Turkish Journal of Electrical Engineering and Computer Sciences

Volume 21 | Number 1 Article 13

1-1-2013

A time--memory trade-off approach for the solution of nonlinear equation systems

HÜSEYİN DEMİRCİ

MAHMUT ŞAMİL SAĞIROĞLU

MUHAMMED OĞUZHAN KÜLEKCİ

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 186 – 197

doi:10.3906/elk-1103-42

A time–memory trade-off approach for the solution of nonlinear equation systems

Hüseyin DEMİRCİ*, Mahmut Şamil SAĞIROĞLU, Muhammed Oğuzhan KÜLEKCİ
TÜBİTAK - UEKAE, POB 74 41470, Gebze, Kocaeli, Turkey

Received: 29.03.2011 • **Accepted:** 27.07.2011 • **Published Online:** 27.12.2012 • **Printed:** 28.01.2013

Abstract: We propose a memory-based method for the solution of a specific type of nonlinear equation systems. We observe that when the equations in a system can be separated into 2 parts, where each subset contains fewer parameters than the whole set of equations, the system can be solved faster with a preprocessing phase. We show that reduced rounds of AES produce such a system under a chosen plaintext scenario. This observation enables us to solve that system within a practically applicable complexity of 2^{37} operations where a brute force approach requires 2^{72} trials. The method can be used for the solution of other equation systems of the same structure. In the optimal case where we can divide the equations into 2, a problem that contains n binary variables can be solved at time $O(\frac{n}{2} \cdot 2^{n/2})$ operations and using $O(2^{n/2})$ units of memory rather than $O(2^n)$ trials of the equation system.

Key words: Time–memory trade-off, solution of nonlinear equations, AES, cryptanalysis

1. Introduction

The solution of nonlinear equation systems is a fundamental question in mathematics, not only in cryptography. There exists yet no general solution algorithm for nonlinear systems as opposed to the situation for linear systems, where one can perform, e.g., Gaussian elimination to reduce the number of unknown variables step by step.

Methods such as Groebner basis finding algorithms can be used as a tool in some cases. We have also methods concentrating on converting a nonlinear system into a linear system. Such linearization/relinearization, XL, and XSL [1, 2, 3] techniques have been proposed for the solution of nonlinear equation systems arising from cryptographic systems. These techniques express the input nonlinear system of n parameters with a linear system having far more parameters, and then deploy the rich set of linear case techniques on the converted system. As it is easier to deal with linear equations, they pay the penalty of enlarging the parameter space to make use of those linear case methods.

The use of time–memory trade-off is well known by the cryptology community for the key search problem both in block ciphers and in stream ciphers. It has also improved variants such as rainbow tables and distinguished points approach [4, 5, 6, 7, 8, 9, 10]. In this work, we suggest the use of a special type of time–memory trade-off for a specific type of equations. Similar approaches have been used in the cryptanalysis of block ciphers DES [11] and FEAL [12, 13], and hash functions [14].

This technique requires the input system to hold the *separable* (definition is given in section 2) property. Since we have fewer parameters to consider in the converted subsystems, we gain the ability to perform a time–memory trade-off, where it is more difficult to apply on the original parameter set. In the optimal case, where

*Correspondence: huseyin.demirci@tubitak.gov.tr

each subsystem would have $n/2$ parameters, we are able to solve the input system in $O(\frac{n}{2} \cdot 2^{n/2})$ operations with $O(2^{n/2})$ units of memory rather than $O(2^n)$ trials of the original equation system with O(1) memory. Thus, the technique can be considered a divide-and-conquer approach combined with time–memory trade-off. The difference between the proposed method and classical time–memory trade-off is the deterministic structure. The separability property enables us to find a solution without using the birthday approach. We also show that the condition of separability can be relaxed for some cases.

We apply the proposed technique to solve the equation system of AES observed by Gilbert and Minier. This enables us to distinguish 3 rounds of AES from a random permutation with only $10-11$ chosen plaintexts. This constitutes a marginal advantage over the classical square attack in terms of data complexity. Previously, Gilbert and Minier equations over 256 chosen plaintexts have been used as a distinguisher for AES [15, 16] (see section 3.2 for details). Having the ability to solve these equations in much less time by our new technique, previous Gilbert-Minier related time–memory trade-off attacks are improved.

This paper proceeds as follows. In Section 2, we discuss the mechanism behind the proposed solution system. Section 3 is dedicated to AES. In Section 3.1 we recall the AES encryption system and previous work. Section 3.2 introduces the nonlinear equation system of AES that we deal with. We apply the solution technique to this equation system in Section 3.3. Section 4 discusses the possible extensions of the method. We discuss future applications and conclude the paper in Section 5.

2. The general idea

The naive solution for a given nonlinear equation system is to verify all possible settings of its variable set. In general, solutions proposed to date concentrate on finding heuristics to speed up this checking mechanism. We propose a new technique that makes use of memory combined with a divide-and-conquer approach. Our method requires the given equation system to be separable according to Definition 1.

Definition 1 *Let $\breve{S}$ denote a system of equations over a set $V = \{x_1, x_2, \ldots, x_n\}$ of n discrete variables as*

$$\begin{aligned} E_1(x_1, x_2, \ldots, x_n) &= h_1 \\ E_2(x_1, x_2, \ldots, x_n) &= h_2 \\ &\vdots \\ E_z(x_1, x_2, \ldots, x_n) &= h_z \end{aligned}$$

We call $\breve{S}$ separable, if there exist 2 disjoint nonempty subsets $A = \{a_1, a_2, \ldots, a_\ell\}$ and $B = \{b_1, b_2, \ldots, b_{n-\ell}\}$ of V such that each equation $E_i, 1 \leq i \leq z$, can be expressed as a sum of 2 other equations over the variable sets A and B respectively, i.e.

$$\begin{aligned} E_1(x_1, x_2, \ldots, x_n) = F_1(a_1, a_2, \ldots, a_\ell) + G_1(b_1, b_2, \ldots, b_{n-\ell}) &= h_1 \\ &\vdots \\ E_z(x_1, x_2, \ldots, x_n) = F_z(a_1, a_2, \ldots, a_\ell) + G_z(b_1, b_2, \ldots, b_{n-\ell}) &= h_z \end{aligned}$$

Without loss of generality, assume that a separable equation system is defined over $GF(2)$. We compute the values of F_i and G_i, $1 \leq i \leq z$, for all possible settings of their corresponding variable sets, and store

these values in matrices denoted by F and G. Each column denotes a unique setting over the corresponding variable set. The matrix F has 2^ℓ columns since there exist 2^ℓ possible distinct settings for its ℓ variables $(a_1, a_2, \ldots, a_\ell)$ in $GF(2)$, and G has $2^{n-\ell}$ columns accordingly. The entry $F_{i,j}$ corresponds to the value of the $F_i(a_1, a_2, \ldots, a_\ell)$ computed for the j^{th} distinct setting of $(a_1, a_2, \ldots, a_\ell)$. Define the matrix $G^{'}$ as $G^{'}_{i,j} = h_i - G_{i,j}$ for all $1 \leq i \leq z$ and $1 \leq j \leq 2^{n-\ell}$. Figure 1 demonstrates the matrices F and G'.[1]

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \ldots & F_{1,2^\ell} \\ F_{2,1} & F_{2,2} & \ldots & F_{2,2^\ell} \\ \ldots & \ldots & \ldots & \ldots \\ F_{z,1} & F_{z,2} & \ldots & F_{z,2^\ell} \end{bmatrix} \qquad G^{'} = \begin{bmatrix} (h_1 - G_{1,1}) & (h_1 - G_{1,2}) & \ldots & (h_1 - G_{1,2^{n-\ell}}) \\ (h_2 - G_{2,1}) & (h_2 - G_{2,2}) & \ldots & (h_2 - G_{2,2^{n-\ell}}) \\ \ldots & \ldots & \ldots & \ldots \\ (h_z - G_{z,1}) & (h_z - G_{z,2}) & \ldots & (h_z - G_{z,2^{n-\ell}}) \end{bmatrix}$$

Figure 1. The sketch of F and $G^{'}$ matrices.

The equation system $\breve{S}$ has a solution if and only if the matrices F and $G^{'}$ have an equal column. Thus, we need to search for the equal columns among F and $G^{'}$ matrices. To achieve this goal, we first make a column-wise sort of F and $G^{'}$ matrices. This will take $O(2^\ell \cdot \log 2^\ell + 2^{n-\ell} \cdot \log 2^{n-\ell}) = O(\ell \cdot 2^\ell + (n-\ell) \cdot 2^{n-\ell})$ time complexity with quick sort. Note that, for a fixed equation system, the matrix F is independent of the solution. Therefore, F can be calculated and sorted in a preprocessing step. As a result, the run time complexity of the algorithm is $O((n-\ell) \cdot 2^{n-\ell})$.

The last step is to search between these sorted lists to find the equal items. If we consider the naive process to achieve this goal, we will traverse all the items over the matrices and search for an equal column. The complexity of this process is $O(2^{(n-\ell)} + 2^\ell)$. An alternative approach may be to sort only the smaller matrix. In this case, instead of sorting the second matrix with complexity $O(2^{(n-\ell)} + log 2^{n-\ell})$, a binary search of values of this matrix in the first (sorted) matrix requires time complexity $O(2^{(n-\ell)} + log 2^\ell)$.

The run time complexity is independent of the calculations for the matrix F. If the complexity is $O(z)$ for each column of G', the total computation time complexity for the matrix G' is $O(z2^{n-\ell})$. The total run time complexity is $O(z \cdot 2^{n-\ell} + (n-\ell) \cdot 2^{n-\ell} + 2^{(n-\ell)} + 2^\ell) = O((n - \ell + z + 1) \cdot 2^{n-\ell} + 2^\ell)$. If we exclude the number of equations z (as it is a constant factor and not a major term in complexity) and constants, the proposed technique's time complexity is $O((n-\ell) \cdot 2^{n-\ell})$. The dominant time consuming process is the sorting operation. The space requirement is mainly storing the F matrix, which is $O(z2^\ell)$. In the case where $\ell = \frac{n}{2}$, time complexity becomes $O(\frac{n}{2} \cdot 2^{n/2})$, and space complexity is approximately $O(2^{n/2})$.

Note that there might be more than one possible candidate at the end of the last step. The method is capable of finding each solution. A verification step may be required to eliminate wrong candidates.

We would like to explain the new idea with the following example.

[1]Note that subtracting H from F is also possible, and in practice the matrix having fewer columns will be selected for subtraction.

Table 1. The F and $G^{'}$ matrices computed for the sample equation system.

x_3	0	0	0	0	**1**	1	1	1
x_2	0	0	1	1	**0**	0	1	1
x_1	0	1	0	1	**0**	1	0	1
$F =$	0	1	0	0	**1**	0	1	1
	0	1	0	0	**0**	1	0	1
	0	0	1	1	**1**	0	0	1
	0	0	0	0	**0**	0	0	1

x_6	**0**	0	0	0	1	1	1	1
x_5	**0**	0	1	1	0	0	1	1
x_4	**0**	1	0	1	0	1	0	1
$G^{'} =$	**1**	0	1	0	1	0	1	1
	0	1	0	1	0	1	1	0
	1	1	0	0	1	0	0	1
	0	0	0	1	1	1	1	0

Example 1 *Let $\breve{S}$ be given as:*

$$\begin{aligned} x_1 + x_1x_2 + x_3 + x_4 + x_4x_5x_6 &= 1 \\ x_1 + x_2 + x_2x_3 + x_4 + x_5x_6 &= 0 \\ x_1x_3 + x_2 + x_3 + x_4x_6 + x_5 &= 1 \\ x_1x_2x_3 + x_4x_5 + x_6 &= 0 \end{aligned}$$

This system is separable according to Definition 1, since each equation can be expressed as the sum of 2 distinct variable sets $A = \{x_1, x_2, x_3\}$ and $B = \{x_4, x_5, x_6\}$. The corresponding F_i and G_i, $1 \leq i \leq 3$, functions are

$$\begin{aligned} F_1(x_1, x_2, x_3) &= x_1 + x_1x_2 + x_3, \\ F_2(x_1, x_2, x_3) &= x_1 + x_2 + x_2x_3, \\ F_3(x_1, x_2, x_3) &= x_1x_3 + x_2 + x_3, \\ F_4(x_1, x_2, x_3) &= x_1x_2x_3, \\ G_1(x_4, x_5, x_6) &= x_4 + x_4x_5x_6, \\ G_2(x_4, x_5, x_6) &= x_4 + x_5x_6, \\ G_3(x_4, x_5, x_6) &= x_4x_6 + x_5, \\ G_4(x_4, x_5, x_6) &= x_4x_5 + x_6 \end{aligned}$$

Table 1 shows the F and $G^{'}$ matrices computed with $h_1 = 1$, $h_2 = 0, h_3 = 1$, and $h_4 = 0$ values as given in $\breve{S}$. We observe that the equal columns from F and $G^{'}$ indicate a solution, e.g., the fifth column of F and the first column of $G^{'}$ are equal to $< 1, 0, 1, 0 >$ with variable settings $< x_1, x_2, x_3 >=< 0, 0, 1 >$, $< x_4, x_5, x_6 >=< 0, 0, 0 >$, which is a valid solution of the system.

3. The solution of AES equations

3.1. The AES encryption algorithm

In this paper we will focus on the Rijndael with block and key size 128. The 128-bit plaintext can be considered as a 4×4 matrix in $GF(2^8)$, and the entries of the matrix are represented as 1-byte values. There is a key whitening layer at the very beginning. Then each round function, except the final one, consists of 4 inner functions: the S-box substitution (SB), shift row (SR), mix column (MC), and add round key (ARK)

operations, respectively. The single S-box substitution is used for all entries in the table, and it is based on the inverse mapping in $GF(2^8)$, and affine mapping, which is strong against differential and linear attacks [17]. In the SR operation, rows are shifted to the left by 0, 1, 2, and 3 bytes, from the first row to the last. MC operation provides efficient confusion on columns of the matrix, since it is an MDS matrix multiplication. In the ARK, the state is simply XORed with the 128-bit round key. This design of AES round function guarantees full diffusion after 2 round function calls. For details of the encryption, decryption, and key scheduling algorithms, we refer to [18].

Being the new encryption standard, AES is probably one of the most attractive targets for cryptanalysis. There have been many attempts to analyze AES. First, the designers of AES break the 6 round version of AES-128 by the square attack using 2^{32} chosen plaintexts with complexity of about 2^{72} encryptions [19]. This attack has been improved and the workload has been reduced to 2^{44} in [20]. For AES-192 and AES-256, with the help of the key schedule, the attack of [21] can be successful for 7 rounds. In [15], Gilbert and Minier found an attack based on a collision property after 3 rounds of encryption. Seven rounds of AES-192 and AES-256 are broken using 2^{32} chosen plaintexts with a complexity of 2^{140} encryptions. Moreover, the attack is faster than an exhaustive search for AES-128. In [22, 23, 24, 25, 26], the impossible differential attack is applied to 7 rounds of AES, but it has higher complexity than the square attack. There are some new impossible differential attacks [27, 28, 29, 30] on AES that reduce the time complexities of the previous attacks. A boomerang attack is applied by Biryukov [31] for the 5 and 6 rounds of the cipher. It breaks 5 rounds of AES-128 using 2^{46} adaptive chosen plaintexts in 2^{46} steps of analysis, whereas the 6-round attack needs 2^{78} chosen plaintexts, 2^{78} steps of analysis, and 2^{36} bytes of memory. A class of algebraic attacks on AES is examined in [3]. In this paper, the AES S-box is written as a system of implicit quadratic equations, resulting in the conversion of the cryptanalysis to solving a huge system of quadratic equations. The XSL method is suggested if the system of equations is overdefined and sparse, which is the case for AES.

There have been recent developments in the cryptanalysis of AES. In [32] there is a related key attack on 10 rounds of AES-192 and AES-256 with practical complexity. Biryukov et al. [33, 34] have constructed related key attacks on the full AES-192 and AES-256. Moreover, chosen plaintext attacks work up to 7 rounds of AES-128 and 8 rounds of AES-192 and AES-256 [19, 15, 16, 28, 27, 29, 30]. Timing cache attacks are also an important threat against AES [35].

Throughout this section, we use $K^{(r)}$ and $C^{(r)}$ to denote the round key and the ciphertext of the r-th round; $K_{ij}^{(r)}$ and $C_{ij}^{(r)}$ denote the byte values at row i, column j. The arithmetic operations among table entries are in $GF(2^8)$, where addition is the same as bit-wise XOR. By a one round AES encryption, we mean an inner round without whitening or exclusion of the mix column operation unless otherwise stated.

3.2. The Gilbert-Minier equation

In [15], Gilbert and Minier showed an interesting distinguishing property for 3 rounds of AES: Consider the evolution of the plaintext over 3 full rounds including the whitening layer. Let a_{ij} denote the i-th row, j-th column of the plaintext. Assume that all the entries are fixed except the first one a_{11}. After the first S-box transformation, define $t_{ij} = S(a_{ij} + K_{ij}^{(0)})$, where $K_{ij}^{(0)}$ is the i-th row, j-th column of the whitening key. The add round key and S-box operations map fixed entries to fixed entries. After the mixed column operation, the elements of the first column are variable, where the other entries are still fixed to other variables depending

Table 2. The state matrix of AES after one round of encryption.

$2t_{11}+c_1$	m_{12}	m_{13}	m_{14}
$t_{11}+c_2$	m_{22}	m_{23}	m_{24}
$t_{11}+c_3$	m_{32}	m_{33}	m_{34}
$3t_{11}+c_4$	m_{42}	m_{43}	m_{44}

on previous values. Therefore, at the end of round 1, the state matrix is as given in Table 2 where m_{ij} and c_i, $1 \leq i \leq 4$, $2 \leq j \leq 4$, are fixed values that depend on the fixed entries and subkey values. Applying the round operations S-box, shift row, mix column, and key addition, at the end of the second round, the first main diagonal entry $C_{11}^{(2)}$ can be determined by the following equation:

$$\begin{aligned} C_{11}^{(2)} &= 2S(2t_{11}+c_1)+3S(m_{22})+S(m_{33})+S(m_{44})+K_{11}^{(2)} \\ &= 2S(2t_{11}+c_1)+c_5, \end{aligned}$$

for some fixed value $c_5 = 3S(m_{22})+S(m_{33})+S(m_{44})+K_{11}^{(2)}$. Applying the round function, similar equations can be written for the other main diagonal entries. At the end of the second round, the main diagonal entries are of the form:

$$\begin{aligned} C_{11}^{(2)} &= 2S(2t_{11}+c_1)+c_5 \\ C_{22}^{(2)} &= S(3t_{11}+c_4)+c_6 \\ C_{33}^{(2)} &= 2S(t_{11}+c_3)+c_7 \\ C_{44}^{(2)} &= S(t_{11}+c_2)+c_8 \end{aligned}$$

for some fixed values c_5, c_6, c_7, c_8. These values are determined by the fixed entries and the related subkey values. Since

$$C_{11}^{(3)} = 2S(C_{11}^{(2)})+3S(C_{22}^{(2)})+S(C_{33}^{(2)})+S(C_{44}^{(2)})+K_{11}^{(3)},$$

we can summarize the above observations with the following proposition:

Proposition 1 ([15]) *Consider a set of 256 plaintexts where the entry a_{11} is variable and all the other entries are fixed. Encrypt this set with 3 rounds of AES. Then the function that maps a_{11} to $C_{11}^{(3)}$ is entirely determined by 10 fixed 1-byte parameters.*

Consider the following equation formed by 3 rounds of AES encryption:

$$\begin{aligned} C_{11}^{(3)} &= 2S(2S(2t_{11}+c_1)+c_5)+3S(S(3t_{11}+c_4)+c_6) \\ &\quad +S(2S(t_{11}+c_3)+c_7)+S(S(t_{11}+c_2)+c_8)+K_{11}^{(3)}. \end{aligned} \tag{1}$$

For the value of t_{11} we also need the value of $K_{11}^{(0)}$. Therefore, the values of the variables $c_1, ..., c_8, K_{11}^{(0)}$ and $K_{11}^{(3)}$ define the function from a_{11} to $C_{11}^{(3)}$.

We call the equation (1) the Gilbert-Minier equation.

At this point, we assume that the value of t_{11} is known. Note that this assumption is equivalent to 2^8 trials of the key value $K_{11}^{(0)}$. We would like to answer the following question:

Question 1 *Assume that we are given the values of the equation (1) for $m \leq 256$ different values of t_{11}, is it possible to find the values of $c1, c2, c3, c4, c5, c6, c7, c8$ and $K_{11}^{(3)}$ from this information?*

Whether this system has a solution can be used as a distinguisher. We know that 3 rounds of AES has such a property, but for instance if we make partial decryption with wrong subkey values, the property will not hold. It is possible to search the possible byte combinations exhaustively and find the value of c_i and $K_{11}^{(3)}$ values or decide there is no solution. On the other hand, it is also possible to precompute a table containing all the possible values of the equation (1) for all possible combinations and ask from the table when a query for the solution is required. Alternatively, one can use a classical time–memory trade-off between these 2 edges. We propose a different time–memory trade-off with the help of the special structure of this equation. ¿From the classical time–memory approach we have some differences. The method works for any query, not for a subset of the values. Therefore there is no coverage concept. As a result, there is no probabilistic approach; we have an exact solution. This approach is similar to the analysis conducted by Matsui et al. for the FEAL cipher [13].

The basic square attack on AES uses the same chosen plaintext scenario. It exploits the fact that the sum of any entry is 0 after 3 rounds of encryption. Therefore the square attack does not need to find internal parameters of the system, but requires 256 chosen plaintexts for a 3-round distinguisher. In our model, we are able to distinguish 3 rounds of AES with about 10 chosen plaintexts.

3.3. Solution of the Gilbert-Minier equation system

In this section, we give the algorithm to find the solution of the equation system in Question 1.

Consider the equation (1):

$$C_{11}^{(3)} = 2S(2S(2t_{11} + c_1) + c_5) + 3S(S(3t_{11} + c_4) + c_6)$$

$$+S(2S(t_{11} + c_3) + c_7) + S(S(t_{11} + c_2) + c_8) + K_{11}^{(3)}.$$

Let

$$F = 2S(2S(2t_{11} + c_1) + c_5) + 3S(S(3t_{11} + c_4) + c_6)$$

and

$$G = S(2S(t_{11} + c_3) + c_7) + S(S(t_{11} + c_2) + c_8).$$

Then the Gilbert-Minier equation can be written as:

$$F(c_1, c_4, c_5, c_6)(t_{11}) + G(c_2, c_3, c_7, c_8)(t_{11}) + K_{11}^{(3)}.$$

Assume that we are given $10 - 11$ ciphertexts after 3 rounds of AES encryption. Let the vector y_i be the value of $C_{11}^{(3)}$ at the point $t_{11} = 0, ..., 8$. Form the vector $y = (y_0, y_2, ..., y_8)$. We can eliminate the linear constant $K_{11}^{(3)}$ and solve the system as follows:

1. Prepare a table of size 8×2^{32} that contains values of the function

$$F(c_1, c_4, c_5, c_6)(t_{11}) + F(c_1, c_4, c_5, c_6)(0)$$

for every c_1, c_4, c_5, c_6 and for $t_{11} = 1, ..., 8$. Call this table A.

2. Similarly, prepare another table of size 8×2^{32} for every value of the function

$$G(c_2, c_3, c_7, c_8)(t_{11}) + G(c_2, c_3, c_7, c_8)(0)$$

for the possible values of c_2, c_3, c_7, c_8 at $t_{11} = 1, ..., 8$. Call this set B.

3. Sort the matrix A according to its columns. Call the sorted matrix A'.

4. Form a table of size 8×2^{32} as follows:

$$C = \begin{bmatrix} y_1 + y_0 & y_1 + y_0 & \cdots & y_1 + y_0 \\ y_2 + y_0 & y_2 + y_0 & \cdots & y_2 + y_0 \\ \cdots & \cdots & \cdots & \cdots \\ y_8 + y_0 & y_8 + y_0 & \cdots & y_8 + y_0 \end{bmatrix}$$

5. Obtain the matrix $D = (C - B)$. Sort this matrix with respect to its columns. Call the sorted matrix D'.

6. Compare the matrices A' and D'. Check if one column of $A1'$ is equal to one column of D'. Return the equal columns as a candidate solution for $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$. If no columns are equal, there is no solution.

7. Using other 1–2 plaintexts, check if the derived c_i values are correct.

8. Put the c_i values in one of the equations and obtain the value of $K_{11}^{(3)}$.

Note that steps 1–3 can be done in a precomputation step since the AES equation system is fixed. Then, in the run time, we need to consider only the sort of the matrix D'.

To actually observe the behavior of the algorithm, we have implemented it on a server having 2.4 GHz Intel Xeon 64-bit CPU running Linux Gentoo. To shrink the memory to a practical size, we have used a banking approach. Our implementation requires 256×2 files, each of which occupies 192 MB storage area. 256×192 MB are required to store the sorted matrix A' and 256×192 MB are required to store matrix B. We have run the algorithm for each bank. The algorithm uses 400 MB of RAM, and spends 25 s on average where the user time is measured with the *getrusage()* command to find the c_i values or decide there is no solution. Therefore, in 25×256 s total time, the algorithm finds all candidate solutions. Since the banking approach is advantageous, we have parallelized our algorithm using all 16 CPUs available. Then the response time decreased to 25×16 s. Running 256 parallel algorithms for each bank is even possible. In this situation, the response time will be only 25 s. The best banking value depends on the practical implementation limits. Finally we would like to note that the information of t_{11} can either be satisfied with a search on the online complexity. This requires the search for $K_{11}^{(0)}$, which brings a factor of 256 in time complexity. Another solution may be to include the search for $K_{11}^{(0)}$ in the precomputation step and prepare the tables A and B accordingly. This will increase the size of each of the tables A and B 256 times, whereas the online complexity does not change.

This distinguisher can be used to attack on reduced rounds of AES. For 5 and 6 rounds of AES, the basic square attack makes the key elimination after partial decryption with the 2^{40} and 2^{72} candidate keys of 256 ciphertexts. In our method, we are able to eliminate the keys after conducting partial decryption with the same number of keys, but we need to perform a partial decryption for only 10–12 ciphertexts. We have also a sorting complexity of these ciphertexts. Therefore our method has an advantage over the basic square attack with a ratio of about $\frac{2^8}{(4\times 2^4)} = 4$ (if $K_{11}^{(0)}$ is calculated and stored in the preprocessing step). On the other hand, the improved square attack [20] performs better than this attack. The main advantage of our method is the possibility to distinguish AES with very few plaintext-ciphertext pairs (10–11 pairs, instead of 256).

4. Possible extensions of the system

There are some possible improvements that loosen the conditions to apply the algorithm. The basic assumption of the proposed method is that each equation of the system should be separable. Again, without loss from generality, assume that we work on $GF(2)$ and the equation system H contains n variables. Let each function H_i in the system be written as a sum of 2 nonlinear functions F_i and G_i, each containing n_1 and n_2 parameters, and the sets F and G contain r common variables. If we have $2^r + 2^{n_1} << 2^n$ and $2^r + 2^{n_2} << 2^n$, we may follow the same steps as in Section 3.3 to produce 2 tables containing all the possible evaluations of the functions F_i and G_i and decide whether there is a candidate solution. The final additional step here is to check that the obtained solutions are consistent, regarding the common variables. In this case, we will also have an advantage over searching 2^n values.

A second improvement comes if the number of equations is large. In this case, it may be possible to choose 2 subsets, which makes some of the equations separable. Using the separated equations, we may reduce the complexity and then return to the original system and solve it. A naive approach is to take arbitrary sets and try to observe if the equations are separable. Assume that we have an arbitrary quadratic binary nonlinear equation system containing $2n$ variables, and assume that the equation is sparse. We arbitrarily divide the set of variables $X = \{x_1, ..., x_n, x_{n+1}, ..., x_{2n}\}$ variables into 2 separate parts containing the elements $X1 = \{x_1, ..., x_n\}$ and $X2 = \{x_{n+1}, ..., x_{2n}\}$. Then define 3 sets of monomials $A1, A2$, and $A3$ as

$$A1 = \{x_i x_j | x_i, x_j \in X1\},$$

$$A2 = \{x_i x_j | x_i, x_j \in X2\},$$

$$A3 = \{x_i x_j | x_i \in X1,\ x_j \in X2\}.$$

Note that the sets $A1$ and $A2$ have cardinality $n(n-1)/2$, and $A3$ has n^2 elements, and the sets are disjoint. Suppose that we have a sparse equation system where there are at most m monomials in each equation. The probability that one equation does not contain an element from the set $A3$ is: $(\frac{n-1}{2n-1})^m$. Hence, if the number of equations is k, we expect to have $k \times (\frac{n-1}{2n-1})^m$ equations, which are separable. On the other hand, the linearization technique requires $2n \times (2n-1)$ (the number of monomials) many linearly independent equations. Table 3 summarizes the number of equations required to get a solution for the linearization and the proposed method. Here we have considered 64-, 80-, and 128-bit variables, which are the common key lengths in cryptographic applications. For instance let $2n = 128$ and $m = 3$. Then we need approximately $128 \times (\frac{63}{127})^{-3} = 1048$ equations to obtain a solution, whereas the linearization method requires about $128 \times 127 = 16256$ linearly independent equations. This observation demonstrates that finding separable

Table 3. Number of required equations.

2n	m = 2	m = 3	m = 4	m = 5	Linearization
64	264	537	1091	2218	4032
80	328	664	1346	2728	6320
128	520	1048	2113	4261	16256

equations can be easier in some special cases (such as equation systems that are very sparse.) It also indicates that this method may require significantly fewer equations corresponding to linearization. We note that the complexity of the linearization technique may be less than that of the proposed algorithm, but if we have only a limited number of equations, the method can be helpful. Finally, we would like to note that finding the best decomposition in general is related to other computational problems.

5. Discussions and conclusion

In this paper, we propose a memory-based method for the solution of nonlinear equation systems of specific type, called separable. We provide an example in which 3 rounds of AES encryption forms such a system of equations. Hence, we are able to solve this nonlinear system practically using very few plaintexts.

If the equation system has a separable structure, the proposed solution algorithm can have a significance advantage over a direct exhaustive search or a precomputation method. As an example, the complexities of the proposed method are compared in Table 4 for some values. We have again considered equation systems with 60, 80, and 128 variables in the binary case and assumed that the nonlinear systems can be separated into 2 equal parts. It is interesting to observe that the precomputation has helped to build a trade-off between time and memory. Therefore, for some cases it becomes practical to divide the complexity into memory and time where a direct search is infeasible.

Table 4. Complexity comparison of the proposed method.

Number of variables	Method	Run time	Memory	Preprocessing time
64	Exhaustive search	2^{64}	-	-
	Precomputation	-	2^{64}	2^{64}
	Proposed method	2^{33}	2^{32}	2^{32}
80	Exhaustive search	2^{80}	-	-
	Precomputation	-	2^{80}	2^{80}
	Proposed method	2^{41}	2^{40}	2^{40}
128	Exhaustive search	2^{128}	-	-
	Precomputation	-	2^{128}	2^{128}
	Proposed method	2^{65}	2^{64}	2^{64}

There are some extensions of the proposed system. First, we show that the idea of the method can still be used for the solution of other equation systems that are not exactly separable. Therefore the separability condition can be relaxed. Moreover, we observe that the method provides advantage in terms of the number of required equations to get a solution with respect to linearization. This could be advantageous in sparse equation systems where we have only a limited number of equations.

This approach could be used in other areas where discrete nonlinear equations are involved. The minimum cut problem, the Boolean satisfiability problem, and nonlinear systems control could be listed as examples of these areas.

In the mincut problem from graph theory one tries to find the cut with the smallest edge weights. For instance, [36] is a paper that introduces mincut ideals of 2-terminal networks that arise in the algebraic analysis of system reliability. It is interesting to observe that in this field there are network structures that have a separable structure.

The Boolean satisfiability problem (SAT) involves determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE or to determine whether no such assignments exist. SAT is known to be an NP-complete problem. We think that the proposed solution method can be advantageous in some instances of the SAT problem.

Nonlinear systems control is an important area where discrete nonlinear variables are used to express the behavior of dynamic systems like robotics and the aerospace industry. If the underlying systems cause a separable equation system where each separated block has a searchable complexity, then the proposed method could be helpful for the solution of these systems.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions, which have improved the quality of the manuscript.

References

[1] A. Kipnis, A. Shamir, "Cryptanalysis of the HFE public key crypto system by relinearization", In CRYPTO 99, LNCS, Vol. 1666, pp. 19–30, Springer-Verlag, 1999.

[2] N. Courtois, A. Klimov, J. Patarin, A. Shamir, "Efficient algorithms for solving overdefined systems of multivariate polynomial equations", In EUROCRYPT 2000, LNCS, Vol. 1807, pp. 392–407, Springer-Verlag, 2000.

[3] N.T. Courtois, J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations", In ASIACRYPT 2002, LNCS, Vol. 2501, pp. 267–287, Springer-Verlag, 2002.

[4] M.H. Hellman, "A cryptanalytic time-memory trade-off", IEEE Transactions in Information Theory, Vol. IT-26, pp. 401–406, 1980.

[5] A. Biryukov, A. Shamir, "Cryptanalytic time/memory/data trade-offs for stream ciphers", In ASIACRYPT 2000, LNCS, Vol. 1976, pp. 1–13, Springer-Verlag, 2000.

[6] A. Biryukov, A. Shamir, D. Wagner, "Real time cryptanalysis of A5/1 on a PC", In FSE 2000, LNCS, Vol. 1978, pp. 37–44, Springer-Verlag, 2001.

[7] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off", In CRYPTO 2003, LNCS, Vol. 2729, pp. 617–630, Springer-Verlag, 2003.

[8] S.H. Babbage, "Improved exhaustive search attacks on stream ciphers", In European Convention on Security and Detection, IEE Conference publication, Vol. 408, pp. 161–166, IEE, 1995.

[9] J.Dj. Golić, "Cryptanalysis of alleged A5 stream cipher", In EUROCRYPT 97, LNCS, Vol. 1233, pp. 239–255, Springer-Verlag, 1997.

[10] D.E. Denning, In Cryptography and Data Security, page 100, Addison-Wesley, 1982.

[11] W. Diffie, M.E. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard", Computer, Vol. 10, 74–84, 1977.

[12] H. Morita, K. Ohta, K. Miyaguchi, "A switching closure test to analyze cryptosystems", In CRYPTO 1991, LNCS, Vol. 576, pp. 183–193, Springer-Verlag, 1992.

[13] M. Matsui, A. Yamagishi, "A new method for known plaintext attack of FEAL cipher", In EUROCRYPT 1992, LNCS, Vol. 658, pp. 81–91, Springer-Verlag, 1993.

[14] D. Khovratovich, I. Nikolić, R.P. Weinmann, "Meet-in-the-middle attacks on SHA-3 candidates", In FSE 2009, LNCS, Vol. 5665, pp. 228–245, Springer-Verlag, 2009.

[15] H. Gilbert, M. Minier, "A collision attack on 7 rounds of Rijndael", In The Third AES Candidate Conference, 2000.

[16] H. Demirci, A.A. Selçuk, "A meet in the middle attack on 8-round AES", In FSE 2008, LNCS, Vol. 5086, pp. 116–126, Springer-Verlag, 2008.

[17] K. Nyberg, L.R. Knudsen, "Provable security against a differential attack", Journal of Cryptology, Vol. 8(1), pp. 27–38, 1995.

[18] "Fips-197: Advanced Encryption Standard", November 2001, available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[19] J. Daemen, L. Knudsen, V. Rijmen, "The block cipher SQUARE". In FSE 1997, LNCS, Vol. 1267, pp. 149–165, Springer-Verlag, 1997.

[20] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, D. Whiting, "Improved cryptanalysis of Rijndael", In FSE 2000, LNCS, Vol. 1978, pp. 213–230, Springer-Verlag, 2001.

[21] S. Lucks, "Attacking seven rounds of Rijndael under 192-bit and 256-bit keys", In The Third AES Candidate Conference, 2000.

[22] E. Biham, N. Keller, "Cryptanalysis of reduced variants of Rijndael", In The Third AES Candidate Conference, 2000.

[23] J.H. Cheon, M.J. Kim, K. Kim, J. Lee, S. Kang, "Improved impossible differential cryptanalysis of Rijndael", In ICISC '2001, LNCS, Vol. 2288, pp. 39–49, Springer-Verlag, 2001.

[24] R.C.W. Phan, M.U. Siddiqi, "Generalized impossible differentials of Advanced Encryption Standard", IEE Electronics Letters, Vol. 37(14), pp. 896–898, 2001.

[25] R.C.W. Phan, "Classes of impossible differentials of Advanced Encryption Standard", IEE Electronics Letters, Vol. 38(11), pp. 508–510, 2002.

[26] R.C.W. Phan, "Impossible differential cryptanalysis of 7-round Advanced Encryption Standard AES", Information Processing Letters, Vol. 91, pp. 33–38, 2004.

[27] B. Bahrak, M.R. Aref, "Impossible differential attack on seven-round AES-128", In IET Information Security Journal, Vol. 2, pp. 28–32, 2008.

[28] B. Bahrak, M.R. Aref, "A novel impossible differential cryptanalysis of AES", In Proceedings of the Western European Workshop on Research in Cryptology, Bochum Germany, 2007.

[29] W. Zhang, W. Wun, D. Feng, "New results on impossible differential cryptanalysis of reduced AES", In Proceedings of ICISC 2007, LNCS, Vol. 4817, pp. 239–250, Springer-Verlag, 2007.

[30] J. Lu, O. Dunkelman, N. Keller, J. Kim, "New impossible differential attacks on AES", In INDOCRYPT '2008, LNCS, Vol. 5365, pp. 279–293, Springer-Verlag, 2008.

[31] A. Biryukov, "Boomerang attack on 5 and 6-round AES", In The Fourth Conference on Advanced Encryption Standard, 2004.

[32] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, A. Shamir, "Key recovery attacks of practical complexity on AES variants with up to 10 rounds", 2009. available at http://eprint.iacr.org/2009/374.pdf.

[33] A. Biryukov, D. Khovratovich, I. Nikolić, "Distinguisher and related-key attack on the full AES-256 (extended version)", In CRYPTO 2009, LNCS, Vol. 5677, pp. 231–249, Springer-Verlag, 2009.

[34] A. Biryukov, D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256", 2009. available at http://eprint.iacr.org/2009/317.pdf.

[35] D.J. Bernstein, "Cache-timing attacks on AES", 2005. URL: http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[36] A. Cabezon, H.P. Wynn, "Mincut ideals of two-terminal networks", Applicable Algebra in Engineering, Communication and Computing, Vol. 21, pp. 443–457, 2010.