Evolving balanced Boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity.

James McLaughlin[*], John A. Clark

Abstract

Using simulated annealing, we derive several equivalence classes of balanced Boolean functions with optimum algebraic immunity, fast algebraic resistance, and maximum possible algebraic degree. For numbers n of input bits less than 16, these functions also possess superior nonlinearity to all Boolean functions so far obtained with said properties.

Keywords: Algebraic immunity, nonlinearity, metaheuristics, simulated annealing, stream ciphers, filter functions, algebraic attacks, fast algebraic attacks.

1 Introduction.

Combiner and filter functions for shift register-based stream ciphers need to satisfy various cryptographic criteria. They must be *balanced* [4], possess high *nonlinearity* to resist fast correlation attacks (see, among others, [7, 18, 2, 29]), and must possess high *algebraic degree*:

- to resist the RØnjom-Helleseth attack [19],
- to resist the Berlekamp-Massey attack [4],
- as a necessary but not sufficient condition to resist fast algebraic attacks [14].

(Furthermore, where n denotes the number of input bits of the Boolean function, an algebraic degree less than $\lceil \frac{n}{2} \rceil$ restricts the degree of so-called *algebraic immunity* that can be achieved.)

In the case of combiner functions, a high order of *correlation immunity* is also necessary [27, 28]. For filter functions, correlation immunity of order 1 is considered sufficient [7]. Unfortunately, the criteria of correlation immunity and algebraic degree are in conflict with one another, and the higher the correlation immunity of f, the lower the value that can be achieved for its degree - which increases the desirability of a model relying on filter instead of combiner functions.

(Correlation immunity of order 1 for a filter function is typically achieved by generating a function g which is as close as possible to the optimum for the other desirable criteria, and then using a shift register state bit x_{n+1} which is not input to g to define a function $f(x_1, \ldots, x_{n+1}) = g(x_1, \ldots, x_n) \oplus x_{n+1}$. It may be necessary to apply an affine transformation to the input bits of f and/or g. [4, 3])

Correlation immunity of order m for a balanced function is also referred to as order-m *resiliency.*

Until the early 21st century, these were the only criteria which a stream cipher's filtering/ combining function needed to satisfy. However, the discovery by Courtois et al. of algebraic attacks [15] and fast algebraic attacks [14] changed this, forcing:

[*]Corresponding author, `jmclaugh@cs.york.ac.uk`

1. An increase in the number of input bits needed by these functions (from "about 10" to "at least 13 and in practice much more, maybe 20" [7].) Where the shift register is a 256-bit LFSR, Braeken et al. [1] show that a balanced filter function on at least 14 input bits is needed to keep the time complexity of fast algebraic attacks below 2^{80}. The "RØnjom-Helleseth attack" [19] is a more recent form of algebraic attack, and it has been claimed [26] that filter functions need more than 30 input bits to resist it.

2. The introduction of two new criteria to quantify the resistance of Boolean functions f to these attacks. The first of these is the aforementioned *algebraic immunity* (AI) [9]. The second criterion, which involves "$(d_g, d_h)-$relations", is unnamed at present; we shall refer to it as *fast algebraic resistance.* A criterion known as *fast algebraic immunity* attempts to unify the two; however we consider it to be flawed.

In [17], algebraic attacks on "augmented functions" are discussed. For a given filter function f, shift register update polynomial L, and integer $m > 1$, the augmented function S_m is a vectorial Boolean function which takes as input the internal state of the shift register and is defined as the concatenation $(f(x)|f(L(x))|f(L^2(x))|\ldots|f(L^m(x)))$. Since the properties of augmented functions depend on both the shift register update function and the filter function, and since this paper only studies filter functions, it is beyond our remit to examine resistance to such attacks here.

Finding Boolean functions which combine optimal or near-optimal resistance to algebraic and fast algebraic attacks with the various other desirable properties has proven difficult. The need for high algebraic degree has led some researchers in this field to abandon the combiner model and focus entirely on filter functions; for which only a few suitable constructions have been found.

The first such was the Carlet-Feng construction [7]. This defines a class of balanced Boolean functions on n variables with algebraic degree $(n-1)$ and algebraic immunity $\lceil \frac{n}{2} \rceil$ (These are the optimal values of AI and degree for balanced f). A lower bound exists on their nonlinearity; this lower bound is not near-optimal but in practice the nonlinearity of the functions in this class was observed to exceed $2^{n-1} - 2^{\lfloor \frac{n}{2} \rfloor + 1}$ for $n \leq 11$. Furthermore, the fast algebraic resistance of the constructed functions was examined and, for functions with less than 10 variables, shown by experimentation to be optimal.

The Carlet-Feng class was independently rediscovered by Wang et al. [23], with a slight increase in the lower bound for nonlinearity. Functions with higher nonlinearity than previously achieved for $n = 8$, $n = 9$ and $n = 16$ were also presented.

Carlet demonstrated [6] that the two constructions were the same. He also stated that the functions could be implemented without needing to store a lookup table in memory; the computation of the Boolean function could be reduced to calculating a discrete logarithm, which, he stated, was feasible using the Pohlig-Hellman algorithm [20] when the function operated on 20 input bits or less. He stated, however, that this was the highest value of n used by such functions, and if we accept the statement in [26] that more than thirty are in fact needed, then it is not clear whether such an implementation is in fact viable for these larger functions.

The construction was improved upon in two later works coauthored by Carlet [8, 10]. The constructions in these papers obtained increased nonlinearity for $n = 10$, and results were obtained for much larger values of n than the previous papers had dealt with. Values of nonlinearity for newly-obtained Carlet-Feng functions corresponding to these higher values of n were given in [8]. In [10], the lower bounds on nonlinearity for the Carlet-Feng functions were tightened, and another balanced construction was presented which improved on the nonlinearity for some values of n but not others.

It is notable that the only apparent way to compute the functions in [8] without needing a lookup table also involves calculating a discrete logarithm, and this would appear also to be a necessary step in computing the functions in [10]. We reiterate that it is not clear whether this is viable for functions on more than 20, or indeed more than 30, input bits; thus motivating a search for alternatives.

In this paper, we apply simulated annealing to the problem of finding balanced Boolean functions with high nonlinearity and optimal AI, FAR and algebraic degree. This technique has achieved success in a similar context, when searching for combiner functions with high nonlinearity, low autocorrelation, and good tradeoffs between high degree and high order of correlation immunity [11, 12, 13]. In particular, it provided a search technique by means of which functions with profiles not hitherto obtained by any construction were found [11, 12].

(Autocorrelation was believed when first defined [30] to be a potential weakness that might lead to attacks on stream ciphers akin to differential cryptanalysis of block ciphers. As this has not subsequently proven to be the case, and as no eSTREAM finalist other than Dragon [16] was designed in a way that took autocorrelation into account, we will not focus on it here.)

Crucial to the method's success was a technique known as "two-stage optimisation". This technique would use one cost/fitness function for the simulated annealing, and would then hill-climb the results using a different cost function. The idea was that the first cost function would guide the annealing into a region of the search space (this space being the set of all Boolean functions of the pertinent size) in which candidate solutions (evolved Boolean functions) of above-average quality as defined by the second cost function were likely to exist. The second cost function would search this region for one of these high quality solutions, and return it. In Clark et al.'s experiments [11], this achieved far more favourable results using two-stage simulated annealing than any previous attempts to construct Boolean functions with metaheuristics.

This paper is structured as follows: Section 2 will provide precise definitions of the various cryptanalytic criteria and the tradeoffs between them, and describe the simulated annealing algorithm. In Section 3, we will describe the search landscape defined by the various properties, and justify our decision to represent the candidate functions using truth tables. We will also discuss the various cost functions used. In Section 4, we will compare the best Boolean functions found by our search to the best found by means of construction, and discuss avenues for future research.

2 Preliminaries

A balanced function is a Boolean function with an equal number of 1s and 0s in its truth table. We are not interested in filter or combiner functions that are not balanced, and the experiments were designed to ensure that these could not be evolved.

The algebraic normal form (ANF) of a Boolean function is its representation as a multivariate polynomial in which the variables are the values $(x_0, \ldots, x_{n-1})$ of the input bits. This representation is unique, and there exist mappings from truth table to ANF and vice-versa (both with matrix representations).

A linear function has algebraic normal form $a_0x_0 \oplus a_1x_1 \oplus \ldots \oplus a_{n-1}x_{n-1}$ $(a_i \in \{0, 1\})$.

The algebraic degree of a Boolean function is defined thus: Let the Hamming weight of a monomial be defined as the number of variables in it - so, for instance, x_1x_4 has weight 2. The algebraic degree of a monomial is defined as being equal to its weight, and the algebraic degree of a Boolean function is equal to the algebraic degree of the highest-weight monomial in its ANF.

The Walsh-Hadamard spectrum of a Boolean function $f \in B_n$ contains information on the correlation between f and the various n-bit linear functions. That is to say, where ω is an integer between 0 and $2^n - 1$, let $\omega_1\omega_2\ldots\omega_n$ be the bitstring representation of ω. Then entry ω in the Walsh spectrum is equal to:

$$\hat{F}(\omega) = \sum_{i=0}^{2^n-1} (-1)^{f(i)} \cdot (-1)^{\omega \cdot i}$$

Given a Walsh spectrum $\hat{F}$, the truth table of the original function f can be recovered from $\hat{F}$ using the Inverse Walsh Transform [12]. It is, therefore, a valid alternate representation.

The nonlinearity of f is defined as

$$2^{n-1} - \frac{\max_{\omega} |\hat{F}(\omega)|}{2}$$

Nonlinearity and algebraic degree are partially in conflict. For functions on an even number of variables, the highest nonlinearity possible is achieved only by *bent* functions, which have degree $\leq n/2$ and cannot be balanced.

The correlation immunity of f is the maximal value m such that $|\hat{F}(\omega)| = 0$ for all ω of Hamming weight $\leq m$ (that is, all ω with m ones or less in their base-2 representations).

The autocorrelation spectrum of f is defined thus. Let ω denote the bitstring representation of an integer between 0 and $2^n - 1$. Then

$$\hat{r}_f(\omega) = \sum_{i=0}^{2^n-1} (-1)^{f(i)}(-1)^{f(i \oplus \omega)}$$

and the autocorrelation spectrum is the sequence $(\hat{r}_f(0), \hat{r}_f(1), \ldots, \hat{r}_f(2^n - 1))$.

There is no inverse transformation allowing the truth table to be recovered from the autocorrelation spectrum.

The autocorrelation of f is the maximum absolute value, $\max_{i \neq 0} |\hat{r}_f(i)|$ in the autocorrelation spectrum.

Algebraic immunity (AI) is defined as the minimum degree of the nonzero functions g such that either $fg = 0$ or $(f \oplus 1)g = 0$. [9]. Such g are known, respectively, as *annihilators* of f or of $(f \oplus 1)$. For this reason, algebraic immunity is sometimes known as "annihilator immunity".

A corollary to Theorem 6.0.1 in [9] is that $AI(f) \leq \lceil \frac{n}{2} \rceil$.

Fast algebraic resistance (FAR) is defined thus:

For two values d_g, $(d_h > d_g)$, we say that a $(d_g, d_h)-$relation exists for f if two nonzero functions, g and h, exist such that $fg = h, deg(g) < deg(h), deg(g) = d_g$ and $deg(h) = d_h$.

The fast algebraic resistance of f is the minimum value of $(d_g + d_h)$ for all $(d_g, d_h)-$relations on f. Clearly, since $f{\cdot}1 = f$, this is upper-bounded by $deg(f)$. From our viewpoint, this means that any cost function dealing with fast algebraic resistance also deals to some extent with algebraic degree, since the FAR lower-bounds the degree.

For a given $(d_g + d_h)$, different values of (d_g, d_h) lead to different attack complexities. Various tradeoffs are discussed in [14] and [1]; however at present the cipher designer simply aims to achieve a $(d_g + d_h)$ too high for any (d_g, d_h) to lead to an attack, and preferably equal to the maximum value (for a balanced function) of $(d_g + d_h) = (n - 1)$.

It is shown in [1] that in any $(d_g, d_h)-$relation, d_h is greater than or equal to the algebraic immunity of f.

Fast algebraic immunity (FAI) is an attempt to unify the criteria of algebraic immunity and fast algebraic resistance. It is defined in [22] as:

$$FAI(f) = \min\{2{\cdot}AI(f), FAR(f)\}$$

We believe that this criterion is inadequate, and illustrate our reasons as follows: Let $f \in B_{13}$ be a Boolean function with fast algebraic resistance 12. Clearly, the optimal value of $AI(f)$ is 7. However, when $AI(f) = 6$, the value for fast algebraic immunity is the same as if it were 7, since in both cases $FAI(f) = 12$.

2.1 The simulated annealing algorithm

Simulated annealing [25] is a local-search based metaheuristic search algorithm, inspired by a technique used in metallurgy to eliminate defects in the crystalline structures in samples of metal.

The pseudocode below describes the workings of the algorithm. At the start, some initial candidate solution, S_0, usually chosen at random (so S_0 would be the truth table of a randomly chosen Boolean function on n input bits in this case), is input to the SA algorithm, along with the following parameters:

- The cost function C, which takes a solution candidate as input, and outputs a scalar value; the "cost". The cost function evaluates the candidate's desirability in terms of whatever criteria the experimenter is interested in (deciding on the relative weighting of the various criteria can be tricky!). The more desirable the candidate, the lower the cost should be.

- The initial value T_0 for the "temperature". The higher the temperature in the current iteration, the more likely the search algorithm is to accept a move which results in a candidate solution with higher cost than the current candidate (that is, to store said candidate solution as the "current candidate"). The temperature drops over time, causing the algorithm to accept fewer non-improving moves and hence to shift away from exploration and towards optimisation. Towards the end of the search, it is extremely rare for the algorithm to accept a non-improving move, and its behaviour is very close to that of a hill-climbing algorithm.

- In choosing the value of T_0, various sources state that it should be chosen so that a particular proportion of moves are accepted at temperature T_0. There is very little information or advice available as to what this proportion should be. In one of the earliest papers on simulated annealing [24] it is stated that any temperature leading to an initial acceptance rate of 80% or more will do; however our initial experiments indicated that this was far too high for most of the experiments in this paper. We eventually settled on an initial acceptance rate of 0.5 instead of 0.8.

  Having chosen the initial acceptance rate, the experimenter executes the annealing algorithm with various T_0 until a temperature is found that achieves a fraction close enough to this. We started with the temperature at 0.1, and repeatedly ran the algorithm, doubled the temperature, and re-ran the algorithm until an acceptance rate at least as high as that specified was obtained. Where T_a was the temperature at which this had been achieved, and $T_b = T_a/2$, we then used a binary-search-like algorithm to obtain a temperature between T_a and T_b that would result in an acceptance rate $\approx 50\%$.

- A value α; the "cooling factor", determining how far the temperature decreases at each iteration of the algorithm.

- An integer value: MAX_INNER_LOOPS, determining the number of moves that the local search algorithm can make at each temperature.

- The stopping criterion must also be specified. We used a MAX_OUTER_LOOPS value, indicating how many times the algorithm was to be allowed to reduce the temperature and continue searching before it stopped.

- We also specified a $MAX\_FROZEN\_OUTER\_LOOPS$ parameter. If the algorithm had, at any stage, executed this many outer loops without accepting a single move, it would be considered extremely unlikely to do anything other than remain completely stationary from then on, and would be instructed to terminate early.

A "move" is a transformation of the current solution candidate into another. Its precise definiton depends on the entity being annealed. Since we are evolving truth tables of balanced Boolean functions, we swap the positions of a zero and a one in the truth table. If we were not interested in preserving

Algorithm 1 Pseudocode for simulated annealing algorithm

```
S ← S_0
bestsol ← S_0
T ← T_0
ZERO_ACCEPT_LOOPS ← 0
for x ← 0, MAX_OUTER_LOOPS − 1 do
    ACCEPTS_IN_THIS_LOOP ← false
    for y ← 0, MAX_INNER_LOOPS − 1 do
        Choose some S_n in the 1-move neighbourhood of S.
        cost_diff ← C(S_n) − C(S)
        if cost_diff < 0 then
            S ← S_n
            ACCEPTS_IN_THIS_LOOP ← true
            if C(S_n) < C(bestsol) then
                bestsol ← S_n
            end if
        else
            u ← Rnd(0, 1)
            if u < exp(−cost_diff/T) then
                S ← S_n
                ACCEPTS_IN_THIS_LOOP ← true
            end if
        end if
    end for
    if ACCEPTS_IN_THIS_LOOP = false then
        ZERO_ACCEPT_LOOPS ← ZERO_ACCEPT_LOOPS + 1
        if ZERO_ACCEPT_LOOPS = MAX_FROZEN_OUTER_LOOPS then
                                                        ▷ Algorithm terminates early.
            return bestsol
        end if
    end if
    T ← T × α
end for
return bestsol
```

balanced functions, we might just flip a bit in the truth table at random. In general, there should be reason to believe that there are bounds on the extent to which the cost can change when a move is made. The "1-move neighbourhood" of solution candidate S is the set of candidate solutions that can be obtained from S by making one move precisely.

2.2 The hill-climbing algorithm

The below pseudocode describes the hill-climbing algorithm used. The value of MAX_INNER_LOOPS is identical to that used by the annealing algorithm.

Algorithm 2 Pseudocode for hill-climbing algorithm

```
S ← initial candidate (output of the simulated annealing algorithm in this case.)
repeat
    S_best ← S
    ACCEPTS_IN_THIS_LOOP ← false
    for x ← 0, MAX_INNER_LOOPS do
        S_x ← some randomly chosen member of the 1-move neighbourhood of S.
        cost_diff ← C(S_x) − C(S)
        if cost_diff < 0 then
            ACCEPTS_IN_THIS_LOOP ← true
            S_best ← S_x
        end if
    end for
    if ACCEPTS_IN_THIS_LOOP = true then
        S ← S_best
    end if
until ACCEPTS_IN_THIS_LOOP = false
return S
```

3 The experiments

3.1 Representing candidates as truth tables

So far, we have referred to three possible representations of Boolean functions:

- Their truth tables.
- Their algebraic normal forms.
- Their Walsh-Hadamard spectra.

An additional representation in the form of a univariate polynomial also exists, in which we treat the value of the n input bits as a single value in $GF(2^n)$. [4, 5].

We have decided to focus on the truth tables, with the positions of a 1 and a 0 being swapped as the move function. Not only does this move function preserve balancedness, but several smoothnesses in the search landscape exist for the truth table representation, as we shall demonstrate below:

Lemma 3.1. *If one element of the truth table of a Boolean function f with more than one input bit changes value, the algebraic immunity of f changes by at most 1.*

Proof. Let x_α be the input value for which the output value flips. Let f be the original function, f' the function after the truth table is altered that differs from f only in the value of $f(x_\alpha)$. Let g be an annihilator of either f or $(f \oplus 1)$ of degree $AI(f)$.

$f'(x) = f(x) \oplus \delta(x_\alpha)$, where $\delta(x_\alpha)$ is the sum of all supermonoms of x_α. (supermonoms being x_α and all multiples thereof, i.e. any monoms containing all the "on" variables of x_α.)

That is, $\delta(x_\alpha) = x_\alpha(1 \oplus x_b \oplus x_c \oplus x_b x_c \oplus \ldots) = x_\alpha(1 \oplus x_b)(1 \oplus x_c)\ldots$ where x_b, x_c, etc are input bits not appearing in the monom x_α. Let us refer to these as "not-in-common inbits", and the others as "in-common inbits". For example, $\delta(10001) = x_1(1 \oplus x_2)(1 \oplus x_3)(1 \oplus x_4)x_5$, where x_1 and x_5 are the in-common inbits, and x_2, x_3, x_4 are the not-in-common inbits.

$\delta(x_\alpha) \cdot$ (one of the not-in-common inbits) $= 0$. (Note that if x_α is the maximum-weight-all-ones input, no not-in-common inbits exist). Furthermore, $\delta(x_\alpha) \cdot (1 \oplus$ any in-common inbit$) = 0$.

If $x_b g = 0$ for all not-in-common x_b, g must be a multiple of $(1 \oplus x_b)(1 \oplus x_c)\ldots$, with algebraic degree $\geq (n - HW(x_\alpha))$.

If $(1 \oplus x_i)g = 0$ for all in-common x_i, g must be a multiple of $(x_i \cdot x_j \cdot \ldots) = x_\alpha$, with algebraic degree $\geq HW(x_\alpha)$.

If $x_b g = 0$ for all not-in-common x_b and $(1 \oplus x_i)g = 0$ for all in-common x_i, g must be $x_\alpha(1 \oplus x_b \oplus x_c \oplus x_b x_c \oplus \ldots)$ with algebraic degree n. Since g has algebraic degree $AI(f)$, which is bounded above by $\lceil \frac{n}{2} \rceil$, this is only possible if $n = 1$. So there exists at least one x_b or $(1 \oplus x_i)$ such that the product of it and g is nonzero, and such that the product of it and $\delta(x_\alpha)$ is zero. Call it z.

(In fact, since g must have algebraic degree $\leq \lceil \frac{n}{2} \rceil$, there exist at least $\lfloor \frac{n}{2} \rfloor$ such candidates for z; however we only need one of them to complete the proof.)

Either g is an annihilator of f, or an annihilator of $(1 \oplus f)$.

If the former: $fg = 0$. Then $zgf' = zg(f \oplus \delta) = zgf \oplus zg\delta$. $gf = 0$, so this $= zg\delta = z\delta g = 0$. Hence zg annihilates f', and $AI(f') \leq deg(zg) \leq AI(f) + 1$.

If the latter: $(1 \oplus f)g = 0$. $zg(1 \oplus f') = zg(1 \oplus f \oplus \delta) = zg(1 \oplus f) \oplus zg\delta = 0z \oplus z\delta g = 0$. Hence zg annihilates $(1 \oplus f')$, and $AI(f') \leq deg(zg) \leq AI(f) + 1$.

We have shown that $AI(f') \leq AI(f) + 1$. It is trivial to swap f' and f and repeat the above procedure to show that $AI(f) \leq AI(f') + 1$. Hence $|AI(f) - AI(f')| \leq 1$. □

Lemma 3.2. *If one of the 0s in the truth table of a Boolean function f on more than one input bit changes to a 1, and if one of the 1s in said truth table simultaneously changes to a 0, the algebraic immunity of the resultant Boolean function f' differs from $AI(f)$ by at most 1.*

Proof. Since this represents two changes to the truth table of f, we know from the above result that $|AI(f) - AI(f')| \leq 2$. Now, let the first change be the one turning a 1 into a 0 in the truth table, and let the Boolean function resulting from this change be denoted f_2. Clearly any annihilators of f are annihilators of f_2, so $AI(f_2) \leq AI(f)$.

The second change, a 0 to a 1, changes f_2 into f'. From result 10 above, we know that $AI(f') \leq (AI(f_2) + 1) \leq (AI(f) + 1)$. By similar reasoning, we can show that $AI(f) \leq (AI(f') + 1)$. Hence $|AI(f) - AI(f')| \leq 1$. □

Lemma 3.3. *Let $DP(f)$ be the minimum value of $d_g + d_h$ such that $f \in B_n$ $(n > 1)$ satisfies a $(d_g, d_h)-$relation. Let f' be a Boolean function differing from f in precisely one truth table position, corresponding to input value x_α.*

Then $|DP(f') - DP(f)| \leq 2$.

Proof. As noted in Lemma 3.1 above, $f' = f \oplus \delta(x_\alpha)$, where $\delta(x_\alpha)$, for all input bits $x_b, x_c, \ldots$ that are not submonoms of x_α, is equal to $x_\alpha(1 \oplus x_b)(1 \oplus x_c)\ldots$

Let g with degree d_g and h with degree d_h be two functions such that a $(d_g, d_h)-$relation exists for f. For a valid $(d_g, d_h)-$relation, since $d_h \geq AI(f)$, $d_g \leq \lfloor \frac{n}{2} \rfloor$.

If $x_b g = 0$ for any input bit x_b that is not a submonom of x_α, g must be a multiple of $(1 \oplus x_b)$.

If $(1 \oplus x_i)g = 0$ for any input bit x_i that is a submonom of x_α, g must be a multiple of x_i.

It follows that there must exist at least $\lceil \frac{n}{2} \rceil$ polynomials $p = x_b$ or $(1 \oplus x_i)$ of the form described above such that pg is a nonzero function, otherwise g would have algebraic degree higher than $\lfloor \frac{n}{2} \rfloor$. Let us choose one, and denote it z.

$z \cdot \delta(x_\alpha)$ must equal zero, since if z is one of the x_b, we have $z \cdot \delta = x_\alpha x_b(1 \oplus x_b) \ldots = x_\alpha \cdot 0 \cdot \ldots = 0$, and if z is one of the $(1 \oplus x_i)$, $z \cdot x_\alpha = (1 \oplus x_i)x_i x_j \ldots = 0$ and hence $z \cdot \delta = 0 \cdot (1 \oplus x_b)(1 \oplus x_c) \ldots = 0$.

Now, $zgf' = zg(f \oplus \delta) = zgf \oplus zg\delta = zh \oplus (gz\delta = 0) = zh$. $deg(zg) \leq deg(g) + 1 = (d_g + 1)$, and $deg(zh) \leq deg(h) + 1 = (d_h + 1)$. We see that $DP(f')$ cannot exceed $(DP(f) + 2)$ since $(zg)f' = zh$ with $deg(zg) + deg(zh) \leq (d_g + 1) + (d_h + 1) = (d_g + d_h + 2)$.

We can similarly show that $DP(f) \leq (DP(f') + 2)$, giving us the result that $|DP(f') - DP(f)| \leq 2$. $\square$

Corollary 3.4. *Let $DP(f)$ be the minimum value of $d_g + d_h$ such that $f \in B_n$ $(n > 1)$ satisfies a $(d_g, d_h)-$relation. Let f' be a Boolean function differing from f in precisely two truth table positions. Then $|DP(f') - DP(f)| \leq 4$.*

Lemma 3.5. *Let f' be a Boolean function differing from f in precisely one truth table position. Then all values in the Walsh-Hadamard spectrum of f' differ from their corresponding values in the spectrum of f by ± 2.*

Proof. Consider that, as stated earlier, entry ω in the spectrum is equal to:

$$\hat{F}(\omega) = \sum_{i=0}^{2^n-1} (-1)^{f(i)} \cdot (-1)^{\omega \cdot i}$$

Since only one value of $f(i)$ changes, only one value of $(-1)^{f(i)} \cdot (-1)^{\omega \cdot i}$ changes, from either $(-1) \cdot (-1)^{\omega \cdot i}$ to $1 \cdot (-1)^{\omega \cdot i}$, or vice versa. In any case, the magnitude of the change is $2 \cdot (-1)^{\omega \cdot i}$, i.e. 2. $\square$

Corollary 3.6. *Let f' be a Boolean function obtained by swapping two differing values in f's truth table. Then all values in the Walsh-Hadamard spectrum of f' differ from their corresponding values in the spectrum of f by $+4, 0,$ or -4.*

Since, as stated earlier, all Walsh-Hadamard spectrum entries for a balanced function are multiples of 4, we have:

Corollary 3.7. *Let f' be a balanced Boolean function obtained by swapping two differing values in f's truth table. Let $MW(f)$ denote the maximal absolute value in the Walsh-Hadamard spectrum of f; that is $MW(f) = \max_\omega |\hat{F}(\omega)|$. Then $MW(f') = MW(f)$ or $MW(f) \pm 4$. In any case, the difference is at most 4. Since nonlinearity is defined as $2^{n-1} - \frac{\max_\omega |\hat{F}(\omega)|}{2}$, we see that the nonlinearities of f and f' differ by at most 2.*

Early experiments on evolving truth tables with 8 or 9 input bits showed that the optimal values for AI and FAR would always be found within two outer loops, even with only 100 inner loops. For this reason, we felt confident in focusing solely on truth tables, and in adding nonlinearity to the cost function, thus covering all the relevant criteria for a filter function in [7].

3.2 Choosing a cost function

In [11], cost functions of this form were experimented with for various values of R and X:

$$cost(f) = \sum_{\omega=0}^{2^n-1} ||\hat{F}_f(\omega)| - X|^R$$

To be more precise, the value $R = 3.0$ was preferred, with 2.0 and 2.5 also experimented with. In devising the part of the cost function that would deal with nonlinearity, however, we opted to utilise $R = 4.0$ (and to divide this part of the cost function by a scalar factor dependent on n), for various reasons:

1. According to Parseval's Theorem, the sum of squares of the entries in a valid Walsh spectrum is constant. It therefore seemed unlikely that exponent 2 would be of much help. Furthermore, we had observed that high-quality solutions tended to have higher costs as defined by the pair $(X = 0, R = 1)$; and although attempts to base a cost function on this observation proved ineffective, this was nonetheless evidence that R would have to exceed 2.

2. In [21], it is shown that applying a matrix transformation to the difference distribution table (DDT) of a vectorial Boolean function yields a table containing the autocorrelation spectra of all linear combinations of the co-ordinate functions, and that applying a further matrix transformation to this yields a table containing the squared entries of the Walsh-Hadamard spectra for these functions. Previous research into evolving substitution boxes had utilised the sum of squares of DDT entries after $(R = 2.0, X = 0)$ for this table turned out to be especially efficient and high-performing, and this suggested that the sum of the squares of the squares of the Walsh entries might be analogous with the sum of the squares of the DDT entries for a vectorial Boolean function in some way.

3. Consistent with the preceding point, dividing the variance of the entries in the "squared Walsh spectra" table by a particular value exponential in n yielded the variance of the DDT; and we had been able to prove that the cost as defined by the DDT variance changed by the same amount as the $(R = 2.0, X = 0)$ DDT cost function whenever a move was made.

4. During initial experimentation, dividing the sum of fourth powers by 2^{n+5} to define a cost was observed to create a situation where each move changed the cost by 3.0 or some integer multiple thereof, raising confidence in the uniform smoothness of the search landscape.

5. Furthermore, when combined with algebraic and fast algebraic qualities, this cost function obtained Boolean functions with comparable algebraic characteristics and superior nonlinearity to a cost function in which $(2^{n-1} - NL)$ - (the number of occurrences of the maximal absolute value in the Walsh spectrum) was used as the nonlinearity component.

The overall cost function, therefore, derived an initial cost using the Walsh spectrum in this fashion, and then subtracted $2 * AI(f) + FAR(f)$ from it to obtain the overall cost. This meant that a one point improvement in the nonlinearity portion of the cost function would subtract 3 from the cost, compared to 1 or 2 for the others. We felt that this was justified to reflect the difficulty of obtaining functions with optimal nonlinearity through simulated annealing compared to functions with optimal algebraic characteristics. In experiments, it was observed that this would allow the cost function to move through candidates with suboptimal algebraic characteristics that might otherwise block off promising search avenues. The additional weight given to AI compared to FAR simply reflected its more restricted range of values.

As stated above, we used a different cost function for hill-climbing. This, again, subtracted $2 * AI(f) + FAR(f)$ from the overall cost, but had a simpler nonlinear component of $(2^{n-1} - NL) - 2/freq(max_f(|\hat{F}(\omega)|))$. That is, we divided 2 by the frequency with which the maximal absolute value in the Walsh spectrum occurred, and subtracted this from $(2^{n-1} - NL)$. On this occasion, however, we reduced the weighting given to the nonlinearity - slightly suboptimal nonlinearity was acceptable, anything less than optimal AI and FAR in the final product was not.

We used 500,000 inner loops for problems of size 9 or higher, and 20,000 for size 8 or less. We used 100 outer loops and 50 trials per problem size, cooling factor 0.97, and initial acceptance rate 0.5. Algebraic immunity was calculated according to Algorithm 2 in [9], and fast algebraic resistance according to the algorithm of [1].

3.2.1 The next cost function

For up to 11 input bits, this was acceptably efficient. The following table compares our results to the previously-known best in the literature ([7, 8, 10, 23]):

n	Previous best (NL, AI, FAR)	(NL, AI, FAR) achieved by annealing
6	(24, 3, 5)	(26, 3, 5)
7	(54, 4, 6)	(56, 4, 6)
8	(114, 4, 7)	(116, 4, 7)
9	(236, 5, 8)	(238, 5, 8)
10	(484, 5, 9)	(486, 5, 9)
11	(980, 6, 10)	(986, 6, 10)

Table 1: Comparisons of previously-known Boolean functions with first set of annealed functions for $n \leq 11$

However, both in memory and time, the cost of calculating algebraic immunity and fast algebraic resistance is exponential. Despite the optimisations we were able to make by taking into account the lemmas in Section 3, both complexities were still exponential, and for 12 input bits the algorithm remained stuck in its first hill-climb for several days without returning a result.

Since most of the results that had been achieved still had optimal algebraic characteristics, and since the speed with which these were achieved suggested that functions with optimal (AI, FAR) were plentiful, we decided to run a new set of experiments in which we would remove all parts of the cost functions that did not focus on nonlinearity. We would evaluate (AI, FAR) at the end of the algorithm, and hope that at least some of the annealed functions were optimal in terms of these criteria.

The parameters remained unchanged up to $n = 15$. For $n = 16$, the increased complexity meant that we reduced the number of inner loops to 200,000; however we later raised this to 400,000 (and later 1,000,000, after discovering the substantial gulf between constructed and annealed results at this size.) We did not go as far as $n = 17$; and note that to do so would require at least $4GB$ of memory for the fast algebraic resistance calculations and the precomputed tables used in the nonlinearity sections of the cost function; this quantity increases approximately fourfold when n is increased by 1.

We also reran the experiments for $n = 9$, $n = 10$ and $n = 11$ using this approach, hoping either to improve on our best results or to increase the number of distinct affine equivalence classes possessing the same set of optimal criteria. For $n = 9$, 8% of functions achieved nonlinearity 240, but all of these had only $FAR = 7$. 32% of the functions for $n = 10$ achieved nonlinearity 488, again at the cost of a slightly suboptimal $FAR = 8$. The new experiment for $n = 11$, after hill-climbing, found functions with nonlinearity 988 on every run, but none of these possessed the necessary $FAR > 9$. What was more, as well as $FAR(f) = (n - 2)$, these functions also had suboptimal algebraic degree $(n - 2)$.

Comparing this to the results for higher sizes; for $n = 12$ 58% of the hill-climbed functions had nonlinearity 1996, but all of these had suboptimal degree and FAR of 10. For $n = 13$ 60% of the hill-climbed functions had nonlinearity 4020, but all of these had FAR 11 and degree 11.

n	Best (NL, AI, FAR) achieved with nonlinearity-only cost function.
12	(1994, 6, 11)
13	(4018, 7, 12)
14	(8082, 7, 13)
15	(16222, 8, 14)
16	(32536, 8, 15)

Table 2: Annealed Boolean functions for $12 \leq n \leq 16$ before incorporation of algebraic degree into the cost function.

3.3 Adding algebraic degree to the cost function.

Since all the functions we had found with nonlinearity in excess of those in Table 2 had suboptimal algebraic degree, we altered the hill-climb cost function to heavily penalise algebraic degree $< (n-1)$, and reran the previous experiments with increased numbers of inner loops (going as far as 32,000,000 for $n = 14$).

The results of this were mixed. For $n \leq 13$, the higher values for nonlinearity observed previously simply did not occur. For $n = 14$, four Boolean functions with nonlinearity 8084 and the desired $(AI,\ FAR)$ value were obtained; all the other functions at this size had nonlinearity 8082. For $n = 16$ (with up to 3,000,000 inner loops) one function with nonlinearity 32540 was found, followed by a total of three more when the number of inner loops was increased to 6,000,000 and then 12,000,000. No functions with higher nonlinearity at this size have yet been obtained through annealing; however all functions with this or lower nonlinearity have so far possessed optimal $(AI,\ FAR)$, suggesting that experiments over a longer time period with more inner loops may obtain higher nonlinearities still.

For $n = 15$ (with up to 6,000,000 inner loops), however, most annealed functions had only suboptimal AI of 7, despite their optimal degree and FAR. Over several experiment runs, five functions with $(NL\ 16226,\ AI\ 8,\ FAR\ 14)$ were nevertheless found, but the reduced AI of most of the results suggested that very few Boolean functions with high nonlinearity possess optimal algebraic degree, algebraic immunity and fast algebraic resistance at this size, and that increasing the computational resources devoted to this problem with the current cost function might primarily have the effect of reducing the number of functions with $AI = 8$. This is consistent with the fact that the only function with $NL = 16228$ we obtained also had $AI = 7$. It should be noted that the evaluation of a Boolean function's algebraic immunity is much slower than the evaluation of its algebraic degree, and hence reintroducing this into the cost function would significantly increase the time required to anneal a single Boolean function, or force a reduction in the number of inner loops (and hence the achievable nonlinearity). This may even result in functions with optimal algebraic degree $(n-1)$ but $FAR \leq (n-2)$.

n	Previous best (NL, AI, FAR)	Best (NL, AI, FAR) achieved with annealing and original hill-climber.
12	(1988, 6, 11)	(1994, 6, 11)
13	(3988, 7, 12)	(4018, 7, 12)
14	(8072, 7, 13)	(8084, 7, 13)
15	(16212, 8, 14)	(16226, 8, 14)
16	(32556, 8, 15)	(32540, 8, 15)

Table 3: Comparisons of the best existing Boolean functions with the last annealing results for the original hill-climbing algorithm $(12 \leq n \leq 16)$

3.4 A more exhaustive hill-climbing algorithm.

The original hill-climbing algorithm (Algorithm 2) does not evaluate the cost of every member of the 1-move neighbourhood of the current candidate. This was a conscious design decision, made due to the high time complexity of the AI/FAI algorithms that were involved initially. However, since these were no longer incorporated into any cost function, this was no longer a factor. Despite the fact that the size of the 1-move neighbourhood increases exponentially with n, we decided that it was worth experimenting with a more exhaustive, deterministic, hill-climbing algorithm (see below pseudocode for Algorithm 3).

Using 500,000 inner loops for the simulated annealing algorithm, we obtained our first $(NL = 988,\ AI = 6,\ FAR = 10)$ functions for $n = 11$, but did not obtain any improvement for $n \leq 10$. For $n = 9$, we increased the number of inner loops to 2,000,000 and later 8,000,000 but still did not obtain

Algorithm 3 Pseudocode for the second hill-climbing algorithm

```
S_0 denotes initial candidate
S ← S_0
repeat
    S_best ← S
    ACCEPTS_IN_THIS_LOOP ← false
    for x ← 0, sizeof(1-move neighbourhood of S) do
        S_x denotes the xth member of the 1-move neighbourhood of S.
        cost_diff ← C(S_x) − C(S_best)
        if cost_diff < 0 then
            ACCEPTS_IN_THIS_LOOP ← true
            S_best ← S_x
        end if
    end for
    if ACCEPTS_IN_THIS_LOOP = true then
        S ← S_best
    end if
until ACCEPTS_IN_THIS_LOOP = false
return S
```

$NL > 238$. For $n = 10$, using $2,000,000$ and then $4,000,000$ inner loops, 2% of our obtained results had $(NL = 488,\ AI = 5,\ FAR = 9)$.

For $n = 12$, again using $2,000,000$ followed by $4,000,000$ inner loops, we obtained several $(NL = 1996,\ AI = 6,\ FAR = 11)$ functions. For $n = 13$, we obtained several functions with $NL = 4020$ and $FAR = 12$. Most of these had $AI = 6$, but we did still obtain several with $AI = 7$. For $n = 14$, we equalled but did not improve on the quality of our best previous results, and it should be noted that the exponential increase in time complexity is such that the full 50 trials have not yet been completed after several months of computation. For $n = 15$ and $n = 16$, the time complexity is such that for neither of these sizes has the hill-climber finished evolving the first candidate one month after the completion of the annealing phase (which took approximately two days in the first case, five in the second.)

n	Previous best (NL, AI, FAR)	Best (NL, AI, FAR) achieved with annealing.
6	(24, 3, 5)	(26, 3, 5)
7	(54, 4, 6)	(56, 4, 6)
8	(114, 4, 7)	(116, 4, 7)
9	(236, 5, 8)	(238, 5, 8)
10	(484, 5, 9)	(488, 5, 9)
11	(980, 6, 10)	(988, 6, 10)
12	(1988, 6, 11)	(1996, 6, 11)
13	(3988, 7, 12)	(4020, 7, 12)
14	(8072, 7, 13)	(8084, 7, 13)
15	(16212, 8, 14)	(16226, 8, 14)
16	(32556, 8, 15)	(32540, 8, 15)

Table 4: Comparisons of the best existing Boolean functions with the final annealing results

3.5 Equivalence classes.

The histograms of the values in the Walsh spectra of the evolved functions differed, even for functions with the same (NL, AI, FAR). Since these frequency histograms are affine invariant, it was clear that several different affine equivalence classes of functions existed with these properties.

(n, NL, AI, FAR)	Number of distinct equivalence classes identified
(6, 26, 3, 5)	2
(7, 56, 4, 6)	2
(8, 116, 4, 7)	20
(9, 238, 5, 8)	62
(10, 488, 5, 9)	2
(11, 988, 6, 10)	6
(12, 1996, 6, 11)	23
(13, 4020, 7, 12)	33
(14, 8084, 7, 13)	7
(15, 16226, 8, 14)	5
(16, 32540, 8, 15)	4

Table 5: Number of non-equivalent functions so far with the best (NL, AI, FAR) obtained through annealing.

4 Conclusions and avenues for future research

In this paper, we have established via theoretical analysis that the search landscape defined by the use of truth table flips as a move function is extremely promising with respect to the search for Boolean functions with cryptographically-relevant properties. In addition to the existing results in this area for nonlinearity and autocorrelation, we have demonstrated the existence of smooth search landscapes for algebraic immunity and fast algebraic resistance, and exploited these in a local-optimisation based metaheuristic, finding Boolean functions with superior properties to the best theoretical constructions for their corresponding values of n.

Truth tables for some of the evolved Boolean functions are presented in the appendix, and any researchers wishing to investigate the full set of evolved truth tables are invited to email the authors.

It would be interesting to see if such a search landscape is also defined for properties such as transparency order which are relevant to side-channel attacks, or indeed for any other properties of Boolean functions that are cryptographically relevant. Or, for that matter, relevant in areas of computer science other than cryptology!

The key issue with the new functions is one of implementation. The Carlet-Feng functions can be implemented using the Pohlig-Hellman algorithm [20] for up to 20 bits (and possibly more) without needing the truth-table to be stored in memory; and for purposes of efficiency, some fast means to calculate one of the new functions without needing to store a large lookup table in memory or requiring a circuit with an overly large number of gates is required for them to be of practical use. Algebraic immunity is not invariant in the case of affine transformations on the outputs, but is invariant under transforms on the function inputs, and all other relevant properties are affine invariant [1]. Hence, a potentially profitable avenue might be to apply various affine transformations to the function inputs and to experiment with the results to find out if any of them are of the types described in [7, 8, 10]. Alternatively, the univariate representations of the affine equivalence subclasses thus defined could be examined for functions with suitably sparse univariate forms.

Perhaps the best way to view this work might be as an existence proof. Boolean functions satisfying all the required properties for use as nonlinear filter functions, and with nonlinearity higher than that achieved by existing constructions, have been shown to exist. Now the question is whether any of them can be shown to be part of an infinite class of Boolean functions with these properties (and, ideally, some more efficient means of implementation). The exponential complexity of the algebraic immunity and fast algebraic resistance algorithms renders the use of the current annealing approach to find such functions for higher values of n increasingly impractical.

References

[1] A. Braeken, J. Lano, and B. Preneel. Evaluating the resistance of stream ciphers with linear feedback against fast algebraic attacks. In L.M. Batten and R. Safavi-Naini, editors, *Proceedings of the Eleventh Australasian Conference on Information Security and Privacy (ACISP 2006)*, volume 4058 of *Lecture Notes in Computer Science*, pages 40–51. Springer, July 2006.

[2] A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In B. Preneel, editor, *Advances in Cryptology - Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. IACR, Springer, May 2000.

[3] C. Carlet. Private communication.

[4] C. Carlet. Boolean functions for cryptography and error-correcting codes. In Y. Crama and P. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge University Press, 2010. The chapter is downloadable from `http://www.math.univ-paris13.fr/~carlet/chap-fcts-Bool-corr.pdf`.

[5] C. Carlet. Vectorial Boolean functions for cryptography. In Y. Crama and P. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge University Press, 2010. The chapter is downloadable from `http://www.math.univ-paris13.fr/~carlet/chap-vectorial-fcts-corr.pdf`.

[6] C. Carlet. Comments on "Constructions of cryptographically significant Boolean functions using primitive polynomials". *IEEE Transactions on Information Theory*, 57(7):4852–4853, July 2011.

[7] C. Carlet and K. Feng. An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity. In J. Pieprzyk, editor, *Advances in Cryptology - Asiacrypt 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 425–440. IACR, Springer, December 2008.

[8] C. Carlet, L. Hu, J. Shan, and X. Zeng. More balanced Boolean functions with optimal algebraic immunity and good nonlinearity and resistance to fast algebraic attacks. *IEEE Transactions on Information Theory*, 57(9):6310–6320, September 2011.

[9] C. Carlet, W. Meier, and E. Pasalic. Algebraic attacks and decomposition of Boolean functions. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. IACR, Springer, May 2004.

[10] C. Carlet, D. Tang, and X. Tang. Highly nonlinear Boolean functions with optimal algebraic immunity and good behavior against fast algebraic attacks. Cryptology ePrint Archive, Report 2011/366. July 2011. `http://eprint.iacr.org/2011/366`.

[11] J.A. Clark, J. Jacob, S. Stepney, S. Maitra, and W. Millan. Evolving Boolean functions satisfying multiple criteria. In A. Menezes and P. Sarkar, editors, *Progress in Cryptology - Indocrypt 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 246–259. Springer, December 2002.

[12] J.A. Clark, J.L. Jacob, S. Maitra, and P. Stanica. Almost Boolean functions: The design of Boolean functions by spectral inversion. *Computational Intelligence*, 20(3):450–462, August 2004.

[13] J.A. Clark, J.L. Jacob, and S. Stepney. Searching for cost functions. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC2004)*, pages 1517–1524. IEEE, June 2004. Volume 2.

[14] N.T. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *Advances in Cryptology - Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. IACR, Springer, August 2003.

[15] N.T. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Advances in Cryptology - Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. IACR, Springer, May 2003.

[16] E. Dawson, M. Henricksen, and L. Simpson. *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, chapter 3, pages 20–38. Springer, 2008. The relevant chapter is entitled "The Dragon Stream Cipher: Design, Analysis, and Implementation Issues".

[17] S. Fischer and W. Meier. Algebraic immunity of s-boxes and augmented functions. In A. Biryukov, editor, *Proceedings of the Fourteenth International Workshop on Fast Software Encryption (FSE 2007)*, volume 4593 of *Lecture Notes in Computer Science*, pages 366–381. IACR, Springer, March 2007.

[18] R. Forré. A fast correlation attack on nonlinearly feedforward filtered shift-register sequences. In J-J Quisquater and J. Vandewalle, editors, *Advances in Cryptology - Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 586–595. IACR, Springer, April 1989.

[19] T. Helleseth and S. RØnjom. A new attack on the filter generator. *IEEE Transactions on Information Theory*, 53(5):1752–1758, May 2007.

[20] M. Hellman and S. Pohlig. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, January 1978.

[21] H. Imai, X-M. Zhang, and Y. Zheng. Relating differential distribution tables to other properties of substitution boxes. *Designs, Codes and Cryptography*, 19(1):45–63, January 2000.

[22] T. Johansson and Q. Wang. A note on fast algebraic attacks and higher order nonlinearities. In Xuejia Lai, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 6th International Conference (Inscrypt 2010)*, volume 6584 of *Lecture Notes in Computer Science*, pages 404–414. Springer, October 2010.

[23] H. Kan, J. Peng, Q. Wang, and X. Xue. Constructions of cryptographically significant Boolean functions using primitive polynomials. *IEEE Transactions on Information Theory*, 56(6):3048–3053, June 2010.

[24] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986, March 1984.

[25] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[26] E. Pasalic. On cryptographically significant mappings over $GF(2^n)$. In Joachim von zur Gathen, José Luis Imaña, and Çetin Kaya Koç, editors, *Proceedings of the Second International Workshop on Arithmetic of Finite Fields (WAIFI 2008)*, volume 5130 of *Lecture Notes in Computer Science*, pages 189–204. Springer, July 2008.

[27] T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–780, September 1984.

[28] T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34(1):81–85, January 1985.

[29] O. Staffelbach and W. Meier. Fast correlation attacks on stream ciphers (extended abstract). In C.G. Günther, editor, *Advances in Cryptology - Eurocrypt '88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. IACR, Springer, May 1988.

[30] X-M. Zhang and Y. Zheng. GAC - the criterion for global avalanche characteristics of cryptographic functions. *Journal of Universal Computer Science*, 1(5):320–337, May 1995.

Appendices

We present, in hexadecimal format, some of the truth tables of the evolved functions. Any researchers who would like the full set of evolved functions with nonlinearities as shown in Tables 1 and 2 are welcome to contact the authors directly (`jmclaugh@cs.york.ac.uk`).

$n = 6$**:** The following two truth tables are representatives of the discovered equivalence classes:

`3502 8c3e f607 f571`

and

`385d b3b3 6f90 58a1`

$n = 7$: The representative truth tables are:

`094f ddf3 299f 8b6c 15a4 42c7 5185 edc8`

and

`58ff 2d3a d029 4127 1958 f4d9 d436 3b53`

$n = 8$:

`fbf2 6023 2e62 c9c7 aec4 d8b6 e4b2 ade5 616e 3c45 03f3 08d5 5baf e9aa 9609 6031`

possesses fewer 24s (the maximal absolute value) in its absolute Walsh spectrum than any other annealed function of this size.

$n = 9$:

```
3011 9f10 b4f0 fce0 ebf1 4a57 fe9c 4d17 663b 8911 321d d1c8 8225 c40c a0bc 5c3b
7b91 9d70 a487 67b5 6c30 28ed c3bf 7e24 4b94 f79f 1175 96c7 1b8a fb33 9574 2d52
```

has the fewest 36s in its absolute spectrum.

$n = 10$:

```
70ea 61ed 92c1 e717 c837 2f1c 83bf 8b97 32ac d5e6 d054 df57 9468 934d 03bc fa0e
0492 8550 d23b 32aa dd61 7ec6 aea6 4189 fa28 1b82 1e20 e2da a2d9 d184 4ad2 a778
bb66 b463 b335 c686 df3e 55db 6f25 f439 1e71 b998 1276 8bc8 a770 ba13 10f0 2ca5
1181 9acf e2d3 d6ee e730 0dd0 19ef 7050 e9f8 9330 a949 142f ad4c efd9 af73 ad12
```

has fewer 48s in its absolute spectrum than any other annealed function of this size and nonlinearity.

$n = 11$:

```
ef3d a74f 8ee4 3066 8eae 24c9 5da2 22ce 031d a6cc 7fcd d712 6c29 6274 1bba d4b2
9bfa 9307 55af c5c9 442e 5933 5dc9 027c 1156 27ae 5896 0ea2 f7bd 3c3a f1a4 fa60
76f1 bd35 9133 1afd df26 d3f0 979f 9ae9 afe6 caac 1b04 10f7 e990 2dfc 8658 a489
82a3 644f ab5e d40a 4f9e 1dba 7bf0 4278 88c1 28e6 30b9 00f4 2b7c 7b59 242c 23a1
dc35 21b8 7096 38d1 096a e824 2478 7e2c f897 a4d5 d593 7a2d 98c6 749f 18e9 8d16
8b8c 1a46 67e5 3bf2 f4b4 1e67 4f15 2152 1857 112c 5371 128c 40c0 349b 70ac cbc9
88e9 bb1b b201 f67b b6ca 3ab2 c41b c153 53a8 02fe f49a 2c10 eed3 fa9a 979b 1a70
d751 a9ba 7f95 5678 623c c750 6789 9bc6 4cfd de0e db07 9bd1 300f ed27 a6b6 4af8
```

has the lowest number of 72s in its absolute Walsh spectrum.

$n = 12$:

3047 d0a3 617a ad1d bd27 c955 c3df 0ba0 2ca6 b256 2f78 92dd c2b1 e417 42ec ce8d
1133 e062 7d87 26ee 20d8 c9e5 f142 c333 9df6 07ec a417 026e c27d 062a ce4c 2a68
ef96 bf1c ddfb 9945 a0cf ce07 155c 3c1b 326b 780e 0d4c 6676 1f93 3245 cb3c 9e33
e217 49f2 c06e 94aa 1f21 836a 0bc9 a674 9ffd ae24 654c af61 3a8f ad74 12f0 a7ae
4631 416b 5fbc b2e7 c124 92ab 0a8c 4541 9aed e66e eba8 349c 3b0f b48a 378a ed9e
ba0a e03d 53b7 a0bd 7895 55d2 13b6 cb62 957d b2e7 b7d5 f87b 2718 1a48 2304 f165
2f39 a1f8 af0f b5b1 7b5f 8501 7471 7d6d bba1 eb67 1e80 6090 aa1e 0133 55bc 4e8a
4be4 5784 08f7 25ba de1a 6a9d 9e60 7efe 12da 9c15 8d27 0f93 cf22 754b 8086 6ac3
9590 719d 424b e466 1ec9 3186 430c 84cc d35e aee6 b184 8898 6af2 edbf 0017 6a75
d76d 7477 7788 0636 0f96 8762 3e8d 4ca1 348a b21e 12f1 7a20 f632 ca85 f0c7 dfed
f8fd a288 0a84 b289 6108 5c16 ed3f 408a fa61 9906 90d8 1faa cae3 f715 2ed0 75d3
3bbe 312a e141 e187 9cd1 9010 b156 18ff 3c13 bf3f 5294 31b2 ef35 d866 25e6 16e5
6fe9 e846 2383 b955 b394 a71c be34 eb50 cf5b 464c f3ef a988 29e7 96a0 b3f0 caad
c631 deb0 bc8f 81a2 4e46 d593 a48b 3217 a755 8064 ed15 3037 04c3 644e 3da5 5eeb
506e 4d9a 7e9b d84e fb40 4b0a a432 1400 93a4 6dfb df11 212d 6056 6db6 43f5 4ccc
41f9 c1af eba9 067c 56f7 534f 6f17 dbf1 6f8f 1789 5f0a 48cd 8523 9fc4 c9e5 f8d3
has the fewest 104s in its Walsh spectrum.

$n = 13$:

f10d 0769 ad35 6249 b1a0 1dbb d733 6786 7d68 0c5d 5632 3687 475a cf43 36ee 8619
0d81 e2a7 bc23 4616 be11 6b4e 4c54 fa9a ba6f a8d3 6472 6cde aa07 3fc7 a4cf 248b
2a17 9259 9d82 9826 e944 5829 20d1 a701 2627 0562 c27d 61ab f7d0 3970 354b d08d
3f5e d387 9c4f 6e35 ecfd 606a 655d c563 5a50 1f4b 4e37 bff0 dfcf 6f63 d2b4 4624
d3a4 87ab a67b cd0d c79e 3ed9 f1b3 c836 2007 011e 9769 11eb 7e6a b5c1 9a04 0b10
f34c 5f87 5c78 c9f4 4aee ab81 7d47 ee4a 6fae 7797 0313 7eb3 ce2e 9e72 2c69 c68a
a0e1 9506 8127 ff05 9b1b cb33 93d9 dd47 c6c7 c477 11f5 9a91 3f64 34e4 5c7e 5e58
0ec3 ac6b 070c 6ae8 007b f913 ad8e a1dc e853 f5db e417 d260 0ddf b01d 0574 431e
de61 4cc3 d346 b59c b757 fbe2 2627 39d4 98ce a94c 2286 6cca cb70 d6db def7 6692
5ecd 4788 9f8f 8770 e7a2 cfad 065c fcd3 a314 8907 8a96 2ea0 8bea 13b4 75d7 44ca
18f6 6eb0 6e19 869d b4c4 3d99 9cea ae19 812a be77 6ca7 8b50 6c58 a5a5 020c 10d8
f0cb df77 255c 28a8 92f3 45fa 330f 51f6 1f84 330f e941 d172 2260 33e8 7a47 01c4
196a a92e 3c14 1b9e 9703 b773 d061 67da c46f 92a7 431d 2e12 6e73 faa2 ebfa 7b70
f8e9 0bdf f2fc 2413 6727 3170 9d1f bf3c 0b73 78bc 9598 cfa6 6fd7 af90 2e14 cdcd
95f5 2fe6 d822 a2d1 16cb cd17 a1b6 f8a2 fd49 0800 b00d e8b6 104b 9489 293e b9f0
58db d61f 8d2b f783 1404 d437 deea 7fdc b614 e44a a46a 909b cc21 5b4f c385 f362
1da0 f966 0df0 b7c9 da13 5ccf 50ee 6516 12c5 5981 69dc 305a 7b49 95f3 4330 66b4
694d d4a4 5316 e13c ba33 97f0 d73c d0c7 ff69 953e dda7 fefe cea8 7496 53ed 94c0
c57a 5201 a34e 046d be40 af3b 7d64 27cd 6951 38aa 6650 bfe5 bc4e 3885 ce04 39b4
c21a e8cf 4ca1 775f 9431 859e edba d692 464b c8c6 5082 a4f2 d98f 2725 32e6 2013
e831 4061 fe2d 5205 9718 6f80 65af 680c 8c1f 3218 dc01 dbc8 7d5e 6e54 0160 deae
3f70 ca59 e72e 8eaa 7a0e ba33 146f 65c3 b9d2 1b8d d8c1 1677 f00d 68bc ca88 e7de
5977 fb18 1c53 9ad3 79df 1d68 7f97 218c 07de 7659 dbba de43 8e0c a832 d3be ce4c
00df 0c3e 9d58 5a74 491b 1c0e d69d 65b1 f90d b13f 6dfc bad2 5a76 a4d7 db0c 4ccd
968d 3d73 88de 5200 f39c e0dd 2847 3a98 f6f2 494b 4847 0c1c e9f6 2fd7 5da8 4517
c14b 851d 3595 301e 8e3e f08e e67d 9d12 86c3 e69a ef0d 8df2 2f07 813a 4cbe 99a3
3e99 fdf8 73d2 ba45 4b41 224c eb36 f224 e691 7ba0 4553 a556 62b6 6940 93d7 10d0
e0ba 0370 88c3 b791 bfb8 d6db 26fb e15d ac82 1b14 bb13 a1cd 164e c315 282d 27aa
8a28 f52f 2735 1dff 24f0 4a5c 38ad c4d7 9f65 c381 406e b0ad b85d d6a0 0098 834d
4625 d050 65fb 7504 603a 4da4 0134 8841 ba92 9edf 8ac6 148a d55c e770 6b91 4846
c1c9 5a80 556f 153f b91f 6623 12de 05c5 fc77 d1c0 5315 28ec 8811 88cf 7363 acf8
a550 a343 775c 26c0 c2c3 49bd 6320 4d21 5319 7e8f fa6b 02df 1c47 685e a669 5020

has the fewest 152s in its Walsh spectrum