A C++ CLASS FOR ANALYSING VECTOR BOOLEAN FUNCTIONS FROM A CRYPTOGRAPHIC PERSPECTIVE

José Antonio Álvarez-Cubero and Pedro J. Zufiria
Depto. Matemática Aplicada a las Tecnologías de la Información, ETSIT, UPM, E-28040 Madrid, Spain

Keywords: C++ library, Walsh transform, Differential profile, Autocorrelation spectrum, Vector boolean function, Nonlinearity, Linearity distance, Balancedness, Resiliency, Propagation criterion.

Abstract: In this paper, a C++ class for analising Vector Boolean Functions from a cryptographic perspective is presented. This implementation uses the NTL library from Victor Shoup, replacing some of the general purpose modules of this library by some more specialized and better suited to cryptography, and adding new modules that complement the existing ones. With this class, we can obtain the classical representation of Vector Boolean Function such as its Truth Table and Algebraic Normal Form (ANF). It is possible to calculate mathematical structures such as the Walsh Spectrum, Linear Profile, Differential Profile and Autocorrelation Spectrum. Cryptographic criteria such as nonlinearity, linearity distance, order of correlation immunity, balancedness, algebraic degree and propagation criterion can be obtained with this class. It permits to find out some interesting cryptologic parameters such as linear structures, linear potential, differential potential and the maximum possible nonlinearity or linearity distance of a Vector Boolean Function with the same dimensions. Finally, operations such as to identify if two Vector Boolean Functions are equal, their sum, direct sum, composition, bricklayering, adding coordinate functions and obtaining the polynomial representation over $GF(2^n)$ of a Vector Boolean Function given the irreducible polynomial and its Truth Table are presented.

1 INTRODUCTION

Nowadays, Vector Boolean functions play an important role in various fields of human activity, such as Coding Theory (McWilliams and Sloane, 1977), Switching Theory (Davio et al., 1978) and Cryptography (Carlet, 2008a), (Carlet, 2008b). Conventional secret key cryptosystems can be expressed as a certain composition of Vector Boolean functions. Thus, in cipher design, it is essential to define criteria which measure the cryptographic strength of Boolean and Vector Boolean functions. Moreover, because of the size and complexity of modern ciphers, an automatic analysis program is very helpful in reducing the time which is necessary to spend on studying cryptographic properties of Vector Boolean Functions.

In this paper, a C++ class for analysing cryptographic properties of Vector Boolean Functions is presented. It is called VBF and is based on the well-known Number Theory Library NTL implemented by Victor Shoup (Shoup, 2009). NTL is a high-performance, portable C++ library providing data structures and algorithms for manipulating signed, arbitrary length integers, and for vectors, matrices, and polynomials over the integers and over finite fields. The decision to use this library is mainly based on four reasons:

1. It is free software, and may be used according to the terms of the GNU General Public License.
2. It provides high quality implementations of state-of-the-art algorithms for the Galois field of order 2.
3. It can be easily installed in a matter of minutes on just about any platform.
4. It provides a clean and consistent interface to a large variety of classes representing mathematical objects which are useful in cryptology.

The VBF class makes use of all the Boolean mathematical objects defined in NTL modules as starting point. However, it was necessary to introduce some new algorithms and cryptographic structures in order to achieve the results described in this paper. The main advantages of this approach are derived from the object oriented implementation and the use of effective algorithms: reusability, maintainability, extensibility and flexibility in the analysis of a broad range

Antonio Álvarez-Cubero J. and J. Zufiria P. (2010).
A C++ CLASS FOR ANALYSING VECTOR BOOLEAN FUNCTIONS FROM A CRYPTOGRAPHIC PERSPECTIVE.
In *Proceedings of the International Conference on Security and Cryptography*, pages 512-520
DOI: 10.5220/0002964505120520

of Vector Boolean Functions employed in symmetric ciphers. In our opinion, there is still a lack of stable and efficient C++ algorithms in cryptographic libraries and this implementation can be very useful tool both for the designer and the cryptanalist of symmetric ciphers. At the present time, either the libraries are commercial and restricted to some cryptographic properties of Boolean functions (such as (Bibliowicz et al., 2003) or (Gammel, 2006)) or they do not benefit from the new paradigms of object orientation and generic programming (Pommerening, 2001). We have performed a full analysis of a 14×14 S-box with the VBF class in less than one second with a Core2 Duo 2.4GHz, 4GB RAM, 2x250GB Debian linux platform.

The paper is organized as follows: Sections 2 is devoted to the presentation of the main Vector Boolean Functions concepts. In section 3, we describe the types of Vector Boolean Functions representations that this class can deal with. In section 4, cryptographic relevant matrices, cryptographic criteria and other useful information for cryptanalysis that this implementation can calculate for an individual Vector Boolean Function are described. In section 5, we enumerate the operations over Vector Boolean Functions that are supported. In section 6, we illustrate how the VBF class can be used with examples. Finally, concluding remarks are summarized in Section 7.

2 PRELIMINARIES

Let $< \mathrm{GF}(2), +, \cdot >$ be the finite field of order 2, where $\mathrm{GF}(2) = \mathbb{Z}_2 = \{0,1\}$, '+' the 'integer addition modulo 2' and '$\cdot$' the 'integer multiplication modulo 2'. $\mathrm{V_n}$ is the vector space of n-tuples of elements from $\mathrm{GF}(2)$. The *direct sum* of $\mathbf{x} \in \mathrm{V_{n_1}}$ and $\mathbf{y} \in \mathrm{V_{n_2}}$ is defined as $\mathbf{x} \oplus \mathbf{y} = (x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}) \in \mathrm{V_{n_1+n_2}}$. The *inner product* of $\mathbf{x}, \mathbf{y} \in \mathrm{V_n}$ is denoted by $\mathbf{x} \cdot \mathbf{y}$, and The inner product of real vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$.

$f : \mathrm{V_n} \to \mathrm{GF}(2)$ is called a *Boolean function* and $\mathcal{F}_n$ is the set of all Boolean functions on $\mathrm{V_n}$. $\mathcal{L}_n$ is the set of all linear Boolean functions on $\mathrm{V_n}$: $\mathcal{L}_n = \{l_\mathbf{u} \; \forall \mathbf{u} \in \mathrm{V_n} \mid l_\mathbf{u}(\mathbf{x}) = \mathbf{u} \cdot \mathbf{x}\}$ and $\mathcal{A}_n$ is the set of all affine Boolean functions on $\mathrm{V_n}$.

The real-valued mapping $\chi_\mathbf{u}(\mathbf{x}) = (-1)^{\sum_{i=1}^{i=n} u_i x_i} = (-1)^{\mathbf{u} \cdot \mathbf{x}}$ for $\mathbf{x}, \mathbf{u} \in \mathrm{V_n}$ is called a *character*. The character form of $f \in \mathcal{F}_n$ is defined as $\chi_f(\mathbf{x}) = (-1)^{f(\mathbf{x})}$. The Truth Table of χ_f is called as the $(1,-1)$-*sequence vector* or *sequence vector* of f and is denoted by $\xi_f \in \mathbb{R}^{2^n}$.

Let a Boolean function $f \in \mathcal{F}_n$, the *Walsh Transform* of f at $\mathbf{u} \in \mathrm{V_n}$ is the n-dimensional Discrete Fourier Transform and can be calculated as follows:

$$\hat{\chi}_f(\mathbf{u}) = \langle \xi_f, \xi_{l_\mathbf{u}} \rangle = \sum_{\mathbf{x} \in \mathrm{V_n}} (-1)^{f(\mathbf{x}) + \mathbf{u}\mathbf{x}} \tag{1}$$

The *autocorrelation* of $f \in \mathcal{F}_n$ with respect to the shift $\mathbf{u} \in \mathrm{V_n}$ is the cross-correlation of f with itself, denoted by $\mathcal{R}_f(\mathbf{u}) : \mathrm{V_n} \to \mathbb{R}$ and defined by:

$$\begin{aligned} \mathcal{R}_f(\mathbf{u}) &= \tfrac{1}{2^n} \textstyle\sum_{\mathbf{x} \in \mathrm{V_n}} \chi_f(\mathbf{x}) \chi_f(\mathbf{x} + \mathbf{u}) \\ &= \tfrac{1}{2^n} \textstyle\sum_{\mathbf{x} \in \mathrm{V_n}} (-1)^{f(\mathbf{x}) + f(\mathbf{u} + \mathbf{x})} \end{aligned} \tag{2}$$

$F : \mathrm{V_n} \to \mathrm{V_m}$, $F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ is called a *Vector Boolean function* and $\mathcal{F}_{n,m}$ is the set of all Vector Boolean functions $F : \mathrm{V_n} \to \mathrm{V_m}$. Each $f_i : \mathrm{V_n} \to \mathrm{GF}(2)$ $\forall i \in \{1, \ldots, m\}$ is a coordinate function of F. The *indicator function* of $F \in \mathcal{F}_{n,m}$, denoted by $\theta_F : \mathrm{V_n} \times \mathrm{V_m} \to \mathbb{R}$, is defined in (Chabaud and Vaudenay, 1994) as $\theta_F(\mathbf{x}, \mathbf{y}) = 1$ if $\mathbf{y} = F(\mathbf{x})$ and $\theta_F(\mathbf{x}, \mathbf{y}) = 0$ if $\mathbf{y} \neq F(\mathbf{x})$. The character form of $(\mathbf{u}, \mathbf{v}) \in \mathrm{V_n} \times \mathrm{V_m}$ can be defined as follows: $\chi_{(\mathbf{u},\mathbf{v})}(\mathbf{x}, \mathbf{y}) = (-1)^{\mathbf{u} \cdot \mathbf{x} + \mathbf{v} \cdot \mathbf{y}}$.

Let the Vector Boolean function $F \in \mathcal{F}_{n,m}$, its *Walsh Transform* is the two-dimensional Walsh Transform defined by:

$$\begin{aligned} \hat{\theta}_F(\mathbf{u}, \mathbf{v}) &= \textstyle\sum_{\mathbf{x} \in \mathrm{V_n}} \sum_{\mathbf{y} \in \mathrm{V_m}} \theta_F(\mathbf{x}, \mathbf{y}) \chi_{(\mathbf{u},\mathbf{v})}(\mathbf{x}, \mathbf{y}) \\ &= \textstyle\sum_{\mathbf{x} \in \mathrm{V_n}} (-1)^{\mathbf{u}\mathbf{x} + \mathbf{v}F(\mathbf{x})} \end{aligned} \tag{3}$$

(Nyberg, 1994) The *autocorrelation* of $F \in \mathcal{F}_{n,m}$ with respect to the shift $(\mathbf{u}, \mathbf{v}) \in \mathrm{V_n} \times \mathrm{V_m}$ is the cross-correlation of F with itself, denoted by $\mathcal{R}_F(\mathbf{u}, \mathbf{v}) : \mathrm{V_n} \times \mathrm{V_m} \to R$, so that:

$$\begin{aligned} \mathcal{R}_F(\mathbf{u}, \mathbf{v}) &= \tfrac{1}{2^n} \textstyle\sum_{\mathbf{x} \in \mathrm{V_n}} \chi_{\mathbf{v}F}(\mathbf{x} + \mathbf{u}) \chi_{\mathbf{v}F}(\mathbf{x}) \\ &= \tfrac{1}{2^n} \textstyle\sum_{\mathbf{x} \in \mathrm{V_n}} (-1)^{\mathbf{v}F(\mathbf{x}+\mathbf{u}) + \mathbf{v}F(\mathbf{x})} \end{aligned} \tag{4}$$

Let $F \in \mathcal{F}_{n,m}$ and $\mathbf{u} \in \mathrm{V_n}$, then the *difference Vector Boolean function* of F in the direction of $\mathbf{u} \in \mathrm{V_n}$, denoted by $\Delta_\mathbf{u} F \in \mathcal{F}_{n,m}$ is defined as follows: $\Delta_\mathbf{u} F(\mathbf{x}) = F(\mathbf{x} + \mathbf{u}) + F(\mathbf{x})$, $\mathbf{x} \in \mathrm{V_n}$. If the following equality is satisfied: $\Delta_\mathbf{u} F(\mathbf{x}) = \mathbf{c}$, $\mathbf{c} \in \mathrm{V_n}$ $\forall \mathbf{x} \in \mathrm{V_n}$ then $\mathbf{u} \in \mathrm{V_n}$ is called a linear structure of F.

We define the simplifying notation for the maximum of the absolute values of a set of real numbers $\{a_\mathbf{uv}\}_\mathbf{u,v}$, characterized by vectors $\mathbf{u}$ and $\mathbf{v}$, as: $\max(a_\mathbf{uv}) = \max_{(\mathbf{u},\mathbf{v})} \{|a_\mathbf{uv}|\}$. Using the same simplifying notation, we define the $\overset{*}{\max}(\cdot)$ operator on a set of real numbers $\{a_\mathbf{uv}\}_\mathbf{u,v}$, as: $\overset{*}{\max}(a_\mathbf{uv}) = \max_{(\mathbf{u},\mathbf{v}) \neq (\mathbf{0},\mathbf{0})} \{|a_\mathbf{uv}|\}$. This notation will be used in some criteria definitions.

3 REPRESENTATIONS OF VECTOR BOOLEAN FUNCTIONS

Let a Vector Boolean Function $F \in \mathcal{F}_{n,m}$, the representations supported by the VBF class are described below together with their member functions:

1. The *Truth Table* of F, denoted by $\mathsf{T}_F \in \mathsf{M}_{2^n \times m}(\mathrm{GF}(2))$ and defined by:

$$\mathsf{T}_F = \begin{bmatrix} f_1(\alpha_0) & \dots & f_m(\alpha_0) \\ f_1(\alpha_1) & \dots & f_m(\alpha_1) \\ \dots & \dots & \dots \\ f_1(\alpha_{2^n-1}) & \dots & f_m(\alpha_{2^n-1}) \end{bmatrix} \quad (5)$$

where $f_i \;\; i \in \{1,\dots,m\}$ are its component functions and $\alpha_i = (x_1,\dots,x_n) \in \mathrm{V_n} \;\; i \in \{1,\dots,2^n-1\}$ is a vector whose decimal equivalent $dec(\alpha_i) = i = \sum_{j=1}^{n} x_j 2^{n-j}$ and we can list all the vectors of $\mathrm{V_n}$ so that $\alpha_0 < \alpha_1 < \dots < \alpha_{2^n-1}$.

```
void TT(NTL::mat_GF2& X, VBF& a)
inline NTL::mat_GF2 TT(VBF& a)
```

2. *m*-tuple of *polynomials in ANF*. Any $F \in \mathcal{F}_{n,m}$ can be uniquely represented by m multivariate polynomial over $\mathrm{GF}(2)$ where each variable has power at most one. This polynomial can be expressed as a sum of all distinct kth-order product terms $(0 < k \le n)$ of the variables:

$$\begin{aligned} &f(x_1,\dots,x_n) = a_0 + a_1x_1 + \dots + a_nx_n \\ &+a_{12}x_1x_2 + \dots + a_{n-1,n}x_{n-1}x_n \\ &+ \dots + a_{12\dots n}x_1x_2\dots x_n \\ &= \textstyle\sum_{\mathsf{I}\in\mathsf{P}(N)} a_{\mathsf{I}} \left(\prod_{i\in\mathsf{I}} x_i\right) = \sum_{\mathsf{I}\in\mathsf{P}(N)} a_{\mathsf{I}} x^{\mathsf{I}}, \;\; a_{\mathsf{I}} \in \mathrm{GF}(2) \end{aligned} \quad (6)$$

where $\mathsf{P}(N)$ denotes the power set of $N = \{1,\dots,n\}$. This representation of f is called the *algebraic normal form (ANF)* of f.

```
void Pol(NTL_SNS ostream& s, VBF& a)
vec_pol getpol()
```

3. The *ANF table* of F, denoted by $\mathsf{ANF}_F \in \mathsf{M}_{2^n \times m}(\mathrm{GF}(2))$ and represents the 2^n coefficients of the polynomials of the m coordinate functions in ANF.

```
void ANF(NTL::mat_GF2& X, VBF& a)
inline NTL::mat_GF2 ANF(VBF& a)
```

If F is Boolean permutation, that is, it is bijective and has the same number of input bits as output bits $(n = m)$, then it can be defined as an array:

$$F = \begin{bmatrix} F(1) & \dots & F(n) \end{bmatrix} \quad (7)$$

having $F(i)$ as the image of the bit i for F.

```
void putper(const NTL::vec_ZZ& a)
NTL::vec_ZZ getper() const
```

If F is an affine Vector Boolean Function with $n \neq m$ (such as the Expansion and Compression DES permutations (NBS, 1977)), then it can be defined as an array with m elements which are the output bits.

```
void putlin(const NTL::vec_ZZ& a)
NTL::mat_GF2 getlin() const
```

The VBF class also supports the definition of F as given in (NBS, 1977) for the DES S-boxes.

```
void putsbox(const NTL::mat_ZZ& a)
NTL::mat_ZZ getsbox()
```

4 CRYPTOGRAPHIC ANALYSIS OF A VECTOR BOOLEAN FUNCTION

This section describes the useful information that can be extracted with the VBF class from the cryptographic point of view. First of all, matrices with cryptographic interest are presented. Then, we enumerate the different cryptographic criteria that are supported and how they can be obtained from these matrices: nonlinearity, correlation immunity, balancedness, linearity distance, propagation criterion and algebraic degree. Finally, some useful information for cryptanalysis supported by the VBF class is described: the linear potential, the differential potential, the linear structures of the Vector Boolean Function and the maximum possible nonlinearity and linearity distance for a Vector Boolean Function with the same dimensions as the one analysed.

4.1 Matrices with Cryptographic Interest

Let $F \in \mathcal{F}_{n,m}$ be a Vector Boolean Function, VBF can calculate all the following matrices associated with it:

1. The *Characteristic Function* of F that can be represented by a matrix whose rows are indexed by $\mathbf{x} \in \mathrm{V_n}$ and whose columns are indexed by $\mathbf{y} \in \mathrm{V_m}$ in lexicographic order, denoted by $\mathsf{Img(F)} \in \mathsf{M}_{2^n \times 2^m}(\mathrm{GF}(2))$ and defined as follows:

$$\mathsf{Img(F)} = \begin{bmatrix} \theta_F(\alpha_0,\alpha_0) & \dots & \theta_F(\alpha_0,\alpha_{2^m-1}) \\ \theta_F(\alpha_1,\alpha_0) & \dots & \theta_F(\alpha_1,\alpha_{2^m-1}) \\ \dots & \dots & \dots \\ \theta_F(\alpha_{2^n-1},\alpha_0) & \dots & \theta_F(\alpha_{2^n-1},\alpha_{2^m-1}) \end{bmatrix} \quad (8)$$

where $\theta_F(\mathbf{x},\mathbf{y})$ is the value of the indicator function at $(\mathbf{x},\mathbf{y})$, defined as $\theta_F : \mathrm{V_n} \times \mathrm{V_m} \to \{0,1\}$:

$$\theta_F(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} = F(\mathbf{x}) \\ 0 & \text{if } \mathbf{y} \neq F(\mathbf{x}) \end{cases} \quad (9)$$

```
mat_ZZ charfunct(const mat_GF2& T, int n, int m)
```

This function has as arguments the Truth Table, the number of inputs and the number of outputs of a vector Boolean function and it calculates its Characteristic Function.

2. The *Walsh Spectrum* of F that can be represented by a matrix whose rows are characterized by $\mathbf{u} \in \mathrm{V_n}$ and whose columns are characterized by $\mathbf{v} \in \mathrm{V_m}$ in lexicographic order, denoted by $\mathsf{WS}(F) \in \mathsf{M}_{2^n \times 2^m}(\mathbb{R})$. It holds that $\hat{\theta}_F(\mathbf{u},\mathbf{v}) = \mathsf{WS}(F)(\mathbf{u},\mathbf{v})$.

```
void Walsh(NTL::mat_ZZ& X, VBF& a)
inline NTL::mat_ZZ Walsh(VBF& a)
```

3. The *Linear Profile* of F that can be represented by a matrix whose rows are characterized by $\mathbf{u} \in \mathrm{V_n}$ and whose columns are characterized by $\mathbf{v} \in \mathrm{V_m}$ in lexicographic order, denoted by $\mathsf{LP}(F) \in \mathsf{M}_{2^n \times 2^m}(\mathbb{R})$. It holds that $\mathsf{LP}(F)(\mathbf{u},\mathbf{v}) = \frac{1}{2^{n+m}} |\mathsf{WS}(F)(\mathbf{u},\mathbf{v})|^2$.

```
void LAT(NTL::mat_ZZ& X, VBF& a)
inline NTL::mat_ZZ LAT(VBF& a)
```

4. The *Differential Profile* that can be represented by a matrix whose rows are characterized by $\mathbf{u} \in \mathrm{V_n}$ and whose columns are characterized by $\mathbf{v} \in \mathrm{V_m}$ in lexicographic order, denoted by $\mathsf{DP}(F) \in \mathsf{M}_{2^n \times 2^m}(\mathbb{R})$. This matrix results from the application of the Walsh Transform to the Linear Profile.

```
void DAT(NTL::mat_ZZ& X, VBF& a)
inline NTL::mat_ZZ DAT(VBF& a)
```

5. The *Autocorrelation Spectrum* whose rows are indexed by $\mathbf{u} \in \mathrm{V_n}$ and whose columns are indexed by $\mathbf{v} \in \mathrm{V_m}$ in lexicographic order, denoted by $\mathsf{R}(F) \in \mathsf{M}_{2^n \times 2^m}(\mathbb{R})$. This matrix results from the application of the Walsh Transform to the Linear Profile. It holds that $\mathsf{R}(F)(\mathbf{u},\mathbf{v}) = \frac{1}{\mathsf{DP}(F)(\mathbf{0},\mathbf{0})}(\mathsf{DP}(F)(\mathbf{u},\mathbf{0}) - \mathsf{DP}(F)(\mathbf{u},\mathbf{v}))$.

```
void AC(NTL::mat_RR& X, VBF& a)
inline NTL::mat_RR AC(VBF& a)
```

Some functions have been implemented in the VBF class to compute one matrix from the knowledge of others such as:

```
mat_GF2 rev(const mat_GF2& X, int n, int m)
```

This function has as arguments the ANF Table, the number of inputs and the number of outputs of a vector Boolean function and it computes its Truth Table.

```
mat_GF2 truthtable(const mat_ZZ& C, int n, int m)
```

This function has as arguments the Characteristic Function, the number of inputs and the number of outputs of a vector Boolean function and it computes its Truth Table.

```
mat_ZZ invwt(const mat_ZZ& X, int n, int m)
```

This function has as arguments the Walsh Spectrum, the number of inputs and the number of outputs of a vector Boolean function and it computes its Characteristic Function. It corresponds with the inverse Walsh Transform.

4.2 Cryptographic Criteria

Let $F \in \mathcal{F}_{n,m}$ be a Vector Boolean Function, the following cryptographic criteria can be obtained by means of the VBF class:

1. *Nonlinearity* defined as the minimum among the nonlinearities of all nonzero linear combinations of the coordinate functions of F and can be obtained from the Walsh Spectrum the following way:

$$\begin{aligned} \mathcal{NL}(F) &= \min_{\mathbf{v} \neq \mathbf{0} \in \mathrm{V_m}} \mathcal{NL}(\mathbf{v} \cdot F) \\ &= 2^{n-1} - \tfrac{1}{2} \overset{*}{\max} (\mathsf{WS}(F)(\mathbf{u},\mathbf{v})) \end{aligned} \quad (10)$$

```
void nl(NTL::RR& x, VBF& a)
inline NTL::RR nl(VBF& a)
```

2. *Linearity distance* defined as the minimum among the linearity distances of all nonzero linear combinations of the coordinate functions of F and can be obtained from the Differential Profile the following way:

$$\mathcal{LD}(F) = \min_{\mathbf{v} \neq \mathbf{0} \in \mathrm{V_m}} \mathcal{LD}(\mathbf{v} \cdot F) \quad (11)$$

```
void ld(NTL::RR& x, VBF& a)
inline NTL::RR ld(VBF& a)
```

3. *Balancedness*, considering that $F \in \mathcal{F}_{n,m}$ is balanced (or to have balanced output) if each possible output m-tuple occurs with equal probability $\frac{1}{2^m}$, that is, its output is uniformly distributed in $\mathrm{V_m}$. This criterion can be obtained from the Walsh Spectrum the following way:

$$\hat{\theta}_F(\mathbf{0},\mathbf{v}) = 0, \ \forall \mathbf{v} \neq \mathbf{0} \in \mathrm{V_m} \quad (12)$$

```
void Bal(int& bal, VBF& a)
inline int Bal(VBF& a)
```

4. *Correlation Immunity*, so that $F \in \mathcal{F}_{n,m}$ is an (n,m,t)-CI function if and only if every nonzero linear combination $f(\mathbf{x}) = \sum_{i=1}^{m} v_i f_i(\mathbf{x})$ of coordinate functions of F is an $(n,1,t)$-CI function, where $\mathbf{x} \in \mathrm{V_n}$, $v_i \in \mathrm{GF}(2)$ $i = 1,\ldots,m$ and not all

zeroes. This criterion can be obtained from the Walsh Spectrum the following way:

$$\hat{\theta}_F(\mathbf{u},\mathbf{v}) = 0,\ \forall \mathbf{u} \in \mathrm{V_n},\ 1 \leq wt(\mathbf{u}) \leq t,\ \forall \mathbf{v} \neq \mathbf{0} \in \mathrm{V_m} \quad (13)$$

```
void CI(int& t, VBF& a)
inline int CI(VBF& a)
```

5. *Propagation criterion,* where $F \in \mathcal{F}_{n,m}$ satisfies the propagation criterion of degree l ($PC(l)$) if any nonzero linear combination of the component boolean functions satisfies the $PC(l)$. This criterion can be obtained from the Autocorrelation Spectrum the following way:

$$\mathcal{R}_F(\mathbf{u},\mathbf{v}) = 0,\ \forall \mathbf{u} \in \mathrm{V_n},\ 1 \leq wt(\mathbf{u}) \leq l,\ \forall \mathbf{v} \neq \mathbf{0} \in \mathrm{V_m} \quad (14)$$

```
void PC(int& k, VBF& a)
inline int PC(VBF& a)
```

6. *Algebraic degree* defined as the maximum among the algebraic degrees of all nonzero linear combinations of the coordinate functions of F (Nyberg, 1992), namely:

$$deg(F) = \min_g \{deg(g) \mid g = \sum_{j=1}^{m} v_j f_j,\ \mathbf{v} \neq \mathbf{0} \in \mathrm{V_m}\} \quad (15)$$

being the algebraic order or degree of a Boolean function, the order of the largest product term which exists in the *ANF*. This criterion is calculated by finding out the ANF table and then analysing the order of all the linear combinations of coordinate functions.

```
void deg(int& d, VBF& a)
inline int deg(VBF& a)
```

4.3 Useful Information in Cryptanalysis

Let $F \in \mathcal{F}_{n,m}$ be a Vector Boolean Function, the following information can be obtained by means of the VBF class:

1. The *Linear Potential* of F, defined as $lp(F) = \frac{1}{2^{2n}} \cdot \overset{*}{\max}(\mathrm{WS}(F)(\mathbf{u},\mathbf{v})^2)$ which is exploited as a measure of linearity in linear cryptanalysis, and satisfies (Chabaud and Vaudenay, 1994) $\frac{1}{2^n} \leq lp(F) \leq 1$ so that the lower bound holds if and only if F has maximum nonlinearity (F is bent) and the upper bound is reached when F is linear or affine.

```
void lp(NTL::RR& x, VBF& a)
inline NTL::RR lp(VBF& a)
```

2. The *Differential Potential* of F, defined as $dp(F) = \overset{*}{\max}(\mathrm{DP}(F)(\mathbf{u},\mathbf{v}))$ which is exploited as a measure of the robustness against differential cryptanalysis. It holds that $\frac{1}{2^m} \leq dp(F) \leq 1$ and the lower bound holds if and only if F is bent and the upper bound is reached when F is linear or affine. The differential uniformity of $F \in \mathcal{F}_{n,m}$ and its differential potential are related as follows: $dp(F) = \frac{1}{2^n} DU(F)$.

```
void dp(NTL::RR& x, VBF& a)
inline NTL::RR dp(VBF& a)
```

3. The *Linear structures* of F, defined as the vectors for which its associated row in the Differential Profile coincides with the vector zero.

```
NTL::mat_GF2 LS(VBF& a)
```

4. The *Maximum possible nonlinearity* for a Vector Boolean Function with the same dimensions as F (when n is even).

```
NTL::RR nlmax(VBF& a)
```

5. The *Maximum possible linearity distance* for a Vector Boolean Function with the same dimensions as F.

```
NTL::RR ldmax(VBF& a)
```

5 OPERATIONS OVER VECTOR BOOLEAN FUNCTIONS

In this section, the operations over Vector Boolean Functions that the VBF class supports are described. Some of them corresponds to secondary constructions, which build (n,m) variable Vector Boolean Functions from (n',m') variable ones (with $n' \leq n, m' \leq m$). The direct sum has been used to construct resilient and bent Boolean functions (Carlet, 2004). Adding coordinate functions and bricklayering are operations used to build modern ciphers such as CAST (Adams and Tavares, 1993), DES (NBS, 1977) and AES (Daemen and Rijmen, 2002). Finally, another operations supported are: identification if two Vector Boolean functions are equal, the sum of two Vector Boolean functions, the composition of two Vector Boolean Functions, derivation of polynomial representation of mappings from $GF(2^n)$ to $GF(2^n)$ by Lagrange interpolation. The definitions of all the supported operation are the following:

1. Let $n \geq 1, m \geq 1$, $F, G \in \mathcal{F}_{n,m}$. F and G are *equal* if their Truth Tables are the same.

```
long operator==(VBF& a, VBF& b)
long operator!=(VBF& a, VBF& b)
```

2. Let $n \geq 1, m \geq 1$, $F, G \in \mathcal{F}_{n,m}$. The *Sum* of F and G (denoted by $F+G$) is the Vector Boolean Function whose Truth Table results from the addition of the Truth Tables of F and G: $T_{F+G} = T_F + T_G$. It can be proved that Walsh Spectrum of the sum can be obtained by the convolution of the columns vectors of the respective Walsh Spectra.

```
void sum(VBF& X, VBF& A, VBF& B)
VBF operator+(VBF& A, VBF& B)
```

3. Let $n = n_1 + n_2, n_1, n_2 \geq 1, m \geq 1$, $F_1 \in \mathcal{F}_{n_1,m}$ and $F_2 \in \mathcal{F}_{n_2,m}$. The *Direct Sum* of F_1 and F_2 is the function:

$$(F_1 \oplus F_2) : \mathrm{V}_{n_1} \times \mathrm{V}_{n_2} \rightarrow \mathrm{V}_m$$
$$(\mathbf{x}, \mathbf{y}) \rightarrow (F_1 \oplus F_2)(\mathbf{x}, \mathbf{y}) = F_1(\mathbf{x}) + F_2(\mathbf{y}) \quad (16)$$

This is a generalization for Vector Boolean functions of the construction of Boolean functions first introduced in (Rothaus, 1976).

```
void directsum(VBF& X, VBF& A, VBF& B)
```

4. Let $n \geq 1, m = m_1 + m_2, m_1, m_2 \geq 1$ and $F \in \mathcal{F}_{n,m_1}$ and $G \in \mathcal{F}_{n,m_2}$. The result of *adding coordinate functions* of F and G is the function $(F,G) \in \mathcal{F}_{n,m_1+m_2}$ where $(F,G)(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_{m_1}(\mathbf{x}), g_1(\mathbf{x}), \ldots, g_{m_2}(\mathbf{x}))$. This is a generalization for Vector Boolean functions of the method used in the CAST algorithm and studied in (Nyberg, 1994) by adding more than one coordinate function at the same time.

```
void addimage(VBF& X, VBF& A, VBF& B)
```

5. Let $n = n_1 + n_2, n_1, n_2 \geq 1, m = m_1 + m_2, m_1, m_2 \geq 1$, $F \in \mathcal{F}_{n_1,m_1}$ and $G \in \mathcal{F}_{n_2,m_2}$. The *Bricklayer* of F and G is the function $F|G \in \mathcal{F}_{n,m}$ where $F|G(\mathbf{x}, \mathbf{y}) = (f_1(\mathbf{x}), \ldots, f_{m_1}(\mathbf{x}), g_1(\mathbf{y}), \ldots, g_{m_2}(\mathbf{y}))$. This construction corresponds to the bricklayer function (Daemen and Rijmen, 2002) as a parallel application of a number of Vector Boolean functions operating on smaller inputs.

```
void concat(VBF& X, VBF& A, VBF& B)
VBF operator|(VBF& A, VBF& B)
```

6. Let F be a mapping from $GF(2^n)$ to $GF(2^n)$, it can always represented by a polynomial function over $GF(2^n)$. A general way to derive this polynomial representation is given by Lagrange interpolation from the knowledge of the irreducible polynomial of degree n over $GF(2)$ associated with the field $GF(2^n)$ and the Truth Table of F.

```
void interpolate(GF2EX& f, VBF& a)
```

7. Let $F \in \mathcal{F}_{n,p}$, $G \in \mathcal{F}_{p,m}$, then the *Composition Function* is $G \circ F \in \mathcal{F}_{n,m}$.

```
void Comp(VBF& X, VBF& A, VBF& B)
VBF operator*(VBF& A, VBF& B)
```

6 EXAMPLES OF PROGRAM CODE

In this section, we present some examples of how the VBF class can be used. All of them are computed with a Core2 Duo 2.4GHz, 4GB RAM, 2x250GB Debian linux platform in less than one second.

6.1 Analysis of a Vector Boolean Function from its Truth Table

The example program below obtains cryptographic information about a Vector Boolean Function from knowing its Truth Table.

```
#include <iostream>
#include <fstream>
#include "VBF.h"

int main(int argc, char *argv[])
{
   using namespace VBFNS;

   VBF F;
   NTL::mat_GF2  mat_F;
   NTL::mat_GF2  A, T;
   NTL::mat_ZZ   W, LP, DP;
   NTL::mat_RR   Ac;
   int t, d, k;

   ifstream input(argv[1]);
   if(!input)
   {
      cerr << "Error opening the input file
      containing the TT" << argv[1] << endl;
      return 0;
   }
   input >> mat_F;
   F.puttt(mat_F);
   input.close();

   ofstream output(argv[2]);
   if(!output)
   {
      cerr << "Error opening the output file"
           << argv[2] << endl;
      return 0;
   }

   output << "Argument Dimension = " << F.n()
          << endl;
   output << "Argument space has " << F.spacen()
          << " elements."<< endl;
   output << "Image Dimension = " << F.m() << endl;
   output << "Image space has " << F.spacem()
          << " elements." << endl << endl;
   output << "1. Algebraic Normal Form Table:"
          << endl;
   A = ANF(F);
   output << A << endl;
```

```
   output << "2. Truth Table:" << endl;
   T = TT(F);
   output << T << endl;

   output << "3. Walsh Spectrum:" << endl;
   W = Walsh(F);
   output << W << endl;

   output << "4. Linear Profile:" << endl;
   LP = LAT(F);
   output << LP << endl;;

   output << "5. Differential Profile:" << endl;
   DP = DAT(F);
   output << DP << endl;

   output << "6. Linearity/nonlinearity measures:" << endl;
   output << "Linear potential: " << lp(F) << endl;
   output << "Differential potential: " << dp(F) << endl;
   output << "Nonlinearity: " << nl(F) << endl;
   output << "Maximum Nonlinearity: " << nlmax(F) << endl;
   output << "Linearity distance: " << ld(F) << endl;
   output << "Maximum Linearity distance: " << ldmax(F)
   << endl;

   output << "7. Correlation immunity:" << endl;
   t = CI(F);
   if (F.getbal())
   {
      output << "It is a (" << F.n() << "," << F.m()
             << "," << t << ")-resilient function"
 << endl;
   }  else
   {
      output << "It is a (" << F.n() << "," << F.m()
             << "," << t << ")-CI function" << endl;
   }

   output << "8. Algebraic degree:" << endl;
   d = deg(F);
   output << "The degree of the function is "
          << d << endl;

   output << "9. Propagation criterion:" << endl;
   k = PC(F);
   output << "The function is PC of degree "
          << k << endl;
   output << "10. The polynomial representation is: "
          << endl;
   Pol(output,F);
   output << "11. Linear structures: " << endl;
   A = LS(F);
   output << A << endl;
   output << "12. Autocorrelation Spectrum: " << endl;
   Ac = AC(F);
   output << Ac << endl;
   input.close();
   output.close();
   return 0;
}
```

If we have the Truth Table of the S-box used in Rijndael S_{RD} (Daemen and Rijmen, 2002) as the input, the output of the program is below. We only include the first rows and columns of the represented matrices, and the first terms of the polynomials:

```
Argument Dimension = 8
Argument space has 256 elements.
Image Dimension = 8
Image space has 256 elements.

1. Algebraic Normal Form Table:
[[0 1 1 0 0 0 1 1]
[0 0 0 1 1 1 1 1]
2. Truth Table:
[[0 1 1 0 0 0 1 1]
[0 1 1 1 1 1 0 0]
3. Walsh Spectrum:
[[256 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[0 24 0 24 28 12 4 -12 24 0 16 24 4 20 4 -12 16 -24 24
4. Linear Profile:
[[65536 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[0 576 0 576 784 144 16 144 576 0 256 576 16 400 16 144
5. Differential Profile:
[[16777216 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[0 131072 0 0 131072 0 131072 0 131072 131072 131072
6. Linearity/nonlinearity measures:
Linear potential: 0.015625
Differential potential: 0.015625
Nonlinearity: 112
Maximum Nonlinearity: 120
Linearity distance: 126
Maximum Linearity distance: 136
7. Correlation immunity:
It is a (8,8,0)-resilient function
8. Algebraic degree:
The degree of the function is 7
9. Propagation criterion:
The function is PC of degree 0
10. Polynom:
x6+x6x8+x6x7+x5x6x8+x5x6x7+x5x6x7x8+x4+x4x7x8+x4x6+
x4x6x7x8+1+x5+x5x7+x5x7x8+x5x6+x4x8+x4x7x8+x4x6x8+
x4x6x7x8+x4x5x7+1+x6x7x8+x5x8+x5x7x8+x5x6x7x8+x4+
x4x7x8+x4x6+x4x6x8+x4x6x7+x8+x7+x7x8+x6+x5+x5x6+
x5x6x8+x4x8+x4x7+x4x6x7+x4x5+x4x5x8+x8+x6x7+x5x8+
x5x7x8+x5x6+x5x6x8+x5x6x7+x5x6x7x8+x4+x4x7x8+x8+x7
+x6x8+x5x8+x5x7x8+x5x6+x5x6x8+x4x8+x4x7+x4x7x8+
x4x6x8+1+x8+x7x8+x6x8+x5+x5x8+x5x7+x5x7x8+x5x6+
x5x6x8+x5x6x7+x4x8+1+x8+x7x8+x6+x6x7+x5+x5x7+x5x6+
x5x6x7+x5x6x7x8+x4+x4x8+x4x7+
11. Linear structures:
[]
12. Autocorrelation Spectrum:
[[0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[0 -0.0078125 0 0 -0.0078125 0 -0.0078125 0 -0.0078125
```

6.2 Direct Sum of Two Vector Boolean Functions

The example program below obtains the polynomial representing of the Vector Boolean Function resulting from the direct sum of two others, knowing their Truth Tables.

```
#include <iostream>
#include <fstream>
#include "VBF.h"

int main(int argc, char *argv[])
{
   using namespace VBFNS;

   VBF F1, F2, F;
   NTL::mat_GF2  T1, T2;

   ifstream input1(argv[1]);
   if(!input1)
   {
      cerr << "Error opening the input file
      containing the TT of the 1st function: "
      << argv[1] << endl;
      return 0;
   }
   input1 >> T1;
   F1.puttt(T1);
   input1.close();

   ifstream input2(argv[2]);
   if(!input2)
   {
      cerr << "Error opening the input file
      containing the TT of the 2nd function: "
      << argv[2] << endl;
      return 0;
   }
   input2 >> T2;
   F2.puttt(T2);
   input2.close();

   ofstream output(argv[3]);
   if(!output)
   {
      cerr << "Error opening the output file "
           << argv[3] << endl;
      return 0;
   }

   directsum(F,F1,F2);
   output << "The polynomial representation
   of the direct sum is:" << endl;
   Pol(output,F);

   output.close();

   return 0;
}
```

6.3 Polynomial Representation of Rijndael S-box

The example program below obtains the polynomial representation over $GF(2^8)$ given the irreducible polynomial $x^8+x^4+x^3+x+1$ and the Truth Table of $S_{RD} \in \mathcal{F}_{8,8}$ by Lagrange interpolation.

```
#include <iostream>
#include <fstream>
#include "VBF.h"

int main(int argc, char *argv[])
{
   using namespace VBFNS;

   VBF S0;
   NTL::mat_GF2  mat_S0;
   GF2X  g;
   GF2EX  f;

   ifstream input1(argv[1]);
   if(!input1)
   {
      cerr << "Error opening the file
      containing the irreducible polynomial"
      << argv[1] << endl;
      return 0;
   }
   input1 >> g;
   S0.putirrpol(g);
   input1.close();

   ifstream input2(argv[2]);
   if(!input2)
   {
      cerr << "Error opening the file
      containing the TT" << argv[2] << endl;
      return 0;
   }
   input2 >> mat_S0;
   S0.puttt(mat_S0);
   input2.close();

   ofstream output(argv[3]);
   if(!output)
   {
      cerr << "Error opening the output file "
           << argv[3] << endl;
      return 0;
   }

   output << "The polynomial representation
    is: " << endl;
   interpolate(f,S0);
   print(output,f);

   input1.close();
   input2.close();
   output.close();
   return 0;
}
```

Taking as input the irreducible polynomial representation $\begin{bmatrix}1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1\end{bmatrix}$ and S_{RD} Truth Table, the output of the program is:

```
The polynomial representation is:
5x^254+9x^253+f9x^251+25x^247+f4x^239+1x^223+
b5x^191+8fx^127+63
```

7 CONCLUSIONS

In this paper we have described a C++ class designed to analyse Vector Boolean Functions from a cryptographic perspective. It represents a very useful tool for analysing cryptographic primitives expressed as Vector Boolean Functions in a question of seconds. This class supports as input a broad range of representations such as Truth Tables, ANF Tables, polynomials in ANF, permutation and linear matrices and DES-like Sboxes. It can obtain cryptographic structures such as the Walsh Spectrum, Differential Profile and Autocorrelation Spectrum among others. Cryptographic criteria such as nonlinearity, linearity distance, correlation immunity, balancedness, algebraic degree and propagation criterion are easily calculated. The behaviour of the cryptographic properties of Vector Boolean Functions can also be studied when they interact by means of the VBF class.

ACKNOWLEDGEMENTS

This work has been partially supported by project MTM2007-62064 of the Plan Nacional de I+D+i, MEyC, Spain, and by project 166/Q06 0930-099 of the Universidad Politécnica de Madrid (UPM), Spain.

REFERENCES

Adams, C. and Tavares, S. (1993). Designing s-boxes for ciphers resistant to differential cryptanalysis. In *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography*, pages 181–190.

Bibliowicz, A., Cohen, P., and Biham, E. (2003). A system for assisting analysis of some block ciphers. Technical report, NESSIE.

Carlet, C. (2004). On the secondary constructions of resilient and bent functions. In *Progress in Computer Science and Applied Logic, vol. 23*, pages 3–28.

Carlet, C. (2008a). *Boolean functions for Cryptography and Error Correcting Codes*. Cambridge University Press.

Carlet, C. (2008b). *Vectorial Boolean functions for Cryptography*. Cambridge University Press.

Chabaud, F. and Vaudenay, S. (1994). Links between differential and linear cryptanalysis. In *EUROCRYPT*, pages 356–365.

Daemen, J. and Rijmen, V. (2002). *The Design of Rijndael*. Springer-Verlag, New York, Inc., Secaucus, NJ, USA.

Davio, M., Deschamps, J., and Thayse, A. (1978). *Discrete and Switching Functions*, volume 1 of *Advanced Book Program*. McGraw-Hill.

Gammel, B. M. (2006). http://www.matpack.de/. In *Matpack C++ Numerics and Graphics Library*.

McWilliams, F. and Sloane, N. (1977). *The Theory of Error Correcting Codes*, volume 1,2. New York, NY: North Holland.

NBS (1977). *Data Encryption Standard*. NBS, Washington, DC, USA.

Nyberg, K. (1992). On the construction of highly nonlinear permutations. In *EUROCRYPT*, pages 92–98.

Nyberg, K. (1994). S-boxes and round functions with controllable linearity and differential uniformity. In *Fast Software Encryption*, pages 111–130.

Pommerening, K. (2001). Analysis of boolean maps (s-boxes).

Rothaus, O. S. (1976). On ”bent” functions. *J. Comb. Theory, Ser. A*, 20(3):300–305.

Shoup, V. (2009). http://www.shoup.net/ntl/. In *NTL: A Library for doing Number Theory*.