Cryptographer's Toolkit for Construction of 8-Bit Bent Functions*

Hans Dobbertin and Gregor Leander

Cryptology and IT Security Research Group
Ruhr-University Bochum
D-44780 Bochum, Germany
{hans.dobbertin, gregor.leander}@ruhr-uni-bochum.de

Abstract Boolean functions form basic building blocks in various cryptographic algorithms. They are used for instance as filters in stream ciphers. Maximally non-linear (necessarily non-balanced) Boolean functions with an even number of variables are called bent functions. Bent functions can be modified to get balanced highly non-linear Boolean functions. Recently the first author has demonstrated how bent functions can be studied in a recursive framework of certain integer-valued functions. Based on this new approach we describe the practical systematic construction of 8-bit bent functions. We outline also how to compute the number of all 8-bit bent functions.[1]

1 Introduction

The design of Boolean functions (filters) and Boolean vector functions (S-boxes) is a significant issue in symmetric cryptography. Surveys and references can be found for instance in Sections 6.5 and 7.8 of Menezes et alii [12] or in Schneier [15] on pages 349–351.

To achieve resistance against certain known attacks, criteria such as high non-linearity and correlation immunity are goals the designer has to follow. The concrete practical constructions are related with very difficult combinatorial optimization problems. Fundamental questions are still open today, for instance:

(Q1) what is the maximal non-linearity of a balanced Boolean function with 7 or more variables,

(Q2) is there an APN (almost perfect non-linear) *permutation* on $\mathbb{F}_2^n$ with even $n \geq 6$,

(Q3) is the multiplicative inversion in finite fields of characteristic 2, i,e

$$\text{the mapping } x \longmapsto x^{-1} \text{ on } \mathbb{F}_{2^n}^*,$$

maximal non-linear for even $n \geq 6$?

* preliminary version

1 to be added in the final version

Already these few selected examples demonstrate that the study of Boolean functions and Boolean vector functions offers great challenges with practical impact in cryptology. We mention that in case $n = 8$ (the open) questions Q2 and Q3 can be restated as whether the S-box in the *Advanced Encryption Standard* (AES) is optimal resistent against the differential attack, resp. the linear attack.

In this paper we focus on the constructions of bent functions. A main reason for the significance of bent functions in cryptography is the fact (see [7]) that

– the highest *known* non-linearity of *balanced Boolean functions* is achieved by making a *normal bent function* balanced.

Whether this non-linearity is actually *maximal* is an open questions, see Q1 above.

Bent functions are Boolean functions on $\mathbb{F}_2^n$ with n even and $\mathbb{F}_2$ denoting the two-element field, which are maximally nonlinear in the sense that their Walsh transform attains precisely the values $\pm 2^{n/2}$. Alternatively bent functions can be defined as ± 1-valued functions on $\mathbb{F}_2^n$ with ± 1-valued Fourier transform.

All bent functions are known for $n = 2, 4, 6$, and for $n = 8$ hill climbing techniques can be used easily to find bent functions with computer support, starting with a random Boolean function and improving its non-linearity step by step. Disadvantages of such a procedure are that one can generate only relatively few bent functions, and even more important that it is not clear whether the selection process of bent functions is actually at least almost random. Thus the intension of the present note is to describe a systematic, mathematically well-founded construction of bent functions with 8 variables. It is based on the first author's recent approach [8] to study bent functions in a recursive framework of more general functions, i.e. $\mathbb{Z}$-bent functions. In the next section we shall cite and summarize the needed concepts and results without proof.

Finally in the appendix we describe the process, mentioned above, how a normal bent function can be modified such that we get a balanced Boolean function and at the same time keep close at the non-linearity of the given bent function.

2 Bent Functions in the Framework of $\mathbb{Z}$-Bent Functions [8]

Bent functions were introduced by Rothaus [14] and Dillon [5]. They have been studied intensively since then (see for instance Carlet and Guillot [2], [3], and [9] for some very recent results).

A main obstacle in the study of bent functions is the lack of recurrence laws. There are only few constructions deriving bent functions from smaller ones. But it seems that most bent functions appear without any roots to bent functions in lower dimensions, which could explain their existence.

To embed bent functions into a recursive context one needs more general structures. To this end we consider integer-valued functions f on $\mathbb{F}_2^n$ with the property that their dual, its Fourier transform $\widehat{f}$, is also integer-valued. These functions f, which we call $\mathbb{Z}$-bent functions, can be separated into different levels. ($\mathbb{Z}$ denotes the set of integers.) Higher level $\mathbb{Z}$-bent functions form building blocks that can be glued together under certain conditions in order to get larger $\mathbb{Z}$-bent functions of lower level (Theorem 1 on page 8). In this recursion we finally get the usual bent functions at level zero.

$\mathbb{Z}$-Bent squares are introduced here in order to represent (pairs of dual) $\mathbb{Z}$-bent functions and to describe the mentioned gluing process in a more transparent way.

Notations. Throughout this paper, n is a positive integer and $k = n/2$. With exception of the next section, we require that n is even.

$\mathbb{F}_{2^n}$ denotes the field with 2^n elements, and $\mathbb{R}$ the field of real numbers. Given a Boolean function $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$, its *complement* $\overline{F} = G : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ is defined by $\overline{F}(x) = F(x) + 1$, $x \in \mathbb{F}_2^n$.

The *affine group* $\mathrm{AGL}(n, \mathbb{F}_2)$ consists of all $\mathbb{F}_2$-affine permutations on $\mathbb{F}_2^n$. If a fixed set X is given, then $\mathrm{AGL}(n, \mathbb{F}_2)$ operates on the set of all functions $f : \mathbb{F}_2^n \longrightarrow X$. We denote the *orbit* of f by

$$\mathrm{orbit}(f) = \{f \circ \Phi \,:\, \Phi \in \mathrm{AGL}(n, \mathbb{F}_2)\}.$$

We call $f, g : \mathbb{F}_2^n \longrightarrow X$ *affine equivalent* if they lie in the same orbit.

2.1 Preliminaries on Fourier Transforms

The multiplicative characters of the additive group $\mathbb{F}_2^n$ are

$$\chi_a(x) = (-1)^{\langle a,x\rangle} \ (a, x \in \mathbb{F}_2^n).$$

Given a real-valued function f on $\mathbb{F}_2^n$ we denote the Fourier transform of f by $\widehat{f}$:

$$\widehat{f}(a) = \frac{1}{2^k} \sum_{x\in\mathbb{F}_2^n} f(x)\,\chi_a(x), \quad a \in \mathbb{F}_2^n.$$

The values $\widehat{f}(a)$ are called Fourier coefficients. (If the factor 2^{-k} is omitted then we use the terms *non-normalized* Fourier transform, resp. *non-normalized* Fourier coefficients.) The set of Fourier coefficients is called Fourier spectrum.

The map $f \mapsto \widehat{f}$ is a self-inverse orthogonal operator on the $\mathbb{R}$-vector space of real-valued functions on $\mathbb{F}_2^n$, endowed with inner-product given by

$$\langle f, g\rangle = \sum_{x\in\mathbb{F}_2^n} f(x)g(x),$$

i.e. we have $\widehat{\widehat{f}} = f$ involution law) and $\langle f, g\rangle = \langle \widehat{f}, \widehat{g}\rangle$ orthogonality). A linear mapping is orthogonal if and only if it is norm preserving. Recall that the norm of f induced by the inner-product is defined as $\|f\| = \sqrt{\langle f, f\rangle}$. Another version of the orthogonality law can therefore be states as

$$\|f\| = \|\widehat{f}\|,$$

which is known as *Parseval's equation.*

2.2 $\mathbb{Z}$-Bent Functions

In the remainder of this paper we suppose that $n = 2k$ is even.

Bent functions are usually defined as Boolean functions $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ such that their *Walsh transform*

$$F^{\mathcal{W}}(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{F(x)} \chi_a(x), \quad a \in \mathbb{F}_2^n,$$

attains precisely the values $\pm 2^k$. In the present context it is more convenient to replace F by $(-1)^F$ and thus consider bent functions as ±1-valued functions defined by the property that their Fourier transform is also ±1-valued.

More generally if T is a subset of $\mathbb{Z}$ we call a mapping f on $\mathbb{F}_2^n$ a *T-bent function* if both f and $\widehat{f}$ are T-valued. In order to separate $\mathbb{Z}$-bent functions into different levels, we define the increasing sequence of sets W_r, $r \geq 0$, as follows:

$$\begin{aligned} &W_0 = \{\pm 1\}, \\ &W_r = \{w \in \mathbb{Z} \mid -2^{r-1} \leq w \leq 2^{r-1}\} \quad (r > 0). \end{aligned}$$

We have $W_r \pm W_r = W_{r+1}$ for $r > 0$.

The union of all W_r-bent functions gives the set of all $\mathbb{Z}$-bent functions, since $\bigcup_{r \geq 0} W_r = \mathbb{Z}$.

> In what follows we shall use the term $\mathbb{Z}$-*bent function of level* r instead of W_r*-bent function.* The usual bent functions are precisely the $\mathbb{Z}$-bent functions of level zero.

We restate an appropriate definition which includes the level:

Definition 1. *We call a function $f : \mathbb{F}_2^n \longrightarrow W_r$ a* $\mathbb{Z}$-bent function of size $k = n/2$ and level r *if $\widehat{f}$ is also mapping into W_r. In this case, since the Fourier transform is self-inverse, $\widehat{f}$ is also a $\mathbb{Z}$-bent function of size k and level r, which is called the* dual *of f.*

We have the rules:

$$\begin{aligned}\widehat{f\circ\tau_a} &= \chi_a\widehat{f},\\ \widehat{\chi_a f} &= \widehat{f}\circ\tau_a,\\ \widehat{f\circ\Phi} &= \widehat{f}\circ(\Phi^*)^{-1},\end{aligned}$$

where $\Phi\in\mathrm{GL}(n,\mathbb{F}_2)$, i.e. $\Phi:\mathbb{F}_2^n\longrightarrow\mathbb{F}_2^n$ is a one-to-one linear mapping, Φ^* denotes the adjoint (the transposed if Φ is represented by a matrix), and $\tau_a(x)=x+a$ $(a,x\in\mathbb{F}_2^n)$ is any translation.

Let $\mathcal{B}F_r^k$ denote the set of all $\mathbb{Z}$-bent functions of size k and level r. Obviously we have

$$\begin{aligned}&\mathcal{BF}_0^k\subset\mathcal{BF}_1^k\subset\mathcal{BF}_2^k\subset...,\\ &\mathcal{BF}_r^k = {W_r}^{\mathbb{F}_2^n}\cap\widehat{{W_r}^{\mathbb{F}_2^n}},\\ \bigcup_{r\geq 0}&\mathcal{BF}_r^k = \bigcup_{r\geq 0}\left({W_r}^{\mathbb{F}_2^n}\cap\widehat{{W_r}^{\mathbb{F}_2^n}}\right)=\mathbb{Z}^{\mathbb{F}_2^n}\cap\widehat{\mathbb{Z}^{\mathbb{F}_2^n}}.\end{aligned}$$

In this formulas $X^{\mathbb{F}_2^n}$ stands for the set of all X-valued mappings on $\mathbb{F}_2^n$, and $\widehat{X^{\mathbb{F}_2^n}}$ abbreviates $\{\widehat{f}: f\in X^{\mathbb{F}_2^n}\}$.

2.3 Decomposition of $\mathbb{Z}$-Bent Functions

Proposition 1 (Decomposition). *Let $f\in\mathcal{B}F_r^k$, and suppose that U is a subspace of F_2^n of codimension 2. Without loss of generality say $U=U_{00}$, where*

$$U_{\epsilon_1\epsilon_2}=\{(y,\epsilon_1,\epsilon_2)\mid y\in\mathbb{F}_2^{n-2}\},\quad \epsilon_1,\epsilon_2\in\mathbb{F}_2.$$

Define the functions $h_{\epsilon_1\epsilon_2}$ as restrictions of f to $U_{\epsilon_1\epsilon_2}$ which is identified with $\mathbb{F}_2^{n-2}$:

$$h_{\epsilon_1\epsilon_2}(y)=f(y,\epsilon_1,\epsilon_2),\quad y\in\mathbb{F}_2^{n-2}.$$

Then for $r\geq 1$ the functions $f_{\epsilon_1\epsilon_2}$ defined as

$$\begin{aligned}f_{00}&=h_{00}+h_{10},\\ f_{10}&=h_{00}-h_{10},\\ f_{01}&=h_{01}+h_{11},\\ f_{11}&=h_{01}-h_{11},\end{aligned}$$

or in matrix notation

$$\begin{pmatrix}f_{00} & f_{01}\\ f_{10} & f_{11}\end{pmatrix}=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}h_{00} & h_{01}\\ h_{10} & h_{11}\end{pmatrix}, \tag{1}$$

lie in $\mathcal{B}F^{k-1}_{r+1}$.

We say that f_{00}, f_{01}, f_{10}, $f_{11} \in \mathcal{B}F^{k-1}_{r+1}$ *form the* canonical decomposition *of* $f \in \mathcal{B}F^k_r$. *In fact,* h_{00}, h_{01}, h_{10}, h_{11} *and therefore* f *can be recovered via*

$$\begin{pmatrix} h_{00} & h_{01} \\ h_{10} & h_{11} \end{pmatrix} = {}^1\!/\!_2 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{pmatrix}. \tag{2}$$

In the case $r = 0$ *the factor* ${}^1\!/\!_2$ *appears in equation (1) instead (2).*

Reversing the decomposition (i.e. gluing) is possible under certain conditions. However, in order to give a smoother way to describe the recursive connection between the classes of $\mathbb{Z}$-bent functions of different levels we shall introduce the concept of a $\mathbb{Z}$-*bent square* in the next subsection. There is a one-to-one correspondence between $\mathbb{Z}$-bent functions and $\mathbb{Z}$-bent squares. Matters often become more transparent in the context of $\mathbb{Z}$-bent squares, for instance

in terms of $\mathbb{Z}$*-bent squares gluing simply means concatenation (see Theorem 1).*

2.4 $\mathbb{Z}$-Bent Squares

We define the sets

$$\mathcal{V}^k_r = \{2^{k/2}\,\widehat{h} \mid h : \mathbb{F}_2^k \longrightarrow W_r\}.$$

The reason to multiply with $2^{k/2}$ is to get the integer-valued non-normalized Fourier transform:

$$2^{k/2}\,\widehat{h}(a) = \sum_{x \in \mathbb{F}_2^k} h(x)(-1)^{\langle a,x \rangle}, \quad a \in \mathbb{F}_2^k.$$

Further define

$$\mathcal{S}^k_r = \{(h_0|h_1|...|h_{2^k-1}) \,:\, h_i \in \mathcal{V}^k_r,\, i < 2^k\}$$

where the functions in $\mathcal{V}^k_r$, written as columns, are concatenated to $2^k \times 2^k$ matrices.

Definition 2. $\mathcal{B}S^k_r$*, the set of all* $\mathbb{Z}$-bent squares of size k and level r *is defined as*

$$\mathcal{B}S^k_r = \mathcal{S}^k_r \cap \left(\mathcal{S}^k_r\right)^{\mathrm{tr}}.$$

where tr *denotes the transposition operation, which is here applied to each matrix of the set* $\mathcal{S}^k_r$.

Thus a function $B : \mathbb{F}_2^k \times \mathbb{F}_2^k$ is a $\mathbb{Z}$-bent square of size k and level r if all vertical (column) functions

$$B_x(y) = B(x, y), \quad x, y \in \mathbb{F}_2^k$$

and all horizontal (row) functions

$$B^y(x) = B(x, y), \quad x, y \in \mathbb{F}_2^k$$

of B lie in $\mathcal{V}_r^k$. (As a consequence B maps into W_{r+k}.) In other words the vertical and horizonal functions are non-normalized Fourier transforms of W_r-valued functions on $\mathbb{F}_2^k$. The latter functions can be recovered via Fourier transforms by setting

$$\begin{aligned} f(x,y) &= \frac{1}{2^{k/2}}\widehat{B_x}(y) \text{ for } x, y \in \mathbb{F}_2^k, \\ g(x,y) &= \frac{1}{2^{k/2}}\widehat{B^y}(x) \text{ for } x, y \in \mathbb{F}_2^k. \end{aligned}$$

This setting implies that f and g form a dual pair, that is $\widehat{f} = g$. We call f the *vertical* $\mathbb{Z}$*-bent function,* resp. g the *horizontal* $\mathbb{Z}$*-bent function,* associated to or underlying B.

The term *bent square* refers to $\mathbb{Z}$-bent squares of level zero, which are associated to bent functions. $\mathbb{Z}$-Bent squares are just another way to describe $\mathbb{Z}$-bent functions. Loosely speaking, the Fourier transform on $\mathbb{F}_2^n = \mathbb{F}_2^k \times \mathbb{F}_2^k$ is divided in a vertical part (along cosets of $0 \times \mathbb{F}_2^k$) and in a horizontal part (along cosets of $\mathbb{F}_2^k \times 0$). Up to the factor $2^{k/2}$, in order to get integer entries, $\mathbb{Z}$-bent squares lie in the middle of computing $f \rightsquigarrow \widehat{f}$ and vice versa $\widehat{f} \rightsquigarrow f$ connecting a pair of dual $\mathbb{Z}$-bent functions.

Interchanging f with its dual means transposition of B. Via

$$\boxed{\left(B, B^{\mathrm{tr}}\right) \longleftrightarrow \left(f, \widehat{f}\right)}$$

one obtains a one-to-one correspondence between adjoint pairs of $\mathbb{Z}$-bent squares and dual pairs of $\mathbb{Z}$-bent functions of size k and level r. For any $\mathbb{Z}$-bent function h of size k we define the *parity matrix* by reducing modulo 2:

$$\mathrm{Par}(h)(x, y) = h(x, y) \bmod 2,\ x, y \in \mathbb{F}_2^k.$$

If f and $g = \widehat{f}$ are associated with B as before and $r > 0$ then we call

$$\begin{aligned} \mathrm{VerPar}[B] &= \mathrm{Par}(f), \\ \mathrm{HorPar}[B] &= \mathrm{Par}(g) \end{aligned}$$

the *horizontal parity matrix* and *vertical parity matrix* of B, respectively.

2.5 Decomposition and Gluing of $\mathbb{Z}$-Bent Squares

Proposition 2 (Decomposition). *Suppose that $B \in \mathcal{BS}_r^k$ is a $\mathbb{Z}$-bent square, then setting*

$$B_{\epsilon_1\epsilon_2}(u,v) = B(u\epsilon_1, v\epsilon_2), \quad u,v \in \mathbb{F}_2^{k-1},$$

that is

$$B = \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix} \tag{3}$$

it follows that $B_{00}, B_{01}, B_{10}, B_{11} \in \mathcal{BS}_{r+1}^{k-1}$. We call (3) the canonical decomposition *of B.*

A main motivation to introduce $\mathbb{Z}$-bent squares is that we can also nicely describe the recursive procedure leading from $\mathcal{BS}_r^k$ to $\mathcal{BS}_{r-1}^{k+1}$, the reverse of the preceding proposition.

Definition 3. *We say that $f, f' \in \mathcal{BF}_r^k$ are* parity equivalent *if* $\mathrm{Par}(f) = \mathrm{Par}(f')$.

For $r > 1$ we say $f, f' \in \mathcal{BF}_r^k$, not both in $\mathcal{BF}_1^k$, are companions, *if*

(C1) *f and f' are parity equivalent and*
(C2) *we have $|f(a)| + |f'(a)| \leq 2^{r-1}$ for all $a \in \mathbb{F}_2^n$.*

We call $f, f' \in \mathcal{BF}_1^k$ companions, *if*

(C3) $\mathrm{Par}(f)$ *and* $\mathrm{Par}(f')$ *are complementary, i.e.* $\mathrm{Par}(f') = \overline{\mathrm{Par}(f)}$.

By a admissible parity pair, *we mean a pair (P, Q) of $2^k \times 2^k$ matrices with entries in $\mathbb{F}_2$ of the form $(P,Q) = (\mathrm{Par}(f), \mathrm{Par}(\widehat{f}))$ with some $f \in \mathcal{BF}_r^k$ such that both, f and $\widehat{f}$ have a companion (see page 9).*

Clearly f and f' are companions if and only if $\frac{f+f'}{2}$ is W_{r-1}-valued for $r > 1$, resp. $f + f'$ is W_0-valued for $r = 1$. But this does not imply that also $\frac{\widehat{f}+\widehat{f'}}{2}$ is W_{r-1}-valued. Being companions does in general not inherit to the dual functions.

It is important to note that for $r = 1$ the companion relation is preserved under parity equivalence, while this is not true for $r > 1$.

Theorem 1 (Gluing). *Suppose that $\mathbb{Z}$-bent squares*

$$B_{00}, B_{01}, B_{10}, B_{11} \in \mathcal{BS}_r^k, \quad r > 0,$$

are given, where f_{ij} and g_{ij} are the (vertical and horizontal) $\mathbb{Z}$-bent functions underlying B_{ij}, respectively. (Recall that $g_{ij} = \widehat{f_{ij}}$.)

Then $B_{00}, B_{01}, B_{10}, B_{11}$ glued together form a $\mathbb{Z}$-bent square

$$B = \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$$

in $\mathcal{BS}_{r-1}^{k+1}$ if and only if the following pairs are companions:

(i). f_{00} *and* f_{10}*,*
(ii). f_{01} *and* f_{11}*,*
(iii). g_{00} *and* g_{01}*,*
(iv). g_{10} *and* g_{11}*.*

When the companion relation is represented by vertical or horizontal double arrows, these conditions can be visualized with B in the middle as follows:

$$\begin{pmatrix} f_{00} & f_{01} \\ \updownarrow & \updownarrow \\ f_{10} & f_{11} \end{pmatrix} \quad \begin{array}{c} \text{vertical} \\ \text{Fourier transform} \\ \leftrightsquigarrow \end{array} \quad \begin{pmatrix} B_{00} & B_{01} \\ & \\ B_{10} & B_{11} \end{pmatrix} \quad \begin{array}{c} \text{horizontal} \\ \text{Fourier transform} \\ \leftrightsquigarrow \end{array} \quad \begin{pmatrix} g_{00} \leftrightarrow g_{01} \\ \\ g_{10} \leftrightarrow g_{11} \end{pmatrix}$$

Gluing via parity matrices. Define the sets

$$\begin{aligned} \mathcal{U}_r^k &= \left\{ \mathrm{Par}(f) : f \in \mathcal{BF}_r^k \right\}, \\ \mathcal{P}_r^k &= \left\{ \left(\mathrm{Par}(f), \mathrm{Par}(\widehat{f}) \right) \ : \ f \in \mathcal{B}F_r^k \right\}, \\ \mathcal{BF}_r^k[P] &= \left\{ f \in \mathcal{B}F_r^k \ : \ \mathrm{Par}(f) = P \right\}, \\ \mathcal{BF}_r^k[P,Q] &= \left\{ f \in \mathcal{B}F_r^k \ : \ \mathrm{Par}(f) = P, \quad \mathrm{Par}(\widehat{f}) = Q \right\}. \end{aligned}$$

Recall that by definition

$$\begin{aligned} \mathrm{VerPar}[B] &= \mathrm{Par}(f), \\ \mathrm{HorPar}[B] &= \mathrm{Par}(g), \end{aligned}$$

where f and g are the $\mathbb{Z}$-bent functions underlying a $\mathbb{Z}$-bent square B.

Set $P_{ii} = \mathrm{Par}(f_{ii})$ and $Q_{ii} = \mathrm{Par}(\widehat{f}_{ii})$, where f_{ij} and $\widehat{f}_{ij}$ are the $\mathbb{Z}$-bent functions underlying B_{ij}. Then we say that the four pairs (P_{ij}, Q_{ij}) of parity matrices form a *parity pattern.*

Case $r > 1$. Here parity patterns have the following form:

$$\begin{array}{|c|c|} \hline P_{00} & P_{11} \\ & \\ P_{00} & P_{11} \\ \hline \end{array} \quad \longleftarrow \quad \begin{array}{|c|c|} \hline B_{00} & B_{01} \\ \hline B_{10} & B_{11} \\ \hline \end{array} \quad \longrightarrow \quad \begin{array}{|cc|} \hline Q_{00} & Q_{00} \\ \hline Q_{11} & Q_{11} \\ \hline \end{array}$$

Case $r = 1$. The parity pattern according to the gluing theorem can be visualized for $r = 1$ as follows:

$$\begin{array}{|c|c|}\hline P_{00} & P_{01} \\ & \\ \overline{P}_{00} & \overline{P}_{01} \\ \hline\end{array} \quad\longleftarrow\quad \begin{array}{|c|c|}\hline B_{00} & B_{01} \\ \hline B_{10} & B_{11} \\ \hline\end{array} \quad\longrightarrow\quad \begin{array}{|cc|}\hline Q_{00} & \overline{Q}_{00} \\ \hline Q_{10} & \overline{Q}_{10} \\ \hline\end{array}$$

For all $f \in \mathcal{BF}_1^k$ we have $\|\operatorname{Par}(f)\| = \|f\|$, since f attains only the values $0, \pm 1$. By Parseval's equation we conclude that $\|P_{00}\| = \|Q_{00}\| = \|P_{11}\| = \|Q_{11}\|$ and $\|P_{00}\| + \|P_{01}\| = \|Q_{00}\| + \|Q_{10}\| = 2^n$.

3 On the structure and size of $\mathcal{U}_r^k$ and $\mathcal{P}_r^k$

In the sequel we shall consider Reed-Muller codes. Recall that the *Reed-Muller code* $\mathrm{RM}(\ell, m)$ is defined as the set of all Boolean functions with m variables of degree at most ℓ.

For each fixed k the $\mathcal{U}_r^k$, $r > 0$, form an increasing sequence of sets.

Theorem 2. *For all r, k we have $\mathcal{U}_r^k \subseteq \mathrm{RM}(k, 2k)$, and moreover*

$$\mathcal{U}_r^k = \mathrm{RM}(k, 2k)$$

for each fixed k and sufficiently large r. In the latter case, where

$$\dim \mathcal{U}_r^k = \dim \mathrm{RM}(k, 2k) = 2^{2k-1} + {}^1\!/_2 \binom{2k}{k},$$

we say that $\mathcal{U}_r^k$ is maximal.

Proof. Suppose that $P = \operatorname{Par}(f)$, with $f \in \mathcal{BF}_r^k$. We have to show that $\deg(P) \le k$. Our proof is essentially the same that shows that bent functions have at most degree k. We denote the coefficients of the algebraic normal form of P by α_ν for $\nu \in \mathbb{F}_2^n$, i.e.

$$\operatorname{Par}(f)(x) = f(x) \bmod 2 = \sum_{\nu \in \mathbb{F}_2^n} \alpha_\nu x^u.$$

It its well known that α_ν can be written as

$$\alpha_\nu = \left(\sum_{a \in V_\nu} \operatorname{Par}(f)(a) \right) \bmod 2 = \left(\sum_{a \in V_\nu} f(a) \right) \bmod 2$$

where $V_\nu = \{x \in \mathbb{F}_2^n \mid x \le \nu\}$. Note that the dimension of V_ν is the binary weight $\|\nu\|$ of ν.

Furthermore for any Boolean function g and any subspace $U \subseteq \mathbb{F}_2^n$ we have

$$\sum_{a \in U} \widehat{g}(a) = 2^{-k} \sum_{x \in \mathbb{F}_2^n} g(x) \sum_{a \in U} (-1)^{\langle a, x \rangle} = 2^{-k} \, \#U \left(\sum_{x \in U^{\perp}} g(x) \right).$$

Using this equation we see that

$$\begin{aligned} \alpha_\nu &= \left(2^{k-(n-\|\nu\|)} \sum_{a \in V_\nu^{\perp}} \widehat{f}(a) \right) \bmod 2 \\ &= 2^{\|u\|-k} \left(\sum_{a \in V_\nu^{\perp}} \widehat{f}(a) \right) \bmod 2. \end{aligned} \tag{4}$$

As f is a $\mathbb{Z}$-bent function $\widehat{f}(a)$ is an integer for all a, and thus whenever $\|\nu\| > k$ we see that $\alpha_\nu = 0$ as stated.

To prove the second statement we have to show, that for every Boolean function P of degree at most k, there exist a function f from $\mathbb{F}_2^n$ to the integers, such that all the Fourier coefficients are integers again and $\mathrm{Par}(f) = P$. First assume that P is a monomial, for instance $P(x_0, \ldots, x_{n-1}) = x_0 x_1 \cdots x_{t-1}$ for some $t \leq k$. As then P is independent of $x_t, ..., x_{k-1}$, that is at least k variables, it is clear that all the Fourier coefficients are integers. Moreover the monomials x^ν with $\|\nu\| \leq k$ form a basis of $\mathrm{RM}(k, 2k)$. Hence

$$\dim \mathrm{RM}(k, 2k) = \sum_{i \leq k} \binom{2k}{i} = 2^{2k-1} + 1/2 \binom{2k}{k}.$$

Now if $P \in \mathrm{RM}(k, 2k)$, say

$$P(x) = \left(\sum_{\|\nu\| \leq k} \lambda_\nu x^\nu \right) \bmod 2, \quad \lambda_i \in \{0, 1\} \subseteq \mathbb{Z},$$

and we can simply define $f(x) = \sum_{\|\nu\| \leq k} \lambda_\nu x^\nu$, where the sum is over the integers. Then, due to the linearity of the Fourier transform, $\widehat{f}$ is integer-valued, which means that $f \in \mathcal{BF}_r^k$ for sufficiently large r. □

In the sequel we shall consider the subcode

$$\mathrm{RM}^*(\ell, m) \subseteq \mathrm{RM}(\ell, m)$$

consisting of all $F \in \mathrm{RM}(\ell, m)$ which are sums of monomials with degree ℓ. (Note that $\mathrm{RM}^*(\ell, m)$ can be canonically identified with $\mathrm{RM}(\ell, m)/\mathrm{RM}(\ell-1, m)$.) For $u \in \mathrm{RM}(\ell, m)$ we set

$$u^* = \text{degree } \ell \text{ part of } u.$$

In this way we get the canonical projection $u \mapsto u^*$ from $\mathrm{RM}(\ell, m)$ onto $\mathrm{RM}^*(\ell, m)$.

Lemma 1. *Suppose $(\ell, m) = (k, 2k)$. For $P = \mathrm{Par}(f)$ ($f \in \mathcal{BF}_r^k$) the setting*

$$P \mapsto \tilde{P} = \mathrm{Par}(\widehat{f})^*, \quad \mathcal{U}_r^k \longrightarrow \mathrm{RM}^*(k, 2k)$$

we get a well-defined mapping (i.e., this setting is correct in the sense that it does not depend on the respective choice of f).

Proof. Suppose $\mathrm{Par}(f) = \mathrm{Par}(f')$ for $\mathbb{Z}$-bent functions f, f' of size k. We have to show that $\mathrm{Par}(\widehat{f})^* = \mathrm{Par}(\widehat{f'})^*$. As Par and $()^*$ are linear, we can assume $f' = 0$, i.e. we suppose that f attains only even values. Then by equation (4) above, with f replaced by $\widehat{f}$, we conclude that $\mathrm{Par}(\widehat{f}) \in \mathrm{RM}(k-1, 2k)$ and hence $\mathrm{Par}(\widehat{f})^* = 0$, since

$$\sum_{a \in V_\nu^\perp} f(a)$$

is even, since it is a sum of even numbers. □

For each fixed k the $\mathcal{P}_r^k$, $r > 0$, form an increasing sequence of sets. Based on the previous lemma and similar arguments as in the proof of Theorem 2, it is now easy to verify the next theorem. (We note that for $u \in \mathrm{RM}(k, 2k)$, one can describe $\tilde{u}$ directly as follows: $\tilde{u}$ is the sum of all complemented degree k monomials, which occur in u.)

Theorem 3. *For all r, k we have*

$$\mathcal{P}_r^k \subseteq \{(u, \tilde{u} + v) \mid u \in \mathcal{U}_r^k,\ v \in \mathrm{RM}(k-1, 2k)\} \cong \mathcal{U}_r^k \times \mathrm{RM}(k-1, 2k).$$

Moreover we have $\mathcal{U}_r^k = \mathrm{RM}(k, 2k)$ and

$$\begin{aligned} \mathcal{P}_r^k &= \{(u, \tilde{u} + v) \mid u \in \mathrm{RM}(k, 2k),\ v \in \mathrm{RM}(k-1, 2k)\} \\ &\cong \mathrm{RM}(k, 2k) \times \mathrm{RM}(k-1, 2k) \end{aligned}$$

for each fixed k and sufficiently large r. In the latter case, where

$$\dim \mathcal{P}_r^k = \dim \mathrm{RM}(k, 2k) + \dim \mathrm{RM}(k-1, 2k) = 2^{2k},$$

we say that $\mathcal{P}_r^k$ is maximal.

We are now prepared to construct bent function with 8 variables by applying two times the recursion via gluing described in Theorem 1:

$$\mathcal{B}F_2^2 \rightsquigarrow \mathcal{B}F_1^3 \rightsquigarrow \mathcal{B}F_0^4.$$

4 From $\mathcal{BF}_2^2$ to $\mathcal{BF}_1^3$

The set $\mathcal{BF}_2^2$ of $\mathbb{Z}$-bent functions of size 2 and level 2 can be computed simply by checking the about 2^{37} functions on $\mathbb{F}_2^4$ with values in $W_2 = \{0, \pm 1, \pm 2\}$.

Fact 1. $\# \mathcal{BF}_2^2 = 488\,090\,305 = 2^{28.86}$.

Fact 2. Our experimental results have shown that $\mathcal{U}_1^2$ and $\mathcal{P}_2^2$ are maximal:

$$\begin{aligned} \mathcal{U}_1^2 &= \mathcal{U}_2^2 = \mathrm{RM}(2,4), & \dim \mathcal{U}_1^2 &= 11, \\ \mathcal{P}_2^2 &\cong \mathrm{RM}(2,4) \times \mathrm{RM}(1,4), & \dim \mathcal{P}_2^2 &= 16. \end{aligned}$$

Note that, to confirm the above, it is sufficient to check the respectively stated dimensions for $\mathcal{U}_1^2$ and $\mathcal{P}_2^2$, which are the maximal dimensions by Theorems 2 and 3.

In order to glue four $\mathbb{Z}$-bent functions together to get functions in $\mathcal{BF}_1^3$, we first consider the conditions on their parity patterns. By condition (C1) of Definition 3 and Theorem 1, given $f_{00}, f_{01}, f_{10}, f_{11} \in \mathcal{BF}_2^2$ can be glued together only if

$$\begin{aligned} \mathrm{Par}(f_{00}) &= \mathrm{Par}(f_{10}), \\ \mathrm{Par}(f_{01}) &= \mathrm{Par}(f_{11}), \\ \mathrm{Par}(\widehat{f_{00}}) &= \mathrm{Par}(\widehat{f_{01}}), \\ \mathrm{Par}(\widehat{f_{10}}) &= \mathrm{Par}(\widehat{f_{11}}). \end{aligned}$$

If we denote $P_{ij} = \mathrm{Par}(f_{00})$ and $Q_{ij} = \mathrm{Par}(\widehat{f_{ij}})$ this gluing condition translates to

$$((P_{00}, Q_{00}), (P_{01}, Q_{01}), (P_{10}, Q_{10}), (P_{11}, Q_{11})) \in \mathcal{V},$$

where we set

$$\begin{aligned} \mathcal{V}_1 &= \mathcal{P}_2^2 \times \mathcal{P}_2^2 \times \mathcal{P}_2^2 \times \mathcal{P}_2^2, \\ \mathcal{V}_2 &= \{\big((P'_{00}, Q'_{00}), (P'_{01}, Q'_{00}), (P'_{00}, Q'_{10}), (P'_{01}, Q'_{10})\big) : P'_{ij}, Q'_{ij} \in \mathcal{U}_2^2\} \end{aligned}$$

and $\mathcal{V} = \mathcal{V}_1 \cap \mathcal{V}_2$. As both $\mathcal{V}_1$ and $\mathcal{V}_2$ are vector spaces, the intersection $\mathcal{V}$ can be easily computed. Our computer calculations have shown:

Fact 3. $\dim(\mathcal{V}) = 26$.

For actually gluing functions together we compute for each parity pair P, Q the set

$$\mathcal{BF}_2^2[P, Q] = \left\{ f \in \mathcal{BF}_2^2 : \Big(\mathrm{Par}(f), \mathrm{Par}(\widehat{f})\Big) = (P, Q) \right\}.$$

By computer calculations we found the distribution of the sizes of these sets:

$\|P\| + \|Q\|$	$\# \mathcal{BF}_2^2[P,Q]$ [number of times]
0	213 249 [1]
8	34 366 [560], 21 072 [60]
12	14 620 [7 168], 12 584 [6 720]
16	5 840 [14 336], 6 400 [480], 1 120 [2], 3 992 [1 120], 6 852 [20 160], 7 368 [420]
20	2 880 [6 720], 3 480 [7 168]
24	896 [60], 2 064 [560]
32	896 [1]

Note that the 896 functions in the last line, where $P = Q$ is the 4×4 matrix with constant entry 1, are precisely the bent functions with 4 variables. (The first line of the table, where $P = Q = 0$, means that

$$\# \mathcal{BF}_1^2 = \# 2\, \mathcal{BF}_1^2 = 213\,249 = 2^{17.70}.$$

In order to generate to the set $\mathcal{BF}_1^3$ by gluing, we take the second condition (C2) of Definition 3 into account. This is not a condition for the parity matrices, but on the functions in $\mathcal{BF}_2^2$ itself. Given the 2^{26} possible parity patterns and the size of the sets $\mathcal{BF}_2^2[P,Q]$ (see table above) we can determine the effort by simply multiplying the corresponding sizes of $\mathcal{BF}_2^2[P,Q]$. This gives a huge number of about 2^{80} quadruples of functions contained in $\mathcal{BF}_2^2$ that have to be considered in order to compute $\mathcal{BF}_1^3$.

But we do not need $\mathcal{BF}_1^3$, it suffices to compute $\mathcal{U}_1^3$ and the size of $\mathcal{P}_1^3$, and this is easy to do as we shall see in the sequel. (Actually $\mathcal{BF}_1^3$ could be computed, not by brute force, but using the technique explained in the next section, suitably modified.)

Obviously Theorem 3 has strong consequences for the form of a parity pattern $(P_{ij}, Q_{ij}) \in \mathcal{P}_r^k$ in case $r > 1$: there are unique $T \in \mathrm{RM}^*(k, 2k)$ and $\varDelta_i \in \mathrm{RM}(k-1, 2k)$, $i = 1, 2, 3, 4$ such that

$$\boxed{\begin{array}{ll} P_{00} = T + \varDelta_1, & P_{01} = T + \varDelta_2, \\ P_{10} = T + \varDelta_2, & P_{11} = T + \varDelta_2, \\[1ex] Q_{00} = \tilde{T} + \varDelta_3, & Q_{01} = \tilde{T} + \varDelta_3, \\ Q_{10} = \tilde{T} + \varDelta_4, & Q_{11} = \tilde{T} + \varDelta_4. \end{array}}$$

Moreover, we can deduce information about the parity matrix of the resulting glued function in $\mathcal{BF}_1^3$. For example if $P_{00}(x) = 0$ (and so $P_{10}(x) = 0$ due to the required parity equivalence) with $x \in \mathbb{F}_2^4$, by considering all possible values of the underlying functions f_{00} and f_{10} we see that the parity matrix of the glued function takes on either the pairs of values $0, 0$ or $1, 1$ at the positions $(x, 0, 0)$ and $(x, 0, 1)$. If $P_{00}(x) =$

$P_{11}(x) = 1$ then the parity of the glued function attains either the pair of values $1, 0$ or $0, 1$.

In addition computing $\mathcal{BF}_1^3[P]$ we can use the fact that f and $\chi_a f$ have the same parity matrix and that $\widehat{\chi_a f} = \widehat{f} \circ \tau_a$ (see page 5). This means that in order to construct $f \in \mathcal{BF}_1^3[P]$ we can prescribe signs of its values for linearly independent arguments.

These observations lead to

$$\boxed{\text{a fast algorithm to compute a } \textit{parity class } \mathcal{BF}_1^3[P] = \{f \in \mathcal{BF}_1^3\,;\, \mathrm{Par}(f) = P\}.}$$

This forms a kind of fast[2] *elementary macro operation* in our algorithms to construct bent functions. We shall refer to it as a measure unit for the complexity of algorithms in the next section.

Computing $\mathcal{U}_1^3$. We known that $\mathcal{U}_1^3$ is a subset of $\mathrm{RM}(3, 6)$. Moreover, $\mathcal{U}_1^3$ is closed under affine equivalence. Thus it suffices to check if $\mathcal{BF}_1^3[P]$ is non-empty, where P runs to a subset, say $\mathcal{S}$, of $\mathrm{RM}(3, 6)$ representing the (few thousand) orbits under the affine group. This works, since $\#\mathcal{U}_1^3 \leq \#\,\mathrm{RM}(3, 6) = 2^{42}$:

$$\mathcal{U}_1^3 = \bigcup \{\mathrm{orbit}(P) \;:\; P \in \mathcal{S} \text{ and } \mathcal{BF}_1^3[P] \text{ is non-empty}\}.$$

Computing $\#\,\mathcal{P}_1^3$. From the first statement in Theorem 3 it follows that $\#\,\mathcal{P}_1^3 \leq 2^{64}$. Compute $\mathcal{BF}_1^3[P]$ for $P \in \mathcal{S}$, and

$$b_P := \#\{\mathrm{Par}(\widehat{f})\,;\, f \in \mathcal{BF}_1^3[P]\}.$$

Then

$$\#\,\mathcal{P}_1^3 = \sum_{P \in \mathcal{S}} \#\,\mathrm{orbit}(P) \times b_P.$$

If $\mathcal{U}_r^k$ and $\mathcal{P}_r^k$ are maximal then

$$\#\,\mathcal{U}_r^k = 2^{2^{2k-1} + \frac{1}{2}\binom{2k}{k}}$$

and

$$\#\,\mathcal{P}_r^k = 2^{2^{2k}}.$$

We shall refer in the next section to the quotients between these values and the corresponding values for $r = 1$, that is we define

$$\mathbf{u}(k) = \#\,\mathcal{U}_1^k \,/ 2^{2^{2k-1} + \frac{1}{2}\binom{2k}{k}}, \tag{5}$$

$$\mathbf{p}(k) = \#\,\mathcal{P}_1^k \,/ 2^{2^{2k}}, \tag{6}$$

$$\mathbf{pu}(k) = \mathbf{p}(k) / \,\mathbf{u}(k). \tag{7}$$

[2] Concretely *fast* means: say a few milliseconds on a PC.

Hence for $k = 3$

$$\mathbf{u}(k) = \#\,\mathcal{U}_1^3\,/2^{42},$$
$$\mathbf{p}(k) = \#\,\mathcal{P}_1^3\,/2^{64}.$$

By Theorem 3 we know that

$$\mathbf{p}(k) \le \mathbf{u}(k) \le 1.$$

5 From $\mathcal{BF}_1^3$ to 8-Bit Bent Functions

Now bent functions in 8 variables can be constructed by gluing together suitable functions selected in $\mathcal{BF}_1^3$. This final step is simpler than the previous gluing in so far as now condition (C3) in Definition 3, a pure parity condition, has to be fulfilled.

From Theorem 3 we conclude for each parity pattern $(P_{ij}, Q_{ij}) \in \mathcal{P}_1^k$ (this is the analogue of (5) for $r = 1$): there are unique $T \in \mathrm{RM}^*(k, 2k)$ and $\mathit{\Delta}_i \in \mathrm{RM}(k-1, 2k)$, $i = 1, 2, 3, 4$ such that

$$\boxed{\begin{array}{ll} P_{00} = T + \mathit{\Delta}_1, & P_{01} = T + \mathit{\Delta}_2, \\ P_{10} = T + \overline{\mathit{\Delta}}_1, & P_{11} = T + \overline{\mathit{\Delta}}_2, \\[1ex] Q_{00} = \tilde{T} + \mathit{\Delta}_3, & Q_{01} = \tilde{T} + \overline{\mathit{\Delta}}_3, \\ Q_{10} = \tilde{T} + \mathit{\Delta}_4, & Q_{11} = \tilde{T} + \overline{\mathit{\Delta}}_4. \end{array}} \tag{8}$$

Recall that moreover

$$\|P_{00}\| = \|Q_{00}\| = \|P_{11}\| = \|Q_{11}\|,$$
$$\|P_{01}\| = \|Q_{01}\| = \|P_{10}\| = \|Q_{10}\|$$

and

$$\|P_{00}\| + \|P_{01}\| = 2^{2k}.$$

For a bent function $f \in \mathcal{BF}_0^{k+1}$ we call the collection of the associated $\mathrm{Par}(f_{ij}) = P_{ij}$, $i, j = 0, 1$ (with $P_{1i} = \overline{P}_{0i}$) the *left parity pattern* of f. (8) is the basis for the following algorithm.

`BasicGluing:`

Algorithm to compute all bent functions with prescribed left parity pattern

- **Initialize.** Either the input left parity pattern (P_{00}, P_{01}) is given or a suitable input can be found as follows: Choose any $T \in \mathrm{RM}^*(k.2k)$. Choose $\Delta_1, \Delta_2 \in \mathrm{RM}(k-1, 2k)$ according to (8), i.e.

$$\begin{aligned} P_{00} &:= T + \Delta_1 \in \mathcal{U}_1^k, \\ P_{01} &:= T + \Delta_2 \in \mathcal{U}_1^k, \\ P_{10} &:= \overline{P}_{00} = T + \overline{\Delta}_1 \in \mathcal{U}_1^k, \\ P_{11} &:= \overline{P}_{01} = T + \overline{\Delta}_2 \in \mathcal{U}_1^k, \end{aligned}$$

  and such that $\|P_{00} + P_{01}\| = 2^{2k}$.
- **Step A1.** Compute all $f_{00} \in \mathcal{BF}_1^k[P_{00}]$, and the set $D_3 \subseteq \mathrm{RM}(k-1, 2k)$ of all

$$\Delta_3 := \mathrm{Par}(\widehat{f}_{00}) + \tilde{T}.$$

  Compute all $f_{01} \in \mathcal{BF}_1^k[P_{01}]$, and the set $D_3' \subseteq \mathrm{RM}(k-1, 2k)$ of all

$$\Delta_3' := \mathrm{Par}(\widehat{f}_{01}) + \tilde{T}.$$

  Compute the collision set

$$E_3 := \{X \,:\, X \in D_3,\ \overline{X} \in D_3'\}.$$

  Store all sets

$$\begin{aligned} &\mathcal{BF}_1^k[P_{00}, X], \quad X \in E_3, \\ &\mathcal{BF}_1^k[P_{01}, X], \quad \overline{X} \in E_3. \end{aligned}$$

  This procedure can be visualized as follows:

$$\boxed{P_{00}} = \mathrm{Par}(f_{00}) \rightrightarrows \mathrm{Par}(\widehat{f}_{00}) = \boxed{Q_{00}} \quad \overset{\text{complement?}}{\longleftrightarrow} \quad \boxed{Q_{11}} = \mathrm{Par}(\widehat{f}_{01}) \leftleftarrows \mathrm{Par}(f_{01}) = \boxed{P_{01}}$$

- **Step A2.** Compute all $f_{10} \in \mathcal{BF}_1^k[P_{10}]$, and the set $D_4 \subseteq \mathrm{RM}(k-1, 2k)$ of all

$$\Delta_4 := \mathrm{Par}(\widehat{f}_{10}) + \tilde{T}.$$

  Compute all $f_{11} \in \mathcal{BF}_1^k[P_{01}]$, and the set $D_3' \subseteq \mathrm{RM}(k-1, 2k)$ of all

$$\Delta_4' := \mathrm{Par}(\widehat{f}_{11}) + \tilde{T}.$$

Compute the "collision" set

$$E_4 := \{Y\ :\ Y \in D_4,\ \overline{Y} \in D_4'\}.$$

Store all sets

$$\begin{aligned} &\mathcal{BF}_1^k[P_{10}, Y], \quad Y \in E_4, \\ &\mathcal{BF}_1^k[P_{11}, \overline{Y}], \quad Y \in E_4. \end{aligned}$$

This procedure can be visualized as follows:

$$\boxed{\overline{P}_{00}} = \mathrm{Par}(f_{10}) \rightrightarrows \mathrm{Par}(\widehat{f}_{10}) = \boxed{Q_{10}} \quad \overset{\text{complement?}}{\longleftrightarrow} \quad \boxed{Q_{11}} = \mathrm{Par}(\widehat{f}_{11}) \leftleftarrows \mathrm{Par}(f_{11}) = \boxed{\overline{P}_{01}}$$

– **Step B.** For each pair $(X, Y) \in E_3 \times E_4$ we get a parity pattern by setting

$$\begin{aligned} Q_{00} &:= X, & Q_{01} &:= \overline{X} \\ Q_{10} &:= Y, & Q_{11} &:= \overline{Y}. \end{aligned}$$

That is, each selection of functions $f_{ij} \in \mathcal{BF}[P_{ij}, Q_{ij}]$ can be glued together, and give a bent function of size $k+1$:

$$f := \begin{pmatrix} h_{00} & h_{10} \\ h_{01} & h_{11} \end{pmatrix} = {}^1\!/_2 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} f_{00} & f_{10} \\ f_{01} & f_{11} \end{pmatrix}.$$

Remark 1. Note that for all k

$$\mathcal{BF}_1^k[\mathbf{1}] = \mathcal{BF}_0^k.$$

Thus the very special case $P_{00} = \mathbf{1}$ means that we glue two *bent functions,* in the sense that f_{00}, f_{11} are bent functions and $f_{01} = f_{10} = \mathbf{0}$. In terms of concatenation (see Proposition 1) this is the well-known construction

$$f = \begin{pmatrix} h_{00} & h_{01} \\ h_{10} & h_{11} \end{pmatrix} = \begin{pmatrix} f_{00} & f_{11} \\ f_{00} & -f_{11} \end{pmatrix}.$$

Only here the Gluing Theorem provides a recursion from bent functions of size k to bent functions of size $k+1$. Otherwise none of the glued functions is bent.

Remark 2. Let F be the Boolean function associated to a bent functions f generated by `BasicGluing`, that is

$$f(z) = (-1)^{F(z)}, \quad x \in \mathbb{F}_2^{n+2}\ (n = 2k).$$

It is easy to see that for all $x = (x_1, ..., x_n) \in \mathbb{F}_2^n$ we have

$$F(x, x_n, x_{n+1}) = x_{n+1}x_{n+2}\,(P_{00}(x) + P_{01}(x)) + x_{n+1}\,(P_{00}(x) + 1) + x_{n+2}U(x) + V(x),$$

where $U \in \mathrm{RM}(k, 2k)$ and $V \in \mathrm{RM}(k+1, 2k)$ depend on the specific choice of the f_{ij} which are glued together.

`BasicGluing` requires 4 parity class computations, and the heuristic average number of found parity pattern is

$$\mathbf{pu}(k)^4\ (\#\,\mathrm{RM}(k-1,2k))^2\,, \quad \mathbf{pu}(k) = \mathbf{p}(k)/\,\mathbf{u}(k). \tag{9}$$

In fact we can estimate that the subsets D_i, D'_i, $i = 3, 4$ of the coset $\tilde{T} + \mathrm{RM}(k-1, 2k)$ have size $(\mathbf{p}(k)/\,\mathbf{u}(k)) \times \#\,\mathrm{RM}(k-1, 2k)$. Thus the "collision" sets E_i have an estimated size of $(\mathbf{pu}(k))^2 \times \#\,\mathrm{RM}(k-1, 2k)$, and therefore the number $\#E_3 \times \#E_4$ of found parity pattern can be estimated as stated in (9).

In the practically relevant case $k = 3$ we get from (9) the estimation of

$$\mathbf{pu}(3)^4 \times 2^{44}$$

found parity patterns. (We only know that $\mathbf{pu}(3) \leq 1$, but not its precise value yet.) Given a parity pattern, say P_{ij}, Q_{ij} we get a bent function for each choice of functions $f_{ij} \in \mathcal{BF}_1^3[P_{ij}, Q_{ij}]$, respectively. We say that these bent functions belong to the same class with respect to parity-pattern equivalence (see definition below). Their number is

$$\prod_{i,j\in\{0,1\}} \#\,\mathcal{BF}_1^3[P_{ij}, Q_{ij}]. \tag{10}$$

See below how this can be used to find the number of all 8-bit bent functions.

Definition 4. *We call bent functions $f, f' \in \mathcal{BF}_0^{k+1}$* parity-pattern equivalent, *resp.* left parity-pattern equivalent, *if we have* $\mathrm{Par}(f_{ij}) = \mathrm{Par}(f'_{ij})$ *and* $\mathrm{Par}(\widehat{f_{ij}}) = \mathrm{Par}(\widehat{f'}_{ij})$, *resp.* $\mathrm{Par}(f_{ij}) = \mathrm{Par}(f'_{ij})$, *for their canonical decompositions f_{ij}, f'_{ij} (see Proposition 1).*

Exchanging f_{ij} by $-f_{ij}$ leads to an equivalent bent function (in the sense of the below definition). In general the structure and size of the sets $\mathcal{BF}_1^3[P, Q]$ has still to be analyzed, theoretical and by experiments (some information can already be found in [8]).

Definition 5. *We call $\mathbb{Z}$-bent functions $f, g \in \mathcal{BF}_r^k$* equivalent *if*

$$g(x) = \pm\chi_a f(\Phi(x)), \quad a, x \in \mathbb{F}_2^n, \Phi \in \mathrm{AGL}(n, \mathbb{F}_2).$$

In the additive setting this means that Boolean functions are equivalent *if they are affine equivalent up to addition of an affine map.*

Computing all 8-bit bent functions up to equivalence. Our intension is to compute a collection of 8-bit bent functions representing all equivalence classes. Therefore we can suppose that P_{00} runs through a system of representatives for $\mathrm{RM}(3,6)$ under equivalence. The number of orbits in $\mathrm{RM}(3,6)/\mathrm{RM}(1,6)$ under the affine group $\mathrm{AGL}(6,\mathbb{F}_2)$ is known to be 34. Define the set

$$\mathcal{I} = \{(P_0, P_1) \,:\, P_0, P_1, \overline{P}_0, \overline{P}_1 \in \mathcal{U}_1^3,\ P_0^* = P_1^*,\ \|P_0\| + \|P_1\| = 2^{2k}\}$$

of allowed input for `BasicGluing`. The number of `BasicGluing` computations, which are necessary to find all 8-bit bent function up to equivalence, is 4 times the number of orbits in $\mathcal{I}$ under the affine group operating as follows:

$$\Phi(P_0, P_1) = (P_0 \circ \Phi, P_1 \circ \Phi), \quad \Phi \in \mathrm{AGL}(6, \mathbb{F}_2).$$

The number of orbits, that is $\#\,(\mathcal{I}/\,\mathrm{AGL}(6,\mathbb{F}_2))$, is bounded by

$$\begin{aligned}&\#\,(\mathrm{RM}(3,6)/\,\mathrm{RM}(1,6) \bmod \mathrm{AGL}(6,\mathbb{F}_2)) \times \#\,\mathrm{RM}(1,6) \times \#\,\mathrm{RM}(2,6)\\ &= 34 \times 2^7 \times 2^{22} = 2^{34.09}.\end{aligned}$$

The size of $\#\,(\mathcal{I}/\,\mathrm{AGL}(6,\mathbb{F}_2))$ is certainly much smaller. (It has still to be computed.)

The critical bottleneck is to handle the huge amount of bent functions, which can be derived in Step B of `BasicGluing`. The effort depends on

- the precise value of $\mathbf{p}(3) \leq 1$ and $\mathbf{u}(3) \leq 1$,
- the precise size of $\#\,(\mathcal{I}/\,\mathrm{AGL}(6,\mathbb{F}_2))$,
- fast equivalence tests for bent functions.

A very rough and optimistic *guess* is that we have to sieve bout 2^{50} bent functions.

Computing the number of 8-bit bent functions. For even m let $\mathbf{B}_m = \#\,\mathcal{BF}_0^{m/2}$ denote the number of bent functions with m variables (of size $m/2$). Adding a Step C to the algorithm `BasicGluing` we can compute the number, say $\mathbf{B}_8[P_{00}, P_{01}]$, of 8-bit bent functions with a prescribed left parity pattern as follows:

Step C. Compute (see [8])

- $\mathbf{N}(X) := \#\,\mathcal{BF}_1^3[P_{00}, X] \times \#\,\mathcal{BF}_1^3[P_{01}, \overline{X}]$ for all $X \in E_3$,
- $\mathbf{M}(Y) := \#\,\mathcal{BF}_1^3[\overline{P}_{00}, Y] \times \#\,\mathcal{BF}_1^3[\overline{P}_{01}, \overline{Y}]$ for all $Y \in E_4$,
- $\mathbf{B}_8[P_{00}, P_{01}] := \big(\sum_{X\in E_3} \mathbf{N}(X)\big) \times \big(\sum_{Y\in E_4} \mathbf{M}(Y)\big)$.

We have to do Step C with input (P_{00}, P_{01}) running through a set of pairs representing the orbits in $\mathcal{I}/\,\mathrm{AGL}(6,\mathbb{F}_2)$. An optimistic estimation of the number of orbits is 2^{15}. We anticipate to determine $\mathbf{B}_8$ as described within the next monthes.

Estimating the number of bent functions. Recall that $\mathbf{B}_4 = 896$, as can be found by hand calculations and that also

$$\mathbf{B}_6 = 5\,425\,430\,528 = 2^{32.34}$$

is known for more than ten years (see [13]).

Theorem 4. *We have the following* heuristic *recursive estimation:*

$$\boxed{\log_2 \#\mathcal{BF}_0^{k+1} = \log_2 \mathbf{B}_{2k+2} \approx 4 \log_2 \#\mathcal{BF}_1^k - 2^{2k+1} - \tbinom{2k}{k}.} \tag{11}$$

Proof. In order to cover all possible parity patterns we consider all triples

$$(T, \Delta_1, \Delta_2) \in \mathrm{RM}^*(k, 2k) \times \mathrm{RM}(k-1, 2k) \times \mathrm{RM}(k-1, 2k)$$

and define P_{ij} $(i, j = 1, 2)$ as in `BasicGluing`. The probability that all $P_{ij} \in \mathcal{U}_1^k$ for $i, j = 1, 2$ is $\mathbf{u}(k)^4$ (otherwise we do not get a parity pattern), and in this case the estimated number of derivable parity patterns is given in (9). Hence the overall number of parity patterns is about

$$\mathbf{p}(k)^4 \times \#\,\mathrm{RM}^*(k, 2k) \times \#\,\mathrm{RM}(k-1, 2k)^4. \tag{12}$$

Since

$$\#\mathcal{BF}_1^k = \sum_{(P,Q) \in \mathcal{P}_1^k} \#\mathcal{BF}_1^k[P, Q],$$

the average size of the sets $\mathcal{BF}_1^k[P, Q]$ is

$$\#\mathcal{BF}_1^k / \#\mathcal{P}_1^k = \#\mathcal{BF}_1^k / \left(\mathbf{p}(k) \times 2^{2^{2k}}\right).$$

The 4-th power of this number has to be multiplied with (12) to get the number of bent functions of size $k+1$ (see Step B in `BasicGluing` and (10)). This proves (11). (Note that we can argue also using the restricted form of parity pattern given in (8).)
□

The reason why we cannot simply continue the recursion in (11) is that we have to take into account a certain reduction factor, to be analyzed, which comes from condition (C2) and which is relevant for $r > 1$. We shall discuss this matter and add some results along this line in the final version.

(11) is rather realistic for $k = 1, 2$. In fact, since $\#\mathcal{BF}_1^1 = 33$ and $\#\mathcal{BF}_1^2 = 213\,249$ we get

$$\log_2 \mathbf{B}_4 = \log_2 \#\mathcal{BF}_0^2 \approx 10.18 \quad \text{(correct value: 9.81)},$$
$$\log_2 \mathbf{B}_6 = \log_2 \#\mathcal{BF}_0^3 \approx 32.80 \quad \text{(correct value: 32.34)}.$$

As it is easy to compute $\#\mathcal{BF}_1^3$ with very low effort (see previous section), we can hope to derive from (11) also a good estimation for $\log_2 \mathbf{B}_8 = \log_2 \#\mathcal{BF}_0^4$ and compare it with the exact value to test the quality of the estimation (11).

Practical application

Summarizing, to construct 8-bit bent functions proceed as follows: First compute $\mathcal{U}_1^3 \subseteq \mathrm{RM}(k,2k)$ (see page 4) and then

$$\mathcal{V}_1^3 := \{P \in \mathrm{RM}(k,2k) \,:\, P \in \mathcal{U}_1^3,\ \overline{P} \in \mathcal{U}_1^3\}.$$

Sort the $P \in \mathcal{V}_1^3$ in sublists according to their degree k part $P^* \in \mathrm{RM}^*(k,2k)$. Choose $P_{00}, P_{01} \in \mathcal{V}_1^3$ from the same sublist, that is with $P_{00}^* = P_{01}^*$, and such that $\|P_{00}\| + \|P_{01}\| = 2^{2k}$. In this way one gets an allowed input to run `BasicGluing`. (Otherwise `BasicGluing` would return the empty set.) Our experiments have shown that `BasicGluing` gives about 2^{35} to 2^{38} bent functions on the average, after reduction of those, which are evidently equivalent.

Suppose you have already constructed an 8-bit bent function, for instance by hill-climbing techniques or a mathematical construction. Then you can use `BasicGluing` to compute its complete left parity-pattern equivalence class.

Since obviously (left) parity-pattern equivalence is *not* invariant under affine equivalence, the latter also works with an affine modification or the dual of a bent function that was obtained by a previous application of `BasicGluing`. In this way `BasicGluing` can also be iterated.

As addition to this paper we shall provide programs, data and results of experiments on the homepage `http://www.ruhr-uni-bochum.de/cits/` of our research group CITS (Cryptography and IT Security).

References

1. C. Carlet, H. Dobbertin, G. Leander, *Normal extensions of bent functions*, IEEE Transactions on Information Theory 50 (2004), 2880 – 2885.
2. C. Carlet, P. Guillot, *A characterization of binary bent functions,* J. Comb. Theory, Ser. A 76 (1996), 328 – 335.
3. C. Carlet, P. Guillot, *An alternative characterization of the bentness of binary functions, with uniqueness,* Des. Codes Cryptography 14 (1998), 133 – 140.
4. M. Daum, H. Dobbertin, G. Leander, *An algorithm for checking normality of Boolean functions,* Proceedings of the Workshop on Coding and Cryptography (WCC 2003), Versailles, France, March 2003, pp. 133 – 142.
5. J.F. Dillon, *Elementary Hadamard difference sets.* Ph.D. Thesis, University of Maryland, 1972.
6. J.F. Dillon, H. Dobbertin, *New cyclic difference sets with Singer parameters,* Finite Fields and Their Applications 10 (2004), 342 – 389.
7. H. Dobbertin, *Construction of bent functions and balanced Boolean functions with high nonlinearity*, Fast Software Encryption (Workshop on Cryptographic Algorithms, Leuven 1994), Lecture Notes in Computer Science, vol. 1008, Springer-Verlag 1995, 61 – 74.
8. H. Dobbertin, *Bent functions embedded into the recursive framework of $\mathbb{Z}$-bent functions,* preprint, January 2005 (submitted to IEEE Transactions on Information Theory).

9. H. Dobbertin, G. Leander, *A survey of some recent results on bent functions,* Proceedings of the Workshop on Sequences and Their Applications 2004 (SETA '04), Seoul, Korea, October 2004, Lecture Notes on Computer Science, Springer Verlag 2005, to appear.
10. J. Fuller: `http://www.isrc.qut.edu.au/people/fuller/`
11. G. Leander, *Normality of bent functions, monomial and binomial bent functions,* Ph.D. Thesis, Ruhr-University of Bochum, December 2004.
12. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography* CRC Press, Boca Raton, 1997.
13. B. Preneel, *Analysis and design of cryptographic hash functions,* Ph.D. thesis, KU Leuven (Belgium), February 1993.
14. O.S. Rothaus, *On "bent" functions,* Journal of Combinatorial Theory, Ser. A, 20 (1976), 300 – 305.
15. B. Schneier, *Applied Cryptography Second Edition: protocols, algoritms, and source code in C,* John Wiley & Sons, New York 1996.

Appendix

Making Normal Bent Functions Balanced

In the sequel it is more convenient to consider bent functions not as ± 1-valued, but as $\mathbb{F}_2$-valued (i.e. as special Boolean functions). The notion of a *normal* bent function was introduced in [7]. By definition such a bent function, say $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$, $n = 2k$, is constant on an affine subspace, say $A \subseteq \mathbb{F}_2^n$, of dimension k. A natural idea to make F balanced is to add (mod 2) a Boolean function, say ϕ, with support $\phi^{-1}(1) \subseteq A$ of size 2^{k-1} (see [7]). Obviously we can consider ϕ as a balanced Boolean function on $\mathbb{F}_2^k$ by identifying A with $\mathbb{F}_2^k$.

Given $H : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ we refer to its Walsh transform (see Section 2.2) in order to define the *Walsh radius*

$$\mathbf{R}_n(H) = \max\{|H^{\mathcal{W}}(a)| \,:\, a \in \mathbb{F}_2^n\}$$

of H as a measure for the linearity of H. By the definition of a bent function, $\mathbf{R}_n(F)$ attains the minimal possible value 2^k. For the balanced Boolean function $G = F + \phi$ one can show easily that $\mathbf{R}_n(G) = 2^k + \mathbf{R}_k(\phi)$ (see [7], Proposition 2). Thus we get the minimal $\mathbf{R}_n(G)$ for this approach precisely if the k-bit Boolean function ϕ is chosen with minimal $\mathbf{R}_k(\phi)$. This construction leads to the smallest linearity (i.e. highest non-linearity) which is known today.

Non-normal bent functions exist, which has been shown only recently for $n = 14$ (see [1], [4], [6]). Based on a collection of 8-bit bent functions representing all of them up to affine equivalence (see previous section), we would be able to decide whether there are non-normal 8-bit bent functions or not. (A fast normality test is described in [4].) Those which are normal can be made balanced in the above described way such that they have Walsh radius $2^4 + 8 = 24$. It remains a challenge to improve this value.