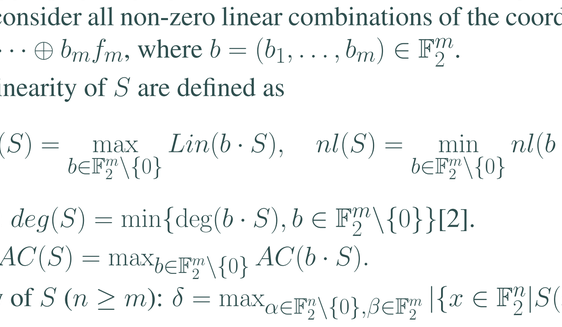

Figure 1: Building blocks structure of BoolSLG

The library is based on a methodology that uses a series of building functions that allow construction of algorithms (procedures) with little effort. Building functions of the procedures are organized in several layers.

www.moi.math.bas.bg/moiuser/~data/Results/Crypto/BoolSPL.html

This library will evolve by improving the algorithms and implementing more algorithms/procedures for computing others cryptographic properties.

BSbox-tools

BSbox-tools is a developed console interface program for the presentation, determination and calculation of the most important cryptographic properties of Boolean and Vector Boolean function. For developing of this program, we use BoolSPLG library. Application can use for research, education, comparatione between CPU and GPU implementation or only checking the GPU characteristic.

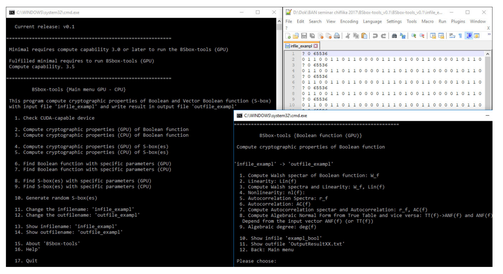

Figure 2: BSbox-tools menu and input file

This application read input data from the file and write output data (result) in different file. Input and output data have suitable format. It is not necessary programming skills to be able to use this application. Function from the application have GPU and CPU implementation. Only CPU functions will be run if is not available/suitable GPU.

Software is under development.

Experimental preliminary results

The test platforms that were used in our experiments are described in Table 1. Here is given preliminary result from experiments made on test platform for implement Fast Walsh transform and Fast Reed-Muller transform (Boolean function). For the purposes of comparison, we implement sequentially algorithms in programming language C++ using development environment MS VISUAL STUDIO 2010, denote by CPU. All CPU examples are executed on Platform 1 (INTEL I3-3110M, in Active solution configuration - Release, and Active solution platform - WIN32.

	Platform 1	Platform 2
CPU	Intel i3-3110M	Intel Xeon E5-2640
Memory	4 GB DDR3 1333 MHz	48GB DDR3 1333 MHz
OS	Win7 x64 SP1	Win7 x64 SP1
Compiler	MSVC 2010 SP1	MSVC 2012
GPU	GeForce GT 740M	GeForce GTX TITAN
Driver	v347.62, SDK 8.0	v347.62, SDK 8.0

Table 1: Description of the test platforms

size	P1 (ms)	P2 (ms)	S_p:P1 vs P2	CPU (ms)	S_p:CPU vs P2
2^{14}	0,036	0,021	1.71x	0,665	31.66x
2^{15}	0,096	0,027	3.55x	1,148	42.51x
2^{16}	0,204	0,048	4.25x	3,116	64.91x
2^{17}	0,537	0,084	6.40x	6,87	81.78x
2^{18}	1.118	0,142	7.87x	14,81	104.30x

Table 2: Fast Walsh Transform (Boolean function): Platform 1 vs. Platform 2 and CPU

#comp. words	P1 (ms)	P2 (ms)	S_p: P1 vs P2	CPU (ms)	S_p: CPU vs P2
$2^{18}/64$	0.014	0.012	1.16x	0.116	9.66x
$2^{20}/64$	0.041	0.023	1.78x	0.505	22x
$2^{22}/64$	0.253	0.040	5.87x	2.253	56.32x
$2^{24}/64$	1.329	0.141	9.42x	9.657	68.49x
$2^{26}/64$	6.689	0.549	12.18x	48.706	90.2x

Table 3: Fast Reed-Muller Transform (Boolean function): Algorithm 1: Platform 1 vs. Platform 2 and CPU

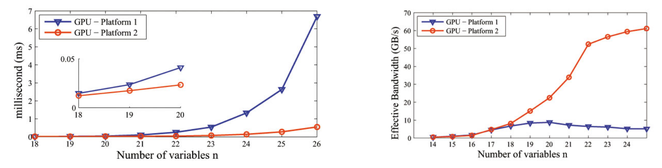

Figure 3: Fast Reed-Muller Transform (Boolean function): Platform 1 vs. Platform 2

Recently we start testing our package (butterfly algorithms) on Titan X Pascal and we active 3-4x speed up (depending of the input size) compering with Platform 2 GPU (Maxwell architecture).

References

[1] C. Carlet. Boolean functions for cryptography and error correcting codes. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 257–397, 2010.

[2] C. Carlet. Vectorial boolean functions for cryptography. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 2010.

[3] I. Bouyukliev D. Bikov and S. Bouyuklieva. S-boxes from binary quasi-cyclic codes. *Electronic Notes in Discrete Mathematics*, 57:67–72, 2017.

[4] R. S. Karpovsky, M. G. Stankovic and J. T. Astola. *Spectral Logic and Its Applications for the Design of Digital Devices*. Wiley Inc., 2008.

Acknowledgement

The Titan X Pascal used for this research was donated by the NVIDIA Corporation.