Biting Boolean Functions

Max Bileschi and Dan Padgett

Mathematics Department, University at Buffalo

University at Buffalo
The State University of New York

University Honors College

Mathematical Background

- A *Boolean function* in n variables is a mapping $f : \{0,1\}^n \to \{0,1\}$.
- Its *weight* is $\Sigma_{\mathbf{x}\in\{0,1\}^n} f(\mathbf{x})$.
- Its *truth table* is the 2^n-tuple $(f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_{2^n}))$, where $\mathbf{x}_i$ are the elements of $\{0,1\}^n$ in increasing order.
- A Boolean function f is *rotation symmetric* (RotS) if $f(x_1, x_2, \ldots, x_n) = f(x_2, \ldots, x_n, x_1)$ for every possible input $x_i \in \{0,1\}$. Intuitively, f is the same even if its inputs are rotated.
- A RotS function f is *cubic monomial rotation symmetric* if it is generated by a single cubic monomial; that is, $f(\mathbf{x}) = \sum_{i=0}^{n-1} x_{1+i}x_{r+i}x_{s+i}$. In this case, we abbreviate f as $(1, r, s)_n$.

Cryptographic Properties

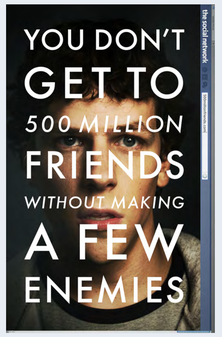
Efficiently computing the weight of rotation symmetric Boolean functions is important in determining their cryptographic properties. We present a method of doing this recursively, by first constructing the truth table for a function $(1, r, s)_n$ in terms of its truth table in one fewer variables; that is, $(1, r, s)_{n-1}$. We then use linear algebra to produce a homogeneous recurrence for the function's weight, allowing rapid computation of weights to an arbitrary number of variables.

Truth Table

The truth table of low values of n must be computed brute force, but once these are computed, we can use some techniques we developed to lessen the computation needed for higher values of n. The idea is to compute the truth table in n variables based on the truth table in $n-1$ variables. For example, for computing the truth table of the function $(1,2,3)_5$, we compute the truth table of the function $(1,2,3)_4$, and split ("bite") it up into green, red and blue parts.

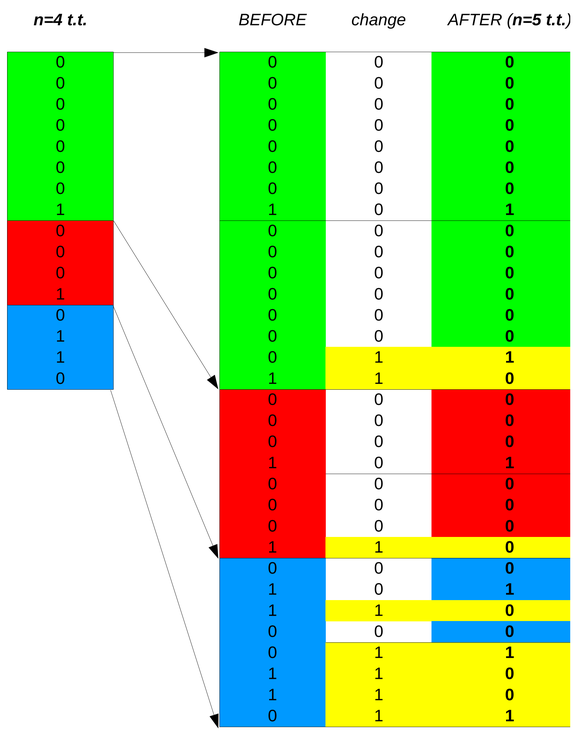
Overview of the Problem

Suppose you want to download a copy of *The Social Network* starring Jesse Eisenberg. However, an adversary wishes to prevent you from downloading the film by sending you false data in place of the true movie. You can check the data as you receive it using a hash function. In order to download the movie as fast as possible, you wish for your hash function to perform quickly. One way to do this is by using a rotation symmetric Boolean function to compute the hash. Therefore, it is advantageous to study the cryptographic properties of such functions.



Truth Table, con't

It is necessary to break (or "bite") up the truth table into green, red, and blue parts due to differences in the terms of the functions $(1,2,3)_4$ and $(1,2,3)_5$. Since the truth table of $(1,2,3)_5$ is twice as long as $(1,2,3)_4$, we simply "double" each of the colors form the $n=4$ column. We then compute the differences in the terms of $(1,2,3)_4$ and $(1,2,3)_5$; these are shown in the column labeled *bitstring*. Finally, we apply the changes in *bitstring* to the column labeled *before*, and we are left with the truth table of $(1,2,3)_5$.



Finding a Weight Recurrence

We can rewrite a split of the truth table in n variables with operations applied as a combination of splits in $n-1$ variables:

$$\begin{array}{rl} m_3^{(n)}: & \mathbf{0\ 0\ 1\ 1\ 0\ 0\ 1\ 1} \\ f^n: & 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array}$$

$$\begin{array}{rl} m_2^{(n-1)}: & \mathbf{0\ 0\ 1\ 1} \\ r_1(f^{n-1}): & 0\ 0\ 1\ 1 \end{array} \qquad \begin{array}{rl} m_2^{(n-1)}: & \mathbf{0\ 0\ 1\ 1} \\ r_2(f^{n-1}): & 0\ 1\ 1\ 0 \end{array}$$

where the r_1 and r_2 functions generate (fixed) additional operations in $n-1$ variables to apply. Since there are only finitely many operations, we can arrange their expansion including operations given by r_1 and r_2 in a table (given a fixed split):

Original		Left	Right
$\emptyset$	$\Rightarrow$	$\emptyset$	m_4
m_2	$\Rightarrow$	$\emptyset$	m_1, m_4
m_3	$\Rightarrow$	m_2	m_2, m_4
m_4	$\Rightarrow$	$\emptyset$	m_3, m_4

$\Downarrow$

	m_2	m_2	m_2	m_2	$\boldsymbol{m_2}$	$\boldsymbol{m_2}$	$\boldsymbol{m_2}$	$\boldsymbol{m_2}$	
	m_3	m_3	$\boldsymbol{m_3}$	$\boldsymbol{m_3}$	m_3	m_3	$\boldsymbol{m_3}$	$\boldsymbol{m_3}$	2^n
	m_4	$\boldsymbol{m_4}$	m_4	$\boldsymbol{m_4}$	m_4	$\boldsymbol{m_4}$	m_4	$\boldsymbol{m_4}$	
$m_2\ m_3\ m_4$	**1**	0	**1**	0	**1**	0	**1**	0	0
$m_2\ m_3\ \boldsymbol{m_4}$	**1**	0	0	0	**-1**	0	0	0	0
$m_2\ \boldsymbol{m_3}\ m_4$	0	**1**	0	**1**	0	**1**	0	**1**	0
$m_2\ \boldsymbol{m_3}\ \boldsymbol{m_4}$	0	**1**	0	0	0	**-1**	0	0	0
$\boldsymbol{m_2}\ m_3\ m_4$	0	0	0	0	0	0	0	0	0
$\boldsymbol{m_2}\ m_3\ \boldsymbol{m_4}$	0	0	**1**	0	0	0	**-1**	0	0
$\boldsymbol{m_2}\ \boldsymbol{m_3}\ m_4$	0	0	0	0	0	0	0	0	0
$\boldsymbol{m_2}\ \boldsymbol{m_3}\ \boldsymbol{m_4}$	0	0	0	**1**	0	0	0	**-1**	0
$\boldsymbol{2^{n-1}}$	0	0	0	0	**1**	**1**	**1**	**1**	**2**

Reduction to a Single Matrix

From these matrices, we can take the minimal polynomial for each split's matrix and find a recurrence by taking the least common multiple. However, we can actually reduce this to finding the minimal polynomial of just the first matrix, since we can show that all of them are *similar*:

Theorem (Bileschi, Padgett)

For each matrix A_i, we can find a matrix P such that $A_1 = PA_iP^{-1}$. Hence, to find a recurrence, it suffices to take the minimal polynomial of the matrix for the first bite alone.

Conclusion

While computing a Boolean function's weight is usually an exponential computation, the recurrence gives a method of polynomial-time computation in terms of n. For future work, the method easily generalizes to any rotation symmetric Boolean function. The polynomials generated also have a deeper connection with the weight than we are able to explain: the weight of a function in n variables is equal to the sum of the roots of the polynomial raised to the nth power, for every n.

References

[1] T. W. Cusick and P. Stănică, Fast evaluation, weights and nonlinearity of rotation symmetric functions, *D*iscrete Mathematics 258 (2002), 289-301.

[2] J. Pieprzyk and C. X. Qu, Fast hashing and rotation-symmetric functions, *J*ournal of Universal Computer Science 5 (1) (1999), 20-31.

Acknowledgements

This work was supported by NSF CSUMS grant 0802994. The presenters would also like to acknowledge the contributions of their faculty mentor, Dr. Thomas Cusick.