IST-2002-507932

ECRYPT

European Network of Excellence in Cryptology

Network of Excellence

Information Society Technologies

D.STVL.2
AES Security Report

Due date of deliverable: 31. September 2005
Revised: 30 January 2006

Start date of project: 1 February 2004 Duration: 4 years

Lead contractor: France Telecom - Research and Development Division

Revision 1.0

Project co-funded by the European Commission within the 6th Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission services)	
RE	Restricted to a group specified by the consortium (including the Commission services)	
CO	Confidential, only for members of the consortium (including the Commission services)	

AES Security Report

Editors
Carlos Cid (RHUL) and Henri Gilbert (FTRD)

Contributors
Daniel Augot (INRIA), Alex Biryukov (KUL), Anne Canteaut (INRIA),
Nicolas Courtois (ALT), Christophe De Cannière (KUL), Cédric Lauradoux (INRIA),
Matthew Parker (UiB), Bart Preneel (KUL), Matt Robshaw (FTRD), and Yannick Seurin (FTRD)

30 January 2006
Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Executive Summary

In October 2000, NIST announced the selection of Rijndael, a symmetric cipher proposed by Joan Daemen and Vincent Rijmen, as the Advanced Encryption Standard (AES). AES was published as a FIPS standard in November 2001 [101], and is gradually becoming one of the most widely used block ciphers worldwide.

This report was prepared by a group of cryptography experts from industry and academia, as part of the joint research achieved within the Symmetric Techniques Virtual Laboratory (STVL) of the European project ECRYPT. It was motivated by the fact that since the selection of the AES five years ago, there have been a variety of research results leading to speculation about the long-term security of the cipher. Thus the purpose of this report is to provide an update on the recent advances in various aspects of the analysis of AES and to assess their impact upon the practical security of the cipher.

In order to minimize overlap with the content of former reports on the security of AES, namely the one published in 2000 by NIST [73] and the one published in February 2003 by the European project NESSIE [102], this report only provides a very short account of results already mentioned in these reviews and mainly focuses on more recent security results. Comprehensive surveys of design and implementation aspects of AES are available in the author's book "The Design of Rijndael" [52] and on the AES lounge maintained by the Crypto group of the Technology University of Graz [72].

The following aspects of the AES security are investigated in this report:

- Resistance to statistical attacks (Chapter 1);
- Resistance to attacks exploiting multiset properties (Chapter 2);
- Resistance to algebraic attacks (Chapter 3);
- Resistance to generic trade off attacks (Chapter 4);
- Special properties not directly related to potential attacks (Chapter 5);
- Potential vulnerability of AES implementations to cache-timing attacks (Chapter 6).

Thus, the main classes of known attacks against block ciphers are considered. For each class of attacks, recent advances in the analysis of the security of AES are summarized.

While issues related to the potential vulnerability of AES to so-called algebraic attacks [48, 49, 99] remain somewhat opaque, the AES cannot currently be considered vulnerable

to such analysis. Instead, security aspects to the implementation of the AES may be the most pressing issue, as is the case for many cryptographic primitives. So, while not directly related to the cryptographic strength of the AES, cache-timing attacks against unprotected implementations seem to present a practical threat, at least when the AES is not executed in a trusted environment.

Implementation attacks aside, after considering the state of the art of block cipher cryptanalysis, the conclusion of this report is that, five years after publication, there are still no discernible cryptographic weaknesses in the AES.

Contents

Chapter 1

Resistance to Statistical Attacks

We discuss here an area of cryptanalysis that is quite recent and far from being broadly understood. It does not make a lot of sense to propose and discuss the "most general attack on block ciphers we can think of" that generalizes all the classical attacks we know. This is because, if it is excessively general, maybe we will not learn much, and fail to see much less general, but also much more interesting special cases of it.

Several authors have proposed (several) approaches that generalize linear and differential cryptanalysis and give interesting attacks on (some) ciphers that are not broken by other means. There are obvious (and less obvious) connections and similarities between these approaches, but, unfortunately, we are at present not ready to formulate a full theory of these attacks. Our goal will be to show the main contributions and ideas, that could be applied to AES in the future. There is moreover little hope that we can quickly evaluate the potential of applying these (already very rich) attacks to AES right now. It will require rather a whole lot of new research.

1.1 Linear and Differential Attacks on AES

There has been little recent work on improved classical linear and/or differential attacks on AES. This is not surprising, as AES is designed to be immune to such attacks. Such immunity arises from the strong diffusion layer that occurs at every round of the SPN (substitution-permutation-network). By employing a byte-level MDS (maximum-distance-separable) linear transformation to realise the diffusion, the designers have ensured that any approximation of the cipher by linear relations and/or linear differentials must activate a majority of the constituent s-boxes, thereby guaranteeing that the product of the biases is so small that the cipher is practically secure [52]. However results by Park et al [106], Sano et al [109], and Keliher [85, 86], have managed to provide lower and upper bounds on the *Maximum Expected Linear Probability* (MELP) and *Maximum Expected Differential Probability* (MEDP) for AES. Table 1.1 shows the best current upper bounds on MELP and MEDP achievable, as collated by [86], where T is the number of core rounds of AES, such that the total number of rounds of AES is $R = T + 1$. The last entry in Table 1.1 is an improvement in MELP and was obtained by Keliher by modifying an algorithm called KMT2 [87, 85]. This algorithm is a general

recursive algorithm to compute an upper bound on MELP (the dual algorithm, KMT2-DC, similarly computes an upper bound on MEDP), and works on any SPN. The modification improves the algorithm by incorporating other superior upper bounds not arising from the natural round-recursion of the algorithm, such as those obtained by Park et al [106] for $T \geq 4$. However no improvement is possible in the upper bound on the differential via a similar modification to KMT2-DC.

Table 1.1: Previous Upper Bounds on the MELP and MEDP for AES

MELP	MEDP	Range of Rounds
2^{-24}	2^{-24}	$T \geq 2$
2^{-75}	2^{-75}	$T \geq 7$
$2^{-92.4}$	$2^{-95.1}$	$T \geq 9$
2^{-96}	2^{-96}	$T \geq 4$
1.06×2^{-96}	1.06×2^{-96}	$T \geq 4$
1.075×2^{-106}[106]	1.144×2^{-111}[106]	$T \geq 4$
1.778×2^{-107}[86]		$T \geq 8$

In addition to these results, Keliher [86] is also able to lower and upper-bound 2-round MELP and MEDP to give,

$$\begin{aligned} 1.638 \times 2^{-28} \leq &\quad \text{MELP} \quad \leq 1.44 \times 2^{-27} \\ 1.656 \times 2^{-29} \leq &\quad \text{MEDP} \quad \leq 1.23 \times 2^{-28} \end{aligned}$$

although Keliher [86] also states that the upper bounds cannot be tight, by identifying 'worst-case' situations which can never occur.

1.2 More General Statistical Models for Block Ciphers

Due to the paucity of new classical linear/differential results for AES, the research community has been forced to consider significant generalisations of linear and differential attacks. One can view the current state-of-the-art in this context to be in a phase of consolidation, where previous statistical techniques are being unified so as to clarify the picture and to act as a springboard for new attacks.

The idea of "statistical cryptanalysis" (formulation proposed by Vaudenay in [120] and further formalised in Section 2.1. of [82]) consists of reducing both

- the space of plaintext and ciphertext pairs denoted $\mathcal{P} \times \mathcal{C}$
- and the space of the keys denoted $\mathcal{K}$

to much smaller spaces called respectively $\mathcal{S}$ (sample space) and $\mathcal{L}$ by projection (i.e. eliminating some information unrelated to the attack). Then we define another small space $\mathcal{Q}$ that describes the information on some bits inside the encryption process that we are looking

at. Then if all the spaces are small (it is an essential requirement) it is possible handle the exhaustive analysis of all possible cases and their relative probabilities. From this one can design "statistical distinguishers" that are defined as finding some function $f_3 : \mathcal{L} \times \mathcal{S} \rightarrow \mathcal{Q}$ such that given some set of samples $s_i \in S, i = 1, \ldots$ the distribution of $\{f_3(s_i, l_r), i = 1, \ldots\}$ should be statistically distinguishable from $\{f_3(s_i, l_w), i = 1, \ldots\}$ with respectively l_r and l_w denoting the right and wrong key.
This formalization encompasses differential, linear and many other known attacks. It is possible to see that there are two distinct phases in designing such attacks.

- First can be described as "finding and combining biases". It amounts to observing some characteristics, and combining them for a whole cipher, to get a "heuristic version" of f_3 (that already works well in practice). This is what (in general) most cryptanalysts do.
- Then, at a later stage, it is possible to design an optimal attack in terms of statistical information derivation (or hypothesis verification), that can refine the first "heuristic" version and decrease (sometimes only slightly) the number of plaintext-ciphertext pairs needed for the attack. For more details on how to handle this statistical analysis and explicit examples see [120, 80, 81, 82, 5].

The linear cryptanalysis found by Matsui (LC) is the best known plaintext attack on DES, see [96, 97, 80]. A straightforward way of extending linear attacks is to consider nonlinear multivariate equations. Exact multivariate equations can give a tiny improvement to the last round of a linear attack, as shown at Crypto'98 [84]. A more powerful idea is to use probabilistic multivariate equations, for every round, and replace Matsui's biased linear I/O sums by nonlinear I/O sums as proposed by Harpes, Kramer, and Massey at Eurocrypt'95 [63]. This is known as Generalized Linear Cryptanalysis (GLC). Another name that can be used to refer to attacks involving non linear multivariate equations is **Multivariate Cryptanalysis** proposed by Jacques Patarin. The meaning of **Multivariate Cryptanalysis** is however much broader, it can describe also many other attacks that use multivariate equations in cryptanalysis of not only block ciphers, but also stream ciphers and multivariate public key cryptosystems.

In [119], Vaudenay proposes generalised models for block cipher design in conjunction with various statistical attack models. The aim is to provide security proofs for relatively generic block cipher models when subject to distinguishing attacks which exploit the notion of *d-wise decorrelation* and associated *decorrelation distance*. Such distance measures are based on certain *matrix norms* where the *d-wise distribution matrix* in question, $[F]$, contains the statistical information regarding the input and output of the function or permutation. Specifically, let F be a random function (or permutation) from a given set $\mathcal{M}_1$ to a given set $\mathcal{M}_2$. Then the (x, y)-entry of $[F]$, corresponding to the multi-points $x = (x_0, \ldots, x_{d-1})$ and $y = (y_0, \ldots, y_{d-1})$, is defined as the probability that one simultaneously has $F(x_i) = y_i$, $\forall i$. [119] compares the function (permutation) under test with an ideal (uniform) function (permutation). The associated decorrelation distances are multiplicative and, therefore, can be applied to product ciphers. [119] provides 'cheap' and 'efficient' decorrelated functions or permutations that can satisfy *perfect d-wise decorrelation* up to $d = 2$. [119] shows how to construct d-limited distinguishers and then shows that there is an upper-limit on d for the resistance of a block cipher against such distinguishers, this being set by the number of bits

of randomness that the cipher uses (i.e. the number of key bits). However [119] demonstrates that a low d-decorrelation resistance is enough to resist a large class of distinguishers that capture many of the existing attack methods. Vaudenay also describes an attack model for which low-d decorrelation resistance is not sufficient. In fact an attack of this type was demonstrated by [13]. In spite of the generality of the statistical model, [119] observes that it remains problematic to estimate the decorrelation bias for AES.

In [20], Biryukov, De Canniere and Quisquater propose a formal statistical framework for block cipher attacks based on *multiple linear approximations*, building on the ideas of Kaliski, Burton and Robshaw [83], who exhibit 10 006 linear characteristics for 14 rounds of DES. The approach derives compact expressions in terms of the bias of the approximations and the amount of data available to the attacker. The results demonstrate that multiple linear approximations can significantly improve on classical linear attacks. The attacks consist of three phases:

- The distillation phase, which is applied to each plaintext/ciphertext pair, resulting in a vector containing all relevant information, and then further reduced to a vector of counters where possible.
- The analysis phase, generating a list of equivalent key classes, then determining which of these classes is most likely to contain the true key.
- The search phase, exhaustively trying all keys in the classes suggested until the correct key is found.

[20] compares attacks by a measure called *gain*, γ, where

$$\gamma = -\log_2 \frac{2M-1}{2^n},$$

where M is the average number of candidate keys checked, and the key is of length n bits. Using Bayesian techniques the authors compute the probability that a particular counter vector is generated by a particular key class, and then approximate the resultant m-binomial distribution with an m-dimensional Gaussian distribution, where m is the number of approximations used. This leads to the conclusion that the relative likelihood of a key class, z, is completely determined by the Euclidean distance $|\hat{\mathbf{c}}-\mathbf{c_z}|$, where $\hat{\mathbf{c}}$ is an m-dimensional vector containing the estimated imbalances derived from the known texts, and $\mathbf{c_z} = ((-1)^{z_0}c_0, \ldots, (-1)^{z_{m-1}}c_{m-1})$. This geometrical interpretation allows the authors to derive estimates on the attack gain. They also define an important quantity, the *capacity*, $\bar{c}^2$, of a system of m approximations, where,

$$\bar{c}^2 = 4\sum_{j=0}^{m-1} \epsilon_j^2,$$

where ϵ_j is the well-known *bias* of the linear approximation. Experiments on DES show that the statistical model of [20] obtains a significant improvement over classical linear cryptanalysis, and the authors suggest that the gains of the scheme for AES could be even greater, due to the artificially flat bias profile of AES, which comes at the expense of a large increase in the number of approximations which have the same bias. However [20] state that, since AES

has such a large security margin against linear cryptanalysis, then they do not expect that multiple linear attacks will pose a practical threat to the security of AES.

Wagner [122] has proposed a statistical framework that unifies many attacks on *product ciphers*, such as linear, differential, differential-linear, mod n, truncated differential, impossible differential, higher-order differential, and interpolation attacks. The unifying framework exploits a so-called *stochastic commutative diagram*, which combines the ideas of *projection* and *Markov transition matrices* so as to facilitate the concatenating of non-trivial local properties into a non-trivial global property over the whole cipher. The compositional behaviour of commutative diagrams can only be pieced together if the local commutative diagrams interface using the same projection. These commutative diagrams each hold with some probability, as encoded by the transition matrices. Wagner [122] then exploits the *decorrelation theory* of Vaudenay [119] to derive the *advantage* of such distinguishers, where the global transition matrix is computed from the round transition matrices by multiplication, The model can then be used as a *distinguisher* between the cipher and a random permutation, and this in turn would lead to a key-recovery technique. Given the global transition matrices, M and M' for the cipher and a random permutation, respectively, then the advantage of one plaintext query is computed as the matrix norm $\frac{1}{2}||M - M'||_\infty$. The success of the technique depends on identifying projections, ρ_0, where the domain is the l-bit round input space, $\mathcal{M} : \{0,1\}^l$, and the range, Y_0, has small dimension. Similarly, one must identify an output projection, ρ_1, whose domain is the l-bit round output space, $\mathcal{M} : \{0,1\}^l$, and the range, Y_1, is again small. For instance, for linear cryptanalysis for one round of the cipher, one identifies preferred input and output linear equations Γ_0 and Γ_1 , from $\{0,1\}^l \to \{0,1\}$, so that $\rho_0 = \Gamma_0$ and $\rho_1 = \Gamma_1$. The associated 2×2 Markov transition matrix, M_0, then completes the commutative diagram so that we have,

$$\begin{array}{ccc} \mathcal{M} & \stackrel{\rho_0}{\rightarrow} & Y_0 \\ f_0 \downarrow & & \downarrow M_0 \\ \mathcal{M} & \stackrel{\rho_1}{\rightarrow} & Y_1 \end{array}$$

Such commutative diagrams are concatenated on condition that the projections at the interfaces match. To model multiple linear approximations one then identifies linear equations $\Gamma_{i,0}, \Gamma_{i,1}, \ldots, \Gamma_{i,d-1}$ which then form a projection down to a space of dimension 2^d. The associated transition matrix, M_i, will then be of dimension 2^d so, clearly, d cannot be too large. The modelling of higher-order approximations such as differential cryptanalysis is then made possible by dealing with more than one plaintext at each step. For instance, for differential cryptanalysis, ρ would take two plaintexts, x and x', at a time and compute the projection $\rho(x, x') = x - x'$. Thus the above scheme is simply generalised to a vectorial version. The generality of the stochastic commutative diagram allows for any combination of the well-known statistical attacks, and generalisations thereof, on condition that suitable projections can be found that interface between rounds. AES is designed to be immune to linear and differential projections, so the challenge is to find novel projections on small sets of plaintexts which hold with sufficiently high probability, and whose range is small in comparison to the domain. These stipulations ensure that a distinguisher based on these projections can be computed tractably and with sufficient advantage.

[122] draws on the prior χ^2 work of Vaudenay [120] which uses linear projections, and

the work of Harpes and Massey [64] which uses nonlinear projections. The χ^2 measure is related to the l_2-matrix-norm, $||M - M'||_2$, as opposed to the l_∞-norm of [122], and can therefore be viewed as a variant of a stochastic commutative diagram. This variant has been further developed by Baigneres, Junod, and Vaudenay in [5]. They develop a metric that is shown to be related to the *relative entropy* between two distributions which, in turn leads to the metric of *Squared Euclidean Imbalance* if one of those distributions is uniform. The authors then identify that, in order to realise practically realisable distinguishers, they must apply the idea of projection, as was done in [122]. This once again leads to the notion of a commutative diagram based on Markov processes and, in particular, to the notion of a *bias matrix* between transition matrices, $B = M_0 - M_1$, so that the optimal distinguisher is based on a metric of the form $||B||_2^2$. The authors state that the classical block cipher design criterion of resistance to linear cryptanalysis, as used by AES, implies a (somewhat weaker) resistance to generalised $\mathcal{Z}_2$-linear projections but does not extend to projections outside this class. In fact, both Parker [107] and Standaert [113] have identified new forms of projection which obtain much higher biases. But Parker [107] notes that it is not at all clear how to piece these approximations together across multiple rounds.
In [65, 66] Harpes introduces partitioning cryptanalysis (PC) and shows that it generalizes both LC and GLC. A numerical relationship between this type of attacks that use partial information, and the best linear attack is shown in [82]. The correlation cryptanalysis (CC) introduced in Jakobsen's master thesis [75] is claimed even more general. These attacks can also be described, studied (and certainly improved) in the aforementioned framework of statistical cryptanalysis [120, 82, 5] that tends nevertheless to "forget" the nature of the attack and how it was found in the first place, and therefore can be applied at a later stage.
In [77] Jakobsen shows that all these attacks, including also Differential Cryptanalysis are closely related and can be studied in terms of the Fast Fourier Transform for the cipher round function. Unfortunately, computing this transform is in general infeasible for a real-life cipher such as AES. As a result, up till now, non-linear multivariate I/O sums still play a small role in attacking real ciphers. Accordingly, following [39] these attacks may be excessively general and there is probably no substitute to finding and studying in details interesting special cases.

The following idea have been proposed in [39]: given a high-level structure of a cipher one should design and study a special dedicated version of GLC that works well, basically only for this kind of ciphers.
For example, for Feistel ciphers we have the "Bi-Linear Cryptanalysis" (or BLC), see [39]. For generalised Feistel Ciphers (with several branches, one updated at a time), such as SHACAL, Skipjack etc, we have "Multi-Linear Cryptanalysis" proposed in [42]. Similarly, we can for example consider the top-level structure of AES in which we fix the linear parts of AES, allow to put an arbitrary S-box, and ask ourselves the following question: what kind of non-linear multivariate equations can be combined for several rounds for this kind of ciphers and how strong is the AES S-box for this type of (non-linear) characteristics.
A closely related question (dual: bottom-up approach) is the question of building insecure (but very special ciphers) that use the AES S-box, or an Inverse S-box, see Section 5.2.3.

Recent work in [53] has sought to characterise the *fixed-key probability distributions* associated with a block cipher of the *key-alternating* type (such as AES), specifically the distributions of differential and of linear probabilitites. [53] show that, for key-alternating ciphers, a

reliance on the *hypothesis of stochastic equivalence* is no longer necessary. [1] Instead, explicit distribution formulae, of normal, Poisson, Gamma or Extreme value type, are presented by disregarding the key-schedule and averaging over independently selected round keys. The results are obtained without the conventional assumptions of perfectly uniform behaviour, independently acting rounds, or averaging over all cipher keys. Bounds on the EDP (*expected differential probability*) and ELP (*expected linear probability*) are developed based on their respective distributions. [53] are also able to characterise the distribution of extreme values, i.e. the maximum over all key values. Using the above distributions, [53] presents bounds on the weight distribution of differential and linear characteristics where 'weight' here means the absolute value of the $\log_2$ of the probability of the characteristic. The authors also derive bounds on the EDP and ELP as a function of the minimum weights.

1.3 Boomerang-style Attacks on AES

In this section we describe recent studies of security of AES against boomerang-style attacks. The *boomerang attack* was developed in 1999 [121] after the AES competition was already running. This attack sometimes allows to break more rounds than the conventional differential or linear attacks, especially for the ciphers with few but carefully designed rounds (for example, see an attack on SAFER++ [19]).

The first result that we will cover here studies the strength of 128-bit key 10-round AES against the boomerang attack [17]. It shows attacks on AES reduced to 5 and 6 rounds, much faster than the exhaustive key search and twice faster than the "Square" attack of the AES designers. The attacks are structural and apply to other SPN ciphers with incomplete diffusion.

The second type of attacks are the recently discovered boomerang-related key attacks studied in [71, 14] and applied to round-reduced versions of 192 and 256-bit key AES.

None of these attacks threaten security of the full-round AES, moreover they are much less practical than the time-memory-key trade-off attacks described in chapter 4.

1.3.1 Boomerang attack on AES with 128 bit keys

In this section we show attacks on AES reduced to 5 and 6 rounds. Six round attack has complexity of 2^{71} data and steps of analysis (measured in 6-round encryptions). The attack is twice faster than the "Square" attack of the designers of the AES in terms of time complexity which is a dominant factor, but has much higher data complexity. The boomerang attack on AES is less efficient than the partial sum attack [59]. See Table 1.2 for comparison of boomerang attacks with the previous results on a 128-bit key AES.

[1] This hypothesis averages over all keys assuming independent round keys, as is done in [119]. By avoiding this hypothesis, the authors of [53] are able to characterise the distribution for fixed-keys.

Table 1.2: Comparison of boomerang attacks with previous attacks on AES.

Attack	Key size	Rounds	Data[a]	Type[b]	Workload[c]	Memory[a]
Square attack [52]	128	5 of 10	2^{11}	CP	2^{40}	2^{11}
Square attack [52]	128	6 of 10	2^{32}	CP	2^{72}	2^{32}
Collision attack [61]	128	7 of 10	2^{32}	CP	2^{128}	2^{80}
Partial sum [59]	128	6 of 10	$2^{34.6}$	CP	2^{44}	2^{32}
Partial sum [59]	128	7 of 10	2^{128}–2^{119}	CP	2^{120}	2^{64}
Imposs. diff. [15]	128	5 of 10	$2^{29.5}$	CP	2^{31}	2^{40}
Imposs. diff. [32]	128	6 of 10	$2^{91.5}$	CP	2^{122}	$2^{?}$
Boomerang attack	128	5 of 10	2^{39}	CP/ACC	2^{39}	2^{33}
Boomerang attack	128	6 of 10	2^{71}	CP/ACC	2^{71}	2^{33}

[a]Expressed in the number of blocks.
[b]CP – Chosen Plaintext, ACC – Adaptive Chosen Ciphertext.
[c]Expressed in equivalent number of encryptions.

1.3.2 Boomerang Attack on SPNs with Incomplete Diffusion

Boomerang attack is a chosen plaintext-adaptive chosen ciphertext attack. It is an extension of differential cryptanalysis and works on quartets of data $(P, P'), (Q, Q')$. The attack works when encryption $E()$ can be split into $E = E_1 \circ E_0$, where E_0 is weak in encryption direction and E_1 is weak in decryption direction. We refer the reader to [121] for further details.

In this section we explain a generic method of breaking five and six round substitution-permutation networks (SPNs) using a boomerang distinguisher. The attacks that we will show are *structural* in the sense that they will not use specific properties of S-boxes or of the mixing layer, but will use only the fact that diffusion is incomplete (which is the case for many ciphers, including the AES).

We will describe this attack on an example of Rijndael-like cipher with layers of 16, 8x8-bit S-boxes, and Rijndael-like diffusion involving `ShiftRows` and `MixColumns` (though exact constants in the `MixColumns` matrix will be irrelevant to the attack).

The five round attack will be as follows:

1. Prepare a pool of plaintexts $\{P_i\}, i = 0, \ldots 2^{32} - 1$ which have all possible values in four bytes (which will appear in the same column before the `MixColumns`) and arbitrary constant in the other bytes. Encrypt the pool and obtain a pool of 2^{32} ciphertexts $\{C_i\}$.

2. Construct a pool of modified ciphertexts: $D_i = C_i \oplus \nabla$, where ∇ is a fixed non-zero difference with only one active S-box (for example, a non-zero difference in the first byte and zero difference in 15 other bytes).

3. Decrypt the pool $\{D_i\}$ to obtain a pool $\{Q_i\}$ of 2^{32} new plaintexts.

4. Sort the pool $\{Q_i\}$ by the bytes corresponding to eight inactive S-boxes. Pick only those pairs Q_i, Q_j which have zero difference in these 8 bytes. If none found go to step 1.

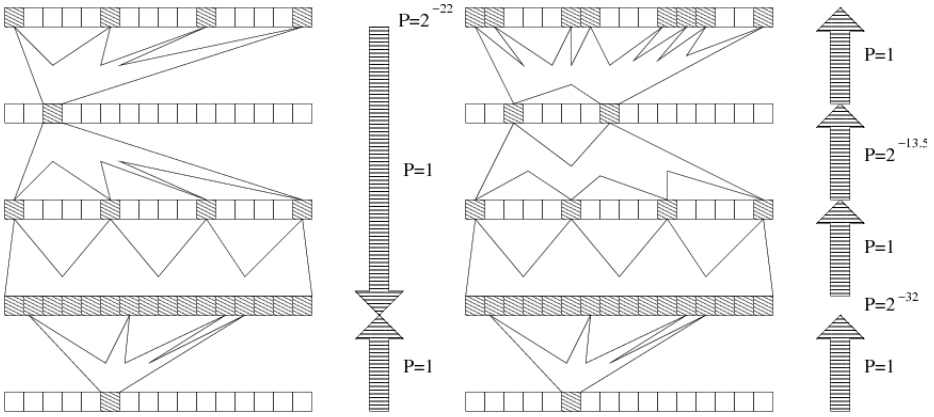

Figure 1.1: Schematic description of a boomerang quartet for Rijndael reduced to five rounds.

5. For each of the quartets P_i, P_j, Q_i, Q_j that pass step 4, guess the 32-bit key value that enters the four S-boxes corresponding to non-constant bytes. Using the guessed key value partially encrypt one round and check that resulting difference is in a single active S-box, which is a 22-bit filtering condition for each pair (P_i, P_j) and (Q_i, Q_j). This gives a 44-bit condition in total for both sides of the boomerang in the case of common 4-tuples of active S-boxes. However with probability half we will have no common 4-tuples, i.e. all the twelve active S-boxes (4 from the (P_i, P_j) pair and 8 from the (Q_i, Q_j) pair) non-overlapping. We will then pick key-candidates that are suggested at least twice.

See Figure 1.1 for a schematic description of the boomerang distinguisher used in this 5-round attack. The rectangles in the figure denote the layers of S-boxes, and the gray squares indicate the active S-boxes. Arrows represent the cost of pattern propagation in terms of probability.

1.3.3 Complexity of the 5-round attack

Analysis of the complexity of the attack described above is as follows: Each pool of 2^{32} texts contains 2^{63} pairs with difference in the four relevant bytes. From these pairs $2^{63}/2^{22} = 2^{41}$ will have a single active S-box after one round[2]. After the second round we will have four active S-boxes. After the 3rd round all bytes will be active. From the bottom up direction we will have one round crossed with probability one, with a truncated differential that starts with one active S-box and ends with four active S-boxes. At this point we need that after the next S-box layer the difference in these four active S-boxes would be the same. This happens with probability 2^{-32}. Then we can switch to the last face of the boomerang, where the effect of the mixing of the 3rd round can then be undone with probability one and we will get four active S-boxes after the S-box layer of the 3rd round. We will have to pay $2^{-13.5}$ in probability for the four active S-box difference to turn into two active S-box difference after this S-box layer and the `MixColumns` which is just above it. From that point we let our truncated differential run freely with probability 1. As a result we will obtain a new pair

[2]The chance of going from 4 active S-boxes to one is $4 \cdot 2^8/2^{32} = 2^{-22}$, since we do not care about the location of the single active S-box.

of plaintexts with some difference in eight bytes and zero difference in the other eight bytes. This is our $64 - \log(6)$-bit filtering condition for the good boomerang quartets (the $-\log(6)$ appears since we do not fix the places of the two active S-boxes).

We pick about 2^6 pools in which we will have $2^{41} \cdot 2^6 \cdot 2^{-13.5} \cdot 2^{-32} \approx 3$ good boomerangs returning back. The average amount of false quartets which satisfy our initial filtering condition is $2^{63} \cdot 2^6 \cdot 2^{-64} \cdot 6 = 192$.

In the simplest case when the boomerang returns in the same four bytes as it was sent we perform a guess of the 32-bit key and check it against two sides of the boomerang P_i, P_j and Q_i, Q_j whether in both cases it leads to a single active S-box after one round. This gives a 44-bit filtration condition which leaves only the correct key guess with probability $1 - 2^{-12}$. However with probability $1/2$ it may happen that for the two boomerang pairs active 4-tuples of the output pair (Q_i, Q_j) will be different from those of the input pair (P_i, P_j). In this case we independently guess 32-bits of the key corresponding to the input four bytes for each pair and leave only those keys that lead to a single active S-box after the first round. We expect at least two good boomerang quartets in our pools and thus the correct key would be counted at least twice. Note that we have about 100 noisy pairs (those with non-overlapping 4-tuples) each of which suggests about $4 \cdot 2^8$ candidates for the 32-bit key. That means that we may have a few wrong keys suggested due to the birthday collisions together with the correct key suggested by the good boomerangs. The same analysis can be performed in parallel on another 4-tuple to produce few candidates for another 32-bit part of the key. Knowing a few candidates for at least half of the first subkey we can repeat the attack with much less data and smaller complexity to achieve the full key-recovery.

Total complexity of this 5-round attack is 2^{38} chosen plaintext/adaptive chosen ciphertext queries and 2^{38} time steps which mainly would be spent on encrypting and sorting the data. The memory required by the attack is 2^{32} blocks or 2^{36} bytes.

1.3.4 Extension to 6-rounds of AES

The attack described above can be extended by one round at the bottom at the cost of guessing 32-bits of the key of the 6th round. We double the number of pools from 2^6 to 2^7 to get 4-6 good quartets for better filtration. We expect that at least 2-3 good quartets will have overlapping 4-tuples between the P's and the Q's which provides 2^{-12} filtration power. Thus we will get about $2^{32} \cdot 100 \cdot 2^{-12} \approx 2^{27}$ candidates for 64-bit partial key: 32-bits at the top and 32-bits at the bottom. The correct key will be suggested at least twice, and the wrong keys would likely be suggested only once, since we are below the birthday bound for a 64-bit event. Thus we expect that all the wrong pairs will be filtered out at the key-recovery step. Finally, complexity of this 6 round attack will be 2^{39} chosen plaintexts, 2^{71} adaptively chosen ciphertexts, the same amount of time steps spent mainly encrypting the texts and 2^{37} bytes of memory.

It seems likely that this attack may be converted to break 7-rounds of the 192-bit key AES.

1.3.5 Conclusions Regarding Boomerang Attacks

We have shown boomerang attacks on 5 and 6 round AES much faster than exhaustive search. We notice that AES has many truncated differentials with probability one spanning up to three rounds, however they are quite expensive in terms of probability when trying to extend them at either end of the boomerang distinguisher. The boomerang attacks on AES are twice more efficient than the "Square" attack but are less efficient than the partial sum attack. This may mean that AES has sufficient security margin against the boomerang attacks. The attacks presented in this paper are *structural* attacks (i.e. they do not use specific properties of the underlying cipher) applicable to arbitrary 5-6 round SPNs with incomplete diffusion. It is an open problem whether the middle-round gaining trick, for example as used in a recent attack on Safer++ [19] would be applicable to the AES.

1.4 Related Key Rectangle Attacks

In this section we describe two recent related key rectangle attacks on round reduced versions of AES.

1.4.1 The First Attack

The related-key rectangle attack on 9-round AES-192 has a data complexity of 2^{87} chosen plaintexts (2^{79} plaintexts encrypted under 256 different keys), and a time complexity equivalent to 2^{125} encryptions [14]. More precisely, the worst case data complexity is 2^{87} chosen plaintexts, while the average case has a data complexity of 2^{86} chosen plaintexts. The paper also presents a related-key rectangle attack on 10-round AES-256 with data complexity of $2^{114.9}$ chosen plaintexts and time complexity of $2^{171.8}$ encryptions.

The 9-round attack on AES-192 attacks rounds 2–10 by suggesting a related-key rectangle distinguisher for rounds 4–9, and performing a key recovery attack on rounds 2,3, and 10. The related-key rectangle distinguisher uses two related-key differentials. One is used for rounds 4–6, while the second one is used in rounds 7–9.

The related-key rectangle attack, just like the rectangle attack, utilizes simultaneously many differentials for each part of the distinguisher. Hence, all the differentials used in rounds 4–6 yield an effective probability of $\hat{p} = 2^{-13.96}$. The second differential used in the attack has probability 1.

Due to the nature of the second differential, and the key schedule algorithm of AES-192, the key difference required for the subkeys' differences of the second differential is unknown. However, by trying all possible 127 key difference, we can find a quartet of keys for which the subkey differences for both differentials hold. Thus, our attack requires 256 related keys for its execution.

The attack generates 128 structures of 2^{72} chosen plaintexts each. Each of these structures is encrypted under the 256 different keys. The attack algorithm assumes that the quartet of keys is one of the possible 127 quartets, and tries to find right rectangle quartets. The attack then tries to find right plaintext quartets that satisfy the rectangle conditions.

Cipher	Number of Rounds	Complexity Data	Time	Memory	Number of Keys	Source
AES-192	8	$2^{128} - 2^{119}$ CP	2^{188}		1	[59]
(12 rounds)	8	2^{89} RK-CP	2^{183}		2	[74]
	8	$2^{86.5}$ RK-CP	$2^{86.5}$		4	[71]
	9	2^{86} RK-CP	2^{125}	2^{85}	256	[14]
AES-256	8	$2^{128} - 2^{119}$ CP	2^{204}		1	[59]
(14 rounds)	9	2^{85} RK-CP	$5 \cdot 2^{224}$		256	[59]
	10	$2^{114.9}$ RK-CP	$2^{171.8}$	$2^{113.9}$	256	[14]

RK – Related-key, CP – Chosen plaintext,
ACPC – Adaptive chosen plaintext and ciphertext
Time complexity is measured in encryption units
Memory complexity is measured in bytes

Table 1.3: Summary of the Previous Attacks and of Our New Attacks

Table 1.4: Attacks on AES-192

Block Cipher	Type of Attack	Number of Rounds	Complexity Data / Time
AES-192	Square	7(0-6)	2^{32}CP / 2^{184} [95]
(12 rounds)	Partial Sums	7(0-6)	$19 \cdot 2^{32}$CP / 2^{155} [59]
		7(0-6)	$2^{128} - 2^{119}$CP / 2^{120}[59]
		8(0-7)	$2^{128} - 2^{119}$CP / 2^{188}[59]
	Related-Key Impossible	7(0-6)	2^{111}RK-CP / 2^{116} [74]
		8(0-7)	2^{88}RK-CP / 2^{183} [74]
	Related-Key Rectangle	8(0-7)	$2^{86.5}$RK-CP / $2^{86.5}$ [71]

CP: Chosen Plaintexts, RK-CP: Related-Key Chosen Plaintexts,
CC: Chosen Ciphertexts, Time: Encryption units

We summarize our results along with previously known results on the respective ciphers in Table 1.3.

1.4.2 The second Attack on AES-192

In [71] they found that in AES with 192-bit keys there exist a related-key truncated differential for rounds $0 \sim 3$ and another related-key truncated differential for rounds $4 \sim 6$ with probability one that are independent of each other. This is due to the incomplete diffusion of AES as well as the simplicity of its key scheduling algorithm with 192-bit keys. The consecutive two related-key truncated differentials can be used in constructing a 7-round related-key rectangle distinguisher based on 4 related keys. This distinguisher is exploited to analyze 8-round AES with 192-bit keys with a data/time/memory complexity of $2^{86.5}$.

1.5 Acknowledgements

We would like to thank Orr Dunkelman and Jongsung Kim for their contribution to the boomerang section of the report.

Chapter 2

Resistance to attacks exploiting multiset properties

Standard differential and linear attacks are typically very sensitive to the exact specification of each component in the block cipher. The choice of the S-boxes determines which characteristics propagate with high probability; whether or not they can be chained together in order to form a linear of differential trail depends on the specific coefficients of the diffusion layer. Another feature of differential and linear attacks is their probabilistic nature. The attacks recover round keys by eliminating values which are statistically unlikely given some property of the observed data. These values are not impossible, though, and hence the possibility remains that correct keys are discarded, in which case the attacks fail.

The attacks described in this section, some of which are currently the most efficient attacks on round-reduced versions of the AES, are much less affected by design choices for individual components. What does matter, is how these components are interconnected. More specifically, the attacks exploit two particular properties of the AES structure: first, the fact that all components of the cipher perform operations on bytes (not on individual bits), and that these operations are bijective; secondly, that a single application of the `ShiftRows` and `MixColumns` transformations does not provide full diffusion, i.e., it takes at least two rounds for any input byte to affect all output bytes. Most of the attacks in this section also differ from differential and linear attacks by the fact that they are not probabilistic, i.e., when values for a round key are discarded, they are guaranteed to be incorrect.

2.1 Square attacks

Square attacks have been given different names in the last few years. S. Lucks proposed the name *saturation attack* [95], A. Biryukov and A. Shamir treated the technique as a special case of *structural cryptanalysis* [22], and L. Knudsen and D. Wagner referred to it as *integral cryptanalysis* [89]. The main idea remains the same, however: instead of analyzing pairs of related plaintexts, as in differential cryptanalysis, the attacker will now study the behavior of larger sets of carefully chosen plaintexts. The different values taken at specific byte positions throughout the cipher are then treated as multisets. A multiset is a list of values, each of

$$\begin{array}{cccc} P & C & C & C \\ C & C & C & C \\ C & C & C & C \\ C & C & C & C \end{array} \rightarrow \begin{array}{cccc} P & C & C & C \\ P & C & C & C \\ P & C & C & C \\ P & C & C & C \end{array} \rightarrow \begin{array}{cccc} P & P & P & P \\ P & P & P & P \\ P & P & P & P \\ P & P & P & P \end{array} \rightarrow \begin{array}{cccc} B & B & B & B \\ B & B & B & B \\ B & B & B & B \\ B & B & B & B \end{array}$$

Figure 2.1: Square distinguisher over three rounds

which can appear multiple times, but the order of which is irrelevant. During the attack, a number of special multisets are considered:

- **Constant multiset (C).** A multiset consisting of a single value repeated an arbitrary number of times.
- **Permutation** or **saturated multiset (P).** A multiset which contains all 256 possible byte values exactly once.
- **Even multiset (E).** A multiset in which each value, if present, occurs an even number of times.
- **Balanced multiset (B).** A multiset such that the XOR of all values (taking into account their multiplicity) is zero.

Notice that some of these properties are implied by others. For example, a constant multiset with an even number of elements is also an even multisets, and any saturated or even multiset is automatically balanced.

2.1.1 The original Square attack

The first attack exploiting multiset properties was presented by L. Knudsen in 1997 and was applied to the direct predecessor of Rijndael, called Square [51] (hence the name of the attack).

The core of the attack consists in encrypting a set of 256 chosen plaintext blocks, which take on all possible values in one byte position, and are constant in the others. The corresponding multiset is depicted at the left-hand side of Fig. 2.1. Due to the byte-oriented structure of the aes (or Square), the properties of this multiset evolve in a very predictable way over three rounds: after the first round, one column of the state will consist of four saturated multisets; after one more round, all byte positions will be saturated; the output bytes of the third round, finally, will be balanced (see Fig. 2.1).

The observation above can easily be exploited to attack a 5-round variant of aes. Thanks to the relatively slow diffusion, it suffices to correctly guess one linear combination of four key bytes in round 4, and four more key bytes in round 5, in order to detect the balancedness of a byte after the third round from the output of the fifth round. This provides the attacker with a means to eliminate incorrect round keys, and, after the analysis of a few sets of plaintexts, to recover the correct one.

Table 2.1: Attacks on round-reduced versions of the AES.

Attack	Key size	Rounds	Plaintext[a]	Type[b]	Workload[c]	Memory[d]
Imposs. Diff.	128–256	5 of 10–14	$2^{29.5}$	CP	2^{31}	2^{40}
Square	128–256	6 of 10–14	2^{32}	CP	2^{72}	2^{32}
Partial sums	128–256	6 of 10–14	$6 \cdot 2^{32}$	CP	2^{44}	2^{32}
Partial sums	192	7 of 12	$19 \cdot 2^{32}$	CP	2^{155}	2^{32}
Partial sums	256	7 of 14	$21 \cdot 2^{32}$	CP	2^{172}	2^{32}
Partial sums	128–256	7 of 10–14	$\lesssim 2^{128}$	CP	2^{120}	2^{64}
Gilbert-Minier	128	7 of 10	2^{32}	CP	$\lesssim 2^{128}$	2^{80}
Gilbert-Minier	192–256	7 of 12–14	2^{32}	CP	2^{140}	2^{96}
Partial sums	192	8 of 12	$\lesssim 2^{128}$	CP	2^{188}	2^{128}
Partial sums	256	8 of 14	$\lesssim 2^{128}$	CP	2^{204}	2^{128}
Related key	256	9 of 14	$2^8 \cdot 32 \cdot 2^{64}$	RK-CP	2^{226}	2^{77}
Exhaustive	128–256	10–14 of 10–14	1–2	KP	2^{128}–2^{256}	1

[a] Expressed in number of blocks.
[b] KP – Known Plaintext, CP – Chosen Plaintext, RK – Related Keys.
[c] Expressed in equivalent number of encryptions.
[d] Expressed in number of entries.

The attack can be extended to six rounds by adding an extra round at the input. The attack now starts with a set of 2^{32} plaintext which differ in four specific positions, such that the output of the first round consists of a column which takes on all 2^{32} possible values, while the three others remain constant. This set of states can be seen as a collection of 2^{24} subsets of 2^8 texts which differ in a single position as before. However, in order to partition the texts into the desired subsets, the attacker needs to guess four bytes of the initial round key. From that point on, the attack proceeds in the same way as described above. In total, the attacker will have to examine all possible values for 9 bytes of the key, resulting in a workload which is comparable to 2^{72} encryption operations (see Table 2.1).

2.1.2 Partial sums

In 2000, Ferguson et al. [59] presented a much more efficient way to perform the original 6-round attack. Their first observation is that the attacker does not really need to partition the 2^{32} plaintexts into subsets. Since all bytes of each of these subsets sum to zero after round 4, this will also hold when summing over all 2^{32} texts. Therefore, the attacker could first encrypt a set of 2^{32} chosen plaintext, run through all possible 2^{40} values for the five round key bytes in rounds 5 and 6, and then check if a particular byte at the output of round 4 sums to 0 over all 2^{32} plaintexts. This test is expected to reduce the number of key candidates by a factor 2^8, and thus has to be repeated at least 5 times in order to isolate the correct key value.

During the attack just described, 2^{32} bytes need to be computed and summed together for 2^{40} different key values. As pointed out by Ferguson et al., this can be carried out in a surprisingly efficient way by gradually guessing the key bytes, and continuously trying to regroup ciphertexts which behave in an identical way by computing "partial sums." This

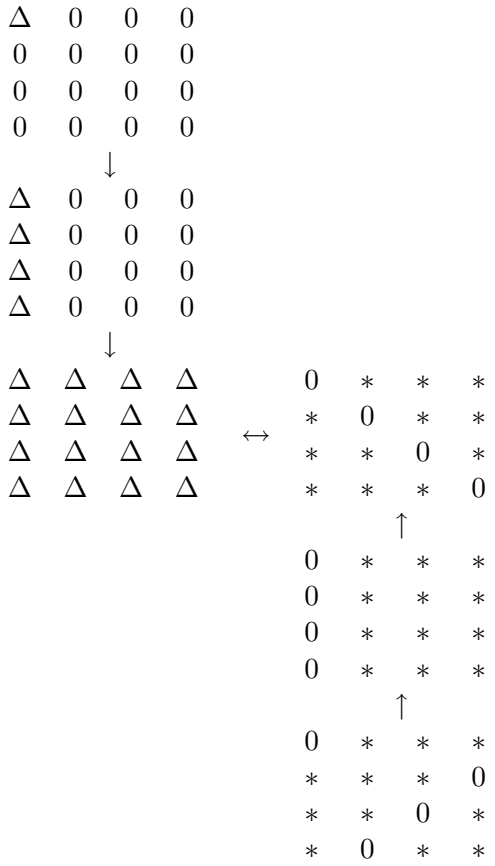

Figure 2.2: A 4-round impossible differential (no `MixColumns` in the last round)

reduces the total workload of the attack to the equivalent of 2^{44} encryptions.

The attack can be extended to 7 rounds by adding a round from which the complete 128-bit round key is guessed. This is clearly not interesting for the 128-bit key version of the AES, but the resulting attack is still more efficient than exhaustive search for the 192-bit and 256-bit versions (in the case of 192-bit AES, there is an additional gain due to the key schedule). An alternative attack, presented in the same paper and based on similar ideas, does apply to the 128-bit key version, but requires almost the complete codebook. The complexities of the different attacks are listed in Table 2.1.

2.2 Impossible differentials

Differential cryptanalysis relies on the existence of differences which propagate with high probability. Differences which cannot occur at all, called impossible differences [15], can be exploited just as well, however. An attack on 5-round AES based on this idea was presented by Biham and Keller [16] in 2000. The way in which the attack exploits the structure of the cipher shows some similarities with the Square attacks, which is why we discuss it here.

The 4-round impossible differential used in the attack is shown in Fig. 2.2. It starts with

two plaintexts which differ in a single byte (byte positions which differ are denoted by 'Δ' in the figure). Because of the bijective nature of the components of the AES, the two texts will necessarily differ in all byte positions after two rounds, i.e., at the input of the third round. Suppose now that the corresponding ciphertexts are equal in the four positions marked by '0' in the figure. Working back, this would imply that at least four bytes were equal at the input of the third round. This contradicts the previous observation.

In order to attack five rounds of AES, Biham and Keller propose to add a round at the input, and to encrypt a set of plaintexts which takes on all 2^{32} values in one column at the input of the first `MixColumns`. The attacker then searches all pairs of ciphertexts which match in the four byte positions, selects the corresponding plaintexts, and eliminates all round key values which would cause a single byte difference after one round, and are hence incorrect. It can be shown that $2^{29.5}$ plaintext would suffice to eliminate all key values but the correct one in a time equivalent to 2^{31} encryptions. This assumes that the attacker uses four similar impossible differentials, and performs a precomputation to fill a table of 2^{40} entries, which is later used to efficiently enumerate incorrect key values.

In 2003, Cheon et al. [32] extended the attack to six rounds by adding an additional round at the output of the cipher. However, the approach requires $2^{91.5}$ chosen texts, and is significantly less efficient (2^{122} encryptions) than the 6-round attacks described in the previous sections.

2.3 Gilbert-Minier attack

The Gilbert-Minier "collision" attack (GM attack) of [61] can be viewed as a multiset attack. It is based upon a 4-round distinguisher between AES and a random permutation. This 4-round distinguisher can be used to mount a key derivation attack of complexity 2^{144} requiring 2^{32} chosen plaintexts on 7-round versions of AES-192 and AES-256, and to derive the key of a 7-round version of AES-128 key marginally faster by an exhaustive search.

2.3.1 Outline of the GM distinguisher

The GM distinguisher has some features in common with the one used in the initial square attack of [52, 51]. The attacker also considers multisets of chosen plaintexts derived by setting all input bytes to constant values except one single one, for which all 256 possible input values are considered. However, the GM distinguisher is quite distinct from the one used in the square attack, and more generally from the distinguishers used in integral attacks and higher order differential cryptanalysis. Instead of considering the multiset properties (e.g. balance) of the intermediate blocks resulting from the input multiset after a limited number of rounds, one considers how the various branches of the propagation path relating the variable input byte and bytes of the consecutive resulting intermediate blocks are affected by key dependent bytes[1]. In other words, one analyzes the key dependence of partial (typically one-byte to one-byte) functions induced by round reduced version of the cipher.

[1] In each branch of the propagation path, the initial byte is iteratively submitted to an S-box, one multiplication by one of the Mix-Column coefficients and exclusive or with a key-dependent byte. This key-dependent byte is constant as long as no folding of the propagation path occurs, i.e. for the 3 first rounds.

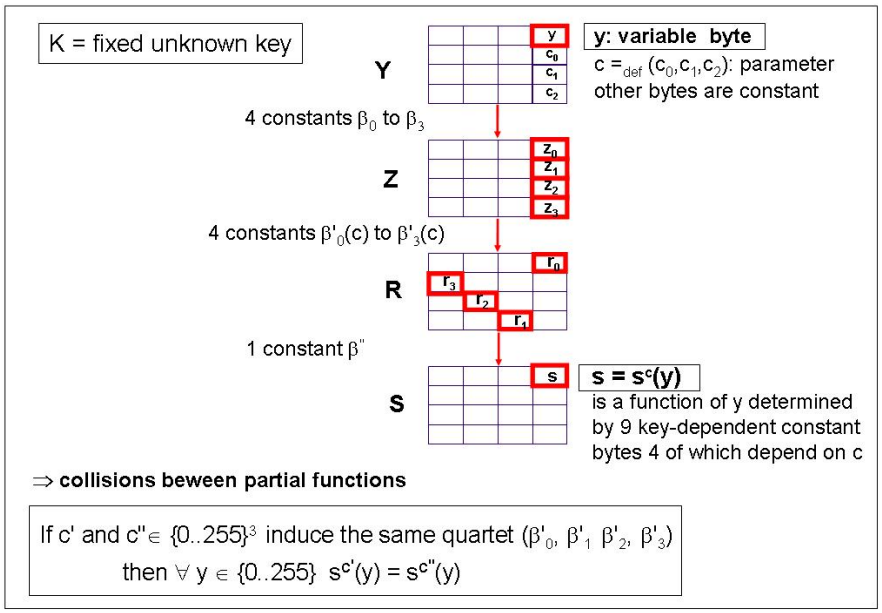

Figure 2.3: The 3-round property used in the GM attack

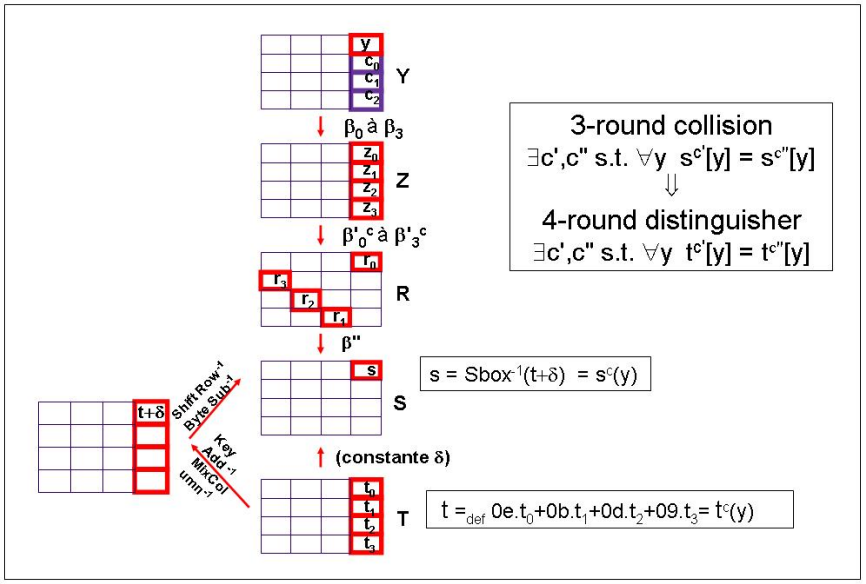

Figure 2.4: The 4-round distinguisher used in the GM attack

It turns out that due to the byte oriented structure of AES, after 3 rounds, each intermediate byte only depends upon the input byte and a quite limited number of key-dependent constants. Moreover, if one modifies some well chosen constant input bytes, only a subset of these key-dependent constants will be affected. The detail of this 3-round property is illustrated in Figure 2.3. The input blocks of rounds 1 to 4 are denoted by Y, Z, R and S, and the variable input byte of Y is denoted by y. The three constant bytes of the Y column containing the variable byte y are denoted by c_1, c_2 and c_3. The (c_1, c_2, c_3) triplet is denoted by c ; c is viewed as a parameter in the analysis of the one-byte to one-byte partial mappings induced by AES. Now it is easy to check on Figure 2.3 that each of the marked bytes of the Z, R and S inputs to rounds 2 (resp. 3 and 4) only depends upon the marked byte(s) of the input to the previous round and on 4 (resp. 4 and 1) key dependent constant(s) β_0 to β_3 (resp $\beta_0'^c$ to $\beta_3'^c$ and β''), which are respectively independent, dependent and independent of c. Consequently, the byte s of block S is a function of y which depends upon 5 key-dependent constants independent of c, and only 4 key and c-dependent constants[2].

In order to exploit the latter property, let us consider about 2^{16} of the 2^{24} possible c values: if the four c-dependent constants do not behave too differently from four random functions of c, then due to the birthday paradox at least one collision is likely to occur between the 4-tuples associated with two of these c values, say c' and c''. Consequently, two of the about 2^{16} $s^c(y)$ partial functions, $s^{c'}(y)$ and $s^{c"}(y)$ are expected to be equal. The resulting 4-round distinguisher used in the GM attack is represented in Figure 2.4. For each of the about 2^{16} considered c values, the decryption of the fourth round allows to express $s^c(y)$, up to an unknown constant δ, as the image by the inverse AES S-box of a linear combination t of four fourth round output bytes t_0 to t_3. Therefore, the existence of a collision between two partial functions $s^{c'}$ and $s^{c''}$ is equivalent to the existence of a collision between the 256-tuples of bytes $t^{c'}(y)_{y=0..255}$ and $t^{c''}(y)_{y=0..255}$. This provides a four-round distinguisher.

2.3.2 Outline of the Key derivation

AES-192 and AES-256

The former 4-round distinguisher can be used to mount an attack against 7-round versions of AES-192 and AES-256, resulting from the 4-round version of the distinguisher by left and right composition with one round and two rounds respectively. The attack is represented in Figure 2.5. The right column (y, c_0, c_1, c_2) of the distinguisher input block Y can be derived, up to an unknown constant which does not matter here from the input block X and a four-byte initial key K_{ini}. As for the bytes of the right column (t_0, t_1, t_2, t_3) of the distinguisher output block T, it is easy to show that t_0 and t_1 can be derived from the 7th round output block V and a 10-byte key K_{01}, and that t_2 and t_3 can be derived[3] from V and another 10-byte key K_{23}.

The initial step of attack consists of an exhaustive search of K_{ini} and (c', c'')[4]. To check

[2]The same property holds for bytes of S other than s, but the properties of other S bytes are not considered in the attack

[3]In order to simplify the representation of the decryption of the two last rounds in Figure 2.5, the 7-th round output bytes are permuted as to obtain an equivalent representation in which there is no shift row in the last round

[4]if K_{ini} is known, c is known up to a constant, whose value does not matter here

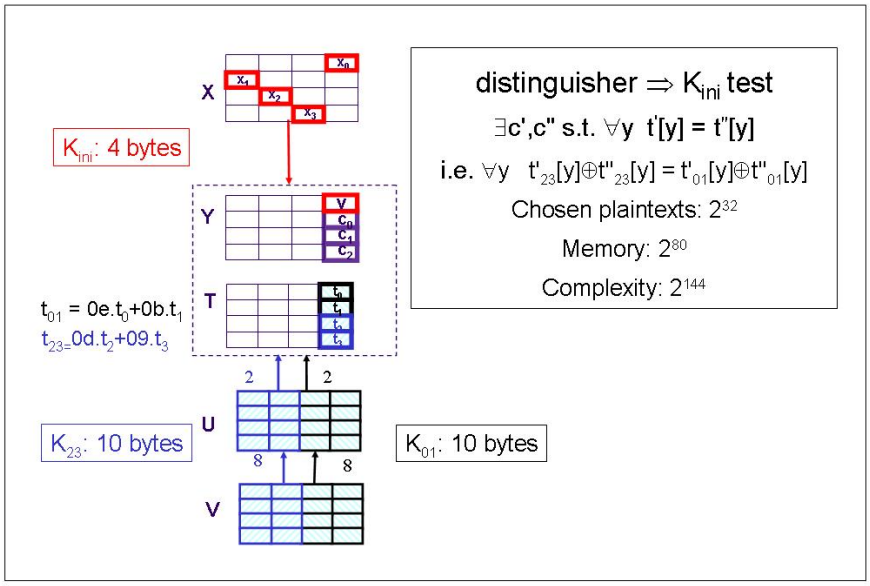

Figure 2.5: Outline of the key derivation for AES-192 and AES-256

a K_{ini} assumption and a (c', c'') pair, one has to pre-compute, for all possible values of K_{01} the resulting contribution to $t_{01}^{c'}(y) \oplus t_{01}^{c''}(y)$ in the decryption of the two last rounds for a few values of y, and the check whether there exists a K_{23} such that the resulting contributions to $t_{23}^{c'}(y) \oplus t_{23}^{c''}(y)$ for the same values of y matches one of the pre-computed ones. The attack requires only 2^{32} chosen plaintexts, and has a complexity of about 2^{144}.

AES-128

We now outline how to use the former 4-round distinguisher to derive the key of a 7-round version of AES-128 key marginally faster than by an exhaustive search. The modified key search procedure is still an essentially exhaustive search of the value of the AES key K, but precomputations allow to speed up the process by replacing the trial encryption one usually has to perform for each tried K value by a faster test. Let us reuse the notation of Figure 2.4. If one looks closer at the dependence between the marked R bytes r_i and the marked Z bytes z_i, one can see that each byte r_i has an expression of the form $r_i = \alpha'_i S(r_i) \oplus \beta'^c_i$, where α'_i is a Mixcolumn coefficient. Moreover, each β'^c_i has an expression of the form $\alpha'_{i0} S(\alpha_{i0} S(c_0) + \beta_{i0}) \oplus \alpha'_{i1} S(\alpha_{i1} S(c_1) + \beta_{i1}) \oplus \alpha'_{i2} S(\alpha_{i2} S(c_2) + \beta_{i2}) \oplus \beta'_i$, where the α_{ij}, the α'_{ij}, the β_{ij} and the β'_i are key-dependent constants which do not depend upon c. It is easy to see that collisions between four-tuples of β'^c_i bytes are determined by the values of 12 key-dependent bytes which value does not depend upon c, namely (β_{ij}), $i = 0$ to 3, $j = 0$ to 2. We denote in the sequel this 12-tuple of key-dependent bytes by $\Phi(K)$.

Now for each of the 2^{12} possible $\Phi(K)$ values, one can precompute a (c', c'') pair of c values such that the quartets $(\beta'^{c'}_i)_{i=0...3}$ and $(\beta'^{c''}_i)_{i=0..3}$ collide, so that $\forall y\ t^{c'}(y) = t^{c''}(y)$. One can also precompute, for each (K_{ini}, c) pair, the subset of 16 of the 2^{32} considered X input blocks leading in the right column of Y to a y byte equal to 0 to 15 and to a (c_0, c_1, c_2) triplet equal

to c, and the corresponding output blocks $V^{c}_{K_{ini}}(y)_y = 0..15$.

These precomputations are used in the following way to speed up the exhaustive search of the AES-128 key K. For each tried value of K, one computes $\Phi(K)$ and K_{ini}, one derives from $\Phi(K)$ a (c', c'') of parameter values leading to a collision, one successively compares, for y=0 to 16, the $t^{c'}(y)$ and $t^{c''}(y)$ linear combinations of key bits obtained by partial decryption of the two last rounds of AES-128 of the output blocks $V^{c'}_{K_{ini}}(y)$ and $V^{c'}_{K_{ini}}(y)$. Only 4 partial (2-round) decryptions have to be performed on average, so the average complexity of the modified exhaustive search is marginally less than one trial encryption per tested key.

Chapter 3

Algebraic Cryptanalysis of the AES

The AES has an extremely rich algebraic structure, and it seems therefore natural that one would attempt to exploit this structure in the analysis of the cipher. In [58], Ferguson *et al* describe how one can express the AES encryption transformation in a compact algebraic formula containing around 2^{50} terms. They acknowledge however that it seems very unlikely that an attacker could take advantage of such algebraic description to mount a *practical attack*.

An alternative, more promising approach is to express the encryption operation as a *system* of polynomial equations. While in theory most modern block ciphers can be fully described by a system of multivariate polynomials over a finite field, for the majority of the cases such systems prove to be just too complex for any practical purpose. Yet there are a number of ciphers that present a highly algebraic structure, and could therefore be more vulnerable to algebraic attacks [18]. Of particular interest is the case of the AES.

3.1 Algebraic Attacks on Block Ciphers

A central question when considering algebraic cryptanalysis of block ciphers is the I/O degree of an S-box. Consider the Boolean function

$$\begin{array}{rccc} f: & \mathrm{GF}(2)^n & \longrightarrow & \mathrm{GF}(2)^m \\ & (x_0, \ldots, x_{n-1}) & \longmapsto & (y_0, \ldots, y_{m-1}). \end{array}$$

We call the *I/O degree* of the Boolean function f the smallest degree of an algebraic relation $g(x_0, \ldots, x_{n-1}; y_0, \ldots, y_{m-1})$ that holds with certainty for every pair (x, y) such that $y = f(x)$.

Cryptographic attacks that exploit this notion (under different names but in full generality) have been proposed in several areas of cryptography, see [40] for a survey. For the HFE cryptosystem a general algebraic attack based on this type of equations was proposed for the first time in [37, 38]. For block ciphers, the idea was first explicitly proposed in [76] and further developed in [46, 47, 41, 40]. For stream ciphers, an attack based on this notion was first proposed in 2003 [45] and later studied and extended by many authors. The basic claim of algebraic cryptanalysis of block ciphers is that encryption systems based on low I/O functions *may* be vulnerable to algebraic attacks that exploit these multivariate relations.

In its general form, an algebraic attack is mounted by expressing the full cipher operation as a system of low-degree multivariate equations involving the (known) plaintext and ciphertext values, the secret key and a large number of intermediate variables arising in the cipher operation. This is often possible when the cipher is based on S-boxes with low I/O degree and all its other components are linear. Furthermore it is well-known that systems that contain a large number of equations compared with the number of variables are generally easier to solve (these systems are called *overdefined*). Thus when designing an algebraic attack of this type, for each non-linear component of the cipher, one attempts to obtain as many low-degree, linearly independent equations as possible. By usually limiting the monomials allowed to some pre-defined set (e.g. quadratic equations only), one can combine all the linear and the non-linear equations in an overdefined system that uniquely describes the cipher operation. Solution of such system represents recovery of the encryption key.

3.2 System of Equations deriving from the AES

3.2.1 AES equations over GF(2)

The only non-linear component of the AES (the S-Box) is based on the inverse map on a finite field. Based on this fact, Courtois and Pieprzyk [46, 47] were able to obtain a small set of quadratic multivariate equations (in the input and output bits) that completely described the S-Box transformation. Indeed, if w and x represent the input and output of an AES S-Box, respectively, then we have that the relations

$$w \cdot x = 1 \qquad w^2 \cdot x = w \qquad w \cdot x^2 = x$$

give rise to 24 quadratic equations, 23 of which are true with probability 1 and one that is true with probability $\frac{255}{256}$. Refer to [47] for the list of all relations[1].

By combining all equations throughout the cipher, Courtois and Pieprzyk were able to express the full AES encryption transformation as a large, sparse and overdefined system of multivariate quadratic equations over GF(2) (in total 8000 quadratic equations with 1600 variables for the AES with 128-bit key).

3.2.2 AES equations over GF(2^8)

The representation of the AES as imbedded in the BES cipher [99] also gives rise to a system of multivariate quadratic equations, but over GF(2^8) rather than GF(2).

The round function of the BES is given by

$$\mathbf{b} \mapsto M_B.\mathbf{b}^{-1} + (\mathbf{k}_B)_i,$$

and thus the encryption can be described by the following system:

$$\begin{array}{rcll} \mathbf{w}_0 & = & \mathbf{p} + \mathbf{k}_0, & \\ \mathbf{x}_i & = & \mathbf{w}_i^{(-1)} & \text{for } i = 0, \ldots, 9, \\ \mathbf{w}_i & = & M_B \mathbf{x}_{i-1} + \mathbf{k}_i & \text{for } i = 1, \ldots, 9, \\ \mathbf{c} & = & M_B^* \mathbf{x}_9 + \mathbf{k}_{10}, & \end{array}$$

[1] By including the relations $w^4 \cdot x = w^3$ and $w \cdot x^4 = x^3$, one can obtain 16 extra, fully quadratic equations.

where $\mathbf{p}$ and $\mathbf{c}$ are the plaintext and ciphertext respectively, $\mathbf{w}_i$ and $\mathbf{x}_i$ are the input and output of the i^{th} invocation of the inversion layer, $\mathbf{k}_i$ is the i^{th} round subkey, and M_B^* is a modified diffusion matrix (the final round in both BES and AES does not use the `MixColumn` operation).

Together with the conjugacy equations (arising from embedding of the AES encryption into the BES framework), the system above gives rise to a system of multivariate quadratic equations over GF(2^8) for the AES encryption, consisting of 7808 equations (5248 from the encryption operation and 2560 from the key schedule), with 4608 variables (2560 state variables, 1408 key variables and 640 auxiliary variables). One can also reduce the sizes of the systems by using the linear equations to substitute for state and key variables, though the resulting system is slightly less sparse.

It is currently not known which of the two systems of equations (over GF(2) and GF(2^8)) would be more suitable for mounting an algebraic attack against the AES. While computations with binary variables can usually be carried out more efficiently, the system over GF(2^8) has a much simpler structure, which could perhaps be exploited by a dedicated method of solution.

3.3 Methods of Solution of Polynomial Systems

3.3.1 Generic Computational Algebra Techniques

Solving multivariate polynomial systems is a typical problem studied in Algebraic Geometry and Commutative Algebra. In this section, we give a brief overview of the main algorithms for solving algebraic systems, in the context of cryptology. Our discussion will go from the simplest to the most efficient algorithms, that is from the linearization principle to F_4 and F_5, through XL and Buchberger algorithms, although this does not respect the chronological order of discovery of these algorithms. We conclude by discussing some recent results on the relationship between these algorithms.

The problem. Let k be a field and $f_1, \ldots, f_m$ be polynomials in n variables with coefficients in k, i.e. $f_i \in k[X_1, \ldots, X_n]$, for $i = 1, \ldots, m$. Let K be an algebraic extension of k. The problem is to find $(x_1, \ldots, x_n) \in K^n$ such that $f_i(x_1, \ldots, x_n) = 0$, for $i = 1, \ldots, m$. Note that the problem may have no solution (inconsistency of the equations), a finite number of solutions, or an infinite number of solutions (when the system is underdefined and K is the algebraic closure of k).

This problem is most often studied in the context of abstract algebra. More precisely, let $I \subseteq k[X_1, \ldots, X_n]$ be the *ideal* generated by $f_1, \ldots, f_m$ and

$$V_K(I) = \{(x_1, \ldots, x_n) \in K^n; \quad f_i(x_1, \ldots, x_n) = 0, \text{ for } i = 1 \ldots m\}$$

be the *variety* over K associated to I. The problem is then to find $V_K(I)$.

When k is a finite field of order q, one can always add to the existing set of equations the so-called field equations $X_i^q = X_i$, for $i = 1 \ldots n$, and obtain $m + n$ equations. For most cryptographic applications, the case of interest is when $k = K = \mathbb{F}2$. In this case, the field

equations are $X_i^2 = X_i$. This preprocessing step has the following consequences: the space of solutions is 0-dimensional (or empty), including at "infinity", and the ideal becomes radical (i.e. the solutions are of multiplicity one).

Linearization

The method of *linearization* is a well-known technique for solving large systems of multivariate polynomial equations. In this method, one considers all monomials in the system as independent variables and tries to solve the system using linear algebra techniques. More precisely, let A be the set of multi-indices $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbf{N}^n$, which represent the exponents of the monomials of $k[X_1, \ldots, X_n]$. Then any polynomial f can be written as $f = \sum_{\alpha \in A} c_\alpha X^\alpha$, where the sum involves only a finite number of monomials $X^\alpha = X_1^{\alpha_1} \cdots X_n^{\alpha_n}$. Using this notation, we can write the following matrix M_{L}:

$$\begin{array}{c} \\ f_1 \\ \vdots \\ f_m \end{array} \begin{array}{c} \begin{array}{ccc} \ldots & X^\alpha & \ldots \end{array} \\ \begin{pmatrix} \ldots & c_\alpha^1 & \ldots \\ & & \\ \ldots & c_\alpha^j & \ldots \end{pmatrix} \end{array} = M_{\mathrm{L}},$$

where $f_i = \sum_\alpha c_\alpha^i X^\alpha$. Note that the columns of the matrix can be arranged in different ways, depending on the order chosen to sort the multi-indices α.

To apply linearization, one now considers each (non-constant) monomial X^α as an indeterminate and attempts to solve the corresponding system of linear equations using linear algebra techniques.

The effectiveness of the method clearly depends of the number of linearly independent polynomials in the system. For example, in the case of boolean functions, the total number of monomials of degree less than or equal to 2 (excluding the constant) is $\binom{n}{2} + n$. Thus if the system consists of m polynomials of degree 2, it can be solved if the matrix M_{L} has this rank. Note that the method also tolerates a smaller rank: it is possible to perform an exhaustive search on the affine space of solutions when the dimension of the kernel of the matrix is not too large.

Concerning the complexity, we observe that the cost of the linear algebra operations is $O(N^3)$, N being the size of the matrix M_{L}. We may theoretically write $O(N^\omega)$, ω being the exponent of linear algebra, and sometimes even optimistically use $\omega \approx 2 + \epsilon$ in the case of sparse matrices.

Linearization has been considered in the cryptanalysis of LFSR-based, filtered, stream ciphers. As stated before, each new bit of the key stream gives rise to a new equation on the key bits, and by using a large number of bits from the key stream, one should have in theory enough equations to directly apply linearization. Note however that no practical attack has been reported to have been implemented using linearization, and the problem of estimating the rank of the linearized system is still unsolved.

The XL algorithm and variants

In order to apply the linearization method, the number of *linearly independent* equations in the system needs to be approximately the same as the number of terms in the system. When this is not the case, a number of techniques have been proposed that attempt to generate enough LI equations. The most prominent is the XL algorithm (standing for *eXtended Linearization*), which was introduced in [44]. The XL algorithm aims at introducing new rows to the matrix M_{L}, by multiplication of the original equations by monomials of prescribed degree. More specifically, the following matrix M_{XL} is constructed:

$$\begin{array}{c} \\ \vdots \\ X^{\beta} f_1 \\ \vdots \\ X^{\beta'} f_m \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & X^{\alpha} & \cdots \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & c_{\alpha}^{1,\beta} & \cdots \\ & \vdots & \\ \cdots & c_{\alpha}^{j,\beta'} & \cdots \\ & \vdots & \end{pmatrix} \end{array} = M_{\mathrm{XL}},$$

where the set of the rows is constructed from all products $X^{\beta} f_j = \sum_{\alpha} c_{\alpha}^{j,\beta} X^{\alpha}$, where β and f_j are such that $\deg(X^{\beta} f_j) \leq D$, D being a parameter of the algorithm. The hope is that at least one univariate equation (say in X_1) will appear after the Gaussian elimination on M_{XL}. This equation should be easily solved over the finite field, the found values substituted in the equations, and the process repeated for the other indeterminates X_i, $i \geq 2$. One expects that after a few iterations, the algorithm will yield a solution for the system.

To estimate the complexity of the XL algorithm, the problem is to find D such that XL succeeds with parameter D. Since the number of monomials of total degree $\leq D$ in $k[X_1, \ldots, X_n]$ is equal to $\binom{n+D}{D}$, there is an exponential dependance on D.

Since the introduction of the XL method, a number of variants have been proposed attempting to exploit some specific properties of the polynomial system [36, 30]. Of particular relevance for the analysis of the AES is the method proposed in [47], called *XSL*, which we discuss in Section 3.3.2

Gröbner bases algorithms

Gröbner bases algorithms are perhaps the best known technique for solving polynomial systems. These algorithms return a basis for the ideal derived from the set of equations, which can then be used to obtain the solutions of the system. The most accessible historical reference is [25], while the book [50] presents a gentle introduction to the topic together with the basics of algebraic geometry (it does not however included the more recent algorithms F_4 and F_5).

We now give a definition of a Gröbner basis of an ideal. Let $\preceq$ be a monomial order, i.e. a total order on the set of monomials X^{α}, $\alpha \in \mathbf{N}^n$, which is compatible with multiplication. Then the set of terms $c_{\alpha} X^{\alpha}$ of a polynomial $f = \sum_{\alpha} c_{\alpha} X^{\alpha} \in k[X_1, \ldots, X_n]$ can be ordered with respect to $\preceq$, and the notion of leading term $\mathrm{LT}(f)$, leading monomial $\mathrm{LM}(f)$ and leading coefficient $\mathrm{LC}(f)$ of the polynomial f are all well defined.

Let $I \subseteq k[X_1, \ldots, X_n]$ be an ideal and $\mathrm{LM}(I) = \{\mathrm{LM}(f)\ ; f \in I\}$ the set of leading monomials of polynomials in I. A Gröbner basis of the ideal I is a set $G = \{g_1, \ldots, g_l\} \subset I$ such that:

$$\mathrm{LM}(I) = \bigcup_{i=1}^{l} \mathrm{LM}(g_i).\{X^\alpha,\ \alpha \in \mathbf{N}^n\}.$$

In other words, G is a Gröbner basis of I if the leading term of any polynomial in I is divisible by the leading term of some polynomial of G. One can show that every non-empty ideal $I \subseteq k[X_1, \ldots, X_n]$ has a Gröbner basis (which however is not unique). It is to be stressed that the notion of Gröbner basis is a mathematical one, independently of any algorithm computing Gröbner bases.

There is also the notion of a *Gröbner basis of degree D* of an ideal I (denoted by G_D), which has the property that the leading monomial of every polynomial in I of degree $\leq D$ is divisible by the leading monomial of a polynomial of G_D. It can be shown that there exists D large enough such that G_D is a Gröbner basis of I.

Gröbner bases algorithms are powerful tools for solving systems of polynomial equations. In most cases, when the Gröbner basis is found, the solution is also found. For most cryptographic applications, we will have a system with unique solution, say $(a_1, \ldots, a_n) \in \mathbb{F}2^n$, and the ideal is radical. Then the *reduced* Gröbner basis of I is $\{X_1 - a_1, \ldots, X_n - a_n\}$.

The Buchberger algorithm

The Buchberger algorithm is the classical algorithm for computing the Gröbner basis of an ideal I. It works based on a generalization of the Euclidean division of polynomials in one variable to the multivariate case. More precisely, given a monomial order, there exists an algorithm `division`$(f, f_1, \ldots, f_l) = (g_1, \ldots, g_l, r)$ with the following properties: $f = f_1 g_1 + \ldots f_l g_l + r$, and no leading monomial of the g_i divides r. Then a Gröbner basis of an ideal generated by $f_1, \ldots, f_l$ can be computed by the following algorithm (Buchberger algorithm):

Initialize: $G = \{f_1, \ldots, f_l\}$

Loop

1. Combine every pair f_i, f_j by cancelling leading terms, to get $S(f_i, f_j)$ (the S-polynomials);
2. Compute the remainders of the $S(f_i, f_j)$ by G;
3. Augment G with the non-zero remainders.

Until all remainders are zero.

Return G.

One can show that this algorithm terminates and computes a Gröbner basis of the ideal generated by $f_1, \ldots, f_l$. It is a fact that most S-polynomials generated in step 1 will reduce to zero, and therefore many useless computations leading to zero remainder are performed. The algorithm can be modified to include *Buchberger's criteria* [24], which are a priori conditions on the pairs (f_i, f_j) to detect the ones whose S-polynomial will have a remainder equal to

zero, and therefore discard them from steps 1, 2 of the algorithm. While a great proportion of pairs will be discarded by the criteria, still many S-polynomials constructed will reduce to zero, as experienced in reported implementations.

The complexity of the Buchberger algorithm is closely related to the total degree of the intermediate polynomials that are generated during the running of algorithm. In the worst case, it is known to run in double exponential time. Regarding implementation, there are number of optimizations that can be made to improve the performance of the algorithm. For example, the main loop of the algorithm can be sliced into loops of finer grain, and instead of combining every possible pair, pairs can be successively selected with respect to some strategy, and steps 2 and 3 can be performed with this selection. For instance, the most recent pairs from G can be chosen; alternatively, the pairs of smallest total degree may also be chosen.

The F_4 and F_5 algorithms

The F_4 algorithm [56] can be roughly sketched as a matricial version of the Buchberger algorithm. To introduce the idea, we first depict the Euclidean division for univariate polynomials $f = f_d X^d + \cdots + f_1 X + f_0$ and $g = g_{d'} X^{d'} + \cdots + g_1 X + g_0$, with $d' \leq d$, as a matrix reduction algorithm. Consider the following:

$$\begin{array}{l} \\ f \\ X^{d-d'}g \\ X^{d-d'-1}g \\ X^{d-d'-2}g \\ \vdots \\ g \end{array} \begin{array}{c} \begin{array}{cccccccc} X^d & X^{d-1} & & & & & & X^0 \end{array} \\ \begin{pmatrix} f_d & f_{d-1} & \cdots & & & & & f_0 \\ g_{d'} & g_{d'-1} & \cdots & g_0 & & & & \\ 0 & g_{d'} & g_{d'-1} & \cdots & g_0 & & & \\ 0 & 0 & g_{d'} & g_{d'-1} & \cdots & g_0 & & \\ \vdots & & & \ddots & & & \ddots & \\ 0 & 0 & 0 & 0 & g_{d'} & g_{d'-1} & \cdots & g_0 \end{pmatrix} \end{array} \tag{3.1}$$

Then successive reductions of the first row by the remaining rows (row echelon reduction by elementary row operations) give the remainder of f by g. Similarly the multivariate division algorithm can be written in a matrix fashion.

At each iteration of the F_4 algorithm, corresponding to each iteration in the Buchberger algorithm, and subject to the selection strategy, the two parts f_i and f_j of the selected pairs (f_1, f_j) are written in a global matrix M_{F_4}. Now the crucial point of the algorithm F_4 it to write, at a given step of the algorithm, all the considered f_i and f_j into this global matrix, together with all already known polynomials of the current basis G, multiplied by fitting monomials, in the same way as the polynomial g is shifted in the matrix in (3.1).

Then, in a single step corresponding to one iteration in the Buchberger algorithm, a huge matrix reduction operation (computation of the row echelon form) is done on the matrix M_{F_4}. In contrast to the Buchberger algorithm, where each remainder is computed separately and sequentially, this global reduction operates all reductions of all polynomials by all multiples of polynomials in the current basis G. It turns out that, properly implemented, this is a big win [56, 114]. Additionally, the algorithm can also benefit from any optimization technique from linear algebra algorithmics which can be applied here.

The F_5 algorithm [57] is an optimized version of the F_4 algorithm, with an optimal F_5 criterion, generalizing Buchberger's criteria, for discarding *any* pair which will reduce to zero

in the remaining computations. That is, the rank of the constructed matrix M_{F_5} is equal to the number of its rows (property of full rank).

Relationship between these algorithms

Recent research has shown that some of the algorithms introduced above are related. In fact, let M_∞ denote the Macaulay matrix with an infinite number of rows and columns, defined as

$$\begin{array}{c} \\ \vdots \\ X^\beta f_i \\ \vdots \\ X^{\beta'} f_j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & X^\alpha & \cdots \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & c_\alpha^{1,\beta} & \cdots \\ & \vdots & \\ \cdots & c_\alpha^{j,\beta'} & \cdots \\ & \vdots & \end{pmatrix} \end{array} = M_\infty,$$

for all monomials X^β, $X^{\beta'}$, of unbound degree. The M_{XL} matrix of the XL algorithm in degree D is therefore just a finite submatrix of the Macaulay matrix, corresponding to all monomials of degree less than or equal to D. Performing a Gaussian elimination on the Macaulay matrix is equivalent to running the Buchberger algorithm [93]. This fact is closely related to the behaviour of the XL algorithm, and it is shown in [3] that the XL algorithm terminates for a degree D if and only if it terminates in degree D for the lexicographical ordering.

Concerning F_4, we can see that the matrix

$$\begin{array}{c} \\ \vdots \\ X^\beta f_i \\ \vdots \\ X^{\beta'} f_j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccc} \cdots & X^\alpha & \cdots \end{array} \\ \begin{pmatrix} & \vdots & \\ \cdots & c_\alpha^{1,\beta} & \cdots \\ & \vdots & \\ \cdots & c_\alpha^{j,\beta'} & \cdots \\ & \vdots & \end{pmatrix} \end{array} = M_{F_4}$$

is constructed only from pairs (f_i, f_j) originating from the previous iterations of the algorithm, and which are not discarded by the Buchberger criteria. This shows that M_{F_4} is a very small submatrix of the matrix M_{XL} constructed by XL. Using an XL description as an F_4 algorithm, it is proven [3] that *a slightly modified XL computes a Gröbner basis.*

We note that things are pushed further in the same vein, when one considers the F_5 algorithm, which constructs a matrix M_{F_5} with even less rows than M_{F_4}. In [3], an example is given for the case of 130 equations in 128 variables, where the number of rows in the matrix M_{XL} will be more than 10 thousands times the number of rows in the matrix M_{F_5}.

Complexity bounds

We now state some results on the complexity of the algorithms introduced above, taken from [54, 31], which focus on the XL algorithm, and from [6], which focus on F_5. In both

cases, the concepts of Hilbert Theory are behind the scenes, and it is important to understand it to properly analyze these algorithms. In the case of XL, the point is to find the degree D such that an univariate equation in the last variable X_1 can be found after the Gaussian elimination process.

For quadratic polynomials, we have the following complexity result: when the number of polynomials is $m = n + c$, then the minimum degree for XL to succeed is

$$D \geq \frac{n}{\sqrt{c-1}+1}.$$

Since the number of monomials is $\binom{n}{D}$ in the binary case, and $\binom{n+D}{D}$ in the general case, this theorem implies that XL has an exponential complexity.

For F_5, the highest degree which appears in the algorithm is when the number of rows is equal to the number of columns (because of the property of full rank). This reasoning is exact since the F_5 criterion eliminates all rows which reduce to zero. In [6], we are presented with the following result for quadratic polynomials over $\mathbb{F}2$: for n equations in n variables (without counting the field equations), the degree for which F_5 stops is approximately $D \approx 0.09n$, the approximation being valid even for small n. This also implies exponential complexity for F_5.

For general systems (over $\mathbb{F}2$) the results are the following [6]: when m grows linearly with n, the size of the matrix is exponential in n, and the complexity of F_5 is exponential; when n/m tends to zero, F_5 is subexponential; and when m grows as Nn^2, F_5 has polynomial complexity, with exponent depending on N.

3.3.2 The XSL algorithm

The XSL algorithm was introduced in [46, 47], and it is derived from the XL algorithm. It is however a method which attempts to exploit the sparsity and specific structure of the equations: whereas in the XL algorithm the equations are multiplied by all monomials up to a certain degree, in the XSL algorithm the equations are multiplied only by "carefully selected monomials". The goal here is to create fewer new monomials when generating the new equations. Additionally, there is a last step (called T′ method), in which we try to obtain new linearly independent equations without creating any new monomials.

Motivations and Criteria Behind the XSL method

According to Courtois and Pieprzyk, a global algebraic attack on block ciphers should contain the following three stages, which in theory can be studied separately:

1. Write an appropriate initial system;
2. Expand the system;
3. Perform final elimination.

They claim that the XSL method is basically a method for handling the second step, although it may also have implications on the first and third steps. Although very little is known about

real-life behaviour of such attacks, they presented some heuristics for the design of the three steps above in [46, 47], which can be summarised as follows:

1. **Write an appropriate initial system:** write a system of equations that, given one or several known plaintext-ciphertext pairs, uniquely determines the secret key. The main heuristics to choose "interesting" systems are that:

   - it should be as overdefined as possible;
   - it should be as sparse as possible.

   Both of these criteria can be reflected by the initial ratio R_{ini}/T_{ini} between the number of equations R_{ini} and the total number of monomials T_{ini} in the system.

2. **Expand the system:** the goal is, by starting with the original R_{ini} equations and T_{ini} monomials, to produce a larger set of R equations with T monomials. This can be done for example, by multiplying the initial equations by some well chosen polynomials. The goal is to have the new ratio R/T close (or bigger than) 1. Achieving such a result as quickly as possible was the main motivation behind the XSL method. The main criterion of "success" does not seem to be the final ratio R/T (that simply must be somewhat close to 1, e.g. 0.9) but rather the size T.

3. **Perform final elimination:** the final step should be an elimination method that given an "almost saturated system" with R/T close to 1, finds a solution of the system. On proposed method to solve such systems is the so-called T$'$ method proposed in [46, 47]. Other alternatives can be the application of other common computational algebra techniques (e.g. the Buchberger algorithm, the F4 or F5 algorithms) to the expanded system.

The XSL algorithm on the AES

Two particular instances of the XSL algorithm were proposed for mounting an algebraic attack against the AES. These are in fact what is widely known in literature as the "XSL method". The main differences between these versions of the XSL algorithm are how one writes the initial system of equations and expands this system (first and second steps above). Both versions use the T$'$ method as the final step.

The first version was presented in [46], where two different attacks were described: the first one eliminating the key schedule equations (but requiring a number of plaintext-ciphertext pairs), and a second, more specific attack, that used the key schedule equations (and should work with a single plaintext-ciphertext pair). Later a different version of the algorithm was introduced in [47] (called "compact XSL"). Only the first attack was described in [47], although it is straightforward to extend the method to the second attack.

The initial claims were that, by applying the XSL algorithm against the system of equations over GF(2) arising from the AES, one could mount a (at least theoretical) successful attack against the AES with 256-bit keys. By using the system of equations over GF(2^8), it was also noticed that the XSL method could in theory provide an efficient attack against the AES with 128-bit keys [99, 100].

Analysis of the XSL algorithm does not seem to be an easy task, and currently very little is known about its behaviour. The reasons for that are the many versions of the algorithm found in the literature, where the description of the method often leave some room for interpretation. Furthermore, given the size of the systems involved, it is very difficult to implement and run experiments even on small examples to verify the heuristics in [46, 47]. Recent results [33] indicate however that, as presented in [47], the algorithm cannot solve the system arising from the AES, and some doubts are cast on whether the algorithm in its current form can provide an efficient method for solving the AES system of equations.

3.3.3 Research Directions

Although algebraic attacks have received a lot of attention of the cryptographic community in the last few years, there has not been too much progress in assessing whether they can be effective against block ciphers in general, and the AES in particular. The main reason is that the size of systems arising from block ciphers are completely out of reach for the current computational power. While for most methods of cryptanalysis it is quite straightforward to perform experiments on reduced versions of the cipher to understand how the attack might perform, this has not been the case for algebraic attacks on block ciphers. One possible direction to follow is the introduction of toy examples of symmetric ciphers, in order to test the effectiveness of the main algorithms in solving the systems of algebraic equations. This is discussed in Section 3.4.

Likewise, much can be done on the algorithmic side. Computer algebra is a well established subject, and some of the algorithms presented earlier have been known and extensively studied for a number of years now. For example, Gröbner bases algorithms are known to be a powerful tool for solving systems of polynomial equations. They are however general-purpose algorithms, which are used to deal with a number of problems arising in algebraic geometry (including computing the solutions of a system). They may well prove to be an overkill when considered in the context of cryptography. The systems arising from symmetric ciphers are very structured and with special characteristics: they are usually sparse, with unique solution over a finite field, structured in blocks of similar format (rounds), etc. Perhaps the most promising approach would be the development of *dedicated* methods for specific block ciphers. These could be built upon known techniques from computer algebra, but aiming to exploit the special properties of a particular system. This theoretical approach together with experiments with small versions of the ciphers can hopefully shed some light on how effective algebraic attacks can be against symmetric ciphers.

3.4 Algebraic Attacks on Small Versions of the AES

A number of small versions of the AES have been proposed with the clear objective of studying the applicability of algebraic attacks against the cipher. Although it is not an easy task to design small versions that can replicate the main cryptographic and algebraic properties of the AES, the hope is that experiments on these small versions can provide a preliminary insight into the behaviour of algebraic cryptanalysis on the AES.

3.4.1 Small Scale Variants of the AES

A family of small scale variants of the AES was proposed in [35]. The main goal of this construction was to provide a fully parameterised framework for the analysis of AES equation systems.

The small scale variants of the AES are called SR(n, r, c, e), and are parameterised in the following way:

- n is the number of (encryption) rounds;
- r is the number of "rows" in the rectangular arrangement of the input;
- c is the number of "columns" in the rectangular arrangement of the input;
- e is the size (in bits) of a word.

The cipher SR(n, r, c, e) has therefore n rounds and a block size of rce bits, where a data block is viewed as an array of $(r \times c)$ "words" of e bits. A round of the small scale variants of the AES consists of small scale variants of the AES operations[2] (i.e. `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`).

Some preliminary experiments were performed in [35], where the solution of the systems of equations arising from some of the small scale variants were computed using MAGMA's implementation of implementation of Faugère's F4 algorithm. These were basic timing experiments, which can however provide a preliminary assessment of algebraic attacks as cryptanalytic techniques. In particular, by comparing attacks on different variants it may be possible to better understand of how various components and representations of the cipher contribute to the complexity of algebraic attacks.

For example, following some of the results using ciphers with different block structures, with systems over GF(2^4) and GF(2^8), the experiments seem to indicate that both the underlying field equations and inter-word diffusion in the cipher play an important role in the complexity of the computations. The experiments were however somewhat inconclusive on which is the best representation for the system: over GF(2) or GF(2^e) (i.e. BES-style equations).

As the systems of the equations arising from block ciphers are very structures, a promising technique to find the overall solution is, in effect, a *meet-in-the-middle* approach: rather than attempting to solve the full system of equations for n rounds (we assume that n is even), one can try to solve two subsystems with $\frac{n}{2}$ rounds, by considering the output of round $\frac{n}{2}$ (which is also the input of round $\frac{n}{2}+1$) as variables. By appropriately pre-processing the system, the solutions of these two systems can then be combined, giving rising to a third smaller system which can be solved to obtain the encryption key.

This technique was applied successfully on many of the small variants [35]. The results also suggested the applicability of a type of *divide-and-conquer* approach to the problem of solving the system of equations arising from the AES, in which some form of (perhaps largely

[2]For the last round of the AES, the operation `MixColumns` is omitted; this operation is however retained for the final round of SR(n, r, c, e).

symbolic) pre-computation can be performed and then combined to produce the solution of the full system.

Although it is clearly not straightforward to carry forward much of the results from these small variants to the AES, experiments with small versions should be able to provide some preliminary insight into the behavior of algebraic attacks. In fact, given the highly structured systems that arise from the AES, a theoretical approach (in designing *dedicated* methods, which can exploit this particular structure), together with experiments with small versions of the ciphers should hopefully shed some light on how effective algebraic attacks can be against symmetric ciphers.

3.4.2 Other Experiments with Small Ciphers

One of the earliest examples of small ciphers that were designed for the purpose of studying algebraic attacks was proposed in [46]. These were constructed using 3-bit S-boxes, with a very simple key schedule. A number of simulations were presented in [46]. In these simulations the expanded system was generated according to the proposed version of the XSL method. The attacks weren't however fully implemented: it was only verified whether the rank of the systems was large enough for application of the T′ method. These systems were also used in experiments presented in [123].

Another series of recent experiments on small ciphers is reported in Gwénolé Ars' doctoral dissertation [2]. We note however that although the structure of the studied parameterized family of small ciphers is inspired on the one of AES, the connection with the actual AES is not as strong as for the small scale variants considered in section 3.4.1.

The family of small ciphers studied in [2] are parameterized by the block size (denoted by n) and the number of rounds (denoted by r). Each round is instantiated by:

- a system of n quadratic equations involving $2n$ variables (corresponding to the n input bits and n output bits), which is intended to model the `ByteSub` transformation of the AES;
- a one-to-one linear mapping which is intended to model the `ShiftRow` and `MixColumn` transformations of the AES;
- the XOR operation of an n-bit subkey derived from the small cipher n-bit key by a bijective mapping.

The quadratic equations of each round are randomly drawn, but include constants selected as to guarantee the existence of a chain of intermediate values relating one arbitrary fixed n-bit input block to one arbitrary fixed n-bit output block[3]. The bijective linear

[3]It can be noticed that a major difference with the non-linear part of AES rounds is that these equations do not generally represent a one-to-one function, or even a function. This alone represents a cryptanalytic weakness that would render even a real-scale version of the considered ciphers breakable, and introduces some additional uncertainty as whether it is licit to transpose results on algebraic attacks of the considered toy ciphers to the actual AES.

mappings used at each round and for the key schedule are also drawn at random.

Gröbner basis algorithms are applied to the quadratic system associated with one input block and the corresponding output block. All the experiments involve block sizes n comprised between 3 and 9 bits per block. For 4 rounds, the system was found to be always solvable, and the degree reached during the Gröbner basis computation was upper bounded by 4. In the particular case $n = 8$, the partitioning of the quadratic equations into two quadratic subsystems involving 4 input bits and 4 output bits each (as to model two contiguous S-boxes) resulted in a decrease of the observed degree from 4 to 3. Last of all, in an experiment where the quadratic equations of each round represent the "inversion" function of $\mathrm{GF}(2^n)$, n values comprised between 4 and 9 and r values comprised between 3 and 10, the obtained system was found to be almost always solvable, and a paradoxical phenomenon of decrease of the reached degree when r is increasing was observed. The two above phenomena are not yet fully understood. Further experiments are still needed, and it seems premature to draw any conclusion concerning the highest degree one would encounter in a Gröbner basis computation for the actual AES.

Chapter 4

Generic Trade-off Attacks

4.1 Trade-off Attacks and AES Key Sizes

Hellman's tradeoff [67] is a well-known way to invert arbitrary one-way functions. In the context of block ciphers with reasonably long keys this attack is typically not considered to be of a threat since its precomputation time is the same as the exhaustive search of the key. Moreover the attack works for a single chosen plaintext encryption and cannot benefit if more plaintext-ciphertext pairs are available to the attacker since the precomputed tables are "wired" to a fixed plaintext. This is contrary to what happens in the case of stream ciphers, where two different tradeoffs both involving data are available: the Babbage-Golic Time-Memory, Data-Memory tradeoff [4, 62] and a more flexible Time-Memory-Data tradeoff by Biryukov-Shamir [21]. More importantly, precomputation in these attack is way below the exhaustive key search complexity.

However it is easy to see that all the reasoning from the Time-Memory-Data tradeoff in the case of stream ciphers [21] can be applied to the block-cipher "Time-Memory-Key" case as well. Namely we no longer need a full coverage of the space N, but rather can cover a fraction N/D_k. Thus we will use t/D_k tables instead of t, which means our memory requirements go down to $M = mt/D_k$ (here m is the number of Hellman's tables). Our time requirements are $T = t/D_k \cdot t \cdot D_k = t^2$ (less tables to check but for more data points), which is the same as in the original Hellman's tradeoff. Finally the matrix stopping rule is: $N = mt^2$ which is the condition to minimize the waste of matrix coverage due to birthday paradox effects. Using the matrix stopping rule and eliminating the parameters m and t we get a tradeoff formula:

$$N^2 = T(MD_k)^2.$$

This is exactly the same formula as the one derived in [21] for the case of stream ciphers. For example, for the case of AES with 128-bit key, assuming that one is given 2^{32} encryptions of a plaintext "all zeroes" (or any other fixed text, like 16 spaces, "Hello Joe, " etc.) under different unknown keys, one can recover one of these keys after a single preprocessing of 2^{96} steps, and using 2^{56} memory for table storage and 2^{80} time for the actual key-search[1]. It is important to note that unlike in Hellman's original tradeoff the preprocessing time is

[1] At the moment of this writing 2^{85} computations is approximately the power of all computers on the Internet during 1 year.

Table 4.1: Several generic TMD attack tradeoff points.

Cipher	Key size	Keys (Data)	Time	Memory	Preprocessing
Any cipher	k	$2^{k/4}$	$2^{k/2}$	$2^{k/2}$	$2^{3k/4}$
Any cipher	k	$2^{k/3}$	$2^{2k/3}$	$2^{k/3}$	$2^{2k/3}$
Any cipher[12]	k	$2^{k/2}$	$2^{k/2}$	$2^{k/2}$	$2^{k/2}$

much lower than the exhaustive search and thus technically this is a break of cipher. Though even better theoretical attacks for block-ciphers exist in this setting [12] they are in direct correspondence to Babbage-Golic "birthday" tradeoff attacks and thus suffer from the same lack of flexibility due to $T = D$. Such attack will require impractical amount of 2^{64} fixed text encryptions as well as high storage complexity of 2^{64}. If one would try to implement these attacks he would prefer to use less data and less memory at the expense of more preprocessing and longer attack time. In Table 4.1 we summarize complexities of TMD attacks. Another important observation is that the attack is not exactly a chosen plaintext attack – since the specific value of the fixed plaintext is irrelevant. Thus in order to obtain the attack faster than exhaustive search the attacker will first check which plaintext is the most frequently used in the specific application, collect the data for various keys and then perform the attack. The attack is technically faster than the exhaustive search even if the attacker obtains a relatively small number of arbitrary fixed text encryptions. For example if the attacker obtains only 2^8 128-bit key AES encryptions, then after preprocessing of 2^{120} steps and using 2^{60} memory and 2^{120} analysis steps one of the keys would be recovered. In practical applications it might be a rather non-trivial task to ensure that the attacker never obtains encryptions of 2^8 fixed known plaintexts. This attack is much better than the existing state of the art attacks on 128-bit AES, which barely break 7-rounds of this cipher. Note that Biham's attack for the same amount of fixed text would formally have the same 2^{120} total complexity but would require unrealistic amount of memory 2^{120} which is probably the reason why such tradeoff attacks have not been viewed as a threat by the crypto community. In addition to all said above note that intentionally malicious protocol design may ensure that some fixed plaintext is always included into the encrypted stream (for example by fixing a header in communication, using communication to a fixed address or using fixed file header as is common in many applications). Note also that scenario of TMD attacks is more practical than that of related key attacks and thus such attacks should be benchmarked against the time-memory-key attacks.

Due to the importance of some tradeoff points we provide Tables 4.2–4.4 for AES with key lengths, 128/192/256 and compare them with best attacks known so far.

4.1.1 Key-size Consideration

Modern symmetric ciphers typically have keys larger or equal to 128 bits and they assume that exhaustive search is the best way one could recover a secret key[2].

However in a *variable key* scenario no k-bit cipher can offer a k-bit security against some

[2]Depending on the mode of operation used, there are also distinguishing attacks which may require about 2^{64} fixed key data, and do not lead to key-recovery. Those attacks are not considered to be of a threat by the community and are typically taken care of by key-change provisions.

Table 4.2: Tradeoff attacks on 128-bit key AES (and any other 128-bit key cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^8	2^{120}	2^{60}	2^{120}
BS TMD	FKP	2^{20}	2^{100}	2^{58}	2^{108}
BS TMD	FKP	2^{32}	2^{80}	2^{56}	2^{96}
BS TMD	FKP	2^{43}	2^{84}	2^{43}	2^{85}
Biham[12]	FKP	2^{64}	2^{64}	2^{64}	2^{64}
GM collision*	CP	2^{32}	2^{128}	2^{80}	?
FSW partial sum*	CP	2^{128}–2^{119}	2^{120}	2^{64}	?

* — only 7 out of 10 rounds. FKP – fixed known plaintext, CP – chosen plaintext.

Table 4.3: Tradeoff attacks on 192-bit key AES (and any other 192-bit key cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^{48}	2^{96}	2^{96}	2^{144}
BS TMD	FKP	2^{64}	2^{128}	2^{64}	2^{128}
Biham[12]	FKP	2^{96}	2^{96}	2^{96}	2^{96}

FKP – fixed known plaintext.

Table 4.4: Tradeoff attacks on 256-bit key AES (and any other 256-bit key cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^{64}	2^{128}	2^{128}	2^{192}
BS TMD	FKP	2^{85}	2^{170}	2^{85}	2^{170}
Biham[12]	FKP	2^{128}	2^{128}	2^{128}	2^{128}

FKP – fixed known plaintext.

quite practical attacks. One may assume that this problem can be cured by introducing the IV which has to be of the same size as the key. However in such popular block-cipher modes of operation like CBC due to a simple XOR of the *known* IV with the first plaintext block the attacker capable of mounting chosen plaintext attack can easily obtain encryptions of arbitrary fixed text under different keys. In the less likely but still not impossible case of a chosen IV attack other modes of operation like CFB, OFB or counter mode become vulnerable as well. A careful study of what should be the IV size in order to avoid tradeoff attacks is given in [70], however a simple rule of a thumb is that the IV size should be at least equal to the key-size, since the state of the cipher at any given moment has to be twice the key-size in order to avoid birthday time-data attacks [12, 4, 62]. XORing of the IV into the plaintext/ciphertext should be avoided.

Following these simple observations it is clear that 80-bit (or less) key ciphers should not be used since they allow for practical attacks in real-life scenarios, while 128-bit ciphers (which in practice provide security of about 80-bits) should not be used when full 128-bit security is required. At least 192-bit keys should be used for this security level.

One may argue that generic tradeoff attacks do not exploit weaknesses of specific designs and thus should be considered separately from other attacks. There are two counter-arguments to this point: first of all we have at the moment no proof that existing tradeoff attacks (such as Hellman's attack) are the best possible and thus a popular maxim "The attacks only become better, they do not get worse" may still apply. Moreover tradeoff attacks may be sped up by specific properties of the design, for example by what is called in a stream cipher case — cipher's sampling resistance [21]. In the case of stream cipher LILI-128 low sampling resistance was used to obtain tradeoff attack [108] with a complexity much lower than a naive application of a tradeoff technique would suggest.

It seems that we will have to give up the convenient world in which we assumed a k-bit security for a good k-bit cipher.

4.1.2 On Software and Hardware Cryptanalysis

Most of the classical cryptanalytic attacks assume that the attack has to be run once to find the key (or part of the key) or to distinguish a cipher from random. Single break in that sense is sufficient to demonstrate that the cipher is broken.

Classical attacks (like differential or linear and others) assume a *software* attack scenario in which the attack is performed on a general purpose computer or on a network of such computers. After the attack the computer can be used for other non-cryptanalytic tasks. This way cost of hardware is not a parameter in the classical attacks, and their complexity is measured as $max(T, M, D)$, as long as the memory requirement M is reasonable (ex. below 2^{45} bytes). The two other important parameters are: probability of success and ability to parallelize the attack.

If the attack is parallelizable, it means that the attacker who has additional resources can bring the attack time down by spending money on additional hardware. Typically linear speedup is expected unless mentioned otherwise. This classical view on cryptanalytic attack allows to avoid such considerations as amortized cost and what happens if the attacker needs to recover many keys in a short fixed period of time (i.e. the value of each key is decreasing

with time).

On the other hand, if the attacker builds special hardware for the cryptanalytic attack he is not interested in a single attack but rather in multiple attacks finding many keys after the machine is built. It is not clear what should be the lifetime as well as the maintenance cost of such device. Such attacker would also have a tradeoff of building an expensive machine which will find more keys per time unit, than a cheaper machine which will find less. This results in a so called *throughput cost* metric, in which the cost per key is:

$$\frac{C \cdot T}{K}$$

where C is the machine's cost, T is the time needed to recover K keys.

The complexity of attack in this metric is thus proportional to the product of the hardware cost and the time to find one key. This measure still ignores the value of the recovered key for the attacker. It is quite possible that this value differs greatly in whether the key is recovered fast (and thus possibly allows for exploitation of the key while it is still in use) or not.

Assymptotics of this approach is studied in [125] and some additional though sometimes misleading remarks are given in [8]. One important conclusion would be that cryptanalytic attacks that have very high memory requirements may be less efficient than exhaustive search which uses many parallel processors rather than a huge memory. One important caveat however is that cost coefficient between CPU and memory hardware is very high and should not be ignored. For example RAM is $2^{20} - 2^{30}$ cheaper than dedicated processor hardware and hard disk memory is further orders of magnitude cheaper than RAM. Note that for example time-memory trade off attacks do not need large RAM but can store their tables in hard disk memory by using tables with distinguished points. Note also that TMD attacks are highly parallelizable and thus comparison of non-parallelized TMD attack with a parallel exhaustive search machine given in [8] would be misleading. A proper comparison can be seen in [125] and it shows that in the throughput metric Hellman's tradeoff is much more efficient than the exhaustive search.

Chapter 5

Structural Properties of the AES

5.1 Representations of the AES

Although the general principle in cryptography is that cryptosystems are presented in a clear and natural manner, it is often the case that a cipher can be represented in some alternative way. Though not always appreciated, there are many benefits in studying a cryptosystem in an alternative way, as these new representations can often reveal mathematical structure that was hidden, permit calculations that were previously considered intractable and encourage the development of ideas about the analysis of the cryptosystem. Additionally, alternative presentations of a cipher may provide more efficient implementations, or protect the cipher against some forms of side-channel attacks, such as timing or power analysis.

A number of alternative representations of the AES have been considered in the last few years [7, 98, 99]. Although none of these has so far provided substantial benefit for the cryptanalysis of the AES, it is not inconceivable that new techniques can be developed that can exploit the AES rich mathematical structure in the analysis of the cipher.

5.1.1 Dual Ciphers of the AES

In [7] Barkan and Biham define a number of alternative representations of the AES, which are based on the concept of *dual cipher*.

Definition 1 ([7])**.** *Two ciphers E and E' are called* dual ciphers *if they are isomorphic, i.e., if there exist invertible transformations f, g and h such that*

$$\forall P, K \qquad f(E_K(P)) = E'_{g(K)}(h(P)).$$

By using the automorphisms of the finite field $\mathbb{F} = \mathrm{GF}(2^8)$ (generated by the Frobenius map $a \mapsto a^2$) and the different representations of the field $\mathbb{F}$ itself (via the explicit isomorphisms between fields of order 2^8), they were able to construct $30 \cdot 8 = 240$ non-trivial dual ciphers of the AES. This list was later extended in [118], who obtained $1170 \cdot 8 = 9360$ dual ciphers.

The dual ciphers of the AES were used to provide an insight into some of the choices made in the design of the AES, such as the use of the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ for

defining the field $\mathbb{F}$. In fact Barkan and Biham conclude in [7] that a change of polynomial (for example, the use of a primitive polynomial) should not affect the strength of the cipher. There has not been however any significant application of the dual ciphers in the cryptanalysis of the AES.

5.1.2 Embeddings of the AES

Some of the representations of the AES were constructed by defining an *embedding* of the AES into another cipher. Embeddings are standard techniques for analysing a mathematical structure, and given its highly algebraic structure, the AES seems to be a prime candidate on which to apply and analyse different embedding methods [34].

The BES Extension of the AES

The embedding of the AES in a larger cipher called Big Encryption System (BES) was introduced in [99]. The main motivation for this construction was to represent the AES within a framework where the cipher could be expressed through simple operations (inversion and affine transformation) in the finite field $\mathbb{F} = \mathrm{GF}(2^8)$.

The AES encryption process is typically described using operations on an array of 16 bytes, where each byte can be regarded as an element of the field $\mathbb{F}$. One the other hand, the BES operates on 128-byte blocks with 128-byte keys.

The embedding function for the BES embedding is based on the vector conjugate mapping $\phi : \mathbb{F} \to \mathbb{F}^8$ [99], which maps an element of $\mathbb{F}$ to a vector of its eight conjugates. Thus ϕ is an injective ring homomorphism given by

$$\phi(a) = \left(a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3}, a^{2^4}, a^{2^5}, a^{2^6}, a^{2^7}\right).$$

If $\mathbf{A} = \mathbb{F}^{16}$ and $\mathbf{B} = \mathbb{F}^{128}$ denote the state spaces of the AES and BES, respectively, the mapping ϕ above can be extended in the obvious way to a vector conjugate mapping $\phi : \mathbf{A} \to \mathbf{B}$ given by

$$\phi(\mathbf{a}) = \phi(a_0, \ldots, a_{15}) = (\phi(a_0), \ldots, \phi(a_{15})),$$

which is an injective ring homomorphism.

The function ϕ is defined in such a way that the diagram below commutes[1], where $\mathbf{B}_A$

$$\begin{array}{ccccccc} & & \mathbf{A} & \xrightarrow{\phi} & \mathbf{B_A} & & \\ & & \downarrow & & \downarrow & & \\ k & \to & \boxed{\text{AES}} & & \boxed{\text{BES}} & \leftarrow & \phi(k) \\ & & \downarrow & & \downarrow & & \\ & & \mathbf{A} & \xleftarrow{\phi^{-1}} & \mathbf{B_A} & & \end{array}$$

denotes the image of ϕ.

[1] Without loss of generality, we consider the version of the AES with 128-bit keys and 10 encryption rounds.

The BES provides a quite concise mathematical alternative presentation of the AES, such that the cipher operations are expressed in this alternative presentation as two simple operations ("inversion" and affine transformation over the field GF(2^8)). This alternative description seems to offer some benefits in the analysis of the AES. An example is the study of the multivariate quadratic equation system over GF(2^8) for the AES that is based on the BES embedding [99], which has a much simpler structure than the one obtained directly from the AES (over GF(2)). More generally, it seems that the BES provides a more natural environment in which to study the algebraic properties of the AES.

"Weak" Extensions of the AES

Monnerat and Vaudenay have considered extensions of the AES and the BES, namely the CES and the Big-BES [98]. These were shown by the authors to be *weak* extensions in which cryptanalytic attacks could be easily mounted. They observed however that the weaknesses in the larger ciphers did not translate to weaknesses in the AES and BES, and were therefore of no consequence to the security of the AES. The main motivation of these "weak" constructions seems to have been to show some of the limitations of cipher embeddings in general.

It is currently the subject of some research whether alternatives representations of the AES can provide additional insight into the cipher. For example, it has been shown that the many different approaches to embeddings in the literature are in fact not algebraically equivalent [34]. So while some embeddings, by their very construction, cannot possibly offer additional insights into the cipher, it is quite possible that some forms of embeddings might bring benefits such as cryptanalytic or implementation insights.

5.2 Structural properties of the substitution layer

The AES substitution layer has been carefully designed in order to provide a high resistance to statistical attacks and in order to yield a low implementation cost on several platforms. The main counterpart of such an elegant and efficient design is that it presents a very rich structure which is likely to be exploited in a cryptanalysis. Besides all properties which have been discussed in the previous chapters, three additional characteristics of the AES substitution layer may influence its security.

5.2.1 Construction by concatenation of smaller S-boxes

A first very particular structure of the substitution layer comes from the fact that it corresponds to the concatenation of several copies of the same 8-bit S-box. This construction is commonly used in block cipher designs since it leads to efficient implementations. Indeed, it provides high performance on 8-bit processors and it allows hardware implementations of the S-boxes. Moreover, it requires a reduced amount of memory when the S-box is implemented by a lookup table, for instance on super-scalar processors.

However, this construction by concatenation has several consequences. First, it originates some multi-set properties, leading to the attacks described in Chapter 2. Then, even

if the number of rounds in the AES makes the classical statistical attacks infeasible, it is worth noticing that a substitution layer obtained by this construction cannot achieve optimal resistance to these attacks. For instance, for a 128-bit key, the 128-bit transformation corresponding to the AES substitution layer achieves the following parameters: its maximum linear probability (MLP) and its maximum differential probability (MDP) are both equal to 2^{-6}. These parameters are not optimal: for instance, the inverse function over $GF(2^{128})$ leads to MLP = MDP = 2^{-126}. The construction by concatenation has also an impact on higher order differential attacks: the AES S-box layer has degree 7 and its Walsh coefficients are all divisible by 2^{32}. These are the best values which can be obtained by concatenating sixteen 8-bit bijective S-boxes, but degree 127 and a divisibility of 4 only of the Walsh spectrum can be achieved for a general 128-bit bijective mapping [29].

Similarly, the use of such a concatenation has an impact on the number of quadratic multivariate equations in the input and output bits of the S-box layer. Each 8-bit S-box in the AES provides 39 linearly independent quadratic equations, leading to 624 equations for the 128-bit S-box layer. As a comparison, the inverse function over $GF(2^{128})$ provides 639 linearly independent quadratic equations [43]. Moreover, any substitution layer obtained by concatenating smaller S-boxes provides a sparse algebraic system. For instance, in the AES, each quadratic equation involves at most 16 of the 256 variables. This property may affect the performance of the underlying algebraic attack as shown in Section 3.4.

5.2.2 Use of a power function

Another major property of the AES S-box which could be exploited in an attack is that the used 8-bit S-box is affinely equivalent to a power function over a finite field, i.e., to a function of the form $x \mapsto x^s$ over $GF(2^n)$. This choice is imposed both by the low implementation complexity in hardware environments and by the existence of theoretical results for this family of functions only. When the AES was designed, the only known S-boxes achieving the best known MDP and MLP were equivalent to power functions. But, a few functions with the same cryptographic properties and which are not of that type have been exhibited recently [27]. Therefore, one can wonder whether the use of a power mapping introduces some weaknesses which could be avoided by using another S-box with the same characteristics.

A first well-known property of power functions is that all its Boolean coordinates (i.e., all output bits) are affinely equivalent. But, it is still an open issue to determine whether this property can be seen as an advantage or as disadvantage for the attacker (one might argue that all coordinates provide the same information) [60]. A second consequence is that the rich algebraic structure of the underlying field $GF(2^8)$ can be extensively used, probably in a much simpler manner for a power function than for a polynomial with many terms.

The impact of the choice of a power function on algebraic attacks is another open question. For instance, it can be checked that all bijective power mappings over $GF(2^8)$ have algebraic immunity at most 2, whereas the existence of 8-bit permutations is not contradicted by classical combinatorial arguments (see e.g. [1]).

5.2.3 Potential weaknesses induced by the Inverse S-box

The inverse S-box used in AES, is known to have very high non-linearity, and one could be tempted to design ciphers in which we only use this S-box, and everything else is excessively simple diffusion and key mixing. Such excessive simplicity may be dangerous, and there are several methods known to construct insecure ciphers of this type, as shown in [41, 79, 78, 39, 26].

Here is a list of reported constructions of insecure ciphers based on the inverse S-box:

1. **Ciphers weak with respect to interpolation attacks**. In these ciphers, the round function is weak because it can be approximated by a low degree polynomial (univariate or multivariate). Such (quite special) ciphers have been studied in [79, 78, 41].
2. **Ciphers weak with respect to homographic approximation attacks**. In these attacks a round function (or a part of it) has an approximation of the form $X \mapsto \frac{\alpha X+\beta}{\gamma X+\delta}$. These attacks and their generalisations are studied in [41] following an early example known already from [79, 78].
3. **Ciphers weak with respect to bi-linear attacks**. These attacks work only for Feistel ciphers (both balanced with two equal branches, and also unbalanced with several branches). Many examples of weak ciphers of this type are proposed in [39, 41]. They use bi-linear expressions that can be summed-up for a whole cipher.
4. **Ciphers weak with respect to multi-linear attacks**. The multi-linear cryptanalysis is an extension of the bi-linear cryptanalysis. It uses equations of degree higher than two. Again, it allows to construct insecure ciphers based on the inverse S-box, and one example has been proposed in [41].

It should be noted that, on one side, all these ciphers have a lot of "special" structure and have been designed to be insecure. They may all appear as an exercise in designing very special insecure ciphers with highly non-linear components, an exercise with no other purpose. On the other side, the variety of insecure ciphers exploiting some multivariate equations is growing. All of them have a probabilistic version, by assuming that the equations/property exploited in the attack holds with some probability $\neq 1$. Thus they could be extended to a rather general attack that could be applied to any block cipher. Therefore, they remain interesting and should be studied.

5.3 Cyclic and Group Properties of the AES

If $\mathcal{E}$ is a block cipher with message space $\mathcal{M} = \{0,1\}^n$ and key space $\mathcal{K} = \{0,1\}^k$, then there exists a mapping $\sigma : \mathcal{K} \longrightarrow S_{\mathcal{M}}$, where each key k corresponds to a permutation $\sigma_k = \sigma(k)$ in the symmetric group on the set $\mathcal{M}$. Study of the subgroup of $S_{\mathcal{M}}$ generated by the set $\mathcal{E} = \{\sigma_{K_1}, \ldots, \sigma_{K_{2^k}}\}$ (where $\mathcal{E}$ can be either the full cipher or the cipher round function) can often provide an insight of the overall structure of the algorithm.

Ralph Wernsdorf showed in [124] that the round functions of the AES generate the whole alternating group on the set $\{0,1\}^{128}$. He points out that this indicates that several thinkable

regularities (like the existence of non-trivial factor groups or a too small diversity of occurring permutations in the Rijndael algorithm) can be excluded.

Further algebraic properties of the AES round function were studied in [94]. Several cycle lengths for the round function were computed, and a special complementation-like property of the round function was presented. Although the results could not be extended in a straightforward manner to the full AES cipher, the authors remark that the AES has many algebraic properties that are not found in other block ciphers. It is not known whether these properties, combined with other approaches, can lead to further insights and analysis methods to be applied against the AES [94].

Chapter 6

Cache timing attacks

Side-channel attacks are a type of cryptanalysis that take advantage of some physical information leaked by a cryptographic device (e.g. timing, power consumption). Since the first work of Kocher [90], several classes of side-channel attacks, such as differential power analysis, have been firmly established [55]. Until recently, most of these attacks were based on specific features of some software implementations of the basic transformations involved in the targeted algorithm. For instance, a timing attack against AES due to Koeune and Quisquater [91] applies if the `MixColumns` transformation uses a particular implementation of the multiplication in $GF(256)$.

However it is usually believed that such weaknesses of AES do not appear for optimized implementations which are dedicated to 32-bit processors. In this case, both `MixColumns` and `SubBytes` transformations are implemented by lookup tables in order to decrease the encryption time. More precisely, one table of 256 words is generally used for representing how an input byte maps to a column of the 4×4 output block. Thus, a total of four tables, each of 1 Kbyte, are needed for the whole encryption. AES encryption and decryption then mainly consist of a sequence of memory accesses to these lookup tables. Unfortunately, memory accesses are not always performed in constant time. In particular, the use of cache memory has a great impact on the latency and on the power consumption of a memory access. Timing analysis or power analysis may then provide some side-channel information that can be exploited by the cryptanalyst, as mentioned by Kocher [90] and Kelsey *et al.* [88]. Side-channel attacks based on the analysis of the cache behavior for a block cipher have drawn much attention during the last three years [105, 115, 9, 92, 10, 103, 28].

6.1 Basics on cache memory

Since the gap between the latency of memory and the speed of processors is still increasing, the bus bandwidth and the access speed to the main memory become the limiting factors in the overall processor throughput. This bottleneck is overcome by cache memory. A cache is a small piece of high speed memory. The goal of a cache is to keep the CPU as busy as possible by minimizing the load/store latency to the main memory. A cache is divided into blocks (or lines) of fixed size and typical block sizes are 32, 64, 128 bytes. The cache associativity

determines how the main memory blocks map into cache blocks. An m-way associative cache is divided into sets of m blocks. A main memory block can be mapped in any block of a given set. The block selection within a set is performed using a replacement algorithm like LRU [68].

The cache parameters affect the execution time of an algorithm that uses a data array of size S as follows. The data array is mapped into the cache according to its alignment and to the size s_e of its elements. The alignment of data can be considered as a constraint on its address. For instance, on most architectures the address of data is a multiple of the size of the addressed data. The data array is divided into $\lceil S/s_b \rceil$ blocks of size s_b and each block can hold at most $\lceil \frac{s_b}{s_e} \rceil$ elements. Then, when an element of the array is accessed, the relative addressing can be seen as a block selection and an offset into the block. The block offset requires $\lceil \log_2(\frac{s_b}{s_e}) \rceil$ bits and the block selection $\lceil \log_2(\frac{S}{s_b}) \rceil$-bits. In modern processors, we find two levels of on-chip cache (respectively called L1 and L2 cache). Those caches have different parameters. Other memory parameters can affect the behavior of the execution such as cache write policies, TLB [68, 69]. Moreover, modern processors feature very complex interactions between instructions. The microscopic execution time of any sequence of instructions depends on these interactions.

6.2 Different types of cache attacks and their applicability

When a data block is accessed, two situations may occur. If the block lies in the cache memory, this access is called a hit. Otherwise the block has to be loaded from the main memory; this situation is referred as a miss. The power consumption and timing of a device may highly vary depending on whether the memory accesses are performed from the cache or from the main memory. This means that when timing a sequence of memory accesses, the number of access hits can be successfully detected.

During the execution, we can distinguish three families of cache misses according to [69]:

- cold start misses, which arise for the first reference to a data;
- capacity misses, which occur if the size of the data array exceeds the size of the cache;
- conflict misses, which may happen only if accesses can provoke the eviction of a recently accessed data.

Capacity misses are usually not relevant in the context of symmetric encryption since a reasonably high encryption throughput can be achieved only if the sizes of the involved lookup tables do not exceed the cache capacity.

Thus the behavior of cache memory during an encryption (or decryption) depends both on the cache initial state and on the sequence of memory accesses which are performed. The assumptions made by the attacker on the cache initial state may highly vary according to the targeted cryptographic device. Since these strongly affect the practicability of the attack, we classify cache attacks according to the corresponding requirements on the initial state of the cache:

- *Empty initial state (reset attack)*: these attacks require that no table involved in the encryption algorithm are contained in the cache before any observation. This class of attacks is mainly based on the observation of cold start misses.
- *Forged initial state (initialization attack)*: in these attacks the adversary must be able to trigger the cache into a known state before the encryption. This means that the attacker will generate a chosen number of cold start misses.
- *Loaded initial state (micro-architecture attacks)*: the cache already holds all the tables involved in the encryption algorithm.

For a given initial state, the sequence of memory accesses performed during the encryption can be observed by timing analysis or by power analysis as suggested in [88]. The power analysis allows the attacker to observe the encryption access by access whereas timing attack gives a global measure of the events that occur during the encryption. As a result, power analysis of a cache memory will give more information than timing.

Another class of attacks has been proposed by Osvik *et al.* in [104]. Here, the attacker is able to directly observe the content of the cache using software probe. This information is mostly obtained by using interactions between two data tables that share the same cache. The first array C will be the lookup table involved in the encryption algorithm. The second array D is controlled by the attacker. During the encryption the attacker performs a sequence of accesses to D. These accesses may suffer from conflict misses due to table C. Therefore by performing chosen accesses to D, the attacker is able to generate some conflict misses which are related to the secret key. This kind of attacks is only possible on multi-threaded processors [117] such as the Pentium 4 HT. To carry such an attack the attacker must be able to read and trigger the cache either both before and after the encryption or during the encryption. In this sense, the practical applicability of these attacks is similar to the one of forged state attacks, and the limits that apply to forged state will also apply to this attack.

6.2.1 Attacks starting from empty cache

The complexity of resetting a cache memory (*i.e.*, a cache flush) depends on the target device. By nature, cache memory is volatile. As a consequence, simply removing the voltage supply of the device will clear the cache. This is only affordable on embedded systems like smart cards. On more complex systems like computers, the attacker needs a user account on the targeted host. Actually, the cache is reset by triggering all cache blocks with many memory accesses before any observation.

In this particular context, a cache miss occurs when the accessed element does not belong to a block which already lies in the cache. A cache block corresponds to a set of elements with the same $\lceil \log_2(\frac{S}{s_b}) \rceil$ most significant bits. Then a cache hit corresponds to a collision on the $\lceil \log_2(\frac{S}{s_b}) \rceil$ most significant bits of the input of the table. In this sense, an attack starting from an empty cache can be seen as a partial collision attack [111, 110] on the addressing function of a cache memory. For instance on Pentium 3 (for which L1 data cache block size is $s_b = 32$), cache misses for the AES lookup tables involve the five most significant bits of each key byte.

The first practical implementations of such attacks have been described against MISTY1 and DES by Page [105] and Tsunoo *et al.* [116]. It was pointed out in [116] that a pair of plaintext/ciphertext which leads to a high miss ratio (*i.e.*, to a long encryption time) provides an improbable value for the difference between some bits of the first and the last round keys. After several observations the correct partial difference can then be deduced from the value which appears the least frequently. This work also illustrates the impact of data mapping into the cache. DES Sboxes have 64 inputs of 1 byte. But better performance may be achieved when each entry is aligned on 4-byte boundary. Unfortunately this decreases the number of Sbox inputs per cache block, and it considerably affects the cryptanalysis as pointed out in [115]. This attack successfully applies to MISTY1 and to the AES.

Another reset cache attack dedicated to embedded devices has been proposed by Lauradoux in [92]. It uses power analysis to recover linear relations on the most significant bits of all bytes of the AES secret key. The attack was demonstrated against different AES implementations and is similar to the timing attack of [115]. It exploits the fact that the first input of the Sbox lookup tables in AES corresponds to the XOR between the plaintext and the secret key.

6.2.2 Attacks starting from initialized cache

In this class of attacks, the adversary must be able to initialize some chosen cache blocks with data from the lookup tables. This can be done by flushing the cache memory and then performing fake encryptions with a known key in order to load certain table blocks into the cache. This requirement limits the scope of the attack to multi-users systems since an access to the cache memory is needed. With several chosen initializations and some power traces, Bertoni *et al.* [10] show how to recover the most significant bits of each key byte in AES.

This class of attacks is similar to the attacks that start from empty cache in the sense that are based on the analysis of cache misses. The main difference is that they use conflict misses instead of cold start misses to gain some information on the most significant bits of the key bytes.

6.2.3 Attack starting from loaded cache

Both classes of attacks previously presented can be thwarted by systematically loading the whole table before any encryption, as proposed in [10, 92]. This simple countermeasure completely removes the possibility to observe cache hits from misses. However, this does not imply that all timing variations have been removed. For instance, the cache memory on superscalar processors can handle more that one access per cycle. Several solutions have been proposed to implement this feature [126, 112]. One solution used in the design of X86 processor consists in splitting the cache into several independently addressed banks. Despite cache banks, memory accesses can suffer from conflict penalties when several simultaneous concurrent references to the same bank occur. Such penalties affect and are affected by the whole micro-architecture of the processor (e.g., pipeline, conflict banks detection logic, load/store queue). Some conflicts between the tables and the inputs (*i.e.*, the plaintext or the ciphertext in the case of decryption) can also caused the data to be moved from the L1 cache to the L2 cache.

Because of all these features, some timing variations still exist as pointed out by Bernstein [9]. For instance, Figure 6.1 exhibits the timing variations for the NIST implementation of AES on a Pentium 3 with a loaded cache initial state.

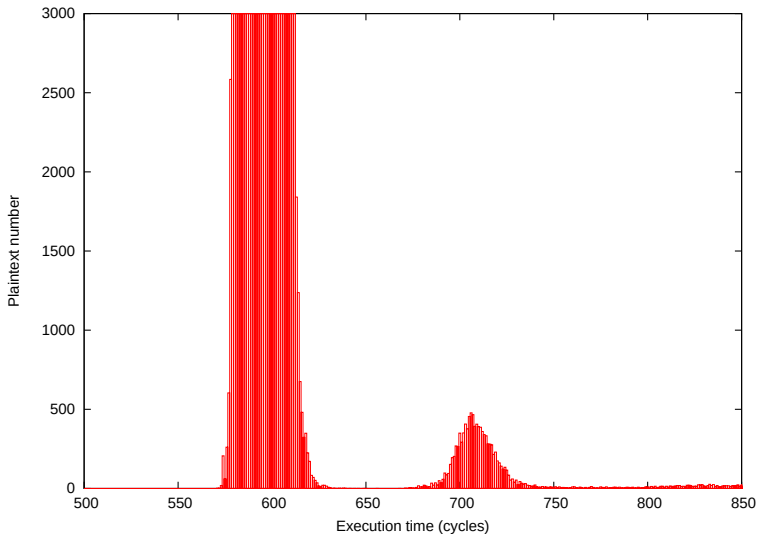

Figure 6.1: Distribution of timing of AES NIST implementation on a Pentium 3 for 2^{20} plaintexts (the peak of the distribution is here truncated but almost reaches 2^{19})

These variations can then be exploited for mounting an attack. These attacks are based on the existence of some relations between the encryption time and the input block of the first AES round (which corresponds to an XOR between the plaintext and the secret key and which is represented by a 4×4 array $(A_{i,j})$). Unfortunately the whole micro-architecture of a processor is a complex system which is difficult to model; moreover, most micro-architecture details are not documented (like the structure of cache banks in the cache). Taking this fact into account, the relations between the encryption time and the input of the first round can only be identified by a pre-computational step which must be performed on the same architecture as the targeted processor. Therefore, the attack consists of both following steps:

1. the attacker tries to identify some relevant relations between the encryption time and the input of the first AES round.

2. the attacker guesses some key bits by comparing observations and the relations learned from the first step.

In Bernstein's attack, the precomputation step consists in determining the parameters (average and standard deviation) of the distributions of the encryption time for each value of the byte A_i. It is observed that these timing distributions may present important variations when the value of A_i changes. Using this precomputation, the attack presented in [9] consists in computing the same parameters when some known plaintexts are encrypted with the unknown secret key. Then, the cross-correlation between both distributions may allow to recover some key bits.

From the simulations given in [9, Page 10], we can observe that 78 key bits are recovered after 2^{25} AES encryptions on a Pentium 3. Most of these key bits correspond to the five most significant bits of the key bytes, *i.e.*, the bits involved in cache misses. Therefore, it seems that most of the recovered key bits in these simulations come from cache misses which are due to some array manipulations performed before encrypting each block (corresponding to lines 20-24 of `server.c`). However, the simulations reported in [9] show that a few of the least significant bits of the key bytes can also be recovered. A variant of Bernstein's attack where all cache misses are clearly avoided can then be mounted as proposed in [28]. This variant uses a modified statistical test and does not involve the same key bits as Bernstein's attack. With the NIST implementation on a Pentium 3 and 2^{30} known plaintext-ciphertext pairs, it correctly recovers 66.75 key bits in average.

A key issue in these attacks is that their efficiency is highly related to the compiler (and to the compiler options), to the operating system and to the whole micro-architecture. For instance, Table 6.1 shows how the performance of the attack described in [28] varies for several processors in the Pentium 4 family.

CPU model (familly/model/ stepping)	Frequency (Ghz)	Predicted key bits	Error Rate	Average AES timing	Standard deviation AES timing
15/4/1	3.2	17	10 %	660	25
15/3/3	3	4	50 %	690	9
15/2/4	2.0	55	10 %	820	33
15/2/7	2.4	80	10 %	659	41
15/2/9	2.6	75	10 %	658	40
15/2/5	2.8	78	10 %	654	40

Table 6.1: Evaluation of the cache attack on Pentium 4 processors against the NIST implementation

6.3 Comparison between all cache attacks against the AES

The previous discussion points out that the applicability of the different families of cache attacks highly depends on the type of the targeted device. For embedded devices, the most natural situation is an attack starting from an empty cache. For this type of target, power analysis is obviously preferred to timing analysis since it enables to observe the memory access sequence step by step. For non-embedded devices (e.g. for PCs), the situation is very different. Initialization attacks are obviously much more powerful but their field of application is limited to multi-user systems. For single user-systems or in the context of remote timing attacks, the only realistic hypothesis is a loaded initial state.

The typical cache attacks depending on the type of the targeted device are summarized in Table 6.2.

All previously proposed cache attacks against the AES (with a 128-bit secret key) are compared in Table 6.3, both in terms of efficiency and applicability. The last column in

Type of devices	Class	Cache state
embedded device	power	reset
multi-user system	memory	initialization
single user	loaded	timing

Table 6.2: Typical cache attacks depending on the targeted device

the table gives the number of information bits on the secret key recovered in the attack, as stated by the authors. In all these attacks, the time complexity roughly corresponds to the encryption cost of all required plaintexts.

Attack	Nature	Cache state	Complexity	Key bits
[9]	Timing	Init/Loaded	2^{27}	91
[28]	Timing	Loaded	2^{30}	66.7
[92]	Power	Reset	2^{5}	75
[115]	Timing	Reset	2^{18}	80
[10]	Power	Init	2^{32}	80
[104]	Memory	Init	2^{14}	128

Table 6.3: Comparison between known cache attacks against the AES. Results in brackets corresponds to Pentium 3 cache parameters

6.4 Remote timing attacks

An interesting extension of timing attacks are remote timing attacks. Since timing attacks apply to weak implementations of a cryptographic algorithm, it is possible to transpose them to remote devices. The assumptions requested to mount a remote timing attack are completely different depending on whether we consider public or secret-key algorithms. In the case of public-key cryptography, assumptions are really weak. The attacking machine (Eve) sends a request to the targeted server (Alice) using Alice's public key. By measuring the response time, Eve tries to deduce Alice's private key. This attack has been first explored by Brumley and Boneh against RSA [23].

However secret-key algorithms can not be attacked in such a way. Indeed there is no particular reason to hope that some data encrypted with the secret key K_{AB} are voluntarily sent by Alice to Eve since Eve does not know the secret key K_{AB}. Therefore, such ciphertexts are available to the attacker only if they are eavesdropped in the context of a man-in-the-middle attack. For instance, Eve can probe a routing element of a communication channel between Alice and Bob. She will monitor the traffic on this communication channel to measure the response time of both parties.

The most difficult task in all remote timing attacks is to evaluate the encryption time of the target. Indeed the attacker has to take into account the transmission time from the target to the attacking machine. The noise added by the transmission delay to the encryption

time is considerable. That is why in practice the only known remote timing attack [23] was mounted on a local network. The attack presented by Bernstein [9] bypasses this problem since it applies on a protocol in which the server's timestamp is also transmitted and available to the attacker. This feature then allows the attacker to directly access the encryption time without the noise added by the transmission through the network. This considerably weakens the security model of the target.

Bibliography

[1] Frederik Armknecht. On the existence of low-degree equations for algebraic attacks. In *ECRYPT Network of Excellence - SASC Workshop Record*, pages 175–189, 2004. Available via `http://www.ecrypt.eu.org/stvl/sasc/record.html`.

[2] Gwénolé Ars. Applications des bases de Gröbner à la cryptographie. Doctoral dissertation, June 2005.

[3] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison Between XL and Gröbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2004.

[4] Steve Babbage. Improved "exhaustive search" attacks on stream ciphers. In ECOS 95 (European Convention on Security and Detection), volume 408 of *IEE Conference Publication*, May 1995.

[5] Thomas Baignéres, Pascal Junod, and Serge Vaudenay. How far can we go beyond linear cryptanalysis. In P.J. Lee, editor, *Proceedings of Asiacrypt'04*, number 3329 in Lecture Notes in Computer Science, pages 432–450. Springer-Verlag, 2004.

[6] M. Bardet, J.-C. Faugère, and B. Salvy. Complexity of Gröbner basis computation for semi-regular overdetermined sequences over F_2 with solutions in F_2. Technical Report 5049, INRIA, December 2003. `http://www.inria.fr/rrrt/rr-5049.html`.

[7] Elad Barkan and Eli Biham. In how many ways can you write Rijndael? In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 160–175. Springer, 2002.

[8] D. Bernstein. Understanding brute force. In *Talk at SKEW'05 workshop*, 2005.

[9] Daniel J. Bernstein. Cache-timing attacks on AES, 2005. `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`.

[10] Guido Bertoni, Vittorio Zaccaria, Luca Breveglieri, Matteo Monchiero, and Gianluca Palermo. AES power attack based on induced cache miss and countermeasure. In *International Symposium on Information Technology: Coding and Computing - ITCC '05*, pages 586–591. IEEE Computer Society, 2005.

[11] Eli Biham, editor. *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*. Springer, 1997.

[12] Eli Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. In *Information Processing Letters*, pages 117–124, 2002.

[13] Eli Biham, Orr Dunkelman, and Nathan Keller. Enhancing differential-linear cryptanalysis. In Yuliang Zheng, editor, *Proceedings of Asiacrypt'02*, number 2501 in Lecture Notes in Computer Science, pages 254–266. Springer-Verlag, 2002.

[14] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In Ronald Cramer, editor, *Proceedings of Eurocrypt'05*, number 3494 in Lecture Notes in Computer Science, pages 507–525. Springer-Verlag, 2005.

[15] Eli Biham and Nathan Keller. Cryptanalysis of reduced variants of Rijndael. In *Official public comment for Round 2 of the Advanced Encryption Standard development effort*, 2000. Available at `http://csrc.nist.gov/encryption/aes/round2/conf3/papers/35-ebiham.pdf`.

[16] Eli Biham and Nathan Keller. Cryptanalysis of reduced variants of Rijndael. In *AES Candidate Conference*, 2000.

[17] Alex Biryukov. The boomerang attack on 5 and 6-round reduced AES. In *Proceedings of AES 2004*, number 3373 in Lecture Notes in Computer Science, pages 42–57. Springer-Verlag, 2005.

[18] Alex Biryukov and Christophe De Cannière. Block Ciphers and Systems of Quadratic Equations. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2003.

[19] Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In Dan Boneh, editor, *Proceedings of Crypto'03*, Lecture Notes in Computer Science. Springer-Verlag, 2003. Full version available at `http://eprint.iacr.org/2003/109/`.

[20] Alex Biryukov, Christophe De Cannière, and Michael Quisquater. On multiple linear approximations. In M. Franklin, editor, *Proceedings of Crypto'04*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[21] Alex Biryukov and Adi Shami. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In Proceedings of Asiacrypt'00, pages 1–13. Springer-Verlag, 2000.

[22] Alex Biryukov and Adi Shamir. Structural cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2001.

[23] David Brumley and Dan Boneh. Remote timing attack are practical. In *12th USENIX Security Symposium*, pages 1–14, 2003.

[24] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of Groëbner basis. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation*, volume 72 of *Lectures Notes in Computer Science*. Springer Verlag, 1979.

[25] B. Buchberger. Gröebner bases: an algorithmic method in polynomial ideal theory. In *Multidimensional Systems Theory*. D. Reidel Publishing Company, 1985.

[26] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. Block ciphers sensitive to gröbner basis attacks. In *CT-RSA*, pages 313–331, 2006.

[27] Lilya Budaghyan, Claude Carlet, and Alexander Pott. New constructions of Almost Bent and Almost Perfect Nonlinear polynomials. In *Workshop on Coding and Cryptography – WCC 2005*, pages 306–315, Bergen, Norway, March 2005.

[28] Anne Canteaut, Cédric Lauradoux, and André Seznec. Understanding cache attacks. INRIA Research Report, 2006. To appear.

[29] Anne Canteaut and Marion Videau. Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 518–533. Springer-Verlag, 2002.

[30] Jiun-Ming Chen and Bo-Yin Yang. All in the XL Family: Theory and Practice. In *Proceedings of the 7th International Conference on Information Security and Cryptology*, volume 3506 of *Lecture Notes in Computer Science*, pages 67–86. Springer, 2004.

[31] Jiun-Ming Chen and Bo-Yin Yang. Theoretical Analysis of XL over Small Fields. In *Proceedings of the 9th Australasian Conference on Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 277–288. Springer, 2004.

[32] Jung Hee Cheon, MunJu Kim, Kwangjo Kim, Jung-Yeun Lee, and SungWoo Kang. Improved impossible differential cryptanalysis of Rijndael and Crypton. In Kwangjo Kim, editor, *Proceedings of ICISC'01*, number 2288 in Lecture Notes in Computer Science, pages 39–49. Springer-Verlag, 2001.

[33] Carlos Cid and Gaetan Leurent. An Analysis of the XSL Algorithm. In *Advances in Cryptology - ASIACRYPT 2005*, Lecture Notes in Computer Science. Springer Verlag, 2005.

[34] Carlos Cid, Sean Murphy, and Matthew Robshaw. An Algebraic Framework for Cipher Embeddings. To appear at the *IMA Conference on Coding and Cryptography*, Cirencester UK, 2005.

[35] Carlos Cid, Sean Murphy, and Matthew Robshaw. Small Scale Variants of the AES. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption - FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 145–162. Springer-Verlag, 2005.

[36] Nicholas Courtois and Jacques Patarin. About the XL algorithm over GF(2). In *Topics in Cryptology - CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer, 2003.

[37] Nicolas Courtois. La sécurité des primitives cryptographiques basées sur les problèmes algébriques multivariables MQ, IP, MinRank, et HFE. PhD Thesis - Paris 6 University - France, 2001.

[38] Nicolas Courtois. The Security of Hidden Field Equations (HFE). In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 266–281. Springer, 2001.

[39] Nicolas Courtois. Feistel schemes and bi-linear cryptanalysis. In *Advances in Cryptology - CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 23–40. Springer Verlag, 2004.

[40] Nicolas Courtois. General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Fourth Conference on the Advanced Encryption Standard - AES4*, volume 3373 of *Lecture Notes in Computer Science*, pages 67–83. Springer-Verlag, 2004.

[41] Nicolas Courtois. The Inverse S-box, Non-linear Polynomial Relations and Cryptanalysis of Block Ciphers. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Fourth Conference on the Advanced Encryption Standard - AES4*, volume 3373 of *Lecture Notes in Computer Science*, pages 170–188. Springer-Verlag, 2004.

[42] Nicolas Courtois. Non-linear polynomial relations and cryptanalysis of block ciphers. In *Advances in Cryptology - AES 4 conference, Bonn, May 10-12 2004*, volume 3373 of *Lecture Notes in Computer Science*, pages 170–188. Springer Verlag, 2005.

[43] Nicolas Courtois, Blandine Debraize, and Eric Garrido. On exact algebraic [non-]immunity of s-boxes based on power functions. IACR ePrint Report 2005/203, June 2005. `http://eprint.iacr.org/2005/203`.

[44] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000.

[45] Nicolas Courtois and Willi Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[46] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, 2002.

[47] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[48] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. Available at `http://eprint.iacr.org/2002/044/`, 2002.

[49] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *Proceedings of Asiacrypt'02*, number 2501 in Lecture Notes in Computer Science, pages 267–287. Springer-Verlag, 2002. Different version of the preprint [48].

[50] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer-Verlag, second edition, 1997. An introduction to computational algebraic geometry and commutative algebra.

[51] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In Biham [11], pages 149–165.

[52] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — The Advanced Encryption Standard*. Springer, 2002.

[53] Joan Daemen and Vincent Rijmen. Statistics of correlation and differentials in block ciphers. Available at `http://eprint.iacr.org/2005/212/`, 2005.

[54] Claus Diem. The XL-Algorithm and a Conjecture from Commutative Algebra. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 323–337. Springer, 2004.

[55] ECRYPT - European Network of Excellence in Cryptology. The side channel cryptanalysis lounge. `http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html`, 2005.

[56] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 1999.

[57] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*. ACM, 2002.

[58] N. Ferguson, R. Shroeppel, and D. Whiting. A simple algebraic representation of Rijndael. In *Proceedings of Selected Areas in Cryptography*, pages 103–111. Springer-Verlag, 2001.

[59] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *Proceedings of Fast Software Encryption – FSE'00*, number 1978 in Lecture Notes in Computer Science, pages 213–230. Springer-Verlag, 2000.

[60] Joanne Fuller and William Millan. Linear redundancy in s-boxes. In Thomas Johansson, editor, *Fast Software Encryption - FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[61] Henri Gilbert and Marine Minier. A collision attack on seven rounds of Rijndael. In *Proceedings of the Third Advanced Encryption Standard Conference*, pages 230–241. NIST, April 2000.

[62] Jovan Dj. Golic. Cryptanalysis of alleged A5 stream cipher. In Walter Fumy, editor, Advances in Cryptology – EUROCRYPT'97, volume 1233, 1997.

[63] C. Harpes, G. Kramer, and J. Massey. A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma. In *Advances in Cryptology - EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 24–38. Springer Verlag, 1995.

[64] C. Harpes and J.L. Massey. Partitioning cryptanalysis. In Eli Biham, editor, *Proceedings of Fast Software Encryption – FSE'97*, number 1267 in Lecture Notes in Computer Science, pages 13–27. Springer-Verlag, 1997.

[65] Carlo Harpes. Cryptanalysis of iterated block ciphers. PhD thesis, No 11625, Swiss Federal Int. of Tech., ETH Series in Information Processing, Ed. J. L. Massey, Hartung-Gorre Verlag Konstanz, 1996, ISBN 3-89649-079-6, ISSN 0942-3044.

[66] Carlo Harpes. Partitioning cryptanalysis. Post-Diploma Thesis, Signal and Information Processing Lab., Swiss Federal Institute of Technology, Zurich, March 1995.

[67] Martin E. Hellman. A cryptanalytic time-memory tradeoff. In IEEE Transactions on Information Theory, 1980.

[68] John Hennessy and David Patterson. *Computer Architecture: a quantitative approach.* Morgan Kaufmann Publisher, Inc, 1996.

[69] Mark Hill. *Aspect of cache memory and instruction buffer performance.* PhD thesis, University of California, Berkeley, 1987.

[70] Jin Hong and Palash Sarkar. Rediscovery of time memory tradeoffs, 2005.

[71] Seokhie Hong, Jongsung Kim, Guil Kim, Sangjin Lee, and Bart Preneel. Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192. In Henri Gilbert and Helena Handschuh, editors, *Proceedings of Fast Software Encryption – FSE'05*, number 3557 in Lecture Notes in Computer Science, pages 368–383. Springer-Verlag, 2005.

[72] Graz University of Technology IAIK Crypto Group.

[73] et al. J. Nechvatal. Report on the development of the advanced encryption standard (aes), Ocotber2, 2000. available at http://csrc.nist.gov/CryptoToolkit/tkencryption.html.

[74] Goce Jakimoski and Yvo Desmedt. Related-key differential cryptanalysis of 192-bit key AES variants. In *Proceedings of Selected Areas in Cryptography – SAC'03*, number 3006 in Lecture Notes in Computer Science, pages 208–221. Springer-Verlag, 2004.

[75] Thomas Jakobsen. Correlation attacks on block ciphers. Master's Thesis, Dept. of Mathematics, Technical University of Denmark, January 1996.

[76] Thomas Jakobsen. Cryptanalysis of Block Ciphers with Probabilistic Non-Linear Relations of Low Degree. In *Advances in Cryptology - CRYPTO 1998*, volume 1462 of *LNCS*, pages 212–222. Springer-Verlag, 1998.

[77] Thomas Jakobsen and Carlo Harpes. Non-uniformity measures for generalized linear cryptanalysis and partitioning cryptanalysis. Proceedings of Pragocrypt'96, 1996.

[78] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Biham [11], pages 28–40.

[79] Thomas Jakobsen and Lars R. Knudsen. Attacks on block ciphers of low algebraic degree. *J. Cryptology*, 14(3):197–210, 2001.

[80] Pascal Junod. On the complexity of Matsui's attack. In *Advances in Cryptology - SAC 2001*, volume 2259 of *Lecture Notes in Computer Science*, pages 199–211. Springer Verlag, 2001.

[81] Pascal Junod. On the optimality of linear, differential and sequential distinguishers. In *Advances in Cryptology - EUROCRYPT'03*, Lecture Notes in Computer Science. Springer Verlag, 2003.

[82] Pascal Junod and Serge Vaudenay. Optimal Key Ranking Procedures in a Statistical Cryptanalysis. In *Advances in Cryptology - FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 235–246. Springer Verlag, 2003.

[83] Burton S. Kaliski, Jr and Matthew J. B. Robshaw. Linear cryptanalysis using multiple approximations. In Yvo Desmedt, editor, *Proceedings of Crypto'94*, number 839 in Lecture Notes in Computer Science, pages 26–39. Springer-Verlag, 1994.

[84] Toshinobu Kaneko and Takeshi Shimoyama. Quadratic relation of S-box and its application to the linear attack of full round DES. In *Advances in Cryptology - CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 200–211. Springer Verlag, 1998.

[85] Liam Keliher. *Linear Cryptanalysis of Substitution-Permutation Networks*. Doctoral dissertation, Queen's University, Kingston Canada, 2003.

[86] Liam Keliher. Refined analysis of bounds related to linear and differential cryptanalysis for the AES. In *Proceedings of AES 2004*, number 3373 in Lecture Notes in Computer Science, pages 42–57. Springer-Verlag, 2005.

[87] Liam Keliher, Henk Meijer, and Stafford E. Tavares. Improving the upper bound on the maximum average linear hull probability for Rijndael. In Serge Vaudenay and Amr M. Youssef, editors, *Proceedings of Selected Areas in Cryptography – SAC'01*, number 2259 in Lecture Notes in Computer Science, pages 112–128. Springer-Verlag, 2001.

[88] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side channel cryptanalysis of product ciphers. *Journal of Computer Security*, 8(2/3), 2000.

[89] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[90] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[91] Francois Koeune and Jean-Jacques Quisquater. A timing attack against Rijndael. Technical Report CG-1999/1, UCL Crypto Group, 1999.

[92] Cédric Lauradoux. Collision attacks on processors with cache and countermeasures. In *Western European WOrkshop on Research in Cryptography - WEWoRC '05*, Lecture Notes in Informatics. Springer-Verlag, 2005. To appear.

[93] D. Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, volume 162 of *Lectures Notes in Computer Science*. Springer-Verlag, 1983.

[94] Tri Van Le, Rüdiger Sparr, Ralph Wernsdorf, and Yvo Desmedt. Complementation-like and Cyclic Properties of AES Round Functions. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Fourth Conference on the Advanced Encryption Standard - AES4*, volume 3373 of *Lecture Notes in Computer Science*, pages 128–141. Springer-Verlag, 2004.

[95] Stefan Lucks. Attacking seven rounds of Rijndael under 192-bit and 256-bit keys. In *Proceedings of the Third Advanced Encryption Standard Conference*, pages 215–229. NIST, April 2000.

[96] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Verlag, 1993.

[97] M. Matsui. The first experimental cryptanalysis of the data encryption standard. In *Advances in Cryptology - CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer Verlag, 1994.

[98] Jean Monnerat and Serge Vaudenay. On Some Weak Extensions of AES and BES. In *Proceedings of Sixth International Conference on Information and Communications Security*, volume 3269 of *LNCS*, pages 414–426. Springer, 2004. Malaga, Spain.

[99] Sean Murphy and Matthew Robshaw. Essential Algebraic Structure within the AES. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 1–16. Springer-Verlag, 2002.

[100] Sean Murphy and Matthew Robshaw. Comments on the Security of the AES and the XSL Technique. *Electronic Letters*, 39:26–38, 2003.

[101] National Institute of Standards and Technology. FIPS-197: Advanced Encryption Standard, November 2001. Available at `http://csrc.nist.gov/publications/fips/`, see also [127].

[102] NESSIE consortium. NESSIE Security report. Deliverable report D20, NESSIE, 2002. Available from `http://www.cryptonessie.org/`.

[103] Mairéad O'Hanlon and Antony Tonge. Investigation of cache timing attacks on AES. `www.computing.dcu.ie/research/papers/2005/0105.pdf`, 2005.

[104] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of AES. Cryptology ePrint Archive, Report 2005/271, 2005. `http://eprint.iacr.org/`.

[105] D. Page. Theoretical use of cache memory as a cryptanalytic side-channel. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, June 2002.

[106] S. Park, S.H. Sung, S. Lee, and J. Lim. Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In Thomas Johansson, editor, *Proceedings of Fast Software Encryption – FSE'03*, number 2887 in Lecture Notes in Computer Science, pages 247–260. Springer-Verlag, 2003.

[107] Matthew G. Parker. Generalised S-Box nonlinearity. Public report, NESSIE, June 2002.

[108] Markku-Juhani Olavi Saarinen. A time-memory trade-off attack against LILI-128. In Joan Daemen and Vincent Rijmen, editors, Proceedings of Fast Software Encryption – FSE'02, volume 2365 of *Lecture Notes in Computer Science*, pages 231–236. Springer-Verlag, 2002.

[109] F. Sano, K. Ohkuma, H. Shimizu, and S. Kawamura. On the security of nested SPN cipher against the differential and linear cryptanalysis. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, (1):37–46, 2003.

[110] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A collision-attack on AES combining side channel- and differential-attack. In *Workshop on Cryptographic Hardware and Embedded Systems - CHES '04*, volume 3156 of *Lecture Note in Computer Science*, pages 163–175. Springer Verlag, 2004.

[111] Kai Schramm, Thomas J. Wollinger, and Christof Paar. A new class of collision attacks and its application to DES. In *Fast Software Encryption - FSE 2003*, volume 2887 of *Lecture Note in Computer Science*, pages 192–205. Springer Verlag, 2003.

[112] Gurindar S. Sohi and Manoj Franklin. High-bandwidth data memory systems for superscalar processors. In *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '91*, pages 53–62. ACM Press, 1991.

[113] F.-X. Standaert, G. Rouvroy, G. Piret, J.-J. Quisquater, and J.-D. Legat. Key-dependent approximations in cryptanalysis: an application of multiple z_4 and non-linear approximations. In *Proceedings of 24th Symposium on Information Theory in the Benelux*, 2003.

[114] A. Steel. Allan Steel's Gröbner basis timings page, 2004. `http://magma.maths.usyd.edu.au/users/allan/gb/`.

[115] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of DES implemented on computers with cache. In *Workshop on Cryptographic Hardware and Embedded Systems - CHES '03*, volume 2779 of *Lecture Note in Computer Science*, pages 62–76. Springer Verlag, 2003.

[116] Yukiyasu Tsunoo, Etsuko Tsujihara, Kazuhiko Minematsu, and Hiroshi Miyauchi. Cryptanalysis of block ciphers implemented on computers with cache. In *International Symposium on Information Theory and Its Applications - ISITA '02*, pages 803–806. IEEE Information Theory Society, 2002.

[117] Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *22nd Annual International Symposium on Computer Architecture - ISCA '05*, pages 392–403. ACM Press, 1995.

[118] Håvard Raddum. More Dual Rijndaels. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Fourth Conference on the Advanced Encryption Standard - AES4*, volume 3373 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[119] S. Vaudenay. Decorrelation: A theory for block cipher security. *Journal of Cryptology*, 16:249–286, 2003.

[120] Serge Vaudenay. An experiment on DES - statistical cryptanalysis. In *Proceedings of Conference on Computer and Communications Security – CCS'95*, pages 139–147. ACM Press, 1995.

[121] David Wagner. The boomerang attack. In Lars Ramkilde Knudsen, editor, *Proceedings of Fast Software Encryption – FSE'99*, number 1636 in Lecture Notes in Computer Science, pages 156–170. Springer-Verlag, 1999.

[122] David Wagner. Towards a unifying view of block cipher cryptanalysis. In B. Roy and W. Meier, editors, *Proceedings of Fast Software Encryption – FSE'04*, number 3017 in Lecture Notes in Computer Science, pages 16–33. Springer-Verlag, 2004.

[123] Ralf-Philipp Weinmann. Evaluating Algebraic Attacks on AES. Diplomarbeit, Darmstadt university, Germany, 2004.

[124] Ralph Wernsdorf. The Round Functions of Rijndael Generate the Alternating Group. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 143–148. Springer, 2002.

[125] M. Wiener. The full cost of cryptanalytic attacks. In *Journal of Cryptology*, volume 17, pages 105–124, 2004.

[126] Kenneth M. Wilson and Kunle Olukotun. High bandwidth on-chip cache design. *IEEE Transaction Computers*, 50(4):292–307, 2001.

[127] Advanced encryption algorithm (AES) development effort, 1997–2000. `http://csrc.nist.gov/CryptoToolkit/aes/`.