Algebraic Techniques in Differential Cryptanalysis

Martin Albrecht* and Carlos Cid

Information Security Group,
Royal Holloway, University of London
Egham, Surrey TW20 0EX, United Kingdom
{M.R.Albrecht,carlos.cid}@rhul.ac.uk

Abstract. In this paper we propose a new cryptanalytic method against block ciphers, which combines both algebraic and statistical techniques. More specifically, we show how to use algebraic relations arising from differential characteristics to speed up and improve key-recovery differential attacks against block ciphers. To illustrate the new technique, we apply algebraic techniques to mount differential attacks against Present-128 reduced to 17, 18 and 19 rounds.

1 Introduction

The two most established cryptanalytic methods against block ciphers are linear cryptanalysis [22] and differential cryptanalysis [3]. These attacks are statistical in nature, in which the attacker attempts to construct probabilistic patterns through as many rounds of the cipher as possible, in order to distinguish the cipher from a random permutation, and ultimately recover the key. Due to their very nature, these attacks require a very large number of plaintext–ciphertext pairs, ensuring that (usually) they rapidly become impractical. In fact, most modern ciphers have been designed with these attacks in mind, and therefore do not generally have their security affected by them.

A new development in block cipher cryptanalysis are the so-called algebraic attacks [14, 23, 9]. In contrast to linear and differential cryptanalysis, algebraic attacks attempt to exploit the algebraic structure of the cipher. In its most common form, the attacker expresses the encryption transformation as a large set of multivariate polynomial equations, and subsequently attempts to solve the system to recover information about the encryption key.

The proposal of algebraic attacks against block ciphers has been the source of much speculation; while a well-established technique against some stream ciphers constructions [13], the viability of algebraic attacks against block ciphers remains subject to debate. On one hand these attack techniques promise to allow the cryptanalyst to recover secret key bits given only one or very few plaintext–ciphertext pairs. On the other hand, the runtime of algebraic attacks against

* This author was supported by the Royal Holloway Valerie Myerscough Scholarship.

block ciphers is not well understood, and it is so far not clear whether algebraic attacks can break any proposed block cipher faster than other techniques.

A promising approach however is to combine both statistical and algebraic techniques in block cipher cryptanalysis. In fact, many proposed algebraic approaches already involve statistical components. For instance, the equation systems usually considered for the AES [23, 9], use the *inversion equation* $xy = 1$ for the S-Box. While this equation only holds with probability $p = 255/256$, it may well offer some advantages when compared with the correct equation $x^{254} = y$ representing the S-Box (which due to its very high degree, is usually considered impractical). Further recent examples include key bit guesses [11], the use of SAT-solvers [1] and the Raddum-Semaev algorithm [24] for solving polynomial equations. In this paper we propose a new attack technique that combines results from algebraic and differential cryptanalysis.

The paper is structured as follows. First, we briefly describe differential and algebraic cryptanalysis and give the basic idea of the attack in Section 2. We then describe the block cipher PRESENT in Section 3 and existing attacks against a reduced round version of PRESENT (Section 3.1). In Section 4 we describe the application of our new attack technique against reduced round versions of PRESENT. We give a brief discussion of the attack and possible extensions in Section 5.

2 Overview of the New Attack Technique

Since our approach combines differential and algebraic cryptanalysis, we briefly describe both techniques below.

2.1 Differential Cryptanalysis

Differential cryptanalysis was first introduced by Eli Biham and Adi Shamir at Crypto'90 [4], and has since been successfully used to attack a wide range of block ciphers. In its basic form, the attack can be used to distinguish a n-bit block cipher from a random permutation. By considering the distribution of output *differences* for the non-linear components of the cipher (e.g. the S-Box), the attacker may be able to construct *differential characteristics* $P^{'} \oplus P^{''} = \Delta P \rightarrow \Delta C_N = C^{'}_N \oplus C^{''}_N$ for a number of rounds N that are valid with probability p. If $p \gg 2^{-n}$, then by querying the cipher with a large number of plaintext pairs with prescribed difference ΔP, the attacker may be able to distinguish the cipher by counting the number of pairs with the output difference predicted by the characteristic. A pair for which the characteristic holds is called a *right pair*.

By modifying the attack, one can use it to recover key information. Instead of characteristics for the full N-round cipher, the attacker considers characteristics valid for r rounds only ($r = N - R$, with $R > 0$). If such characteristics exist with non-negligible probability the attacker can guess some key bits of the last rounds, partially decrypt the known ciphertexts, and verify if the result matches the one predicted by the characteristic. Candidate (last round) keys are counted, and as random noise is expected for wrong key guesses, eventually a peak may be observed in the candidate key counters, pointing to the correct round key[1].

Note that due to its statistical nature, differential cryptanalysis requires a very large number of plaintext–ciphertext pairs (for instance, approximately 2^{47} chosen plaintext pairs are required to break DES [5]). Many extensions and variants of differential cryptanalysis exist, such as the Boomerang attack [26] and truncated and higher-order differentials [21]. The technique is however very well understood, and most modern ciphers are designed to resist to differential cryptanalysis. This is often achieved by carefully selecting the cipher's non-linear operations and diffusion layer to make sure that if such differential characteristics exist, then $r \ll N$ which ensures that backward key guessing is impractical. The AES is a prime example of this approach [15].

2.2 Algebraic Cryptanalysis

Algebraic cryptanalysis against block ciphers is an attack technique that has recently received much attention, particularly after it was proposed in [14] against the AES and Serpent block ciphers. In its basic form, the attacker attempts to express the cipher as a set of low degree (often quadratic) equations, and then solve the resulting system. As these systems are usually very sparse, overdefined, and structured, it is conjectured that they may be solved much faster than generic non-linear equation systems. Several algorithms have been used and/or proposed to solve these systems including the Buchberger algorithm, XL and variants [12, 29, 14] , the F_4 and F_5 algorithm [17, 18], and the Raddum-Semaev algorithm [24]. Another approach is to convert these equations to Boolean expressions in Conjunctive Normal Form (CNF) and use off-the-shelf SAT-solvers [2]. However, these methods have had so far limited success in targeting modern block ciphers, and no public modern block cipher, with practical relevance, has been successfully attacked using algebraic cryptanalysis faster than with other techniques.

2.3 Algebraic Techniques in Differential Cryptanalysis

The first idea in extending algebraic cryptanalysis is to use more plaintext–ciphertext pairs to construct the equation system. Given two equation systems F' and F'' for two plaintext–ciphertext pairs (P', C') and (P'', C'') under the same encryption key K, we can combine these equation systems to form a system $F = F' \cup F''$. Note that while F' and F'' share the key and key schedule variables, they do not share most of the state variables. Thus the cryptanalyst gathers almost twice as many equations, involving however many new variables. Experimental evidence indicates that this technique may often help in solving a system of equations at least up to a certain number of rounds [19]. The second step is

[1] In some variants, as described in [5], no candidate key counters are required; see Section 5 for a brief discussion of this attack.

to consider probabilistic relations that may arise from differential cryptanalysis, giving rise to what we call *Attack-A*.

Attack-A. For the sake of simplicity, we assume the cipher is an Substitution-Permutation-Network (SP-network), which iterates layers of non-linear transformations (e.g. S-Box operations) and affine transformations. Now consider a differential characteristic $\Delta = (\delta_0, \delta_1, \ldots, \delta_r)$ for a number of rounds, where $\delta_{i-1} \rightarrow \delta_i$ is a one-round difference arising from round i and valid with probability p_i. If we assume statistical independence of one-round differences, the characteristic Δ is valid with probability $p = \prod p_i$. Each one-round difference gives rise to equations relating the input and output pairs for active S-Boxes. Let $X'_{i,j}$ and $X''_{i,j}$ denote the j-th bit of the input to the S-Box layer in round i for the systems F' and F'', respectively. Similarly, let $Y'_{i,j}$ and $Y''_{i,j}$ denote the corresponding output bits. Then we have that the expressions

$$X'_{i,j} + X''_{i,j} = \Delta X_{i,j} \rightarrow \Delta Y_{i,j} = Y'_{i,j} + Y''_{i,j},$$

where $\Delta X_{i,j}, \Delta Y_{i,j}$ are known values predicted by the characteristic, are valid with some non-negligible probability q for bits of active S-Boxes. Similarly, for non-active S-Boxes (that are not involved in the characteristic Δ and therefore have input/output difference zero), we have the relations

$$X'_{i,j} + X''_{i,j} = 0 = Y'_{i,j} + Y''_{i,j}$$

also valid with a non-negligible probability.

If we consider the equation system $F = F' \cup F''$, we can combine F and all such linear relations arising from the characteristic Δ. This gives rise to an equation system $\overline{F}$ which holds with probability p. If we attempt to solve such a system for approximately $1/p$ pairs of plaintext–ciphertext, we expect at least one non-empty solution, which should yield the encryption key. For a full algebraic key recover we expect the system $\overline{F}$ to be easier to solve than the original system F' (or F''), because many linear constrains were added without adding any new variables. However, we do not know *a priori* how difficult it will be to solve the system approximately $1/p$ times. This system $\overline{F}$ may be used however to recover some key information, leading to an attack we call *Attack-B*.

Attack-B. Now, assume that we have an SP-network, a differential characteristic $\Delta = (\delta_0, \delta_1, \ldots, \delta_r)$ valid for r rounds with probability p, and (P', P'') a right pair for Δ (so that $\delta_0 = P' \oplus P''$ and δ_r holds for the output of round r). For simplicity, let us assume that only one S-Box is active in round 1, with input $X'_{1,j}$ and $X''_{1,j}$ (restricted to this S-Box) for the plaintext P' and P'' respectively, and that there is a key addition immediately before the S-Box operation, that is

$$S(P'_j \oplus K_{0,j}) = S(X'_{1,j}) = Y'_{1,j} \text{ and } S(P''_j \oplus K_{0,j}) = S(X''_{1,j}) = Y''_{1,j}.$$

The S-Box operation S can be described by a (vectorial) Boolean function, expressing each bit of the output $Y'_{1,j}$ as a polynomial function (over $\mathbb{F}_2$) on the input bits of $X'_{1,j}$ and $K_{0,j}$. If (P', P'') is a right pair, then the polynomial equations arising from the relation $\Delta Y_{1,j} = Y'_{1,j} \oplus Y''_{1,j} = S(P'_j \oplus K_{0,j}) \oplus S(P''_j \oplus K_{0,j})$ give us a very simple equation system to solve, with only the key variables $K_{0,j}$ as unknowns (and which do not vanish identically because we are considering nonzero differences, cf. Section 5). Consequently, if we had an effective distinguisher to determine whether (P', P'') is a right pair, we could learn some bits of information about the round keys involved in the first round active S-Boxes.

Experimentally, we found that, for some ciphers and up to a number of rounds, *Attack-A* can be used as such a distinguisher. More specifically, we noticed that finding a contradiction (i.e. the Gröbner basis equal to $\{1\}$) was much faster than computing the full solution of the system if the system was consistent (that is, when we have a right pair). Thus, rather than fully solving the systems to eventually recover the secret key as suggested in *Attack-A*, the *Attack-B* proceeds by measuring the time t it maximally takes to find that the system is inconsistent[2], and assume we have a right pair if this time t elapsed without a contradiction. One needs to be able to experimentally estimate the time t, but for some ciphers this appears to be an efficient form of attack.

An alternative form of *Attack-B* is to recover key bits from the last round. Assume that the time t passed for a pair (P',P''), i.e. that we probably found a right pair. Now, if we guess and fix some subkey bits in the last rounds, we can check whether the time t still passes without a contradiction. If this happens, we assume that we guessed correctly. However, for this approach to work we need to guess enough subkey bits to detect a contradiction quickly. An obvious choice is to guess all subkey bits involved in the last round, which effectively removes one round from the system.

Attack-C. Experimental evidence with PRESENT (cf. Section 4) indicates that *Attack-B* in fact only relies on the differential $\delta_0 \rightarrow \delta_r$ rather than the characteristic Δ when finding contradictions in the systems. The runtimes for finding contradictions for $N = 17$ and differential characteristic of length $r = 14$ did not differ significantly from the runtimes for the same task with $N = 4$ and $r = 1$ (cf. Appendix C). This indicates that the computational difficulty is mostly determined by the difference $R = N - r$, the number of "free" rounds. We thus define a new attack (*Attack-C*) where we remove the equations for rounds $\leq r$.

This significantly reduces the number of equations and variables. After these equations are removed we are left with R rounds for each plaintext–ciphertext pair to consider; these are related by the output difference predicted by the differential. As a result, the algebraic computation is essentially equivalent to solving a related cipher of $2R - 1$ rounds (from C' to C'' via the predicted difference δ_r) using an algebraic meet-in-the-middle attack [9]. This "cipher" has a symmetric key schedule and only $2R - 1$ rounds rather than $2R$ since the S-Box applications after the difference δ_r are directly connected and lack a key addition and diffusion layer application between them. Thus we can consider

[2] Other features of the calculation — like the size of the intermediate matrices created by F_4 — may also be used instead of the time t.

these two S-Box applications as one S-Box application of S-Boxes S_i defined by the known difference δ_r: $S_i(x_{i,\dots,i+s}) = S(S^{-1}(x_{i,\dots,i+s}) + \delta_{r,(i,\dots,i+s)})$ for $i \in \{0, s, \dots, n\}$ and s the size of the S-Box.

Again, we attempt to solve the system and wait for a fixed time t to find a contradiction in the system. If no contradiction is found, we assume that the differential $\delta_0 \rightarrow \delta_r$ holds. Note that we cannot be certain about the output difference of the first round active S-Boxes (as the attack distinguishes pairs that satisfy *differentials* rather than *characteristics*). However, the attack can be adapted such that we can still recover key bits, for instance by considering multiple right pairs. A second option is to attempt to solve the resulting smaller system, to recover the encryption key. Alternatively, we can execute the guess-and-verify step described above.

To study the viability of these attacks, we describe experiments with reduced-round versions of the block cipher Present.

3 The Block Cipher PRESENT

Present [6] was proposed by Bogdanov et al. at CHES 2007 as an ultra-lightweight block cipher, enabling a very compact implementation in hardware, and therefore particularly suitable for RFIDs and similar devices. There are two variants of Present: one with 80-bit keys and one with a 128-bit keys, denoted as Present-80 and Present-128 respectively. In our experiments, we consider reduced round variants of both ciphers denoted as Present-K_s-N, where $K_s \in \{80, 128\}$ represents the key size in bits and $1 \leq N \leq 31$ represents the number of rounds.

Present is an SP-network with a blocksize of 64 bits and both versions have 31 rounds. Each round of the cipher has three layers of operations: `keyAddLayer`, `sBoxLayer` and `pLayer`. The operation `keyAddLayer` is a simple subkey addition to the current state, while the `sBoxLayer` operation consists of 16 parallel applications of a 4-bit S-Box. The operation `pLayer` is a permutation of wires.

In both versions, these three operations are repeated $N = 31$ times. On the final round, an extra subkey addition is performed. The subkeys are derived from the user-provided key in the key schedule, which by design is also quite simple and efficient involving a cyclic right shift, one ore two 4-bit S-Box applications (depending on the key size) and the addition of a round constant. We note that the difference between the 80-bit and 128-bit variants is only the key schedule. In particular, both variants have the same number of rounds (i.e. $N = 31$). The cipher designers explicitly describe in [6] the threat model considered when designing the cipher, and acknowledge that the security margin may be somewhat tight. Although they do not recommend immediate deployment of the cipher (especially the 128-bit version), they strongly encourage the analysis of both versions.

3.1 Differential Cryptanalysis of 16 Rounds of PRESENT

In the original proposal [6], the designers of Present show that both linear and differential cryptanalysis are infeasible against the cipher. In [27, 28] M. Wang provides 24 explicit differential characteristics for 14 rounds. These hold with probability 2^{-62} and are within the theoretical bounds provided by the Present designers. Wang's attack is reported to require 2^{64} memory accesses to cryptanalyse 16 rounds of Present-80. We use his characteristics (see Appendix B for an example of one of these characteristics) to mount our attack. Furthermore, we also make use of the filter function presented in [27], which we briefly describe below.

Consider for example the differential characteristic provided in Appendix B. It ends with the difference $\delta = \texttt{1001} = \texttt{9}$ as input for the two active S-Boxes of round 15. According to the difference distribution table of the Present S-Box, the possible output differences are `2`, `4`, `6`, `8`, `C` and `E`. This means that the least significant bit is always zero and the weight of the output difference (with the two active S-Box) is at most 6. It then follows from `pLayer` that at most six S-Boxes are active in round 16. Thus we can discard any pair for which the outputs of round 16 have non-zero difference in the positions arising from the output of S-Boxes other than the active ones. There are ten inactive 4-bit S-Boxes, and we expect a pair to pass this test with probability 2^{-40}.

Furthermore, it also follows from `pLayer` that the active S-Boxes in round 16 (which are at most six, as described above) will have input difference `1` and thus all possible output differences are `3`, `7`, `9`, `D` (and `0`, in case the S-Box is inactive). Thus we can discard any pair not satisfying these output differences for these S-Boxes. We expect a pair to pass this test with probability $\frac{16}{5}^{-6} = 2^{-10.07}$. Overall we expect pairs to path both tests with probability $2^{-50.07}$. We expect to be able to construct a similar filter function for all the 24 differential characteristics presented in [28].

4 Experimental Results

To mount the attacks, we generate systems of equations $\overline{F}$ as in Section 2 for pairs of encryptions with prescribed difference as described in Section 3.1, by adding linear equations for the differentials predicted by the 14-round characteristic given in the Appendix. For Present this is equivalent to adding 128 linear equations per round of the form $\Delta X_{i,j} = X'_{i,j} + X''_{i,j}$ and $\Delta Y_{i,j} = Y'_{i,j} + Y''_{i,j}$ where $\Delta X_{i,j}$ and $\Delta Y_{i,j}$ are the values predicted by the characteristic (these are zero for non-active S-Boxes).

To perform the algebraic part of the attack, we use either Gröbner basis algorithms or a SAT-solver: the Singular 3-0-4-4 [20] routine `groebner` with the monomial odering *degrevlex*, the PolyBoRi 0.5rc6 [8] routine `groebner_basis` with the option `faugere=True` and the monomial ordering `dp_asc`, or MiniSat 2.0 beta [16]. We note the maximal time t these routines take to detect a contradiction for a given differential length of r, and assume we have a pair satisfying the characteristic (or differential, in *Attack-C*) if this time t elapsed without a contradiction.

We performed experiments for *Attack-B* and *Attack-C*. Runtimes for *Attack-B* and *Attack-C* are given in Appendix C and D respectively. We note that Attack-C requires about 1GB of RAM to be carried out. The times were obtained on a 1.8Ghz Opteron with 64GB RAM. The attack was implemented in the mathematics software Sage [25].

If a characteristic Δ is valid with probability p, then after approximately $1/p$ attempts we expect to find a right pair and can thus set up our smaller systems for each first round active S-Box. These equations are given in Appendix A. After substitution of $P'_i, P''_i, \Delta Y_i$ and elimination of the variables X'_i, X''_i in the system in Appendix A, we get an equation system with four equations in the four key variables. If we compute the reduced Gröbner basis for this system we recover two relations of the form $K_i + K_j(+1) = 0$ for two key bits K_i,K_j per S-Box, i.e. we recover 2 bits of information per first round active S-Box[3].

Alternatively, we can recover key bits from the last rounds using the guess-and-verify step described above.

4.1 PRESENT-80-16

To compare with the results of [27], we can apply *Attack-C* against reduced round versions of PRESENT-80. Using this approach we expect to learn 4 bits of information about the key for PRESENT-80-16 in about $2^{62-50.07} \cdot 6$ seconds to perform the consistency checks using about 2^{62} chosen plaintext–ciphertext pairs, where 6 seconds represents the highest runtime to find a contradiction we have encountered in our experiments when using POLYBORI. Even if there are instances that take slightly longer to check, we assume that this is a safe margin because there are many shorter runtimes. This time gives a complexity of about 2^{62} ciphertext difference checks and about $2^{11.93} \cdot 6 \cdot 1.8 \cdot 10^9 \approx 2^{46}$ CPU cycles to find a right pair on the given 1.8 Ghz Opteron CPU. We assume that a single encryption costs at least two CPU cycles per round – one for the S-Box lookup and one for the key addition – such that a brute force search would require approximately $16 \cdot 2 \cdot 2^{80} = 2^{85}$ CPU cycles and two plaintext–ciphertext pairs due to the small blocksize.

In [28], 24 different 14-round differentials were presented, involving the 0th, 1st, 2nd, 12th, 13th and 14th S-Boxes in the first round, each having either `7` or `15` as plaintext difference restricted to one active S-Box. From these we expect to recover 18 bits[4] of key information by repeating the attack for those S-Box configurations. We can then guess the remaining $80 - 18 = 62$ bits, and the complete attack has a complexity of about $6 \cdot 2^{62}$ filter function applications, about $6 \cdot 2^{46}$ CPU cycles for the consistency checks and 2^{62} PRESENT applications

[3] This is as expected, since the probability of the differential used in the first round S-Box is 2^{-2}; see Lemma 1.

[4] We are not able to recover 24 bits because we learn some redundant information.

to guess the remaining key bits[5]. (Alternatively, we may add the 18 learned linear key bit equations to any equation system for the related cipher and attempt to solve this system.) The attack in [27] on the other hand requires 2^{64} memory accesses. While this is a different metric — memory access — from the one we have to use in this case — CPU cycles — we can see that our approach has roughly the same time complexity, since the 2^{62} filter function applications cost at least 2^{62} memory accesses. However, our attack seems to have a slightly better data complexity because overall six right pairs are sufficient. When applying the attack against PRESENT-128-16, we obtain a similar complexity. We note however that for PRESENT-K_s-16, we can also make use of backward key guessing to recover more key bits. Because we have distinguished a right pair already we expect the signal to noise ratio to be quite high and thus expect relatively few wrong suggestions for candidate keys.

4.2 PRESENT-128-17

Note that we cannot use the filter function for 17 rounds, thus the attack against PRESENT-80-17 gives worse performance when compared to exhaustive key search. However, it may still be applied against PRESENT-128-17. Indeed, we expect to learn 4 bits of information for PRESENT-128-17 in about $2^{62} \cdot 18$ seconds using about 2^{62} chosen plaintext–ciphertext pairs. This time is equivalent to about $2^{62} \cdot 18 \cdot 1.8 \cdot 10^9 \approx 2^{97}$ CPU cycles. If this approach is repeated 6 times for the different active S-Boxes in the PRESENT differentials, we expect to learn 18 bits of information about the key. We can then guess the remaining $128 - 18 = 110$ bits and thus have a complexity in the order of 2^{110} for the attack.

A better strategy is as follows. We identify one right pair using $2^{62} \cdot 18 \cdot 1.8 \cdot 10^9 \approx 2^{97}$ CPU cycles. Then, we guess 64 subkey bits of the last round and fix the appropriate variables in the equation system for the consistency check. Finally, we attempt to solve this system again, which is equivalent to the algebraic part of the $2R$ attack. We repeat this guess-and-verify step until the right configuration is found, i.e. the system is not inconsistent. This strategy has a complexity of 2^{97} CPU cycles for identifying the right pair and $2^{64} \cdot 6 \cdot 1.8 \cdot 10^9 \approx 2^{98}$ CPU cycles to recover 64 subkey bits. Finally, we can either guess the remaining bits or repeat the guess-and-verify step for $1R$ to recover another 64 subkey bits.

4.3 PRESENT-128-18

We can also attack PRESENT-128-18 using *Attack-C* as follows. First note that the limiting factor for the attack on PRESENT-128-18 is that we run out of plaintext–ciphertext pairs due to the small blocksize. On the other hand, we have not yet reached the time complexity of 2^{128} for 128-bit keysizes. One way

[5] Note that the attack can be improved by managing the plaintext–ciphertext pairs more intelligently and by using the fact that we can abort a PRESENT trial encryption if it does not match the known differential.

to make use of this fact is to again consider the input difference for round 15 and iterate over all possible output differences. As discussed in Section 3.1, we have six possible output differences and two active S-Boxes in round 15, which result in 36 possible output differences in total. We expect to learn 4 bits of information about the key for PRESENT-128-18 in about $36 \cdot 2^{62} \cdot 18$ seconds using about 2^{62} chosen plaintext–ciphertext pairs. This time is equivalent to about $36 \cdot 2^{62} \cdot 18 \cdot 1.8 \cdot 10^9 \approx 2^{102}$ CPU cycles. Again, we can iterate this process six times to learn 18 bits of information about the key and guess the remaining information with a complexity of approximately 2^{110} PRESENT applications.

However, this strategy might lead to false positives for each guessed output difference. To address this we need to run the brute-force attack for the remaining 110 bits for each possible candidate. Thus the overall complexity of the attack is in the order of $36 \cdot 2^{110}$ PRESENT applications. The final brute-force run will require for 2-3 plaintext-ciphertext pairs due to the large key size compared to the blocksize. This hardly affects the time complexity since only candidates passing the first plaintext-ciphertext pair need to be tested against a second and potentially third pair and these candidates are few compared to 2^{110}.

The best approach appears to be the guess-and-verify step from the $3R$ attack, which results in an overall complexity of about $36 \cdot 1.8 \cdot 10^9 (2^{62} \cdot 18 + 2^{64} \cdot 6) \approx 2^{103}$ CPU cycles.

Note that we were unable to reliably detect contradictions directly if $R = N - r \geq 4$ within 24 hours (compared to 18 seconds for $R = 3$).

4.4 PRESENT-128-19

Similarly, we can use the filter function to mount an attack against PRESENT-128-19 by iterating our attack $2^{64-50.07} = 2^{13.93}$ times (instead of 36) for all possible output differences of round 16. The overall complexity of this attack is about $2^{13.97} \cdot 1.8 \cdot 10^9 \cdot (18 \cdot 2^{62} + 6 \cdot 2^{64}) \approx 2^{113}$ CPU cycles.

5 Discussion of the Attack

While the attack has many similarities with conventional differential cryptanalysis, such as the requirement of a high probability differential Δ valid for r rounds and the use of filter functions to reduce the workload, there are however some noteworthy differences. First, *Attack-C* requires fewer plaintext–ciphertext pairs for a given differential characteristic to learn information about the key than conventional differential cryptanalysis, because the attacker does not need to wait for a peak in the partial key counter. Instead one right pair is sufficient. Second, one flavour of the attack recovers more key bits if many S-Boxes are active in the first round. This follows from its reliance on those S-Boxes to recover key information. Also note that while a high probability differential characteristic is required, the attack recovers more bits per S-Box if the differences for the active S-Box in the first round are of low probability. This is a consequence of the simple Lemma below:

Lemma 1. *Given a differential Δ with a first round active S-Box with a difference that is true with probability 2^{-b}, then* Attack-B *and* Attack-C *can recover b bits of information about the key from this S-Box.*

Finally, key-recovery differential cryptanalysis is usually considered infeasible if the differential Δ is valid for r rounds, and r is much less than the full number of rounds N, since backward key guessing for $N-r$ rounds may become impractical. In that case the *Attack-C* proposed here could *possibly* still allow the successful cryptanalysis of the cipher. However, this depends on the algebraic structure of the cipher, as it may be the case that the time required for the consistency check is such that the overall complexity remains below the one required for exhaustive key search.

We note that *Attack-C* shares many properties with the differential cryptanalysis of the full 16-round DES [5]. Both attacks are capable of detecting a right pair without maintaining a candidate key counter array. Also, both attacks use active S-Boxes of the outer rounds to recover bits of information about the key once such a right pair is found. In fact, one could argue that *Attack-C* is a generalised algebraic representation of the technique presented in [5]. From this technique *Attack-C* inherits some interesting properties: first, the attack can be carried out fully in parallel because no data structures such as a candidate key array need to be shared between the nodes. Also, we allow the encryption keys to change during the data collection phase because exactly one right pair is sufficient to learn some key information. However, if we try to learn further key bits by repeating the attack with other characteristics we require the encryption key not to change. We note however that while the attack in [5] seems to be very specific to the target cipher DES, *Attack-C* can in principle be applied to any block cipher. Another way of looking at *Attack-C* is to realise that it is in fact is a quite expensive but thorough filter function: we invest more work in the management of the outer rounds using algebraic techniques.

In the particular case of Present-80-N, our attack seems to offer only marginal advantage when compared with the differential attack presented in [27]: it should require slightly less data to distinguish a right pair and similar overall complexity. On the other hand, for Present-128-N this attack seems to perform better than the one in [27]. As in this case the limiting factor is the data and not the time complexity of the attack, i.e. we run out of plaintext–ciphertext pairs before running out of computation time, the attack has more flexibility.

The use of Gröbner bases techniques to find contradictions in propositional systems is a well known idea [10]. In the context of cryptanalysis, it is also a natural idea to try to detect contradictions to attack a cipher. However, in probabilistic approaches used in algebraic attacks, usually key bits are guessed. This is an intuitive idea because polynomial systems tend to be easier to solve the more overdefined they are and because the whole system essentially depends on the key. Thus guessing key bits is a natural choice. However this simplification seems to bring few benefits to the attacker, and more sophisticated probabilistic approaches seem so far to have been ignored. The method proposed in this paper can thus highlight the advantages of combining conventional (statistical) cryptanalysis and algebraic cryptanalysis. By considering differential cryptanalysis we showed how to construct an equation system for a structurally weaker and shorter related "cipher" which can then be studied independently. To attack this "cipher" algebraic attacks seem to be the natural choice since very few "plaintext–ciphertext" pairs are available but the "cipher" has few rounds (i.e. $2R-1$). However, other techniques might also be considered.

Future research might also investigate the use of other well established (statistical) cryptanalysis techniques in combination with algebraic cryptanalysis such as linear cryptanalysis (defining a version of *Attack-A* in this case is straightforward), higher order and truncated differentials, the Boomerang attack or impossible differentials.

We note that this attack may also offer a high degree of flexibility for improvements. For example, the development of more efficient algorithms for solving systems of equations (or good algebraic representation of ciphers that may result in more efficient solving) would obviously improve the attacks proposed. For instance, by switching from SINGULAR to POLYBORI for *Attack-B*, we were able to make the consistency check up to 60 times faster[6]. As an illustration of the forementioned flexibility, if for instance an attacker could make use of an optimised method to find contradictions in $t \ll 2^{128-62} = 2^{66}$ CPU cycles for PRESENT-128-20, this would allow the successful cryptanalysis of a version of PRESENT with 6 more rounds than the best known differential, which is considered "a situation without precedent" by the cipher designers [6]. This task is equivalent to mount a meet-in-the-middle attack against an 11 round PRESENT-like cipher with a symmetric key schedule. Unfortunately with the available computer resources, we are not able to verify whether this is currently feasible.

We ran simulations with small numbers of rounds to verify that the attack indeed behaves as expected. For instance, when using a 3R *Attack-C* against PRESENT-80-6 and PRESENT-80-7 we found right pairs with the expected number of trials. Also, as expected we saw false positives, i.e. the attack suggested wrong information. However, a majority vote on a small number of runs (e.g., 3) always recovered the correct information. We are of course aware that it is in general difficult to reason from small scale examples to bigger instances.

Finally, as our results depend on experimental data and the set of data we evaluated is rather small due to the time consuming nature of our experiments, we make our claims verifiable by providing the source code of the attack online `http://bitbucket.org/malb/algebraic_attacks/src/tip/present.py`.

6 Conclusion

We propose a new cryptanalytic technique combining differential cryptanalysis and algebraic techniques. We show that in some circumstances this technique can be effectively used to attack block ciphers, and in general may offer some advantages when compared to differential cryptanalysis. As an illustration, we

[6] We did not see any further speed improvement by using e.g. MAGMA 2.14 [7]

applied it against reduced versions of Present-80 and Present-128. While this paper has no implications for the security of either Present-80 or Present-128, it was shown that the proposed techniques can improve upon existing differential cryptanalytic methods using the same difference characteristics. In particular we were able to recover key information for Present-128 reduced to 19 rounds. Also we pointed out promising research directions for the field of algebraic attacks.

Acknowledgements

The work described in this paper has been supported in part by the European Commission through the IST Programme under contract ICT-2007-216646 ECRYPT2. We would like to thank William Stein for allowing the use of his computers[7]. We also would like to thank Sean Murphy, Matt Robshaw, Ludovic Perret, Jean-Charles Faugère and anonymous referees for helpful comments.

References

1. Gregory V. Bard. *Algorithms for Solving Linear and Polynomial Systems of Equations over Finite Fields with Applications to Cryptanalysis.* PhD thesis, University of Maryland, 2007.
2. Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson. Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers. Cryptology ePrint Archive, Report 2007/024, 2007. Available at `http://eprint.iacr.org/2007/024`.
3. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
4. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology — CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 3–72. Springer Verlag, 1991.
5. Eli Biham and Adi Shamir. Differential Cryptanalysis of the Full 16-round DES. In *Advances in Cryptology — CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496, Berlin Heidelberg New York, 1991. Springer Verlag.
6. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, Matthew Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *CHES 2007*, volume 7427 of *Lecture Notes in Computer Science*, pages 450–466. Springer Verlag, 2007. Available at `http://www.crypto.rub.de/imperia/md/content/texte/publications/conferences/present_ches2007.pdf`.
7. Wieb Bosma, John Cannon, and Catherine Playoust. The MAGMA Algebra System I: The User Language. In *Journal of Symbolic Computation 24*, pages 235–265. Academic Press, 1997.
8. Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner basis computations with Boolean polynomials. In *Electronic Proceedings of MEGA 2007*, 2007. Available at `http://www.ricam.oeaw.ac.at/mega2007/electronic/26.pdf`.

[7] purchased under National Science Foundation Grant No. 0555776 and National Science Foundation Grant No. DMS-0821725

9. Carlos Cid, Sean Murphy, and Matthew Robshaw. *Algebraic Aspects of the Advanced Encryption Standard*. Springer Verlag, 2006.
10. Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 174–183, 1996. Available at `http://www.cse.yorku.ca/~jeff/research/proof_systems/grobner.ps`.
11. Nicolas T. Courtois and Gregory V. Bard. Algebraic Cryptanalysis of the Data Encryption Standard. In Steven D. Galbraith, editor, *Cryptography and Coding – 11th IMA International Conference*, volume 4887 of *Lecture Notes in Computer Science*, pages 152–169, Berlin Heidelberg New York, 2007. Springer Verlag. Available at `http://eprint.iacr.org/2006/402`.
12. Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer Verlag, 2000.
13. Nicolas T. Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer Verlag, 2003.
14. Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, 2002. Available at `http://eprint.iacr.org/2002/044`.
15. Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES - the Advanced Encryption Standard*. Springer Verlag, 2002.
16. Nicklas Een and Nicklas Sörensson. An extensible SAT-solver. In *Proceedings of SAT '03*, pages 502–518, 2003. Available at `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/`.
17. Jean-Charles Faugère. A New Efficient algorithm for Computing Gröbner Basis (F4), 1999. Available at `http://modular.ucsd.edu/129-05/refs/faugere_f4.pdf`.
18. Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). In *Proceedings of ISSAC*, pages 75–83. ACM Press, 2002.
19. Jean-Charles Faugère. Groebner bases. Applications in cryptology. FSE 2007 – Invited Talk, 2007. Available at `http://fse2007.uni.lu/v-misc.html`.
20. Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. Singular 3.0. A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra, University of Kaiserslautern, 2005. Available at: `http://www.singular.uni-kl.de`.
21. L.R. Knudsen. Truncated and higher order differentials. In *Fast Software Encryption 1995*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer Verlag, 1995.
22. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology — EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Verlag, 1993. Available at `http://homes.esat.kuleuven.be/~abiryuko/Cryptan/matsui_des.PDF`.
23. Sean Murphy and Matthew Robshaw. Essential Algebraic Structure Within the AES. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer Verlag, 2002. Available at `http://www.isg.rhul.ac.uk/~mrobshaw/rijndael/aes-crypto.pdf`.

24. Håvard Raddum and Igor Semaev. New technique for solving sparse equation systems. Cryptology ePrint Archive, Report 2006/475, 2006. Available at `http://eprint.iacr.org/2006/475`.
25. The SAGE Group. *SAGE Mathematics Software (Version 3.3)*, 2008. Available at `http://www.sagemath.org`.
26. David Wagner. The boomerang attack. In *Fast Software Encryption 1999*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer Verlag, 1999. Available at `http://www.cs.berkeley.edu/~daw/papers/boomerang-fse99.ps`.
27. Meiqin Wang. Differential Cryptanalysis of reduced-round PRESENT. In Serge Vaudenay, editor, *Africacrypt 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 40–49. Springer Verlag, 2008.
28. Meiqin Wang. Private communication: 24 differential characteristics for 14-round present we have found, 2008.
29. Bo-Yin Yang, Jiun-Ming Chen, and Nicolas T. Courtois. On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis. In *Proceedings of Information and Communications Security; 6th International Conference 2004*, volume 3269 of *Lecture Notes in Computer Science*, pages 401–413. Springer Verlag, 2004.

A Small Key Bit Recovery System

$$
\begin{aligned}
X_0' &= K_0 + P_0', \quad X_1' = K_1 + P_1', \quad X_2' = K_2 + P_2', \quad X_3' = K_3 + P_3',\\
Y_0' &= X_0'X_1'X_3' + X_0'X_2'X_3' + X_0' + X_1'X_2'X_3' + X_1'X_2' + X_2' + X_3' + 1,\\
Y_1' &= X_0'X_1'X_3' + X_0'X_2'X_3' + X_0'X_2' + X_0'X_3' + X_0' + X_1' + X_2'X_3' + 1,\\
Y_2' &= X_0'X_1'X_3' + X_0'X_1' + X_0'X_2'X_3' + X_0'X_2' + X_0' + X_1'X_2'X_3' + X_2',\\
Y_3' &= X_0' + X_1'X_2' + X_1' + X_3',\\
X_0'' &= K_0 + P_0'', \quad X_1'' = K_1 + P_1'', \quad X_2'' = K_2 + P_2'', \quad X_3'' = K_3 + P_3'',\\
Y_0'' &= X_0''X_1''X_3'' + X_0''X_2''X_3'' + X_0'' + X_1''X_2''X_3'' + X_1''X_2'' + X_2'' + X_3'' + 1,\\
Y_1'' &= X_0''X_1''X_3'' + X_0''X_2''X_3'' + X_0''X_2'' + X_0''X_3'' + X_0'' + X_1'' + X_2''X_3'' + 1,\\
Y_2'' &= X_0''X_1''X_3'' + X_0''X_1'' + X_0''X_2''X_3'' + X_0''X_2'' + X_0''X_1''X_2''X_3'' + X_2'',\\
Y_3'' &= X_0'' + X_1''X_2'' + X_1'' + X_3'',\\
\Delta Y_0 &= Y_0' + Y_0'', \quad \Delta Y_1 = Y_1' + Y_1'', \quad \Delta Y_2 = Y_2' + Y_2'', \quad \Delta Y_3 = Y_3' + Y_3'',
\end{aligned}
$$

where ΔY_i are the *known* difference values predicted by the characteristic.

B 14-round Differential Characteristic for PRESENT

Rounds		Differences	Pr	Rounds		Difference	Pr
I		$x_2 = 7$,$x_{14} = 7$	1				
R1	S	$x_2 = 1$,$x_{14} = 1$	2^{-4}	R8	S	$x_8 = 9$,$x_{10} = 9$	2^{-4}
R1	P	$x_0 = 4$,$x_3 = 4$	1	R8	P	$x_2 = 5$,$x_{14} = 5$	1
R2	S	$x_0 = 5$,$x_3 = 5$	2^{-4}	R9	S	$x_2 = 1$,$x_{14} = 1$	2^{-6}
R2	P	$x_0 = 9$,$x_8 = 9$	1	R9	P	$x_0 = 4$,$x_3 = 4$	1
R3	S	$x_0 = 4$,$x_8 = 4$	2^{-4}	R10	S	$x_0 = 5$,$x_3 = 5$	2^{-4}
R3	P	$x_8 = 1$,$x_{10} = 1$	1	R10	P	$x_0 = 9$,$x_8 = 9$	1
R4	S	$x_8 = 9$,$x_{10} = 9$	2^{-4}	R11	S	$x_0 = 4$, $x_8 = 4$	2^{-4}
R4	P	$x_2 = 5$,$x_{14} = 5$	1	R11	P	$x_8 = 1$, $x_{10} = 4$	1
R5	S	$x_2 = 1$,$x_{14} = 1$	2^{-6}	R12	S	$x_8 = 9$, $x_{10} = 9$	2^{-4}
R5	P	$x_0 = 4$,$x_3 = 4$	1	R12	P	$x_2 = 5$, $x_{14} = 5$	1
R6	S	$x_0 = 5$,$x_3 = 5$	2^{-4}	R13	S	$x_2 = 1$, $x_{14} = 1$	2^{-6}
R6	P	$x_0 = 9$,$x_8 = 9$	1	R13	P	$x_0 = 4$, $x_3 = 4$	1
R7	S	$x_0 = 4$,$x_8 = 4$	2^{-4}	R14	S	$x_0 = 5$, $x_3 = 5$	2^{-4}
R7	P	$x_8 = 1$,$x_{10} = 1$	1	R14	P	$x_0 = 9$, $x_8 = 9$	1

C Times in seconds for *Attack-B*

N	K_s	r	p	#trials	SINGULAR	#trials	POLYBORI
4	80	4	2^{-16}	20	$11.92 - 12.16$	50	$0.72 - 0.81$
4	80	3	2^{-12}	10	$106.55 - 118.15$	50	$6.18 - 7.10$
4	80	2	2^{-8}	10	$119.24 - 128.49$	50	$5.94 - 13.30$
4	80	1	2^{-4}	10	$137.84 - 144.37$	50	$11.83 - 33.47$
8	80	5	2^{-22}	0	N/A	50	$18.45 - 63.21$
10	80	8	2^{-34}	0	N/A	20	$21.73 - 38.96$
10	80	7	2^{-30}	0	N/A	10	$39.27 - 241.17$
10	80	6	2^{-26}	0	N/A	20	$56.30- > 4$ hours
16	80	14	2^{-62}	0	N/A	20	$43.42 - 64.11$
16	128	14	2^{-62}	0	N/A	20	$45.59 - 65.03$
16	80	13	2^{-58}	0	N/A	20	$80.35 - 262.73$
16	128	13	2^{-58}	0	N/A	20	$81.06 - 320.53$
16	80	12	2^{-52}	0	N/A	5	> 4 hours
17	80	14	2^{-62}	10	$12,317.49 - 13,201.99$	20	$55.51 - 221.77$
17	128	14	2^{-62}	10	$12,031.97 - 13,631.52$	20	$94.19 - 172.46$
17	80	13	2^{-58}	0	N/A	5	> 4 hours
17	128	13	2^{-58}	0	N/A	5	> 4 hours

D Times in seconds for *Attack-C*

N	K_s	r	p	#trials	SINGULAR	#trials	POLYBORI	#trials	MINISAT2
4	80	4	2^{-16}	10	$0.07 - 0.09$	50	$0.05 - 0.06$	0	N/A
4	80	3	2^{-12}	10	$6.69 - 6.79$	50	$0.88 - 1.00$	50	$0.14 - 0.18$
4	80	2	2^{-8}	10	$28.68 - 29.04$	50	$2.16 - 5.07$	50	$0.32 - 0.82$
4	80	1	2^{-4}	10	$70.95 - 76.08$	50	$8.10 - 18.30$	50	1.21 - 286.40
16	80	14	2^{-62}	10	$123.82 - 132.47$	50	$2.38 - 5.99$	0	N/A
16	128	14	2^{-62}	0	N/A	50	$2.38 - 5.15$	0	N/A
16	80	13	2^{-58}	10	$301.70 - 319.90$	50	$8.69 - 19.36$	0	N/A
16	128	13	2^{-58}	0	N/A	50	$9.58 - 18.64$	0	N/A
16	80	12	2^{-52}	0	N/A	5	> 4 hours	0	N/A
17	80	14	2^{-62}	10	$318.53 - 341.84$	50	$9.03 - 16.93$	50	$0.70 - 58.96$
17	128	14	2^{-62}	0	N/A	50	$8.36 - 17.53$	50	$0.52 - 8.87$
17	80	13	2^{-58}	0	N/A	5	> 4 hours	5	> 4 hours