Advanced cryptology (GBX9SY06)

Block ciphers 2

Pierre Karpman

2017-11

1 Divide & conquer attacks

A historically and practically important technique used to attack block ciphers is the so-called Meet-in-the-middle attack (MiTM; not to be confused with the *Man*-in-the-middle attack). This was first mentioned in the seventies by Diffie and Hellman [DH77] to show that composing two block ciphers with different keys did not increase the security as much as one could naively think; namely, the expected time complexity of the best generic attack only doubles, while one could have assumed that it goes to the square. In a nutshell, using the notation $\mathscr{E}_k$ for $\mathscr{E}(k,\cdot)$ when $\mathscr{E}$ is a block cipher, a MiTM attack on $\mathscr{E}_{k_1} \circ \mathscr{E}_{k_0}$ simply works as follows:

1. Obtain the ciphertext c corresponding to a known plaintext m for the unknown keys k_0 and k_1.
2. Create a list L of pairs $(k', \mathscr{E}(k', m))$ for all possible keys k'.
3. For all possible k'', compute $x := \mathscr{E}^{-1}(k'', c)$. If x is found as the second element of a pair $(k', \cdot)$ of L, output (k'', k') as a key candidate for (k_1, k_0).

If $\kappa := |k| > n := |m|$, one may use more than one plaintext-ciphertext pair in order to avoid an exponentially-growing number of false positives. Apart from that, this algorithm minimizes the time complexity in generically attacking "double-$\mathscr{E}$". Note however that if one restricts the running time of the adversary to t, the above attack succeeds with probability at most $t^2/2^{2\kappa}$, which is generally lower than the generic complexity of $t/2^\kappa$ one can achieve in attacking $\mathscr{E}$ alone. It can in fact be proven that this gap cannot be filled [ABCV98]

The basic property exploited in the above algorithm is that a collision between two a priori random lists suggests a candidate value for the key. If the lists are of size N and N', this allows to test up to $O(N \cdot N')$ candidates for a certain condition, allowing for a quadratic gain in time complexity. This feature appears in many other collision-based "MiTM" attacks, such as the slide attacks on Even-Mansour constructions or the many-related-keys attack on any block cipher.

MiTM attacks do not only apply in such generic settings. For instance, they can be useful in exploiting weaknesses in the key-schedule of a concrete iterated block cipher.

Definition 1 (Iterated block cipher)**.** An *iterated block cipher* is a block cipher $\mathscr{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ that can be defined as the composition of a *round function* $\rho : \mathbf{N} \times \{0,1\}^{\kappa'} \times \{0,1\}^n \to \{0,1\}^n$ whose *round-keys* are generated by a *key-schedule* or *key-expansion* algorithm $\Gamma : \mathbf{N} \times \{0,1\}^\kappa \to \{0,1\}^{\kappa'}$. That is, the r-round $\mathscr{E}(k,\cdot)$ is equal to $\rho(r, \Gamma(r,k), \cdot) \circ \ldots \circ \rho(0, \Gamma(0,k), \cdot)$.

The huge majority of actual block ciphers are iterated.

One can see that the double-encryption construction $\mathscr{E}_{k_1} \circ \mathscr{E}_{k_0}$ can be redefined as a new block cipher $\mathscr{E}'_k$ with twice as much rounds as $\mathscr{E}$ and a "bipartite" key-schedule Γ' s.t. $\Gamma'(0 \leq i \leq$

$r,k) = \Gamma(i,k_0)$ and $\Gamma'(r < i \leq 2r+1,k) = \Gamma(i-r-1,k_1)$. While one does not expect a concrete design to exhibit such a strong independence property of two parts of its key schedule, weaker cases of independence remain possible. Consider for instance an imaginary block cipher $\mathscr{E}$ which is such that with high probability (over the keys and the plaintexts), a certain subset $\mathscr{S}$ of the bits of the intermediate ciphertext after r_f rounds, i.e. the image of $\rho(r_f,\cdot,\cdot)\circ\ldots\rho(0,\cdot,\cdot)$, only depends on a subset $\mathscr{K}_0$ of the key bits. Assume further that with high probability (over the keys and the ciphertexts), this same subset only depends on a subset $\mathscr{K}_1$ of the key bits when computed as a partial decryption, i.e. as the image of $\rho^{-1}(r_f-1,\cdot,\cdot)\circ\ldots\circ\rho^{-1}(r,\cdot,\cdot)$. Then if $\mathscr{K}_0-\mathscr{K}_1 \neq \emptyset$ and $\mathscr{K}_1-\mathscr{K}_0 \neq \emptyset$, one can mount a MiTM attack that independently searches for the bits of either set difference. This kind of approach is for instance the basis for the currently best attacks on the lightweight block cipher KATAN [CDK09, FM14].

2 Statistical attacks & distinguishers

We now introduce an altogether different approach that is very successful in attacking many block ciphers, namely *statistical*, distinguisher-based attacks. This denomination typically regroups *differential* and *linear* cryptanalysis, which are related in many ways. We will mostly focus on differential attacks, but also briefly mention the linear case. both of these approaches were successfully used to attack des in the early nineties [BS90, Mat93].

Note on signature types. Fundamentally, block ciphers operate on binary data that does not correspond to any abstract element such as a vector (in the mathematical sense), a finite field element, etc. This is why we typically have $\mathscr{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. However, in the same way as concrete designs sometimes benefit from algebraic constructions by seeing their inputs as mathematical objects, attacks may also be defined as operating on, say, vectors of $\mathbf{F}_2^n$ rather than binary strings of length n. When this is the case, one needs to agree on a mapping from one set to another. Fortunately, this is usually straightforward.

2.1 The Differential case

Definition 2 (Differential)**.** Let $\mathscr{E}$ be a block cipher, a *differential* is a pair $(\Delta \neq 0,\delta)$ of input and output *differences* for $\mathscr{E}$, according to some group law + (generally taken to be the addition in $\mathbf{F}_2^n$, i.e. the XOR or $\oplus$).

An input (k,m) to $\mathscr{E}$ is said to *verify* the differential (Δ,δ) if $\mathscr{E}(k,m+\Delta)-\mathscr{E}(k,m) = \delta$. If the key is fixed, we may also call *differential pair* for (Δ,δ) an ordered pair $((m,c),(m',c'))$ of two plaintext-ciphertext pairs for $\mathscr{E}(k,\cdot)$ s.t. $m'-m = \Delta$ and $c'-c = \delta$. Note that over $\mathbf{F}_2^n$, addition coincides with subtraction, and the following expressions can be simplified. We always consider such a case from now on.

The usefulness of the notion of differential comes from the fact that it is often a good distinguisher between mappings that are "ideally random" or not. That is, one will exploit statistical properties of $x \mapsto \mathscr{P}(x)\oplus\mathscr{P}(x\oplus\Delta)$ (for some Δ) in the hope of deciding whether $\mathscr{P} = \mathscr{E}(k,\cdot)$ for some unknown key k or if it is a random permutation. This can be done by considering the *differential probability* of the differential.

Definition 3 (Differential probability)**.** The *differential probability* of a differential (Δ,δ) w.r.t. a permutation $\mathscr{P}$ is the probability of obtaining a differential pair for $\mathscr{P}$:

$$\mathrm{DP}^{\mathscr{P}}(\Delta,\delta) := \Pr_{m\in\{0,1\}^n}[\mathscr{P}(m)\oplus\mathscr{P}(m\oplus\Delta) = \delta].$$

Note that the differential probability is a function of both the differential itself and the permutation. In particular, this means that for a block cipher, we may (and usually do) have

$\mathrm{DP}^{\mathscr{E}(k,\cdot)}(\Delta,\delta) \neq \mathrm{DP}^{\mathscr{E}(k'\neq k,\cdot)}(\Delta,\delta)$, for an arbitrary (Δ,δ). This leads to the notion of *expected differential probability* for block ciphers, which is simply the average over k of the differential probability of $\mathscr{E}(k,\cdot)$:

$$\mathrm{EDP}^{\mathscr{E}}(\Delta,\delta) := 2^{-\kappa} \sum_{k\in\{0,1\}^\kappa} \mathrm{DP}^{\mathscr{E}(k,\cdot)}(\Delta,\delta).$$

A convenient (unfortunately not necessarily true) hypothesis is that any fixed-key DP for $\mathscr{E}$ is "close" to the EDP. In order to distinguish $\mathscr{E}$ from a random permutation $\mathscr{P}$, then one only needs the difference between $\mathrm{EDP}^{\mathscr{E}}(\Delta,\delta)$ and $\mathrm{DP}^{\mathscr{P}}(\Delta,\delta)$ to be sufficiently large for a given (known) (Δ,δ). This begs the question of what is the expected value of $\mathrm{DP}^{\mathscr{P}}$ for a random differential. In the non-injective case, replacing $\mathscr{P}$ by a random function $\mathscr{F}$, the answer is easy. By definition, $\mathscr{F}(x) \xleftarrow{\$} \{0,1\}^n$, thus $\Pr[\mathscr{F}(x)\oplus\mathscr{F}(x\oplus\Delta)=\delta] = \Pr[\mathscr{F}(x)=\delta\oplus\mathscr{F}(x\oplus\Delta)] = 2^{-n}$. The injective case is more complex, but one can show that the number of differential pairs is approximately drawn according to a Poisson distribution of mean and variance 2^{-1} [O'C95, DR07]. This means that the expected DP is also equal to 2^{-n}; note however that it can only take values multiple of 2^{-n+1} (as differential pairs are symmetric and thus come by two). It is also worthwhile to remark that if $\mathscr{P}$ is *linear* w.r.t. the difference operation (here $\oplus$), then by definition, $\mathrm{DP}^{\mathscr{P}}(\Delta,\delta) = 1$ if $\mathscr{P}(\Delta)=\delta$, 0 otherwise.

We now have everything we need to describe a differential distinguisher on a block cipher. This is done in Algorithm 1. This distinguisher only succeeds with some probability, that can

Input: $\mathscr{O} \xleftarrow{\$} \{\mathscr{P}, \mathscr{E}(k \xleftarrow{\$} \{0,1\}^\kappa,\cdot)\}$; (Δ,δ) of $\mathrm{EDP}^{\mathscr{E}} = p$
Output: 1 if $\mathscr{O}=\mathscr{P}$, 0 otherwise

```
1  begin
2      count := 0
3      mult := 10 /* Can be set to other values                    */
4      for i := 0; i < mult/p do
5          m <-$ {0,1}^n /* Without replacement                     */
6          if O(m) ⊕ O(m ⊕ Δ) = δ then
7              count := count + 1
8          end
9      end
10     if count = 0 then
11         return 1
12     end
13     else
14         return 0
15     end
16 end
```

Algorithm 1: Differential distinguisher for $\mathscr{E}$

among other things depend on the actual value of $\mathrm{DP}^{\mathscr{E}(k,\cdot)}$ for the random key k. The multiplicative constant on line 3 can be set to ensure that at least one differential pair is found with high probability. However, if the EDP is close to 2^{-n}, selecting a large constant may decrease the success probability, by increasing the likelihood that a differential pair is found for a random permutation (this issue may be slightly reduced by having the algorithm returning zero as soon as a differential pair is found). Finally, the data complexity is of $2\cdot\mathrm{mult}/p = \Theta(p^{-1})$ chosen plaintexts, which directly dictates the time complexity as well, while the memory complexity is negligible.

Finding a distinguisher is enough to break the PRP security of a block cipher. However, when possible, it is even better for an attack to recover some key material. It is also possible to do so with differential attacks, by using a distinguisher as a test for the possible values of (part of) the unknown key.

We describe this process in a case where the block size n is equal to half of the key size κ and where round keys are as long as the block. We also assume that there are only 2^n keys corresponding to a fixed n-bit round key, and that these are easy to enumerate. A simple key schedule for which this statement is true is $\Gamma(i, k_1||k_0) := k_{(i \mod 2)}$. In that case, an attacker might enumerate all 2^n possible values for the last round key and; if he is able to uniquely determine the right one thanks to a distinguisher, the full key can be determined by again enumerating all possible 2^n values and checking them w.r.t. a few plaintext-ciphertext pairs.

The crucial step is the first one, that is trying to distinguish the right guess for the last round key from all the other ones. In a differential attack, this might be done by having a "good" distinguisher up to the before-last round. Say that the expected probability of the differential used in this distinguisher is $p \gg 2^{-n}$, i.e. $\mathrm{EDP}^{\mathscr{E}/(r-1)} = p$, where $\mathscr{E}/(r-1)$ denotes $\mathscr{E}$ reduced to $r-1$ rounds. One assumes that the EDP of the same differential is much smaller after r rounds, i.e. $\mathrm{EDP}^{\mathscr{E}/r} \ll p$. Going further, we assume that composing $\mathscr{E}/r$ with *the inverse of the round function with a random round key* does not increase the EDP of the differential (ideally increasing it); that is, if we let $\mathscr{E}'(k,\cdot) := \rho^{-1}(r, k' \xleftarrow{\$} \{0,1\}^{\kappa'}, \cdot) \circ \mathscr{E}/r(k,\cdot)$, we assume that $\mathrm{EDP}^{\mathscr{E}'} \lll p$. With these assumptions, the attacker may then run the following Algorithm 2, at the end of which a further 2^n candidates for the entire key need to be tried. The time complexity of this algorithm

Input: $\mathscr{O} = \mathscr{E}(k \xleftarrow{\$} \{0,1\}^{\kappa}, \cdot)\}$; (Δ, δ) of $\mathrm{EDP}^{\mathscr{E}/(r-1)} = p$
Output: A candidate for the last round key

```
1  begin
2      max := 0
3      cand := 0
4      forall k' ∈ {0,1}^n do
5          count := 0
6          mult := 10
7          for i := 0; i < mult/p do
8              m ←$ {0,1}^n
9              c_0 := ρ^{-1}(r, k', (O(m)))
10             c_1 := ρ^{-1}(r, k', (O(m ⊕ Δ)))
11             if c_0 ⊕ c_1 = δ then
12                 count := count + 1
13             end
14         end
15         if count > max then
16             max := count
17             cand := k'
18         end
19     end
20     return cand
21 end
```

Algorithm 2: Differential key-recovery attack for $\mathscr{E}$

is $\Theta(2^n \cdot p^{-1})$; its memory and data complexity depend whether one is able to reuse data for different key guesses; its success probability depends on the validity of all the involved hypotheses (which unfortunately might be hard to test). Note that in practice, one does not need to return a single candidate: it is perfectly reasonable to keep all candidates whose counter is above a certain threshold. Of course, this requires some additional memory and subsequent tests for the correct full key, so it is still best to keep this number quite low.

The basic approach as sketched above is unlikely to be used as is in an actual attack. For instance, the process used to guess (and eliminate) (part of) a round key may be more complex;

several differentials may be used; *truncated* differentials may be employed; some early-abort strategies may be deployed; etc. We do not describe these in these notes and refer an interested reader to e.g. [KR11, Din14].

So far, we have assumed that a suitable differential (Δ, δ) was known to the attacker. Most of the time, finding such a differential is in fact the crux of the attack. We will not address this problem, but briefly mention one of the starting points to do so, that is finding differential *characteristics* (or trails).

In all genericity, the differential probability $\mathrm{DP}^{\mathscr{P}}(\Delta, \delta)$ for $\mathscr{P} : \{0,1\}^n \to \{0,1\}^n$ and a single differential requires to enumerate all possible inputs to $\mathscr{P}$, i.e. with time complexity $\Theta(2^n)$ and negligible memory; computing *all* differentials can be done in time $\Theta(2^n)$ and memory $(\Theta(2^n)$. When n is large (for instance because $\mathscr{P}$ is the round function of a block cipher), this (doubly-)exponential complexity is intractable. However, if $\mathscr{P}$ is a small component used as part of a round function, for instance an 8-bit S-box, the associated complexity is perfectly reasonable. If $\mathscr{E}$ is a substitution-permutation-network block cipher, this fact can be used to efficiently compute the differential probability of its round function for an arbitrary differential: the DP of the S-boxes are explicitly computed, while the DP of the linear layer is known. This is not directly useful, as attacking one round of a block cipher is usually not an impressive feat. However, one may hope to chain several one-round differentials to obtain a characteristic.

Definition 4 (Differential characteristic)**.** Let us write $\Delta \xrightarrow{\rho} \Delta'$ the fact that (Δ, Δ') is a differential for the round function ρ (where we drop the round index and the key for simplicity). An r-round *differential characteristic* for ρ is an $(r+1)$-tuple $(\Delta_0, \ldots, \Delta_r)$ s.t. $\Delta_0 \xrightarrow{\rho} \Delta_1 \xrightarrow{\rho} \ldots \xrightarrow{\rho} \Delta_r$.

Note that if an input follows a characteristic[1] $(\Delta_0, \ldots, \Delta_r)$, then it is also a differential pair for the r-round differential (Δ_0, Δ_r). While the converse is not true in general, one sometimes assume that if a characteristic has a high probability of being followed by a random input, relatively to the block size, then it is dominating the other characteristics that lead to the same differential and the probability of the latter is close to the one of the former. This hypothesis is quite tempting, especially as estimating the probability of a characteristic is much easier than for a differential, even if it requires its own hypotheses.

Let us consider a characteristic $C := (\Delta_0, \ldots, \Delta_r)$. We assume that if all round keys of ρ are independent and random, the probability that an input follows the C is equal to the product of the one-round differential probabilities, that is equal to $\Pi_{0 \leq i < r} \mathrm{DP}^{\rho}(\Delta_i, \Delta_{i+1})$. This is the *Markov assumption* [LMM91]. In practice, for want of a better model, we assume the same even if the round keys are not independent (which is more often the case than not). Using this assumption, one may use various ways to find r-round characteristics and their associated probabilities, for instance "by hand", or using Matsui's branch-and-bound algorithm [Mat94].

2.2 The Linear case

Linear cryptanalysis presents many similarities with the differential approach. The main difference comes from the nature of the statistical property exploited in the underlying distinguishers: in the linear case, one is interested in how *biased* is a linear equation in the input and output bits of a permutation.

Definition 5. Walsh transform The *Walsh transform* $\mathscr{W}^{\mathscr{P}} : \{0,1\}^n \times \{0,1\}^n \to \mathbf{Z}$ of $\mathscr{P} : \{0,1\}^n \to \{0,1\}^n$ is defined extensively by:

$$\mathscr{W}^{\mathscr{P}}(a,b) := \sum_{m \in \{0,1\}^n} -1^{\langle b, \mathscr{P}(m) \rangle + \langle a, m \rangle},$$

where $\langle x, y \rangle$, $x, y \in \{0,1\}^n$ denotes the $\{0,1\}$-valued dot product between x and y when seen as vectors of $\mathbf{F}_2^n$.

[1] Defined in the obvious way.

One can note the similarity of the expressions defining $\mathcal{W}^{\mathscr{P}}(a,b)$ and $\mathrm{DP}^{\mathscr{P}}(\Delta,\delta)$: both involve an expression relating the input and output of $\mathscr{P}$, that is evaluated over all its possible inputs. A difference however is that in the linear case, we only consider a single value $\mathscr{P}(m)$ at a time whereas the differential looks at pairs of input related in a certain way. A consequence of this is that linear attacks may only rely on *known* plaintexts, versus chosen plaintexts for differential ones.

The equation defining the Walsh transform can be defined in words as counting how often a given *linear approximation* $\langle a, m\rangle = \langle b, \mathscr{P}(m)\rangle$ with *linear masks* (a,b) holds, versus how often it does not, when m ranges over all possible values. In the extreme case where $\mathscr{P}$ is linear, this equality is either always true or always false, and $\mathcal{W}^{\mathscr{P}}(\cdot,\cdot) = \pm 2^n$. On the other hand, if the approximation with masks (a,b) holds approximately the same number of time that it does not, then $\mathcal{W}^{\mathscr{P}}(a,b)$ is small.

The *correlation* $C_{\mathscr{P}}(a,b)$ of a linear approximation (a,b) for $\mathscr{P}$ is defined directly from the Walsh transform as $C_{\mathscr{P}}(a,b) = \mathcal{W}^{\mathscr{P}}(a,b)/2^n$; this quantity thus ranges from -1 to 1. One can show that for a non-trivial approximation (a,b) and $n \geq 5$, the distribution of $C(a,b)$ over all n-bit permutations can be approximated by a normal distribution $\mathcal{N}(0, 2^{-n})$ [DR07, BT13]. This means that if a certain $\mathscr{P}$ is such that $C = C_{\mathscr{P}}(a,b)$ is sufficiently away from 0 for a given approximation (a,b), one may be able to distinguish it from a random permutation in a way similar to Algorithm 1: an adversary may count how many times the approximation holds over sufficiently many $D := \Theta(C^2)$ input values, and decide that he is interacting with $\mathscr{P}$ if this is sufficiently away from $D/2$. Finally, to continue the analogy, such distinguishers can also be used in key-recovery attacks, and linear approximations can also be chained over several rounds in a way similar to differential characteristics.

3 Algebraic attacks

We will for a moment step back from n-to-n-bit mappings, and consider the n-to-one case of *Boolean functions*.

Definition 6 (Boolean function)**.** A *Boolean function* with n variables is a mapping $f : \mathbf{F}_2^n \to \mathbf{F}_2$.

Note that we could have alternatively used a signature $f : \{0,1\}^n \to \{0,1\}$, which would be equivalent for most purposes. Our choice was motivated by the fact that we will precisely treat Boolean functions as algebraic objects. However, we will nonetheless adopt a compact representation for vectors of $\mathbf{F}_2^n$, writing $011\ldots$ for the vector $[0,1,1,\ldots]$.

A natural way to represent an n-variable Boolean function f is to define its "truth table", that is, to list all its 2^n possible inputs and their associated evaluation. This simply requires a string s_f of 2^n bits, where the value of the i^{th} bit of the string $s[i]$ is given by the evaluation of f on the vector that maps to i as a binary integer.

Example 1. The 3-variable Boolean function f given by $f(000) = 1$, $f(001) = 0$, $f(010) = 0$, $f(011) = 1$, $f(100) = 1$, $f(101) = 1$, $f(110) = 0$, $f(111) = 1$, can be represented by the truth table `10011101`, read from left to right (i.e. the bit of smallest index is on the left).

With this representation in mind, it is particularly obvious that there are 2^{2^n} distinct n-variable Boolean functions: one for each 2^n-bit string.

An important fact about Boolean functions is that they possess another "natural" representation, as multivariate polynomials over $\mathbf{F}_2$.

Definition 7 (Algebraic normal form and degree of a Boolean function)**.** The *algebraic normal form* (ANF) of the n-variable Boolean function f is the unique polynomial $g \in \mathbf{F}_2[X_0, X_1, \ldots, X_{n-1}]/\langle X_i^2 - X_i\rangle_{i<n}$ such that for all $x \in \mathbf{F}_2^n$, $f(x) = g(x[0], x[1], \ldots, x[n-1])$.

The *degree* of f is the degree of its ANF g.

We can prove existence and unicity with these statements: 1) any polynomial of $\mathbf{F}_2[X_0, X_1, \ldots, X_{n-1}]/\langle X_i^2 - X_i\rangle_{i<n}$ maps to an n-variable Boolean function defined by its evaluation; 2) this mapping is injective, as two distinct such polynomials g, g' define different Boolean functions, since their difference $g - g'$ is not the zero polynomial (and hence does not uniformly evaluate to zero); 3) this mapping is bijective, as the two sets have the same cardinality.

We now have at least two ways of representing a Boolean function: by its truth table and by its ANF. However, while going from the ANF to the truth table is easy, we still need an efficient way of computing the ANF of a function given its truth table. Fortunately, it turns out that doing so is easy too.

First, stating the obvious, computing the ANF means retrieving the value of the coefficient in front of all possible 2^n monomials (which is either one or zero). This is trivial for the constant monomial (written $g_{00\ldots0}$): the ANF g of f has a constant term iff $f(00\ldots0) = 1$. Indeed, by definition, $g(0,0,\ldots,0) = g_{00\ldots0} = f(00\ldots0)$. This is not too hard either for the n degree-one monomials $X_0, \ldots X_{n-1}$. Say that we want to compute the coefficient $g_{10\ldots0}$ in front of X_0: we just need to evaluate f on $10\ldots0$ and add the result (modulo two) to $g_{00\ldots0}$; this is again simply by definition, as $f(10\ldots0) = g_{10\ldots0}X_0 + g_{00\ldots0}$, so $g(1,0,\ldots0) = f(10\ldots0) + g_{00\ldots0} = f(10\ldots0) + f(00\ldots0)$. We then simply proceed inductively for higher-degree monomials. For instance, $g_{110\ldots0}$ is given by $f(110\ldots0) + f(100\ldots0) + f(010\ldots0) + f(000\ldots0)$, and more generally $g_u = \sum_{v\preccurlyeq u} f(v)$, where $a \preccurlyeq b$, $a, b \in \mathbf{F}_2^n$ if $\forall i, b[i] = 0 \Rightarrow a[i] = 0$. The mapping $u \mapsto g_u$ is called the *Möbius transform(ation)*.

Example 2. The ANF of the Boolean function f of Example 1 is given by $g = 1 + X_1 + X_2 + X_0X_2 + X_0X_1X_2$.

This approach calls for three major comments. The first is that to compute a coefficient g_u, of a degree-d monomial X_u, we are in effect differentiating d times the d-variate polynomial g' whose variables are the ones appearing in X_u. This polynomial is of degree at most d, with only X_u as a possible degree-d monomial. Thus, the result of the differentiation is either a non-zero constant polynomial (i.e. 1) if g' is of degree exactly d, or the zero function. This in turns indicates if $X_u \in g'$, i.e. the value of g_u.

Formally, the notion of differentiation that we use is defined as follows.

Definition 8 (Derivative of a Boolean function)**.** The *derivative* of a Boolean function $f : \mathbf{F}_2^n \to \mathbf{F}_2$ *along* $\Delta \in \mathbf{F}_2^n$ is defined as $\partial f/\partial\Delta := x \mapsto f(x) + f(x+\Delta)$. The order-$d$ derivative $\partial^d f/\partial\Delta_0, \ldots \Delta_{d-1}$ is defined as $\partial(\ldots(\partial f/\partial\Delta_0)\ldots)/\partial\Delta_{d-1}$.

The second comment is related to the complexity of computing the ANF of f from its truth table. A naïve implementation might simply compute the value of each coefficient independently; there are 2^n of them, and the coefficient of a monomial of degree d requires 2^d evaluations of f. The total complexity of this approach is thus $\sum_{0\le i\le n}\binom{n}{i}2^i = 3^n$. However, one can do much better by observing that computing the coefficient of g_u uses as intermediate results the values of all of the g_v, $v \preccurlyeq u$. In particular, the computation of $g_{111\ldots1}$ uses the values of *all* of the other g_us. Thus, we only need to compute $g_{111\ldots1}$ while computing all intermediate values only once and storing them along the way. This can be done conveniently in a recursive way: first compute the ANF of the two degree-$(n-1)$ functions $f^{(1)} := g(1,\cdot,\cdot,\ldots,\cdot)$ and $f^{(0)} := g(0,\cdot,\cdot,\ldots,\cdot)$, obtaining $g^{(1)}$ and $g^{(0)}$, then return $g(X_0,\ldots,X_{n-1}) = X_0\,g^{(1)}(X_1,\ldots,X_{n-1}) + g^{(0)}(X_2,\ldots,X_{n-1})$. This algorithm can be transformed into an efficient iterative and in-place procedure [Jou09], given as Algorithm 3. The inner loop 5–10 is executed n times by the outer loop. Its i^{th} execution takes 2^{n-i} executions of the innermost loop 6–8 that itself performs 2^i elementary computations. The total complexity of Algorithm 3 is thus $n2^n$ elementary operations. This compares very favourably with the cost of $3^n \approx 2^{1.58n}$ of the naïve algorithm, even for small values of n.

The third and last comment we make is that the Möbius transform is its own inverse, i.e. an involution. One can prove this statement recursively by seeing that $f(00\ldots0) = g(00\ldots0)$ and

Input: The truth table s of an n-variable Boolean function f
Output: The table s is overwritten with the coefficients of g, the ANF of f

```
1  begin
2      for i := 0; i < n do
3          L := 2^i
4          R := 0
5          while R < 2^n do
6              for j := 0; j < L do
7                  s[L + R + j] := s[L + R + j] ⊕ s[R + j]
8              end
9              R := R + 2L
10         end
11     end
12 end
```

Algorithm 3: Fast Möbius transform computation (iterative version)

that

$$\begin{aligned} & g(1,0,\ldots,0) = f(10\ldots0) + f(00\ldots0) \\ \Leftrightarrow\ & g(1,0,\ldots,0) = f(10\ldots0) + g(0,0,\ldots,0) \\ \Leftrightarrow\ & f(10\ldots0) = g(1,0,\ldots,0) + g(0,0,\ldots,0). \end{aligned}$$

One should notice the relation between a Möbius and a Fourier transform: both allow to interpolate a function from its values and vice-versa. The Walsh transform of Definition 5 could also be seen as a kind of Fourier transform, and can also be computed with a fast algorithm.

We now go back to n-to-n-bit mappings $\mathscr{P}$. As any such mapping can be written as the collection of n n-to-one-bit Boolean functions, all of the results obtained so far in this section extend smoothly. In particular, one may define the ANF of $\mathscr{P}$ as the collection of the ANFs of its constituent Boolean functions (e.g. projected on the canonical basis). This allows to naturally define the degree of $\mathscr{P}$ to be the maximal degree of its n Boolean functions (obtained in turn thanks to their respective ANFs). An important fact is that if $\mathscr{P}$ is a permutation, then it is of degree at most $n-1$. This is simply a consequence of the fact that the coefficient of the unique degree-n monomial (for each of the projected functions) is given by a projection of $\bigoplus_{x\in\{0,1\}^n} \mathscr{P}(x) = 0$.

The degree of a mapping $\mathscr{P}$ can be used as a distinguisher. A random mapping should be of maximal degree (i.e. $n-1$) with high probability: one ANF of its constituent functions may be drawn at random (with the constraint that it is not of degree n), and the probability that all the coefficients of the n degree-$(n-1)$ monomials is zero is 2^{-n}. Thus, if a concrete mapping is of a lower degree, we can use this as a distinguishing criterion, provided that we have an efficient test for the degree.

The simplest way to compute the degree of $\mathscr{P}$ is to compute its ANF. However, this is exponential in the block size of $\mathscr{P}$, and thus quickly becomes intractable in a cryptographic context. If the degree d to be tested for is not too high, an efficient alternative is to differentiate $\mathscr{P}$ at order $d+1$ along $\Delta_0,\ldots,\Delta_d$ in a way that is equivalent to computing one coefficient g_u of the ANFs of each of the constituent functions $\mathscr{P}$, i.e. by having all the Δ_is of weight one and disjoint support (e.g. $\Delta_0 = 100\ldots$, $\Delta_1 = 0100\ldots$, etc.) and evaluating the resulting function on an arbitrary point. Doing thusly, if $\mathscr{P}$ is of degree d, the result is necessarily zero over all the n functions of $\mathscr{P}$; on the other hand, if it is of degree (much) larger than d, the result is zero iff X_u is not a given (sub)monomial in the ANF of any of the functions of, say, a random permutation. By repeating the test a few time with different monomials, one can distinguish the high and low degree cases with high probability. Note that the fast Möbius transform can be used

both to compute a single test and to efficiently bundle several one into the test for an even higher-degree monomial.

Algebraic attacks such as the above are somewhat less common than differential and linear cryptanalysis, but they can nonetheless be very effective. Some examples are given by attacks on the Trivium stream cipher [DS09, FV13], reduced-round Keccak [BCC11] or the ASASA construction [MDFK15, BK15]. Some of these attacks use a particularly nice theorem due to Boura, Canteaut and De Cannière, that provides a non-trivial upper-bound for the degree of an iterated mapping. Quite trivially, if ρ is of degree d, then ρ^r is of degree less than d^r. This already provides a lower-bound on the necessary number of rounds for, say, an iterative cipher, to resist the above attack. However, what Boura et al. showed is that if ρ uses "small" S-boxes of non-maximal degree, the degree of ρ^r may be much lower. We state a particular case of their theorem as Theorem 1

Theorem 1 ([BCC11])**.** *Let $\rho : \{0,1\}^n \to \{0,1\}^n$ be a function corresponding to the concatenation of m smaller invertible S-boxes of size $n_0 \geq 3$ and degree at most $n_0 - 2$, then for any function $\rho' : \{0,1\}^n \to \{0,1\}^n$, we have:*

$$\deg(\rho' \circ \rho) \leq n - \frac{n - \deg(\rho')}{n_0 - 2}$$

For instance, if ρ uses 32 degree-2 4-bit S-boxes, the naïve bound tells us that at least 7 rounds are necessary to reach the maximal degree $127 < 2^7$, while Theorem 1 gives a much higher number of at least 12 rounds (the successive upper-bounds being 2, 4, 8, 16, 32, 64, 96, 112, 120, 124, 126, 127).

A well-known example of distinguisher that can be thought of in an algebraic way is the square/integral/saturation distinguisher on 3-round AES. Recall that this distinguisher works by fixing the input to fifteen of the sixteen AES S-boxes to a constant and summing the 256 3-round ciphertexts obtained when the input to the last S-box ranges over all possible values. The distinguishing criterion is then for this sum to be zero. In other words, one is simply computing the presence of certain degree-8 monomials that are guaranteed to be absent in the ANF of 3-round AES, unlike in the case of a random function.

Finally, one should also mention the relation between the "deterministic" version of algebraic attacks that we have introduced and their statistical counterparts of *higher-order differentials*. The astute reader will have noticed that differential cryptanalysis exploits statistical properties of order-one derivatives along the input differences Δ, where the fact that (Δ, δ) is a high-probability differential means that this derivative is biased towards δ. There is no reason why this could not be generalized to higher-order derivatives as well, exploiting the bias of, say, $x \mapsto \mathscr{P}(x) \oplus \mathscr{P}(x \oplus \Delta_0) \oplus \mathscr{P}(x \oplus \Delta_1) \oplus \mathscr{P}(x \oplus \Delta_0 \oplus \Delta_1)$. Because each round of differentiation may decrease the degree of the resulting mapping, it may become more biased towards certain values than higher-degree ones (for instance, a degree zero mapping is indeed very biased towards its constant value!). A drawback of this approach, however is that the evaluation of a derivative grows exponentially in its order.

References

[ABCV98] William Aiello, Mihir Bellare, Giovanni Di Crescenzo, and Ramarathnam Venkatesan. Security amplification by composition: The case of doubly-iterated, ideal ciphers. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 390–407. Springer, 1998.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of keccak and *Luffa*. In Antoine Joux, editor, *Fast Software Encryp-*

tion - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers, volume 6733 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 2011.

[BK15] Alex Biryukov and Dmitry Khovratovich. Decomposition attack on SASASASAS. *IACR Cryptology ePrint Archive*, 2015:646, 2015.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[BT13] Andrey Bogdanov and Elmar Tischhauser. On the wrong key randomisation and key equivalence hypotheses in matsui's algorithm 2. In Moriai [Mor14], pages 19–38.

[CDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.

[DH77] Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer*, 10(6):74–84, 1977.

[Din14] Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2014.

[DR07] Joan Daemen and Vincent Rijmen. Probability distributions of correlation and differentials in block ciphers. *J. Mathematical Cryptology*, 1(3):221–242, 2007.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[FM14] Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against KATAN. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2014.

[FV13] Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In Moriai [Mor14], pages 502–517.

[Jou09] Antoine Joux. *Algorithmic cryptanalysis*. CRC Press, 2009.

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[LMM91] Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[Mat94] Mitsuru Matsui. On correlation between the order of s-boxes and the strength of DES. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994.

[MDFK15] Brice Minaud, Patrick Derbez, Pierre-Alain Fouque, and Pierre Karpman. Key-recovery attacks on ASASA. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2015.

[Mor14] Shiho Moriai, editor. *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*. Springer, 2014.

[O'C95] Luke O'Connor. On the distribution of characteristics in bijective mappings. *J. Cryptology*, 8(2):67–86, 1995.