Masaryk University
Faculty of Informatics

Mobile cryptography

Diploma thesis

Dušan Klinec

Brno, 2013

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Dušan Klinec

Advisor: RNDr. Petr Švenda, Ph.D.

Acknowledgement

[TODO] Thanks here

Abstract

[TODO] Abstract here

Keywords

white box attack resistant cryptography, look up tables form, AES

Table of contents

1 Introduction

[TODO] Introduction here

2 Area overview

2.1 Overview

[TODO] Overview, setting picture in cryptographic world

2.2 Mobile cryptography

[TODO] Motivation for white box cryptography

- computation with encrypted data
- computation with encrypted function

2.3 Homomorphic encryption

[TODO] Homomorphic encryption follows computation with encrypted data function. Motivation: cloud computation. Short history, recent state of the art...

1. security point of view - optimal
2. short description, computing with encrypted data - use slides from OwnTalk
3. practical usability
4. state of the art practical results

3 Whitebox cryptography

3.1 Introduction

In this part of a cryptography we are studying the cryptographic algorithms with a much stronger attacker model, saying it is executed in a whitebox context.

Whitebox context (also abbreviated as WBC) is itself defined by the attacker model, which was introduced by Chow *et al.* [1] in 2002. An WBC attacker has full control over execution of the particular algorithm. Namely attacker has the following abilities:

- can observe execution:
  - access to the instructions processing at the moment of the computation
  - trace the algorithm flow
  - sees the memory used
- controls the execution environment - runtime modification:
  - tamper the program memory
  - execute only a specified part of the algorithm (one round of the cipher)
  - modify if-conditions
  - change a cycle counters
  - fault induction

It is in contrast to *blackbox context* (also abreviated as BBC), the standard cryptographic model, where attacker has only access to the output of the cryptographic algorithm. In BBC the cryptographic algorithm is considered as an oracle/blackbox evaluating some function (analogy to executing algorithm in secure environment). Depending on a finer granularity of an attacker model, one can have access only to algorithm output (ciphertext), or attacker can also query an oracle (chosen plain-text attack) and so on, but has no access to computation itself.

The cryptographic algorithms (we are mainly interested in symmetric ciphers in this thesis) were extensively studied for attacks in BBC in past. They were originaly designed to resist attacks considering only BBC. But if the context is wrong, it can be possible entry point for an attacker. Typical example is DRM [1], where software of a vendor (representing the rights owner) is executed in potentially hostile environment, where user can have motivation to extract protected content without restrictions added by DRM software. In this situation we cannot consider DRM software to be executed in BBC.

---

1. Digital rights management, <`http://en.wikipedia.org/wiki/Digital_rights_management`

Let's take some symmetric block cipher as an another example. Usually it is constructed as a keyed permutation (round function) that is repeated several times to add randomness and to improve statistical results of the cipher, increasing security. But if we can inspect such execution, it is very easy to extract encryption keys, since we can read memory during execution or trace algorithm flow.

One such whitebox attack is *Key Whitening Attack* [2]. Key whitening is technique intended to increase the security of the iterated block cipher. It is typically implemented as adding a key material to the data (usually by simple operation, such as XOR) in the first and the last round. Such key whitening is used by Twofish [3] and in modified version (only adding the key material in the last round) also AES [4]. In Key Whitening Attack cipher's binary is modified (we are in whitebox context) in such a way that the output of the cipher will be the key material itself.

The main goals of a whitebox attack is to extract a key material (usually the symmetric key) or to build an inverse cipher i.e. perform decryption with tables for encryption with a embedded key.

The definition of whitebox cryptography could be: "The challenge that white-box cryptography aims to address is to implement a cryptographic algorithm in software in such a way that cryptographic assets remain secure even when subject to white-box attacks. Software implementations that resist such white-box attacks are denoted white-box implementations." [5].

3.2 History

The whitebox cryptography is quite new field of a cryptography. The study of the whitebox implementation of the ciphers started by first whitebox implementation of the AES [1] and DES [6] in 2002 by Chow *et al.*.

At first the cryptanalysis of DES focused on simplified variant of DES. The first published in 2002 by Jacob *et al.*uses fault injection [7], another one published in 2005 by Link*et al.*. uses statistical analysis [8]. Later cryptanalysis of fully encoded variant of DES was published by Wyseur *et al.* in 2007 using truncated differentials.

The similar case is AES. Two years after publishing whitebox AES scheme the successful cryptanalysis [9] was published by Billet *et al.* that enabled to recover embedded symmetric key in less that 2^{30} steps. Later in 2008 the generalized version of the previous attack was published [10] by Michiels *et al.* affecting the larger family of ciphers using the same structure as AES.

There were also attempts to fix whitebox AES scheme by adding additional linear mappings and increasing size of the implementation in [11] as a response to the Billet's attack. The attack using linear equivalence algorithm was published in 2012 [12].

The another attempt how to fix whitebox AES was introducing random pertrubations, complicating algebraic cryptanalysis but the effective attack was published by Mulder *et al.*[13].

Last not least a whitebox AES scheme using dual ciphers [14] was published in 2011. The paper claimed the scheme is robust enough to resist practical attacks on the implementation what we proved is false. Thesis is mainly focused on this scheme, what is generalization of the first whitebox AES scheme.

3.3 Whitebox AES scheme

The first whitebox AES implementation, published by Chow *et al.* [1] is based on the look-up tables implementation, that was also mentioned in original AES paper [4]. Note that it is easy to transform AES cipher with embedded encryption key to a network of look-up tables and to use these computed tables for encryption (or decryption). But this implementation is vulnerable in whitebox context, since it is possible to extract encryption key with algebraic analysis of the look-up tables, recall all building blocks (except key schedule) of AES are key-independent and publicly available. Thus these look-up tables has to be further protected to resist algebraic attacks.

We are mainly focused on AES-128, for simplicity but the same strategy can be applied also to AES-256.

For further explanation we will need the following definitions:

Definition 1. *Linear mapping is mapping* $L(x)$ *over* $GF(2)^n$ *that satisfies* $\forall\, x, y \in GF(2)^n$: $L(x+y) = L(x) + L(y)$.

Definition 2. *Affine mapping is mapping* $A(x)$ *over* $GF(2)^n$ *such that* $A(x) = L(x) + c, c \in GF(2)^n$ *and* $L(x)$ *is linear mapping.*

3.3.1 AES-128

The brief introduction of AES is required for further understanding of whitebox implementation and implementation based on dual AES in section 4.

AES-128 is a symmetric, iterated, block cipher that maps $128 \rightarrow 128$ bits (block length) using a 128-bit encryption key. It has 10 rounds and operates over 4×4 byte array. AES works with GF (2^8) and operations used in AES have quite algebraic nature. For each round is generated a key material called round keys by key-schedule routine from encryption key.

The key-schedule is not important operation from our perspective and can be abstracted in further explanations. Important fact about the key-schedule is it is reversible i.e. if we have round keys from 2 consecutive rounds it is possible to derive all other round keys, even the encryption key (the encryption key is used as round key in the first round).

Besides the key-schedule there are 4 operations used in main AES body:

- *AddRoundKey* adds round keys to the state array. It is simple XOR of two 4×4 byte arrays.

- *ShiftRows* performs a simple shift of each row of the state array to the left, by the row index (indexing from 0).

- *SubByte* is $8 \rightarrow 8$ bijection, the only non-linear operation, performing *confusion*[2]. It uses inversion in GF (2^8).

- *MixColum* is the main *diffusion*[3] operation. It corresponds to a matrix multiplication in GF (2^8). State array column is multiplied from left by MixColumn 4×4 matrix (also denoted as MC). Due to this operation after one round of the cipher, one byte of the state array depends on 4 bytes of the input state array.

3.3.2 AES table implementation

Algorithm 1 is AES in a form suitable for whitebox implementation, i.e. some operations were regrouped in such a way that round keys addition is right before *SubByte* operation. This enables to merge these two operations to one 8×8 look-up table as is described in equation 3.1. Resulting tables following this construction are called T-boxes.

$$T^r_{i,j}(x) = S\left(x \oplus k^r_{i,j}\right) \tag{3.1a}$$

$$T^9_{i,j}(x) = S\left(x \oplus k^9_{i,j}\right) \oplus k^{10}_{i,j} \tag{3.1b}$$

ShiftRows has no counter-part in table implementation, it is implemented as the way how tables between rounds are connected together (taking *ShiftRows* into account).

The *MixColumn* is problematic to transform into a look-up table since it is $32 \rightarrow 32$ mapping. A naive transformation would take $2^{32} \cdot 4 = 16\ GB$. Thus linearity of a matrix multiplication is used, as illustrates equation 3.3.

Let

$$MC = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \tag{3.2}$$

2. *Confusion* is a property of secure cipher defined by Shannon [15], it should make relation between ciphertext and symmetric encryption key complex.
3. *Diffusion* is a property of secure cipher defined by Shannon [15], it should make relation between ciphertext and plaintext complex, dissipating small change into large range. Diffusion contributes to the avalanche effect.

Algorithm 1 AES algorithm, form suitable for whitebox implementation

```
1: function AES(plaintext, k)                          ▷ k is array of round keys
2:     state ← plaintext                               ▷ state is 4 × 4 byte array
3:     for r ← 0, to 8 do
4:         ShiftRows(state)
5:         AddRoundKey(state, k_r)
6:         SubByte(state)
7:         MixColumn(state)
8:     end for
9:     ShiftRows(state)
10:    AddRoundKey(state, k_9)
11:    SubByte(state)
12:    AddRoundKey(state, k_10)
13:    return state
14: end function
```

Then from linearity holds:

$$\begin{aligned} MC \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} &= MC \cdot \begin{bmatrix} x_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus MC \cdot \begin{bmatrix} 0 \\ x_1 \\ 0 \\ 0 \end{bmatrix} \oplus MC \cdot \begin{bmatrix} 0 \\ 0 \\ x_2 \\ 0 \end{bmatrix} \oplus MC \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ x_3 \end{bmatrix} \\ &= x_0 \cdot \begin{bmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{bmatrix} \oplus x_1 \cdot \begin{bmatrix} a_{0,1} \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \end{bmatrix} \oplus x_2 \cdot \begin{bmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{bmatrix} \oplus x_3 \cdot \begin{bmatrix} a_{0,3} \\ a_{1,3} \\ a_{2,3} \\ a_{3,3} \end{bmatrix} \\ &= x_0 \cdot MC_0 \oplus x_1 \cdot MC_1 \oplus x_2 \cdot MC_2 \oplus x_3 \cdot MC_3 \end{aligned} \tag{3.3}$$

Thus the $32 \rightarrow 32$ linear mapping represented by a matrix multiplication is decomposed to 4 $8 \rightarrow 32$ look-up tables connected by 3 32-bit XOR tables. The tables performing this operation are denoted as T_y tables in a further text.

Figure 3.1 illustrates evaluation of the AES round function using look-up tables for one column of state array.

3.3.3 Whitebox AES

As mentioned before, special techniques are needed to protect look-up tables from algebraic attacks. The most important ones are described in following paragraphs.

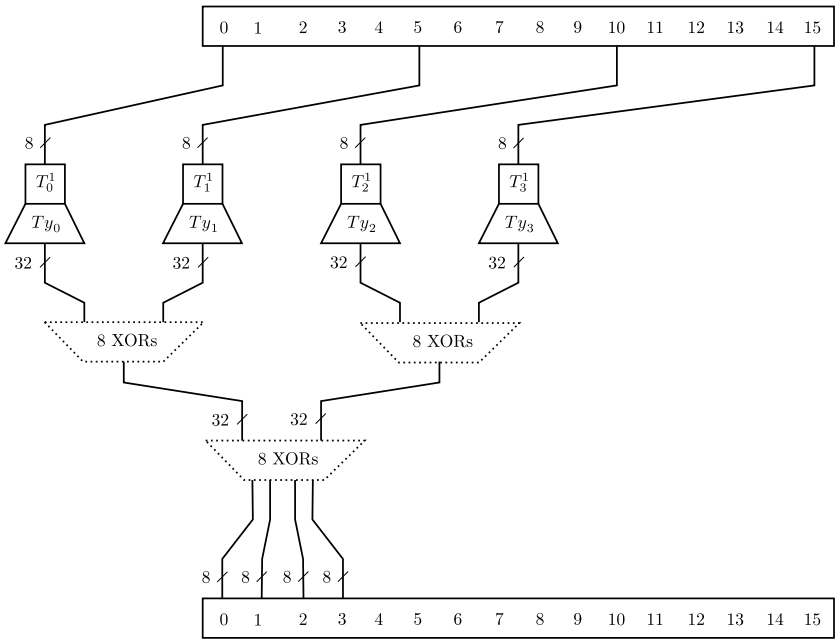

Figure 3.1: Table AES implementation - rounds 2–8, taken from [16]

Input/output bijections. One of the techniques used in whitebox implementation is use of *input/output bijections* (also abbreviated as IO bijections). According to Chow *et al.* [1] IO bijection is a random $n \to n$ bijection[4]. Consider $n \to n$ IO bijections $F_1, \ldots, F_k$, they can be *concatenated* to form a $kn \to kn$ bijection $F_1 || \ldots || F_k$. This concatenation enables to build a large bijections with small look-up tables, in a further explanations the size of basic IO bijection is $4 \to 4$.

Consider the table implementation of AES as mentioned in the section 3.3.2. To protect tables, each table is wrapped by the concatenated IO bijections in such a way the composition of two connected tables cancels the effect of IO bijections as illustrates equation 3.4.

$$T' = g \circ T \circ f^{-1} \tag{3.4a}$$

$$R' = h \circ R \circ g^{-1} \tag{3.4b}$$

$$R' \circ T' = \left(h \circ R \circ g^{-1}\right) \circ \left(g \circ T \circ f^{-1}\right) = h \circ (R \circ T) \circ f^{-1} \tag{3.4c}$$

Where $T,\ R$ are look-up tables and $g,\ h$ IO bijections realizing confusion step and make analysis of a single table harder.

4. usually $n = 4$

Mixing bijections. Another whitebox building block is a *mixing bijection*. It is a linear transformation (represented as a multiplication by a non-singluar mixing bijection matrix) that realizes the diffusion. It is used together with the the IO bijections to increase a security level of the concatenated bijections, since it diffuses a single change in the one sub-bijection to the whole range of the concatenated bijection. In order to fulfill this purpose properly, the mixing bijections has to be constructed in a special way. Zhou *et al.*describes in [17] the algorithm that generates large random non-singular matrices with blocks of a full-rank. The size of the sub-blocks is 4 what matches the size of basic IO bijection what gives good diffusion properties for concatenated bijections.

Note that full-rank property of matrix blocks provides good level of diffusion. It lies somewhere between two extreme cases, random non-singular matrices without any requirements on diffusion power and parity-check matrices of MDS[5] codes that has optimal diffusion power (for a linear transformation), but it is harder to generate them systematically. Note that MDS matrices are often uses as a diffusion element in ciphers.

External input/output encoding. Consider AES protected with aforementioned whitebox techniques. The possible place where to attack is input and output of the algorithm, since it is not protected from a whitebox attack. To mitigate this weakness Chow *et al.*also introduces *external* encoding that wraps the whole AES. Usually the cipher is not standalone element, but part of a system. This technique helps to tie AES to its context and to prevent from using the cipher separately as an oracle.

Whitebox implementation computes:

$$H \circ AES \circ G^{-1} \tag{3.5}$$

If AES is used as a part of a system, a previous element has to apply G transformation on data before passing them to the WB AES. Similarly, the next element has to apply H^{-1} to cancel effect of the previous transformation.

Usually G, H are defined as a multiplication by 128×128-bit matrix followed by the $128 \rightarrow 128$ concatenated bijection.

Table types. To fully transform AES to WB AES the following 4 table types are needed. As mentioned above, external encoding uses 128×128 a matrix multiplication to protect input and output of the cipher. The matrix decomposition technique introduced in section 3.3.2 is used to make table implementation feasible. Figure 3.2 depicts $8 \rightarrow 128$ table used in the first and the last round for the external encoding.

Recall already mentioned T-boxes in section 3.3.2 that performs *SubByte* and *AddRoundKey* operations. The *MixColumn* operation is implemented by the matrix multiplication de-

5. (n, k, d)-code is Maximum Distance Separable if $d = n-k+1$, `http://www.mth.msu.edu/~jhall/classes/codenotes/Linear.pdf`

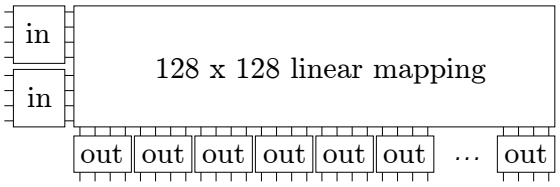

Figure 3.2: Table type I, $8 \rightarrow 128$ mapping, used for external encoding, taken from [18]

composition, these 8×32 tables are called T_y boxes. In order to save space T and T_y boxes are composed to resulting type II table as illustrates figure 3.3.

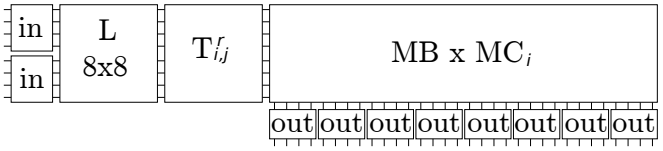

Figure 3.3: Table type II, $8 \rightarrow 32$ mapping, taken from [18]

Note 32×32 mixing bijection added after the *MixColumn* operation to increase diffusion. To cancel this transformation the table type 3 is used as illustrates figure 3.4. Also recall the matrix multiplication decomposition requires addition operation, this is done by exclusive-OR (XOR), cascade of table type 4 is used for this purpose. Note that performing XOR by table look-up is rather ineffective, but it is required to protect look-up tables by IO bijections and mixing bijections (to form protected network of look-up tables without revealing true values on table boundaries).

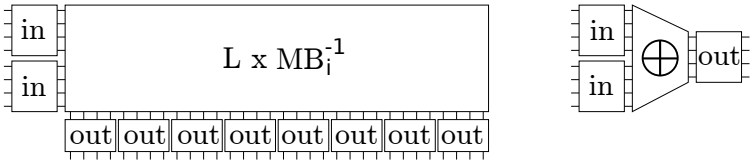

Figure 3.4: Table type III and IV, taken from [18]

Overall scheme. Figure 3.5 illustrates how the round function of whitebox AES implementation for one column looks, using aforementioned tables.

On the diagram in figure 3.5 are the following mappings:

- MB stands for Mixing Bijection. It is 32×32 matrix over GF(2) representing linear

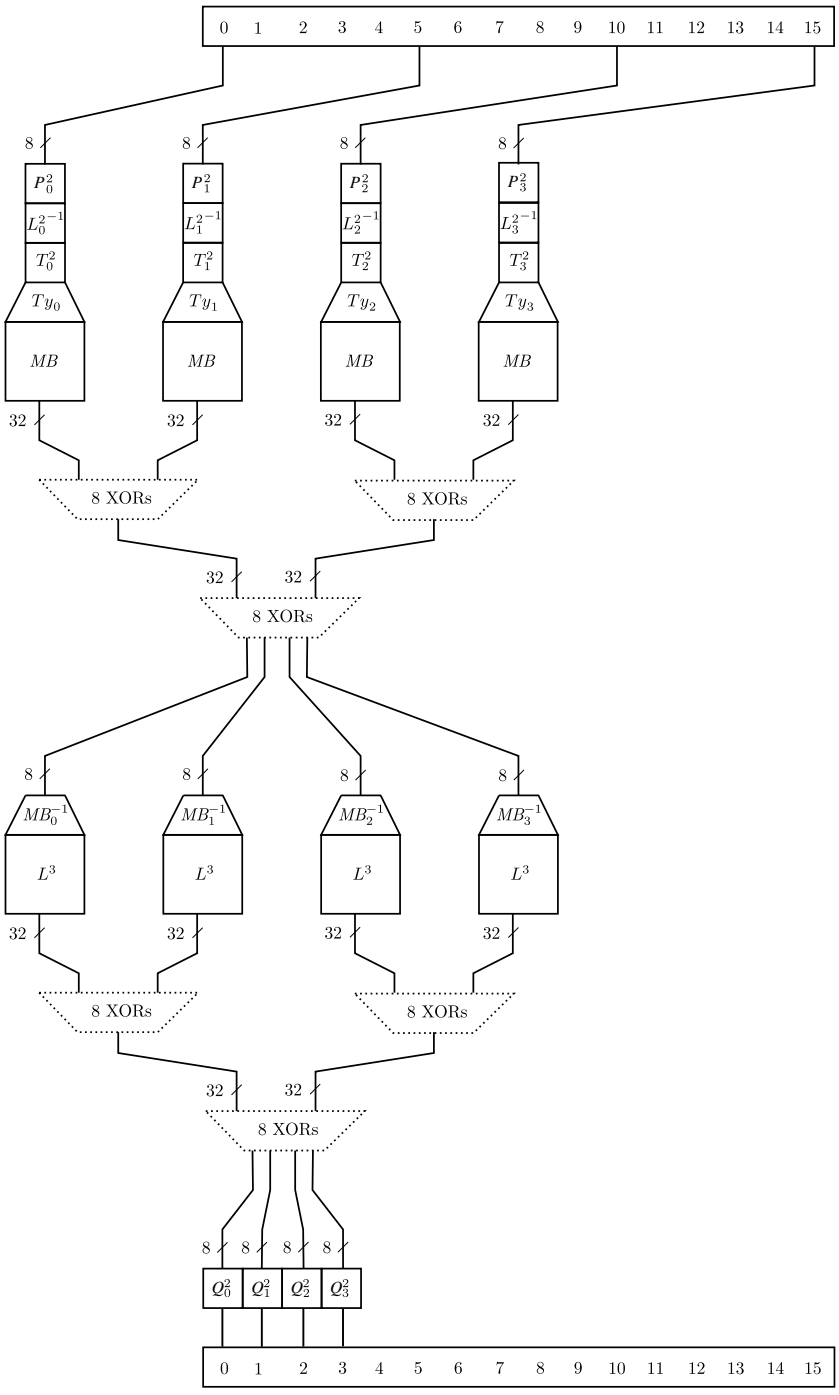

Figure 3.5: Whitebox AES implementation - round #2, based on diagram from [16]

transformation over GF(2). $\mathrm{MB}^{-1}_{\{0,1,2,3\}}$ are then column stripes of corresponding MB inverse matrix (the matrix multiplication decomposition). This transformation cancels out within one round.

- L stands also for Mixing Bijection but in this case it is 8×8 matrix.
- Q,P. These are random IO bijections. It holds that $P^{r+1}_{i,j} \circ Q^{r}_{i,j} = id$

3.3.4 Cipher invertibility

One of the requirements on the whitebox cipher implementation is usually a *non-invertibility*. It means that given an encryption part of the cipher with embedded key one should not be able to use it also for decryption and vice versa. This property is especially useful if one wants to use symmetric cipher to simulate an asymmetric. But it is important to realize that this goal is difficult to achieve in whitebox context.

As an example take AES whitebox implementation Inverting the cipher in blackbox context is rather computationally difficult. Using brute-force one would need 2^{128} operations to invert the cipher. The whitebox context is in contrast to blackbox advantageous for an attacker. One of the problems here is that ShiftRows operation can be very easily canceled in whitebox context and that attacker can execute only particular round of the cipher. We propose some improvement addressing this problem in section 5.3.

There are 4 columns of state array within one round independent on each other. Thus cipher can be easily inverted running the cipher backwards and finding inversion for each column separately, assuming 128×128 external mixing bijections are I_{128}. Thus the task is to find inversion of 32-bit wide function representing one round of the cipher on one column of the state array by running through the space GF $\left(2^8\right)^4$, evaluating the round function and comparing with wanted result.

Computational complexity to invert the cipher is $10 \cdot 4 \cdot 2^{32}$ operations [6].

One can also pre-compute tables for inverted cipher, that would occupy $10 \cdot 4 \cdot (2^{32} \cdot 4)$ B $\approx$ 69 GB. We have implemented inversion of WB AES. In non-optimized version it takes 13 hours on my hardware[7] to invert WB AES with negligible memory requirements.

3.4 The BGE attack

The paper [9] by Billet *et al.* demonstrated that whitebox AES implementation as described in the previous section is vulnerable to an algebraic attack. The attack is named after authors, the BGE attack. It recovers the symmetric key from whitebox AES implementation and negligible memory requirements in 2^{30} computational steps.

6. 10 (rounds) ·4 (columns) $\cdot 2^{32}$ (column function input space size)
7. For hardware specifications see A.2

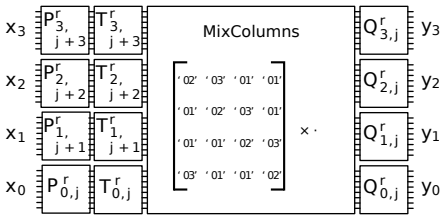

Figure 3.6: AES round function from the BGE attack perspective, taken from [9]

The BGE attack does not analyze look-up tables locally, but instead the whole AES round function is analyzed as a single look-up table. This has few benefits:

- structure of the round function is fixed and well-known, thus it is easy to model it as an algebraic equation.
- 32×32 mixing bijections MB is canceled within one round, so they can be neglected.
- 8×8 mixing bijection L can be easily merged with the IO bijections on round boundary, what simplifies further algebraic analysis.

Figure 3.6 illustrates the round function of AES for one column from the BGE attack perspective. Depicted function can be described as a mapping $(x_0, x_1, x_2, x_3) \xrightarrow{R_j^r} (y_0, y_1, y_2, y_3)$, $j = 0, \ldots, 3$.

Since transformation L, L^{-1} is performed byte-wise on the state array, it can be composed with corresponding IO bijections, as in equation 4.15.

$$Q_{i,j}^{r\,\prime} = Q_{i,j}^r \circ L_j^{r+1} \tag{3.6a}$$

$$P_{i,j}^{r\,\prime} = (L_j^r)^{-1} \circ P_{i,j}^r \tag{3.6b}$$

The IO bijections are generally non-linear, thus composing it with another linear transformation (L) results again in non-linear bijection. Note that in the BGE attack, IO bijections are considered to be general $8 \to 8$ bijections, neglecting the fact they are concatenated from 2 $4 \to 4$ bijections what allows composition with L.

The BGE attack proceeds in three steps:

1. Transform the non-linear IO bijections $Q_{i,j}^r$ to unknown GF (2)-affine transformation (i.e. determine non-linear part up to unknown affine part).
2. Fully determine $Q_{i,j}^r$ bijection, using algebraic analysis and known form of the round function.
3. Obtain round keys from 2 consecutive rounds and recover symmetric key using reversibility of the AES key-schedule.

3.4.1 Recovering non-linear parts

This step is particularly interesting because of its universality. It can be applied on different whitebox implementation.

The main goal of this step is to recover non-linear parts of $(Q_i^r)_{i=0,\dots,3}$. Consider y_0 as a function of (x_0, x_1, x_2, x_3). If x_1, x_2, x_3 are fixed as constants c_1, c_2, c_3 it is easy to see that the following holds:

$$\begin{aligned} y_0(x_0, c_1, c_2, c_3) &= Q_{0,j}^r\left(\alpha T_{0,j}^r\left(P_{0,j}^r(x)\right) \oplus \beta_{c_1,c_2,c_3}\right) \\ &= Q_{0,j}^r \circ \oplus_{\beta_{c_1,c_2,c_3}} \circ \alpha \cdot T_{0,j}^r \circ P_{0,j}^r(x) \end{aligned} \tag{3.7}$$

The function y_0 from the equation 3.7 now takes only 256 input values, thus the functions y_0 and y_0^{-1} can be easily evaluated (as look-up tables). For the rest of the chapter consider constants c_2, c_3 fixed, without loss of generality let $c_2 = c_3 = 0$. For the simplification the r superscript is dropped from the equations if it is clear from the context.

Now assume $c_1' \neq c_1$, also denote $\beta_0 = \beta_{c_1,c_2,c_3}, \beta_1 = \beta_{c_1',c_2,c_3}$, it is visible that:

$$\begin{aligned} &y_0(x_0, c_1', c_2, c_3) \circ y_0^{-1}(x_0, c_1, c_2, c_3) = \\ &= (Q_{0,j} \circ \oplus_{\beta_1} \circ \alpha \cdot T_{0,j} \circ P_{0,j}) \circ \left(P_{0,j}^{-1} \circ (\alpha \cdot T_{0,j})^{-1} \circ \oplus_{\beta_0} \circ Q_{0,j}^{-1}\right) \\ &= Q_{0,j} \circ \oplus_{\beta_1} \circ \oplus_{\beta_0} \circ Q_{0,j}^{-1} \\ &= Q_{0,j} \circ \oplus_{(\beta_0 \oplus \beta_1)} \circ Q_{0,j}^{-1} \end{aligned}$$

Thus by fixing c_1 and iterating c_1' we obtain the set of 256 bijections, represented as look-up tables, of the form $Q_{0,j} \circ \oplus_\beta \circ Q_{0,j}^{-1}$, where β takes all values from GF (2^8).

Theorem 1. *Given a set of functions $\mathcal{S} = \{Q \circ \oplus_\beta \circ Q^{-1}\}_{\beta \in GF(2^8)}$ given by values, where Q is a permutation of $GF(2^8)$ and the $\oplus_\beta$ is the translation by β in $GF(2^8)$, one can construct a particular solution $\widetilde{Q}$ such that there exists an affine mapping A so that $\widetilde{Q} = Q \circ A$.*

For the proof see the original paper [9]. In order to recover non-linear part of the IO bijection it is needed to generate the set $\mathcal{S}$ (by evaluating y_0 functions) and to apply the theorem.

Note that the set $\mathcal{S}$ contains 256 functions, but only as look-up tables so the β is unknown. One easily verifies the set $\mathcal{S}$ together with operation of a function composition form a group.

Observe there exists isomorphism φ :

$$\begin{array}{rccc} \varphi & : & \mathcal{S} & \longrightarrow & \mathrm{GF}(2)^2 \\ & & Q \circ \oplus_\beta \circ Q^{-1} & \longmapsto & [\beta] \end{array}$$

Since we don't know the values β for the functions in $\mathcal{S}$ the φ cannot be constructed directly, however it is possible to recover φ up to unknown linear transformation L i.e. $\psi =$

$L^{-1} \circ \varphi$. Without going into details, the ψ is determined by finding base functions $f_1, \ldots, f_8$ that spans the whole $\mathcal{S}$ under composition. Mapping ψ is then constructed by assigning standard basis vectors $e_1, \ldots, e_8 \in \mathrm{GF}(2)^8$ to these functions.

It is then possible to obtain $\widetilde{Q}$ by using the mapping ψ from the theorem 1 according to the following formula:

$$\widetilde{Q}(\psi(g)) = g(0), \; g \in \mathcal{S} \tag{3.9}$$

Note that the value $g(0)$ is known, the function was evaluated during ψ construction. The term $\psi(g)$ is also easy to evaluate, since we have constructed mapping ψ. Thus in order to recover $\widetilde{Q}$ as a look-up table we just run over the set $\widetilde{Q}$ and compute the values of mapping $\widetilde{Q}$ according to the equation 3.9.

3.4.2 Recovering the symmetric key

As the rest of the BGE attack is very AES and implementation specific it is not covered in detail. In this phase, we are still in situation as depicted in figure 3.6 with a difference $(Q_i^r)_{i=0,\ldots,3}$ and $\left(P_i^{r+1}\right)_{i=0,\ldots,3}$ are affine, still matching, bijections. This fact makes further algebraic analysis of round function possible.

Consider following useful proposition:

Proposition 1. *For any pair (y_i, y_j) exists a unique linear mapping L and a unique constant c such that:*

$$\forall x_0 \in GF\left(2^8\right) : \; y_i(x_0, 0, 0, 0) = L(y_j(x_0, 0, 0, 0)) \oplus c \tag{3.10}$$

By using 1 one can obtain the linear parts of Q_1, Q_2, Q_3 from the knowledge of the Q_0 linear part in 2^{16} steps by simple running over c and testing the resulting mapping for linearity in 2^8 steps. Thus the rest of the attack focuses on Q_0 determination.

First of all, the linear part of Q_0 up to $[\gamma], \gamma \in \mathrm{GF}\left(2^8\right) \setminus \{0\}$ is determined, where $[\gamma]$ denotes the matrix in GF (2) corresponding to multiplication by γ in GF $\left(2^8\right)$, for more details see appendix A.4.

Then γ and constant part q_0 of Q_0 are determined. During this steps, the public knowledge of S-boxes and MixColumn matrix coefficients is used.

With completely determined Q_0 the round keys are extracted. The attack ends using the fact the key-schedule of AES is reversible, so from two consecutive round keys one can derive original symmetric key.

4 WBCAR AES using dual ciphers

WBCAR stands for whitebox context attack resistant, meaning cipher implementation should resist attacks like key-extraction, cipher inversion and others against attacker in whitebox context.

In this chapter I describe whitebox scheme proposed in [14] that make use of AES dual ciphers. It is supposed that using dual AES, different in each round, will increase security of whitebox implementation of the cipher. Paper says that this modification results in raising BGE attack [9] complexity to 2^{91} computational steps, making it unfeasible to perform it in practice. It is shown in section 4.2 that this assumption is false due to existence of an attack we performed.

4.1 Scheme

In the original paper [14] the explanation how to obtain dual AES ciphers and how to construct mapping from one to another is not given. This is important part since it plays crucial role in a proof that this scheme is vulnerable. At first is described generalization of the AES and how to construct mappings between them.

4.1.1 Generic AES

It is possible to generalize AES by changing its irreducible polynomial and generator to obtain generic form of AES.

Generic AES can be represented as a $\{R(x), \beta\}$, where $R(x) \in (\mathbb{Z}/p\mathbb{Z})[x]$ is irreducible polynomial of degree 8, $\beta \in \mathrm{GF}(2^8)$ is a generator of the field $\mathrm{GF}(2^8)$.

Default AES (as in NIST standard) in this notation is represented as $\{\{11B\}_x, \{03\}_x\}$. Polynomial is expressed in hexadecimal notation, each bit corresponds to polynomial coefficient, LSB corresponds to constant term.
Thus $0x11B_{16} = 1\ 0001\ 1011_2 \Rightarrow \{11B\}_x \sim x^8 + x^4 + x^3 + x + 1$.

It is known that there are 30 irreducible polynomials over $\mathrm{GF}(2^8)$. For each of them there are 8 possible generators that can be used to generate field and to preserve duality mentioned in the next section.

4.1.2 AES duality

Definition 3. *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*

$$\forall p, k : E_k(p) = f^{-1}\left(E'_{g(k)}(h(p))\right) \tag{4.1}$$

where p is the plaintext, and k is the secret key.

4.1.3 Generic AES duality

Let's assume we have some arbitrary generic AES $\{R(x), \beta\}$.

All elements of the field GF $(2^8) = \{01, 02, \ldots, FF\}$ can be expressed in terms of the generator β, GF $(2^8) = \{\beta^i \mid i \in [0, 254]\} = \{\beta^0, \beta^1, \ldots, \beta^{254}\}$.

We can then construct 8×8 matrix $\Delta = \begin{bmatrix}\beta^0 & \beta^{25} & \beta^{50} & \beta^{75} & \beta^{100} & \beta^{125} & \beta^{150} & \beta^{175}\end{bmatrix}$ where $\beta^i \in$ GF $(2^8) \cong$ GF$(2)^8$ is a column vector. Then Δ is a base change matrix:

$$\Delta : \{\{11B\}_x, \{03\}_x\} \longrightarrow \{R(x), \beta\} \tag{4.2a}$$
$$\Delta^{-1} : \{R(x), \beta\} \longrightarrow \{\{11B\}_x, \{03\}_x\} \tag{4.2b}$$

For default AES $\{\{11B\}_x, \{03\}_x\}$ holds

$$\begin{aligned}\Delta &= \begin{bmatrix}03^0 & 03^{25} & 03^{50} & 03^{75} & 03^{100} & 03^{125} & 03^{150} & 03^{175}\end{bmatrix} \\ &= \begin{bmatrix}01 & 02 & 04 & 08 & 16 & 32 & 64 & 128\end{bmatrix} \\ &= I_8\end{aligned}$$

as expected.

From this it is clear that following duality holds: $E \sim \{\{11B\}_x, \{03\}_x\},\ E' \sim \{R(x), \beta\}$ then:

$$\forall p, k : E_k(p) = \Delta^{-1}\left(E'_{\Delta(k)}(\Delta(p))\right) \tag{4.3}$$

4.1.4 Constructing Dual AES

We can construct arbitrary dual AES from default AES. Recall there are 4 operations used in single AES round: *ShiftRows*, *AddRoundKey*, *SubByte*, *MixColumn*.

Default AES *ShiftRows*, *AddRoundKey* functions are quite simple and remain same after AES generalization. *SubByte*, *MixColumn* are affected by generalization, for proper understanding of construction generic AES, description follows.

SubByte

$$\begin{aligned}S : \text{GF}\left(2^8\right) &\longrightarrow \text{GF}\left(2^8\right) \\ x &\longmapsto A \times x^{-1} \oplus c\end{aligned} \tag{4.4}$$

where x^{-1} is element inverse in GF (2^8), A is 8×8 matrix over GF(2), c is column vector GF$(2)^8$. A, c are constants defined in NIST standard.

Equation for affine transformation used in S-box:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{4.5}$$

where $x_i, y_i \in \mathrm{GF}(2)$.

MixColumn

- columns considered as polynomials over GF (2^8)
- $p(x) \cdot c(x) \pmod{x^4 + 1}$
  where $c(x)$ is fixed polynomial $c(x) = 03x^3 + 01x^2 + 01x + 02$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4.6}$$

where $x_i, y_i \in \mathrm{GF}\left(2^8\right)$.

Generic AES In the generic AES, the operations *ShiftRows*, *AddRoundKey* work same as in default AES, they are not affected by base change operation.

SubByte

$$\begin{aligned} S_{dual} : \mathrm{GF}\left(2^8\right) &\longrightarrow \mathrm{GF}\left(2^8\right) \\ x &\longmapsto \Delta \times A \times \Delta^{-1}\left(x^{-1}\right) \oplus \Delta\left(c\right) \end{aligned} \tag{4.7}$$

MixColumn MixColumn matrix coefficients are expressed in terms of generator $\beta = 03$.

$$\begin{bmatrix} \beta^{25} & \beta^1 & \beta^0 & \beta^0 \\ \beta^0 & \beta^{25} & \beta^1 & \beta^0 \\ \beta^0 & \beta^0 & \beta^{25} & \beta^1 \\ \beta^1 & \beta^0 & \beta^0 & \beta^{25} \end{bmatrix}$$

Round function - default AES We consider whole AES round as a single function R of a state array. Let's define

$$\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix} \xrightarrow{R} \begin{bmatrix} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,3} \\ y_{1,0} & y_{1,1} & y_{1,2} & y_{1,3} \\ y_{2,0} & y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,0} & y_{3,1} & y_{3,2} & y_{3,3} \end{bmatrix}$$

From this we define $y_{i,j}$ as a function with 4 arguments from GF (2^8):

$$y_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) = \bigoplus_{l=0}^{3} \alpha_{l,j} \cdot S(x_{i,l} \oplus k_{i,l}) \tag{4.8}$$

where $\alpha_{k,j}$ is MixColumn matrix coefficient in k-th row and j-th column. We are abstracting here *ShiftRows* operation, it won't be needed for our further argumentation. To make it clear here are equations for the first column of state array:

$$y_{0,0}\left(x_{0,0}, x_{1,0}, x_{2,0}, x_{3,0}\right) = 02 \cdot T_{0,0}(x_{0,0}) \oplus 03 \cdot T_{1,0}(x_{1,0}) \oplus 01 \cdot T_{2,0}(x_{2,0}) \oplus 01 \cdot T_{3,0}(x_{3,0}) \tag{4.9a}$$
$$y_{1,0}\left(x_{0,0}, x_{1,0}, x_{2,0}, x_{3,0}\right) = 01 \cdot T_{0,0}(x_{0,0}) \oplus 02 \cdot T_{1,0}(x_{1,0}) \oplus 03 \cdot T_{2,0}(x_{2,0}) \oplus 01 \cdot T_{3,0}(x_{3,0}) \tag{4.9b}$$
$$y_{2,0}\left(x_{0,0}, x_{1,0}, x_{2,0}, x_{3,0}\right) = 01 \cdot T_{0,0}(x_{0,0}) \oplus 01 \cdot T_{1,0}(x_{1,0}) \oplus 02 \cdot T_{2,0}(x_{2,0}) \oplus 03 \cdot T_{3,0}(x_{3,0}) \tag{4.9c}$$
$$y_{3,0}\left(x_{0,0}, x_{1,0}, x_{2,0}, x_{3,0}\right) = 03 \cdot T_{0,0}(x_{0,0}) \oplus 01 \cdot T_{1,0}(x_{1,0}) \oplus 01 \cdot T_{2,0}(x_{2,0}) \oplus 02 \cdot T_{3,0}(x_{3,0}) \tag{4.9d}$$

where $T_{i,j}(x) = S\left(x \oplus k_{i,j}\right)$.

Round function - generic AES Using aforementioned generic form of *SubByte* and *MixColumn* functions we can define round function also for generic AES in the same way, using base change transformation. From this we define $y_{i,j}$ as a function with 4 arguments from GF (2^8):

$$y_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) = \bigoplus_{l=0}^{3} \Delta(\alpha_{l,j}) \cdot \left(\Delta \times A \times \Delta^{-1}\left(\left(x_{i,l} \oplus \Delta\left(k_{i,l}\right)\right)^{-1}\right) \oplus \Delta\left(c\right)\right) \tag{4.10}$$

4.1.5 Whitebox Dual AES

The paper describing AES whitebox implementation with use of dual AES [14], claimed that this implementation should be harder (in terms of time complexity) to break using the BGE attack on whitebox AES. On the figure 4.1 is scheme for one round, one column of state array, whitebox dual AES implementation for round 2. According to the original paper, in each column is used different generic AES. This implementation is compatible with default AES, so after computing in dual AES we have to transform the result to default AES with base change transformation Δ.

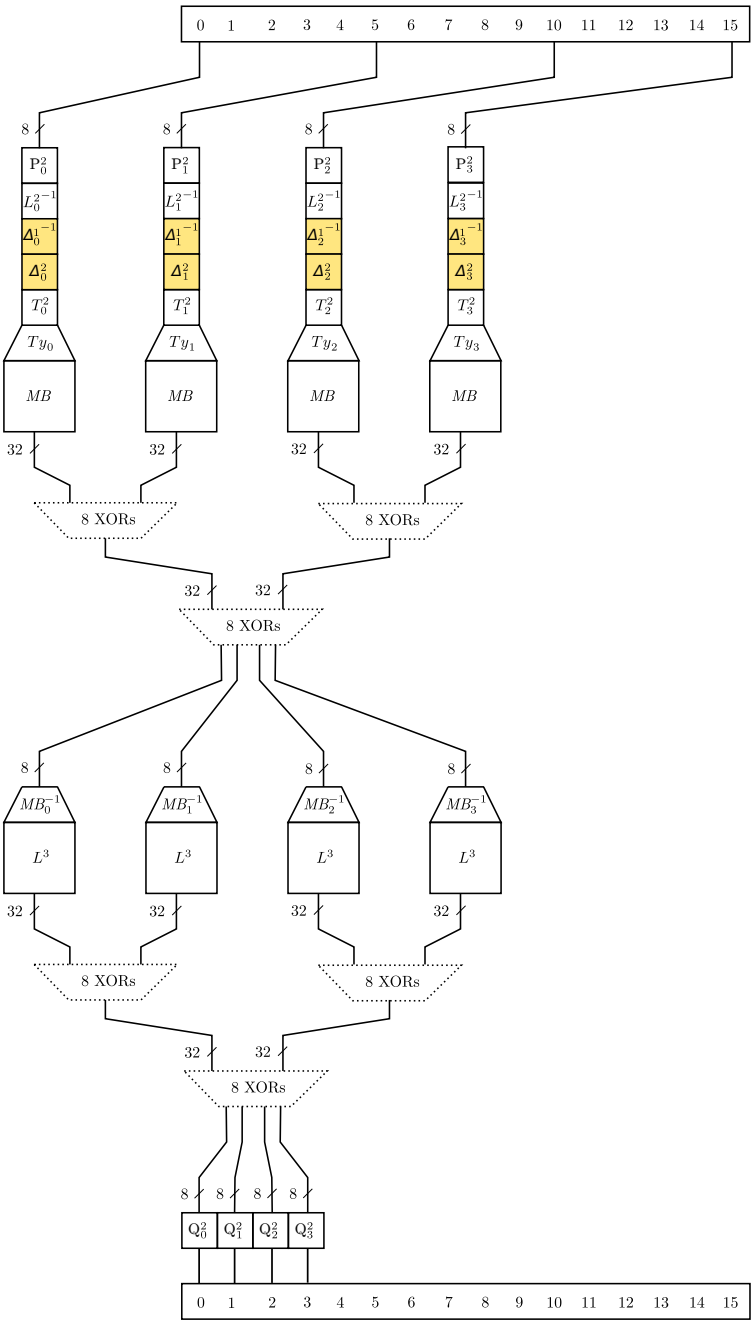

Figure 4.1: Whitebox Dual AES implementation - round #2, based on diagram from [16]

By changing irreducible polynomial and generator we obtain $30 \cdot 8 = 240$ different generic AES ciphers. The bigger is set of possible ciphers to use, the harder is for attacker to break the dual scheme, since he has to try all possible combinations, according to [14]. But in [14] is assumed there are 61200 different generic AES ciphers but it is not said how they are constructed and how such construction influences whitebox implementation.

In order to generate 61200 different AES representations it is needed to study AES S-Box affine self-equivalences [19].

Definition 4. *$n \times n$-bit mapping S is affine self-equivalent if there exist $n \times n$ affine relations $A_1,\ A_2$ such that:*

$$A_2 \circ S \circ A_1 = S \tag{4.11}$$

In [19] are published effective algorithms for finding linear and affine equivalences for permutations (S-boxes). I have implemented them [1] to verify number of self-equivalences for AES S-Boxes for encryption and decryption algorithm. This algorithm can be further used to study modified S-boxed or basic building blocks of the cipher (see chapter 5.1) for equivalences, what can lead to revealing potential weaknesses.

Algorithm found 2040 affine self-equivalences, together with 30 possible irreducible polynomials there are 61200 dual ciphers. Biryukov *et al.* derived general expressions for $A_1,\ A_2$:

$$A_1(x) = [a] \cdot Q^i \cdot x \tag{4.12a}$$

$$A_2(x) = A\left(Q^{-i} \cdot [a] \cdot A^{-1}(x)\right) \tag{4.12b}$$

Where

- A is fixed affine mapping from AES S-box definition (see section 4.1.4), A^{-1} it's inverse.
- $[a]$ denotes 8×8 matrix with coefficients from GF (2) representing *multiplication* by $a \in \text{GF}(2^8) \setminus \{0\}$ in GF (2^8). From the fact $\text{GF}(2)^8 \simeq \text{GF}(2^8)$ (all finite fields with same number of elements are isomorphic), multiplication is linear transformation in GF (2^8), it can be expressed in matrix form. Refer to A.4 to see how to construct such matrix.
- Q denotes 8×8 matrix with coefficients from GF (2) representing *squaring* in GF (2^8). Squaring is *linear* operation in GF (2^8) so it is possible to represent it as a matrix. Refer to A.3 for construction and proof. Note that $Q^8 = I \Rightarrow Q^{-i} = Q^{8-i}$. Thus there are 8 different powers of Q, $Q^i,\ i \in [0, 7]$.

1. path: implementation/LinearAffineEq.{h,cpp}

It is visible that with general expressions we can obtain $8 \cdot 255 = 2040$ different $A_1,\ A_2$ relations[2] confirming output of the algorithm.

Observe that by inserting $A_1,\ A_2$ before and after S-Box the cipher is not affected. Note that A_1 is linear, this can be used to push input mapping A_1 through the mixing layer and combine it with A_2 from previous round, then we obtain 2040 different AES tables evaluating the same function. Note that they are not dual according to definition 3, it is just different table implementation of AES, thus $\Delta = I$. This is potential flaw of whitebox scheme using dual ciphers, since it does not increase security against known attacks at all. This construction is neglected in the original paper [14].

Layers of mixing bijections (L, MB) cancel between rounds so they can be neglected in pushing A_1 to previous round. Recall $A_2\left(S\left(A_1\left(x\right)\right)\right) = S(x)$. Input to S-box is output of previous round, so apply A_1 on previous round. It is shown only for one column (equation 4.13), others are analogical. It is easy to verify that $[a] \cdot Q^i \cdot [c] = \left[c^{2^i}\right] \cdot [c] \cdot Q^i$. Redefine $T_r(x) = A_2^r(S(x \oplus k))$ to take affine relations into account, where k is particular round key byte. Then we can write:

$$A_1^{r+1} \cdot \begin{bmatrix} 02 \cdot T_r(x) & 01 \cdot T_r(x) & 01 \cdot T_r(x) & 03 \cdot T_r(x) \end{bmatrix}^T \tag{4.13a}$$

$$\begin{aligned} &= \begin{bmatrix} A_1^{r+1}(02 \cdot T_r(x)) & A_1^{r+1}(01 \cdot T_r(x)) & A_1^{r+1}(01 \cdot T_r(x)) & A_1^{r+1}(03 \cdot T_r(x)) \end{bmatrix}^T \\ &= \begin{bmatrix} 02^{2^i} \cdot A_1^{r+1}(T_r(x)) & 01^{2^i} \cdot A_1^{r+1}(T_r(x)) & 01^{2^i} \cdot A_1^{r+1}(T_r(x)) & 03^{2^i} \cdot A_1^{r+1}(T_r(x)) \end{bmatrix}^T \\ &= \begin{bmatrix} 02^{2^i} \cdot A_1^{r+1}(T_r(x)) & 01 \cdot A_1^{r+1}(T_r(x)) & 01 \cdot A_1^{r+1}(T_r(x)) & 03^{2^i} \cdot A_1^{r+1}(T_r(x)) \end{bmatrix}^T \end{aligned} \tag{4.13b}$$

This gives us different tables for AES round function when using different $A_1,\ A_2$ relations. Note that table of type 2 with A_1 pushed from next round is still the same, no matter which form of equation 4.13 is used, due to linearity of A_1. For simplicity in further proofs and description we can assume form 4.13a.

4.2 Attacking Dual AES scheme

According to [14] whitebox scheme using Dual AES is considered to be more difficult to crack with BGE attack and thus it is consider safer than original scheme proposed in [1]. But we show that it is not true. This result was not published yet.

Proposition 2. *Whitebox Dual AES scheme can be broken with the BGE attack with the same time complexity as Whitebox AES scheme.*

2. by running through possible values for a, i

Proof. Let's define round function for whitebox AES and for whitebox dual AES and compare it. Note that MB mixing bijections are left out since their effect is canceled within one round. If we also assume use of $A_1,\ A_2$ affine relations, they can be merged together with L mixing bijection to input/output encodings. Thus for simplicity $A_1,\ A_2$ is omitted from the proof (it is clearly visible it does not increase resistance against BGE attack - input/output encodings are fully determined in the attack).

Round function - whitebox AES. There are additional Q', P' functions, input and output bijections, for details see [1] [9].

$$y_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) = Q'^{r}_{i,j}\left(\bigoplus_{l=0}^{3} \alpha_{l,j} \cdot S\left(P'^{r}_{i,l}\left(x_{i,l}\right)\right)\right) \tag{4.14a}$$

$$= Q'^{r}_{i,j}\left(\bigoplus_{l=0}^{3} \alpha_{l,j} \cdot \left(A\left(\left(P'^{r}_{i,l}\left(x_{i,l}\right) \oplus k_{i,l}\right)^{-1}\right) \oplus c\right)\right) \tag{4.14b}$$

$$= Q'^{r}_{i,j} \circ R_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) \tag{4.14c}$$

Round function - whitebox dual AES. For simplicity define:

$$P''^{r}_{i,j} = \left(\Delta^{r-1}\right)^{-1} \circ P'^{r}_{i,j} = \left(\Delta^{r-1}\right)^{-1} \circ \left(L^{r}_{j}\right)^{-1} \circ P^{r}_{i,j} \tag{4.15}$$

Also we have to distinguish in which dual AES is element encoded, so define x^{Δ} as a element of the dual AES which base change matrix is Δ from standard AES field. The same holds for inversion operation $^{-1}$. Denote $^{-1^{\Delta}}$ inversion in field of dual AES which has base change matrix Δ.

For simplicity assume that $\Delta = \Delta^r$ for round r if it is obvious from context and is not

defined otherwise. According to figure 4.1 the equation for one round is:

$$
\begin{aligned}
& y_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) = \\
& = Q_{i,j}^{r\,\prime}\left(\bigoplus_{l=0}^{3}\Delta(\alpha_{l,j})\cdot\left(\Delta\times A\times\Delta^{-1}\left(\left(\Delta\circ P_{i,l}^{r}{}''\left(x_{i,l}\right)\oplus\Delta\left(k_{i,l}\right)\right)^{-1^{\Delta}\,\mathrm{GF}\left(2^{8}\right)}\right)\oplus\Delta\left(c\right)\right)\right) \\
& = Q_{i,j}^{r\,\prime}\circ\Delta\left(\bigoplus_{l=0}^{3}\alpha_{l,j}\cdot\left(A\times\Delta^{-1}\left(\left(\Delta\circ P_{i,l}^{r}{}''\left(x_{i,l}\right)\oplus\Delta\left(k_{i,l}\right)\right)^{-1^{\Delta}\,\mathrm{GF}\left(2^{8}\right)}\right)\oplus c\right)\right) \\
& = Q_{i,j}^{r\,\prime}\circ\Delta\left(\bigoplus_{l=0}^{3}\alpha_{l,j}\cdot\left(A\times\Delta^{-1}\left(\left(\Delta\left(P_{i,l}^{r}{}''\left(x_{i,l}\right)\oplus k_{i,l}\right)\right)^{-1^{\Delta}\,\mathrm{GF}\left(2^{8}\right)}\right)\oplus c\right)\right) \qquad (4.16a) \\
& = Q_{i,j}^{r\,\prime}\circ\Delta\left(\bigoplus_{l=0}^{3}\alpha_{l,j}\cdot\left(A\times\Delta^{-1}\left(\Delta\left(P_{i,l}^{r}{}''\left(x_{i,l}\right)\oplus k_{i,l}\right)^{-1\,\mathrm{GF}\left(2^{8}\right)}\right)\oplus c\right)\right) \qquad (4.16b) \\
& = Q_{i,j}^{r\,\prime}\circ\Delta\left(\bigoplus_{l=0}^{3}\alpha_{l,j}\cdot\left(A\times\left(\left(P_{i,l}^{r}{}''\left(x_{i,l}\right)\oplus k_{i,l}\right)^{-1\,\mathrm{GF}\left(2^{8}\right)}\right)\oplus c\right)\right) \\
& = Q_{i,j}^{r\,\prime}\circ\Delta\left(\bigoplus_{l=0}^{3}\alpha_{l,j}\cdot\left(A\times\left(\left(P_{i,l}^{r}{}''\left(x_{i,l}\right)\oplus k_{i,l}\right)^{-1\,\mathrm{GF}\left(2^{8}\right)}\right)\oplus c\right)\right) \\
& = Q_{i,j}^{r\,\prime}\circ\Delta\circ R_{i,j}^{\prime}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) \qquad (4.16c)
\end{aligned}
$$

Now it is easy to see whitebox dual AES correctness, moreover it is visible that the same attack breaking whitebox AES breaks whitebox dual AES scheme.

Transformation from 4.16a to 4.16b is possible due to base change matrix properties and fields we are computing in.

$$\forall x, y\in\mathrm{GF}\left(2^{8}\right)\ :\ y=x^{-1}\Rightarrow\Delta y=(\Delta x)^{-1^{\Delta}} \qquad (4.17)$$

Note that element inversion GF (2^8) has changed from one field to another.

Now if we compare equations 4.14c and 4.16c, they are very similar, the only difference here is the application of base change matrix Δ.

Here we can do the similar thing we did in equations 3.6, 4.15 where we composed two transformations, non-linear and linear to non-linear transformation, with equation 4.16c.

We can thus define:

$$
\begin{aligned}
Q_{i,j}^{r\,\prime\prime} = Q_{i,j}^{r\,\prime} = Q_{i,j}^{r}\circ\Delta = Q_{i,j}^{r}\circ L_{j}^{r+1}\circ\Delta \qquad (4.18a) \\
y_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) = Q_{i,j}^{r\,\prime}\circ\Delta\circ R_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) \qquad (4.18b) \\
y_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) = Q_{i,j}^{r\,\prime\prime}\circ R_{i,j}\left(x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3}\right) \qquad (4.18c)
\end{aligned}
$$

Now it is evident that equations for whitebox AES 4.14c and 4.18c are the same, the only difference is only in non-linear transformations Q, but it is important they are both non-linear.

Conclusion is if attack can break whitebox AES scheme with round function 4.14c it can also break whitebox dual AES scheme. During the attack is transformation Q fully determined, we verified that if Dual AES scheme is used, transformation Q is the exact form as described above. □

4.3 Implementation of the cipher

[TODO] Cipher implementation description, generalization of oorschot design. Mixing bijections.

4.4 Results

[TODO] Practical results for implementation, performance statistics, results.

4.5 Implementation of the attack

[TODO] Attack implementation description

4.6 Attack results

[TODO] Practical results for attack implementation, time to break.

5 Scheme improvement

The BGE attack [9] strongly relies on publicly known constants and building blocks used in the AES cipher (MixColumn constants, fixed S-box). This leads us to an idea of turning constant part of cipher into key dependent ones, according to Kerckhoffs's principle.

It should increase computational complexity of the attack since attacker would have to try all combinations of key dependent part of the cipher. In the ideal scenario the attack will be unfeasible due to high computational complexity.

As we know AES S-box is constant and has relatively simple algebraic form. In blackbox context, it is quite difficult to construct algebraic equations for whole AES (this was one of design criterion of an AES in order to prevent possible algebraic attacks), but BGE attack aims only on one round of the cipher and from this perspective it is quite easy to construct algebraic equations for 1 round - as we seen in the BGE attack, this is what makes AES vulnerable to algebraic attacks in whitebox context.

In whitebox implementation of cipher we have two contrary goals - to minimize table size and to prevent attack in whitebox context. Table size is what puts quite limitations in implementation and on security boundaries. In one extreme case we would build look-up table for whole AES for every possible input with total size $(2^{128} \cdot 16) > 10^{39}$ bytes. This scheme is no weaker than AES in blackbox context, therefore with the same level of security in whitebox context, but rather unfeasible in practice.

As we seen in BGE attack it is easy to turn random non-linear bijections (input/output encodings), protecting table contents, to affine transformations between rounds of cipher, what helps further algebraic analysis, so constructing more complicated non-linear bijections is probably not the way how to solve this problem. Whitebox construction should not solely rely on non-linear bijections [9, 10].

Widely used ciphers nowadays were designed also with the goal to be effectively implemented in a hardware (one of the criterion in AES contest). It influenced design of the cipher building blocks such as size of diffusion layer, matrix coefficients (lower are better), recycling of cipher primitives in key schedule algorithm, etc... Such additional restrictions do not lower security level of the cipher usually, because of iterating round function, but this can be problem in WBC.

To summarize weaknesses of the ciphers used to mount whitebox attacks:

- simple key schedule (reversible, forward/backward)
- key whitening technique
- publicly known, constant (key-independent) primitives

- weak round function, simple algebraic description
- use of diffusion elements that are easy to remove in whitebox context (e.g. ShiftRows in the AES)
- diffusion element with low diffusion power with respect to the one round of the cipher, i.e. low dependency of output byte from round function on input bytes (1 output byte of the AES round function depends on 4 input bytes)

As the AES in standard form is not suitable for WBC implementation, we are proposing to break the backward compatibility with AES (or any other well known cipher)- as it does not have proper structure for whitebox implementation, what is also visible from the fact there are no non-broken whitebox scheme of AES exists nowadays, as far as we know. Whitebox schemes using new techniques to protect cipher implementation (but still preserving backward compatibility with the cipher) were successfully cryptanalyzed effectively [9, 10, 13, 12]. In literature was already proposed to design a new cipher with whitebox implementation issues in mind [9, 18].

In our proposal of a new cipher suitable for WBC implementation we are addressing aforementioned weaknesses. The base of our cipher is AES since it uses widely accepted cipher building blocks and structure, it was extensively analyzed and is in general considered as a secure cipher.

Our modifications of the AES consist of:

- key-dependent S-Boxes
- non-reversible, strong, key schedule
- stronger (output byte dependence on input bytes), key-dependent diffusion element

In our proposal, we took inspiration from Twofish [3] cipher which has key dependent S-boxes with rather complicated algebraic representations. As I emphasized before, the key idea here is to make expressing one round of cipher as algebraic equations more difficult for an attacker. Our first scheme is to use Twofish key dependent S-boxes in AES algorithm.

5.1 Twofish S-boxes

Here observe Twofish S-boxes (from [3]) and their algebraic representation.

$$s_{0,k_0,k_1}(x) = q_1[q_0[q_0[x] \oplus k_0] \oplus k_1] \tag{5.1a}$$
$$s_{1,k_2,k_3}(x) = q_0[q_0[q_1[x] \oplus k_2] \oplus k_3] \tag{5.1b}$$
$$s_{2,k_4,k_5}(x) = q_1[q_1[q_0[x] \oplus k_4] \oplus k_5] \tag{5.1c}$$
$$s_{3,k_6,k_7}(x) = q_0[q_1[q_1[x] \oplus k_6] \oplus k_7] \tag{5.1d}$$

function	hashes per second
SHA1	63 G/s
MD5	180 G/s
BCrypt	71 K/s

Table 5.1: Hash functions performance comparison

Where q_0, q_1 are fixed 8-bit permutations, $k_i,\ i \in [0,7]$ are key bytes, $s_{j,k_a,k_b},\ j \in [0,4]$ are resulting S-boxes.

Thus instead of fixed AES S-box we use Twofish key dependent S-boxes. In particular we use $s_{j,k_a,k_b},\ j \in [0,4]$ instead of 4 the same constant S-boxes in computation of one column of state matrix - approach consistent with use of S-boxes Twofish algorithm (diffusion element is connected to the output of S-box).

In blackbox context there is disadvantage for key dependent S-boxes since it takes some time to generate them, for each encryption key, but in whitebox context the whole cipher is generated before use, including S-boxes, so during encryption/decryption there is no such disadvantage anymore.

5.2 Key schedule

Part of the BGE attack also make use of reversible AES key schedule to obtain encryption key. It is only needed to obtain round keys for two consecutive rounds of cipher in order to obtain full encryption key.

In order to avoid this reversing we propose to modify key schedule. In particular we suggest to use hash-chains as round keys, so attacker would not be able to combine knowledge of two consecutive rounds as is done in the BGE attack.

We suggest to use *bcrypt* [20] or *scrypt* [21] as a hash function for generating hash chains. We could also use iterated SHA hash function for this purpose, but the main reason we are proposing bcrypt or scrypt is the fact SHA hash been very successfully implemented in hardware (ASICS chips, for Bitcoin mining), with performance 1500 G hashes per second for one device [22]. Also the Bitcoin economy is based on SHA hash function, so there is motivation to build machines specialized on computing SHA hashes - making brute force attacks on SHA easier.

As an illustration consider [23], where M. Gosney used cluster made of GPUs (general purpose hardware) and benchmarked hash functions from performance perspective, for details see table 5.1. *bcrypt* is by orders of magnitude slower than SHA1, almost by factor 10^6. This makes brute-force attack practically unfeasible on general purpose hardware.

From this reason we propose to use hash functions that were specifically designed to take

a long time to evaluate (bcrypt, scrypt) and/or to be difficult to implement in hardware (contrary to standard hash functions like SHA, MD5).

Note that as key schedule is done during preparation of whitebox instance (generating tables) of a cipher, there is no time penalty during usage of the implementation.

In AES-128 we have 128 bit cipher key, $k_0, \ldots, k_{15}$. We define k_i^r to be round key byte $i \in [0, 15]$ used in round $r \in [0, 10]$.

We define hash function used further in our modified key schedule

$$hash\,(inp, salt)_{N_{bc},N_{sha}} = bcrypt\left(N_{bc}, salt, \text{SHA-256}^{N_{sha}}\,(inp)\right) \tag{5.2}$$

Where we have 2 security parameters in this scheme. N_{bc} is work load for bcrypt, determines computation complexity of bcrypt hash function. N_{sha} is number of nested iterations of SHA-256 function.

With this we define key schedule for our new cipher:

$$k_i^r = \begin{cases} hash_{N_{bc},N_{sha}}(key, salt)_i & \text{if } r = 0 \\ hash_{N_{bc},N_{sha}}(k^{r-1} \mid\mid key, salt)_i & \text{otherwise} \end{cases} \tag{5.3}$$

Where

i subscript on right side stands for i-th byte of resulting hash

key is encryption key, 128 bits

k^{r-1} is whole round key for round $r - 1$

$||$ symbol is concatenation of two binary arguments

$salt$ is arbitrary 128 bit salt used in bcrypt algorithm. This can be publicly known - published together with ciphertext or in particular whitebox cipher instance.

Equation for k_i^r is chosen with two primary goals in mind, attacker is not able to:

1. derive encryption key from two (or more) consecutive round keys. This results from infeasability of reversing hash chain. We are also using computational intensive hash function so even brute-forcing is unfeasible.

2. derive round key for $r_1 - 1$ or $r_2 + 1$ if he already have round keys for rounds $[r_1, r_2]$. Unavailability of deriving round key for $r_1 - 1$ results from the previous argument, but here is also important that from already derived round keys we are not able to derive next ones (when compared to standard AES) since it also depends on encryption key directly.

5.2.1 Key bytes for S boxes

In order to increase strength of proposed scheme we don't use round key bytes for S-box computation directly. If someone succeeds in determining this round key bytes by computing proposition 3 from BGE attack for all key bytes possibilities it could help to derive the round keys.

From this reason we use completely different keys for key-dependent S-boxes that in rest of the cipher.

$$k_i^{r\prime} = \begin{cases} hash_{N_{bc},N_{sha}}(key \mid\mid \text{"magicConstant"}, salt)_i & \text{if } r = 0 \\ hash_{N_{bc},N_{sha}}(k^{r-1\prime} \mid\mid key \mid\mid \text{"magicConstant"}, salt)_i & \text{otherwise} \end{cases} \tag{5.4}$$

The equation 5.4 is the same as 5.3 with only difference of concatenation of "magicConstant". This makes two hash chains (1 for round keys, 1 for S-boxes) completely different and non-transformable one to another.

5.3 Diffusion layer modification

In section 3.3.4 was mentioned cipher invertibility. I suggest to extend input/output space of the round function from 32-bits to 128-bits, raising complexity of mentioned inverting attack to $10 \cdot 4 \cdot 2^{128}$ operations. In AES one byte of state array depends only on 4 bytes = one column of state array. Round function of AES acts independently on 4 columns, making it easy to invert it.

Proposed improvement is in changing MDS (Maximum Distance Separable) matrix from 4×4 to 16×16. Then would one byte of state array depend on 16 bytes, making round function 128-bit wide.

MDS matrix acts as diffusion element in the cipher, since our cipher is of type substitution-permutation cipher, our MDS matrix represents invertible linear mapping. The important metric for its security is *branch number* [24], it gives measure on worst case diffusion. If the diffusion matrix has a maximal possible branch number, it is *optimal*. AES [4], Twofish [3] and SHARK [25] ciphers are using MDS matrices optimizing branch number as main security measure of diffusion layer.

For generating such MDS matrices is particularly interesting following proposition from SHARK cipher paper [25] (for proof see original paper).

Proposition 3. *Let C be a $(2n,\ n,\ n+1)$-code over the Galois field $GF(2^m)$. Let G_e be the generator matrix of C in echelon form:*

$$Ge = \begin{bmatrix} I_{n\times n} & B_{n\times n} \end{bmatrix} \tag{5.5}$$

Then C defines an optimal invertible linear mapping γ:

$$\gamma\ :\ GF(2^m)^n \rightarrow GF(2^m)^n = X \mapsto Y = B \cdot X \tag{5.6}$$

Recall that $(2n,\ n,\ n+1)$-code is MDS [1]. Reed-Solomon codes are subset of MDS codes, so their parity check matrix can be used as MDS matrix, in the cipher acting as a strong diffusion element. MDS matrices derived from Reed-Solomon codes are used by many ciphers, for example Twofish, Shark.

In our case we would be interested in $(32,\ 16,\ 17)$-code, to obtain 16×16 MDS matrix with wanted properties.

Another way how to generate MDS matrices with is described in [26] using Cauchy matrices.

It is important to mention that ciphers using MDS matrix as diffusion element usually puts additional requirements on the MDS matrix, also optimizing performance and simplicity in hardware implementation. In blackbox context it is usually security/performance trade off. In whitebox context we need to have diffusion element very strong, so we can neglect performance point of view to increase security.

Also article [27] mentions AES diffusion layer modification from 4×4 to 16×16 MDS matrix, argumenting with stronger security within one round, what is particularly interesting in whitebox context. They are constructing MDS matrix using Cauchy matrices. Cauchy matrices depends on the first row only, this increases possible diversity of 16×16 MDS matrices helping the following idea - key dependent diffusion.

Consider also idea to have key-dependent MDS matrices. If we can generate set S_{MDS} of MDS matrices representing optimal linear mapping, their selection can be based on key-dependent criteria. Set S_{MDS} can be also extended using following proposition from [28].

Proposition 4. *Let $B = [b_{i,j}]_{n \times n}$, $b_{i,j} \in \mathbb{F}_q$ an MDS matrix, for an element $e \in \mathbb{F}_q$, $e \neq 0$, $e \cdot B$ is an MDS matrix.*

Having key-dependent diffusion layer also complicates whitebox attacks, namely the BGE attack [9] and the Generic attack by Michiels [10] requires known MDS matrix coefficients (thus key-independent).

5.4 Analysis

In this chapter we try to analyze suggested scheme improvement from whitebox point of view, particularly we try to mount BGE attack to this modified variant.

S-box definitions are needed in proposition 3 in BGE attack where we obtain 4 affine

1. $(n,\ k,\ d)$-code is MDS iff $d = n - k + 1$

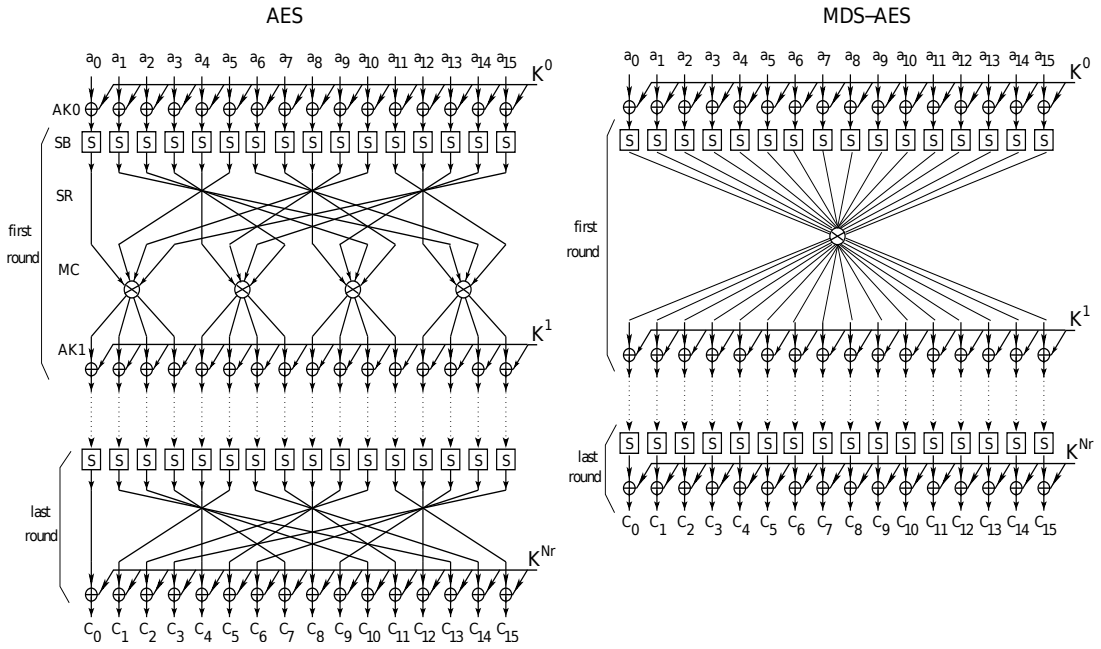

Figure 5.1: Computational diagrams of AES and MDS-AES (taken from [27])

mappings.

$$\widetilde{P}_0 \ : \ x \mapsto \left(S^{-1} \circ \Lambda_{\delta_0} \circ \widetilde{A}_0^{-1}\right)(y_0\,(x, 00, 00, 00)) \tag{5.7a}$$

$$\widetilde{P}_1 \ : \ x \mapsto \left(S^{-1} \circ \Lambda_{\delta_1} \circ \widetilde{A}_0^{-1}\right)(y_0\,(00, x, 00, 00)) \tag{5.7b}$$

$$\widetilde{P}_2 \ : \ x \mapsto \left(S^{-1} \circ \Lambda_{\delta_2} \circ \widetilde{A}_0^{-1}\right)(y_0\,(00, 00, x, 00)) \tag{5.7c}$$

$$\widetilde{P}_3 \ : \ x \mapsto \left(S^{-1} \circ \Lambda_{\delta_3} \circ \widetilde{A}_0^{-1}\right)(y_0\,(00, 00, 00, x)) \tag{5.7d}$$

In our implementation of the BGE attack we iterate over $(\delta_i, c_i)_{i=0,\ldots,3} \in \mathrm{GF}\left(2^8\right) \times \mathrm{GF}\left(2^8\right)$ what gives complexity 2^{16} for one mapping. In each step is mapping checked for affinity in 2^8 steps (for affinity check algorithm see A.5), altogether one relation takes 2^{24} steps, for all relations 2^{26} steps.

Here is the place where we use public knowledge of AES S-Box definitions. One way how to mount BGE attack to this modified variant is to guess also particular S-box mapping for each $\widetilde{P}$ and to test for its affinity.

Equations 5.1 describe Twofish S-boxes. There are 2^{16} possible s_0 S-boxes. One S-box mappings stored as look-up table takes 2^8 bytes. Thus pre-computed s_0 s-box for all possible key bytes would take $2^8 \cdot 2^{16} = 2^{24} > 10^7$ bytes.

Even if attacker determines round keys for S-boxes it will be completely useless for further extraction of cipher key since hash chains are different.

Twofish S-boxes thus increase complexity of proposition 3 from BGE from 2^{24} to 2^{40}. This is still not strong enough, it is highly parallelizable problem. In order to increase work needed to mount proposition 3 attack one could redefine key-dependent S-boxes to increase attack complexity to level 2^{128} what is larger than best known attack on AES [29]. We then would S-box need to depend on 13 bytes derived from encryption key. Upper bound on number of different non-linear S-boxes is $256! \approx 8,5 \cdot 10^{506}$ so there are still options to expand S-box space.

$$s'_j(x) = sboxgen(j, 12, x) \tag{5.8}$$

S-box is generated recursively by *sboxgen* (defined in 5.9), following the idea of nesting fixed permutations with addition of round key:

$$sboxgen(j, l, x) = \begin{cases} q'_{j,0}[x] \oplus k_{j+1} & \text{if } l = 0 \\ q'_{j,l}[sboxgen(j, l-1, x)] \oplus k_{(j+1)\cdot(l+1)} & \text{otherwise} \end{cases} \tag{5.9}$$

Where $q'_{j,l}$ is one of two fixed Twofish 8-bit permutations. Particular sequence of nested permutations requires deeper analysis, to avoid possible weaknesses induced by composing inappropriate permutations together, but we can choose for example:

$$q'_0 = \begin{bmatrix} q_1 & q_0 & q_1 & q_0 & q_1 & q_0 & q_1 & q_0 & q_1 & q_0 & q_1 & q_0 & q_0 \end{bmatrix} \tag{5.10a}$$

$$q'_1 = \begin{bmatrix} q_1 & q_1 & q_0 & q_0 & q_1 & q_1 & q_0 & q_0 & q_1 & q_1 & q_0 & q_0 & q_1 \end{bmatrix} \tag{5.10b}$$

$$q'_2 = \begin{bmatrix} q_0 & q_0 & q_1 & q_1 & q_0 & q_0 & q_1 & q_1 & q_0 & q_0 & q_1 & q_1 & q_0 \end{bmatrix} \tag{5.10c}$$

$$q'_3 = \begin{bmatrix} q_0 & q_0 & q_0 & q_1 & q_1 & q_1 & q_0 & q_0 & q_0 & q_1 & q_1 & q_1 & q_1 \end{bmatrix} \tag{5.10d}$$

We suggest to study linear/affine self-equivalences [19] during analysis of S-boxes generated by this algorithm.

Proposed S-Boxes give us good security margin for the BGE attack, raising computational complexity to 2^{128}. Pre-computed s_0 S-box for all possible key bytes would take $2^8 \cdot 2^{8 \cdot 13} = 2^{112} > 10^{33}$ bytes.

Also as long as are SHA-256 and bcrypt uncracked, key extraction should be unfeasible, since it is not possible to invert hash easily.

Cipher modification that affects only S-boxes and round keys, so other parts of the BGE attack are not affected, namely transforming non-linear parts to affine, propositions 1 and 2 work as before.

Cipher modification don't increase table sizes, because modifications made are only in S-box definitions and key schedule - both parts of the cipher are already evaluated and stored in the look-up tables. Our modifications also don't affect encryption/decryption performance

since the only difference is made during particular whitebox instance (look-up tables) generation. Encryption/decryption algorithm itself is not affected.

5.5 Analysis of diffusion layer modification

Taking also section 5.3 into account will result also in bigger look-up tables. Mainly type 2 tables are affected, previously it was mapping $2^8 \rightarrow 2^{32}$, with cascade of XOR tables to sum $4 \times$ 32-bit values to obtain one column of state array. Now type 2 tables are $2^8 \rightarrow 2^{128}$, with cascade of XOR tables to obtain whole state column vector. Cipher with modified structure uses type 1 tables instead of type 2 tables and cascade of XOR tables is working in the same manner as in external encodings in the first and the last round.

Table 5.2 summarizes changes in table sizes of WB AES. From this is visible that they come at a substantial price (215%) compared to the original whitebox AES implementation.

Type	original			modified		
	# of tables	bit-width	total size	# of tables	bit-width	total size
I	$4 \cdot 4 \cdot 2 = 32$	$8 \rightarrow 128$	131072 B	$4 \cdot 4 \cdot 2 = 32$	$8 \rightarrow 128$	131072 B
II	$4 \cdot 4 \cdot 9 = 144$	$8 \rightarrow 32$	147456 B	$4 \cdot 4 \cdot 9 = 144$	$8 \rightarrow 128$	589824 B
III	$4 \cdot 4 \cdot 9 = 144$	$8 \rightarrow 32$	147456 B	$4 \cdot 4 \cdot 9 = 144$	$8 \rightarrow 32$	147456 B
IV	$8 \cdot 6 \cdot 4 \cdot 9 = 1728$	$8 \rightarrow 4$	221184 B	$8 \cdot 3 \cdot 4 \cdot 9 = 864$	$8 \rightarrow 4$	110592 B
IV	$8 \cdot 4 \cdot 15 \cdot 2 = 960$	$8 \rightarrow 4$	122880 B	$8 \cdot 4 \cdot 15 \cdot 2 = 960$	$8 \rightarrow 4$	122880 B
IV				$8 \cdot 4 \cdot 15 \cdot 9 = 4320$	$8 \rightarrow 4$	552960 B
Total			770048 B 752 kB			1654784 B 1616 k B

Table 5.2: Whitebox implementation size with/without our modifications

From security point of view this modifications also prevents inverting attack mentioned in section 3.3.4. Function is now too wide to be inverted by running over GF $\left(2^8\right)^{16}$.

Key-dependent MDS matrix also prevents mounting the BGE attack. In particular we don't know MixColumn matrix coefficients so we are not able to construct set β from section 3.3 in [9]. This leads to further ambiguities so proposition 2 and 3 won't work anymore. This also makes recovering affine parts of I/O encodings unfeasible, using this attack. But proposition 1 still works, random I/O bijections can be converted of affine ones quite easily.

As noted above, also Generic attack by Michiels [10] requires known MDS matrix coefficients.

5.6 Drawbacks

By modification of AES design we are coming up with new cipher, what brings also some possible problems. AES and Twofish have advantage of being well analyzed from blackbox context and being relatively secure. Designing a new cipher may help with increasing security in whitebox context but there also may be weaknesses in blackbox context. It would be needed to analyze the new cipher from this point of view, for example for resistance to linear or differential cryptanalysis. There is no guarantee that proposed cipher modifications are strong enough to resist classical blackbox context cryptanalysis.

But when designing scheme improvements we tried to follow standard principles in designing secure block cipher, inspired by AES, Twofish, Shark and others.

6 Future work

As a future work I would like to study blackbox properties of suggested improved cipher. In particular it is worth to study S-box construction, to test it for possible weaknesses. It will be needed to model S-boxes as algebraic equations and to test it for fixed points, for example. One can also study this S-boxes and their differential / linear probability coefficients (measuring resistance of an S-box to differential / linear cryptanalysis). One can test S-boxes for Square and linear approximation attacks.

Resistance to the BGE attack results from dependence on 13 round key bytes. If one shows that S-box space is smaller than assumed, it can be vulnerability that can be exploited by whitebox attack.

I would like to examine attacks on whitebox implementations, when each round of the cipher is considered as a single mini-cipher with its own key. Whitebox cipher implementation then would be network of serially connected mini-ciphers. From this point of view I would be interested in resistance of the mini-ciphers to linear and differential cryptanalysis.

7 Conclusion

[TODO]

A Appendix A

A.1 Attachments

CD directory structure, [TODO].

A.2 Hardware specifications

Several performance tests were performed in this thesis, here is detailed description of used hardware.

Component	Specifications	Additional information
CPU	Intel® Core™ i5 M 560 @ 2.67GHz `http://ark.intel.com/products/49653`	64-bit, 4 thread cores 3 MB cache
RAM	2 x 4 GiB SODIMM DDR3 Synchronous 1067 MHz Kinston 9905428-005.A02LF	width: 64 bits

A.3 Squaring matrix

Here is shown how to compute matrix Q from section 4.1.5 that represents squaring operation.

Proposition 5. $\forall a, b \in GF(2^8) \ : \ (a+b)^2 = a^2 + b^2$.

Proof. From binomial theorem, assume general case, $a, b \in \mathrm{GF}(p^m)$, where p is prime.

$$(a+b)^p = a^p + \sum_{k=1}^{p-1} \binom{p}{k} a^{p-k} b^k + b^p = a^p + b^p$$

since $\binom{p}{k}$ is multiple of p for $0 < k < p$ (from binomial coefficient definition) and multiples of p are 0 in $\mathrm{GF}(p^m)$. ($\binom{p}{k} \notin \mathrm{GF}(p^m)$ is coefficient, so it is reduced mod p). To finish proof let $p = 2$. □

Proposition 6. $\forall a, b \in GF(2^8) \ : \ (a+b)^2 = a^2 + b^2 \Rightarrow$ *squaring in* $GF(2^8)$ *is linear operation which is equivalent to matrix multiplication with coefficients from* $GF(2)$.

Proof. Let's have $a \in \mathrm{GF}(2^8)$. Elements from this field are represented as polynomials, in polynomial basis $\begin{bmatrix} x^0 & x^1 & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{bmatrix}$. Thus we can write: $a = \sum_{i=0}^{7} a_i \cdot x^i$ where $a_i \in \mathrm{GF}(2)$. Also note that $a_i = a_i^2$, because $\mathrm{GF}(2) = (\{0,1\}, +, \cdot)$, so $0 = 0^2 \wedge 1 = 1^2$.

Thus we can write

$$a^2 = \left(\sum_{i=0}^{7} a_i \cdot x^i\right)^2 = \sum_{i=0}^{7} a_i^2 \cdot x^{i^2} = \sum_{i=0}^{7} a_i \cdot x^{i^2} = \sum_{i=0}^{7} x^{i^2} \cdot a_i = \left(\sum_{i=0}^{7} x^{i^2}\right) \cdot a = Q \cdot a$$

□

Thus the squaring matrix is Q. It holds that $\mathrm{GF}\left(2^8\right) \simeq \mathrm{GF}\left(2\right)^8$ since finite fields with the same number of elements are isomorphic. We use this isomorphism to obtain matrix Q. It is enough to transform polynomial base in $\mathrm{GF}\left(2^8\right)$ to vector base in $\mathrm{GF}\left(2\right)^8$. It is easy to see that:

$$\begin{bmatrix} x^0 & x^1 & x^2 & x^3 & x^4 & x^5 & x^6 & x^7 \end{bmatrix}_{\mathrm{GF}(2^8)} \mapsto \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{bmatrix}_{\mathrm{GF}(2)^8} = I_{8,\,\mathrm{GF}(2)^8}$$

Where e_i is i-th base vector from standard base.

Now it is obvious how to construct matrix Q:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}_{\mathrm{GF}(2)^8}$$

A.4 Multiplication matrix

In this section is described how to construct a 8×8 matrix $[a]$ with coefficients from $\mathrm{GF}\left(2\right)$ that represents multiplication by constant $a \in \mathrm{GF}\left(2^8\right)$ in $\mathrm{GF}\left(2^8\right)$. It uses the same technique as in section A.3, using isomorphism.

Recall $u_{\mathrm{GF}(2^8)} = \sum_{i=0}^{7} u_i \cdot x^i \mapsto \begin{bmatrix} u_0 & \cdots & u_7 \end{bmatrix}^T_{\mathrm{GF}(2)^8} = \sum_{i=1}^{8} u_{i-1} \cdot e_i$ where $u_i \in \mathrm{GF}\left(2\right)$. Multiplication is linear transformation, so let's denote multiplication by a as $L_a : \mathrm{GF}\left(2\right)^8 \to \mathrm{GF}\left(2\right)^8$. Now from linearity:

$$\begin{aligned} L_a(u) &= L\left(\sum_{i=1}^{8} u_{i-1} \cdot e_i\right) = \sum_{i=1}^{8} u_{i-1} \cdot L_a(e_i) \\ &= \sum_{i=1}^{8} L_a(e_i) \cdot u_{i-1} = \begin{bmatrix} L_a\left(e_1\right) & \ldots & L_a\left(e_8\right) \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ \vdots \\ u_7 \end{bmatrix} \end{aligned}$$

Thus matrix $[a]$ has form:

$$[a] = \Big[L_a(e_1) \quad \dots \quad L_a(e_8) \Big]_{\mathrm{GF}(2)^8}$$

A.5 Affinity check

In this section we describe affinity check needed in proposition 3 of BGE attack. We are given relation $\widetilde{P}$ as a look-up table and the task is to test it for affinity. If $\widetilde{P}$ is affine it must hold:

$$\widetilde{P}(x) = M \times x \oplus c \tag{A.1}$$

For some square matrix M with coefficients from GF(2) and constant $c \in \mathrm{GF}\left(2^8\right)$ (or equivalently 8×1 vector with coefficients from GF(2)).

By evaluating $\widetilde{P}(0) = c$ we obtain affine constant c so we derive new mapping $\widetilde{P'}$, reducing the problem to test $\widetilde{P'}$ for being linear.

$$\widetilde{P'}(x) = \widetilde{P}(x) \oplus c \tag{A.2}$$

And from linearity the following formula must hold:

$$\forall x \in \mathrm{GF}\left(2^8\right),\ \exists!\ k_j \in \{0,1\},\ j \in [0,7]\ :\ x = \sum_{j=0}^{7} k_j \cdot \widetilde{P'}(e_j) \tag{A.3}$$

It says that each element from the field has to be unique sum of its basis vectors. Assuming that $\widetilde{P'}$ is linear, we can obtain mapped base vectors for this transformation easily as $g_j = \widetilde{P'}(e_j)$. Now it is visible that time complexity is 2^8.

Algorithm 2 Algorithm for testing given mapping for being affine

1: **function** ISAFFINE($\widetilde{P} : \mathrm{GF}\left(2^8\right) \mapsto \mathrm{GF}\left(2^8\right)$) ▷ Determine if P is affine mapping
2: $c \leftarrow \widetilde{P}[0]$ ▷ c is affine constant
3: $\widetilde{P'}[x] \leftarrow \widetilde{P}[x] + c$ ▷ 2^8 time complexity
4: $isAffine \leftarrow true$
5: **for** $x \leftarrow 0, (2^8 - 1)$ **do**
6: $px \leftarrow \widetilde{P'}[x]$
7: $cx \leftarrow 0$
8: **for** $i \leftarrow 0, 7$ **do**
9: **if** $x_i = 1$ **then** ▷ x_i is i-th bit of x in binary
10: $cx \leftarrow cx \oplus \widetilde{P'}\left[e_i\right]$ ▷ $\widetilde{P'}\left[e_i\right]$ is mapped base vector
11: **end if**
12: **end for**
13: **if** $px \neq cx$ **then** ▷ cx is expressed via mapped base vectors
14: $isAffine \leftarrow false$
15: **return**
16: **end if**
17: **end for** ▷ All elements from field checked for linearity
18: **return** $isAffine$
19: **end function**

Bibliography

[1] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot. White-box cryptography and an aes implementation. In *Proceedings of the Ninth Workshop on Selected Areas in Cryptography (SAC 2002*, pages 250–270. Springer-Verlag, 2002.

[2] Tim Kerins and Klaus Kursawe. A cautionary note on weak implementations of block ciphers. In *In 1st Benelux Workshop on Information and System Security (WISSec 2006*, page 12, 2006.

[3] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. In *in First Advanced Encryption Standard (AES) Conference*, 1998.

[4] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES—the advanced encryption standard*. Springer-Verlag, 2002. ISBN 3–540–42580–2.

[5] Brecht Wyseur. White-box cryptography: Hiding keys in software. [online], 2012. URL `http://whiteboxcrypto.com/files/2012_misc.pdf`.

[6] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot. A white-box des implementation for drm applications. In *In Proceedings of ACM CCS-9 Workshop DRM*, pages 1–15. Springer, 2002.

[7] Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an obfuscated cipher by injecting faults. In Joan Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2002. ISBN 3-540-40410-4. URL `http://dblp.uni-trier.de/db/conf/ccs/ccsdrm2002.html#JacobBF02`.

[8] Hamilton E. Link and William D. Neumann. Clarifying obfuscation: Improving the security of white-box des. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I - Volume 01*, ITCC '05, pages 679–684, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2315-3. doi: 10.1109/ITCC.2005.100. URL `http://dx.doi.org/10.1109/ITCC.2005.100`.

[9] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box aes implementation. In *Proceedings of the 11th international conference on Selected Areas in Cryptography*, SAC'04, pages 227–240, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-24327-5, 978-3-540-24327-4. doi: 10.1007/978-3-540-30564-4_16. URL `http://dx.doi.org/10.1007/978-3-540-30564-4_16`.

[10] W. Michiels and P. Gorissen. Mechanism for software tamper resistance: an application of white-box cryptography. In *Proceedings of the 2007 ACM workshop on Digital Rights Management*, DRM '07, pages 82–89, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-884-8. doi: 10.1145/1314276.1314291. URL http://doi.acm.org/10.1145/1314276.1314291.

[11] Yaying Xiao and Xuejia Lai. A secure implementation of white-box aes. In *Computer Science and its Applications, 2009. CSA '09. 2nd International Conference on*, pages 1–6, 2009. doi: 10.1109/CSA.2009.5404239. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05404239.

[12] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the xiao - lai white-box aes implementation. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2012. ISBN 978-3-642-35999-6. URL http://dblp.uni-trier.de/db/conf/sacrypt/sacrypt2012.html#MulderRP12.

[13] Yoni De Mulder, Brecht Wyseur, and Bart Preneel. Cryptanalysis of a perturbated white-box aes implementation. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT*, volume 6498 of *Lecture Notes in Computer Science*, pages 292–310. Springer, 2010. ISBN 978-3-642-17400-1. URL http://dblp.uni-trier.de/db/conf/indocrypt/indocrypt2010.html#MulderWP10.

[14] Mohamed Karroumi. Protecting white-box aes with dual ciphers. In *Proceedings of the 13th international conference on Information security and cryptology*, ICISC'10, pages 278–291, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-24208-3. URL http://dl.acm.org/citation.cfm?id=2041036.2041060.

[15] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal, Vol 28, pp. 656–715*, Oktober 1949.

[16] James A. Muir. A tutorial on white-box aes, 2011. URL http://eprint.iacr.org/2013/104.pdf.

[17] James Xiao and Yongxin Zhou. Generating large non-singular matrices over an arbitrary field with blocks of full rank. *IACR Cryptology ePrint Archive*, 2002:96, 2002. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2002.html#XiaoZ02.

[18] Brecht Wyseur. *White-Box Cryptography*. Phd. thesis, Katholieke Universiteit Leuven, 2009. URL http://docserver.bis.uni-oldenburg.de/publikationen/dissertation/2001/aneacr01/aneacr01.html.

[19] Alex Biryukov, Christophe De Cannièere, An Braeken, and Bart Preneel. A toolbox for cryptanalysis: linear and affine equivalence algorithms. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, pages 33–50, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-14039-5. URL http://dl.acm.org/citation.cfm?id=1766171.1766175.

[20] Niels Provos and David Mazieres. A future-adaptable password scheme. In *In Proceedings of the 1999 USENIX, Freenix track (the on-line version)*, page 99, 1999.

[21] Colin Percival. Stronger key derivation via sequential memory-hard functions, 2009. URL http://www.daemonology.net/papers/scrypt.pdf.

[22] Butterfly labs. 1,500 GH/s Bitcoin Miner. https://products.butterflylabs.com/homepage/1500gh-bitcoin-miner.html, 2013. [Online; accessed 15-May-2013].

[23] Jeremi M. Gosney. Password cracking hpc. Passwords ^12 Security Conference, [online], December 2012. URL http://passwords12.at.ifi.uio.no/Jeremi_Gosney_Password_Cracking_HPC_Passwords12.pdf.

[24] Joan Daemen. *Cipher and hash function design. Strategies based on linear and differential cryptanalysis.* Phd. thesis, Katholieke Universiteit Leuven, 1995. URL http://docserver.bis.uni-oldenburg.de/publikationen/dissertation/2001/aneacr01/aneacr01.html.

[25] Vincent Rijmen, Joan Daemen, Bart Preneel, Antoon Bosselaers, and Erik De Win. The cipher shark. In Dieter Gollmann, editor, *FSE*, volume 1039 of *Lecture Notes in Computer Science*, pages 99–111. Springer, 1996. ISBN 3-540-60865-6. URL http://dblp.uni-trier.de/db/conf/fse/fse96.html#RijmenDPBW96.

[26] Ron M Roth and Gadiel Seroussi. On generator matrices of mds codes. *IEEE Trans. Inf. Theor.*, 31(6):826–830, nov 1985. ISSN 0018-9448. doi: 10.1109/TIT.1985.1057113. URL http://dx.doi.org/10.1109/TIT.1985.1057113.

[27] Jorge Nakahara Jr. and Elcio Abrahao. A new involutory mds matrix for the aes. *I. J. Network Security*, 9(2):109–116, 2009. URL http://dblp.uni-trier.de/db/journals/ijnsec/ijnsec9.html#JrA09.

[28] Muhammad Yasir Malik and Jong-Seon No. Dynamic mds matrices for substantial cryptographic strength. *IACR Cryptology ePrint Archive*, 2011:177, 2011. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2011.html#MalikN11.

[29] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full aes. Cryptology ePrint Archive, Report 2011/449, 2011. http://eprint.iacr.org/.