

Dynamic Injection Model

Mohamed Abo El-Fotouh and Klaus Diepold
 Institute for Data Processing (LDV)
 Technische Universität München (TUM)
 80333 München, Germany

[mohamed, kldi]@tum.de

Abstract

In this paper, we present the Dynamic Injection Model (DIM) and its variant the Static Injection Model (SIM). DIM is a model that allows any iterative block cipher to accept a variable length secondary key, this is achieved by injecting the block cipher with keyed transformation functions. A transformation function can vary in strength from a simple xor function to a complete block cipher. The transformation functions modify the input or the output of some (all) round functions of that cipher using the secondary key. SIM is a variant of DIM, where the transformation functions and the secondary key length are determined in the design time. We used the Advanced Encryption Standard (AES) to demonstrate the usage of DIM and SIM models.

Keywords—AES, DIM, SIM, multiple keys encryption.

I. Introduction

USUALLY a block cipher accepts the plaintext and the encryption key to produce the ciphertext. Modes of operations have introduced other inputs, some of these modes use initial vectors like in CBC, CFB and OFB modes [1], counters like in Counter mode [2] or nonces like in OCB mode [3]. The idea of using an extra input to change the way the block cipher behaves, was suggested in HPC [4] and used in Mercy [5]. In [6], the formal definition of tweakable block ciphers has been introduced, since then a lot of tweakable block ciphers modes of operations have been developed (XTS, EME, CMC, XCB, TET [7], [8], [9], [10], [11]) are to name a few. Most of these modes are dedicated to the disk encryption applications. All these modes of operations use *effective* tweaks of fixed length.

The extra input to the block cipher, which we named

secondary key, is either introduced to cheaply increase the security like in DESX [12] or to cheaply select a different permutation when the same key is used like in XTS [7] mode of operation. The main idea of having a secondary key, is to change the block ciphers behavior without the need to change the primary key, which is usually associated with an expensive key scheduling routine. In this paper, the encryption key is divided to a primary key and a secondary key, this means that the secondary key is a part of the encryption key and must kept secret.

In this paper, we present the Dynamic Injection Model (DIM) and its variant the Static Injection Model (SIM). DIM is a model that allows any iterative block cipher to accept variable length secondary key. DIM achieves this by injecting the block cipher with keyed transformation functions. These transformation functions can vary in strength from simple xor functions to complete block ciphers. The transformation functions modify the input or the output of some round functions of the block cipher using the secondary key. SIM is a variant of DIM, where the transformation functions and the secondary key length are determined in the design time. We used the Advanced Encryption Standard (AES) [13] to demonstrate the usage of DIM and SIM models. In [14], the Dynamic Substitution Model (DSM) and the Static Substitution Model (SSM) were presented, DIM and SIM models are considered extensions to these models.

This paper is structured as follows. In Section 2 we present the terminologies used in this paper. In Section 3 we present the DSM and SSM models. In Section 4 we present the DIM model. In Section 5 we present DI-AES, which is a variant of the AES based on the DIM model. In Section 6 we present the SIM model and two variants of AES. In Section 7 we propose an enhanced key scheduling for AES. In Section 8 we present security analysis of DIM/SIM models and the proposed variants. We conclude in Section 9.

II. Terminologies

These terminologies are used to describe the encryption models in this paper:

P: is the input plaintext.

E: is a block cipher.

EK: is the encryption expanded key of **E**.

m:

- in DSM/SSM models is the number of words in EK that will be replaced.
- in DIM/SIM models is the number of the transformation functions.

SK:

- in DSM/SSM models is the secondary key.
- in DIM/SIM models is the secondary key, it is structured as an array that holds the keys of the injected transformation functions.

I:

- in DSM/SSM models is an array of indices that determine which words of **EK** will be replaced.
- in DIM/SIM models is an array of indices that determine where the **F[m]** functions should be injected. Note that the values of **I** should be unique and should be sorted in an ascending order.

F: is an array that holds **m** keyed transformation functions used in encryption process.

C=**E**_{EK}(**P**): encrypts **P** with the block cipher **E**, using **EK** as the encryption expanded key and returns the result.

F_{*i*}(**X**,**K**): applies the *i*th transformation function on **X** using **K** as its key and returns the result.

R_{*i*}(**X**,**EK**): applies the *i*th round of **E** on **X** using the proper subkey from **EK** and returns the result, where the proper subkey is the part of the expanded key that is consumed by the *i*th round function.

C: is the output ciphertext.

III. Dynamic Substitution Model (DSM)

DSM is a model that can provide a block cipher with variable length secondary key. The secondary key is used to replace some words of the cipher's expanded key. Two extra inputs are introduced by DSM to the normal encryption model. The first input (**SK**) is the secondary key that will substitute some words in the expanded key of the cipher, the second input (**I**) holds the indices which specify which words in the expanded key will be substituted, note that the substituted words can be of any size.

The listing of Encrypt-DSM, which is used for encryption in DSM, is in table I.

Encrypt-DSM executes as follows:

- The words of the secondary key **SK** replace certain words in **EK** (determined by the index array **I**).

TABLE I. Encrypt-DSM function.

```
Encrypt-DSM(P, E, EK, SK[m], I[m])
  for i=0 to m-1
    EK[I[i]] = SK[i]
  end for
  C = EEK(P)
  return C
```

TABLE II. Encrypt-DIM function.

```
Encrypt-DIM(P, E, EK, F[m], SK[m], I[m])
X = P
t = 0
If (I[0] == 0) then
  X = F0(X, SK0)
  t++
End if

For i=1 to n
  X = Ri(X, EK)
  If (I[t] == i) then
    X = Ft(X, SKt)
    t++
  End if
End for
return X
```

- The updated **EK** is used as the new expanded key for the encryption function, that encrypts the input plaintext (**P**) to produce the ciphertext (**C**).

Static Substitution Model (SSM) is a variant of DSM, where it uses a fixed length secondary key. It is a special case of DSM, where the secondary key length and the index array **I** are determined in the design time. SSM can build more efficient implementations than DSM.

IV. Dynamic Injection Model (DIM)

DIM is a model that can provide a block cipher with variable length secondary key. The secondary key is consumed by some keyed transformation functions that is injected into the cipher. A transformation function can vary in strength from a simple xor function to a complete cipher. The transformation functions modify the input and/or the output of some of the cipher's round functions (using the secondary key as their key). Three extra inputs are introduced by DIM model to the normal encryption model. The first input (**F**) is the transformation functions, the second input (**SK**) holds the secondary key that will be consumed by the transformation functions, the third input (**I**) holds the indices, which determine where the transformation functions should be injected.

The listing of Encrypt-DIM, which is used for encryption in our proposed model, can be found in table II.

Encrypt-DIM executes as follows:

- 1) Before applying the first round of the cipher, the plaintext P is copied to the text X and Encrypt-DIM checks if the first transformation function should update the text X and if this is the case X is modified using the first transformation function and its corresponding part of the secondary key.
- 2) Encrypt-DIM applies a round function of E (using the proper round key from the expanded encryption key EK) and updates X , where the proper round key is the part of the expanded key that is consumed by this round function.
- 3) Encrypt-DIM checks if the next transformation function should update the text X and if this is the case X is updated.
- 4) The last two steps are repeated another $n-1$ times (where n is the number of rounds of the cipher E).
- 5) Encrypt-DIM returns X as the ciphertext.

A. Transformation functions

A transformation functions can vary from strength from a simple xor function to a complete cipher. But care should be considered, as these function are supposed to add strength to the cipher or in the modest case not to weaken that cipher. So functions that reverse the effect of the cipher's round functions (or even a part of it) should be avoided.

The total number of bits consumed by these functions is the secondary key length. Each transformation function can modify the entire input or only a part of it.

Modifying the same input more than once is also possible, as more than one function can be injected in the same place, in this case they are treated as one function. But care should be taken, that these functions do not cancel any part of the original cipher's round functions nor form a group (e.g. injecting two xor functions with $K1$ and $K2$ as their key, is equivalent to injecting only one xor function with $K3$, where $K3 = K1 \oplus K2$).

B. Pros of DIM

Generality: DIM can be applied to any iterative block cipher.

High throughput: DIM can build ciphers' variants with high-speed, when high-speed functions like xor are used as the transformation functions.

Low memory consumption: Only the secondary key is needed to be stored pro instantiation, when the primary key is shared among n instances like in [15], [16] (if the transformation functions do not require key expansion, which is assumed).

TABLE III. DI-AES encryption function.

```

Encrypt-DI-AES (P, EK, F [m], SK [m], I [m])
X=P
t=0
If (I[0]==0) then
    X= F0(X, SK0)
    t++
End if

X= xor(X, EK)
For i=1 to n - 1
    If (I[t]==i) then
        X= Ft(X, SKt)
        t++
    End if
    X= round(X, EK + 4 × i)
End for

If (I[t]==n-1) then
    X= Ft(X, SKt)
    t++
End if

X= final(X, EK + 4 × (n-1))
If (I[t]==n) then
    X= Ft(X, SKt)
End if
return X

```

Security: DIM can build more secure variants of the cipher (e.g. by increasing the number of rounds of that cipher).

V. DI-AES

In order to demonstrate the DIM model, we present the Dynamic Injection AES (DI-AES). The listing of the DI-AES encryption function is found in table III, where:

- n is the number of rounds: typically 14, 12 and 10 for the primary key of size 256-bit, 192-bit and 128-bit respectively.
- $\text{xor}(X, Y)$: xors X with the first 128-bit of Y and returns the result.
- $\text{round}(X, Y)$: performs an AES round function on X using Y as the key and returns the result.
- $\text{final}(X, Y)$: performs an AES final round function on X using Y as the key and returns the result.
- EK is the expanded AES encryption key, and is an array of 32-bit words.

After presenting DI-AES, let us have a look at some examples for its use :

- 1) Let us suppose that we need the secondary key to be 128-bit, the direct way is to inject one function that accepts 128-bit key (e.g. an AES round function or an xor function) in the middle of the cipher (i.e. $I[0]=$

TABLE IV. AESI1 and AESI2 encrypt functions.

<pre> Encrypt-AESI1(X,EK,SK) X=P xor(X,EK) for i=1 to 7 round(X,EK+4 × i) end for round(X,SK) for i=8 to 14 round(X,EK+4 × i) end for final(X,EK+14 × 4) return X </pre>	<pre> Encrypt-AESI2(X,EK,SK) X=P xor(X,EK) for i=1 to 5 round(X,EK+4 × i) end for xor(X,SK) for i=6 to 10 round(X,EK+4 × i) end for xor(X,SK+4) for i=11 to 13 round(X,EK+4 × i) end for final(X,EK+14 × 4) return X </pre>
---	---

$n/2$). Another solution is to inject two functions, each consumes 64-bit in two different positions.

- 2) To construct AESX (DESX [12] like cipher), the secondary key should be 256-bit and two xor functions should be injected, one at the beginning and other at the end (i.e. $I[0]=0$ and $I[1]=n$).
- 3) Let us suppose that for a certain application 140 bits as a secondary key is needed, one way is to inject the DI-AES with one AES round (e.g. $I[0]=n/2$) and inject an xor function (that modifies only 12-bit) in another place (e.g. $I[1]=n/3$).

VI. Static Injection Model (SIM)

SIM uses a fixed length secondary key. It is a special case of DIM, where the keyed transformation functions and their positions are determined in the design time. SIM can build more efficient implementations than DIM. We used the SIM mode to construct two variants of the AES-256. We named this variant AESI1 and AESI2.

AESI1 accepts 256-bit primary key and 128-bit secondary key. In AESI1, we injected the AES with an extra AES round function. Our choice was to inject AES after the 7th round. AESI2 accepts 256-bit primary key and 256-bit secondary key. In AESI2, we injected the AES with two xor function. Our choice was to inject AES after the 5th and 10th rounds.

Table IV presents the encrypt functions of AESI1 and AESI2, where SK is the secondary key.

VII. Enhanced AES Key Scheduling

To simplify our analysis and to enhance the security of DSM and SSM, we propose a modified key schedule

to AES that eliminate related-key attacks along with some other attacks on AES.

Compared to the cipher itself, the AES key schedule appears to be more of an ad-hoc design. It has a much slower diffusion structure than the cipher, and contains relatively few non-linear elements [17].

To be able to compute the rounds key on the fly, Rijndael's key scheduling has an interesting property, that is if you have the subkeys of a round you can calculate all the other round keys and retrieve the master key. Although this property increase the key agility of Rijndael, it has been used in many theoretical and side channel attacks. For example in the square attack [18], by recovering the round keys of the last round, one can recover all the other round keys including the encryption key. In [19], [20], [21] cache timing attacks have been used to recover either the first or the last round subkeys, that were used to extract the encryption key.

In [22], the authors introduced a new classification scheme for iterative block ciphers based on their key schedules. In essence, this scheme creates two categories of ciphers based on whether or not knowledge of a round subkey generated by the key schedule reveals any information about other round subkeys or the master key. Those that do, fall into *Category 1* and those that do not, fall into *Category 2*. Each of these categories is further subdivided into three Types A, B and C.

In [23], the key schedule of AES candidates were classified. Rijndael's key schedule was classified to be 1C, where the knowledge of a round subkey yields bits of other round subkeys or the master key after some simple arithmetic operations or function inversions.

Our goal is to increase the security of AES's key scheduling, by increasing its classification number, namely to a 2B, where all master key bits are used in the determination of all round subkeys, thus maximizing the entropy of the subkeys and by knowing one of the round keys no information about any other round key or the master key can be extracted.

Our idea is to use the current AES implementation to increase the security of AES's key scheduling, precisely we propose to use AES in counter mode [2] to generate AES's expanded key. The listing of our proposal is in table V, where *inc32* function increments the least significant 32-bit of the input and **Counter** is a 128-bit block. The decryption expanded key can be generated by applying InvMixColumn to all Round Keys except the first and the last one, if the equivalent inverse cipher construction [24] is used for decryption.

It is straightforward to see that the security of our proposed key schedule is inherited of that of the AES. In other words, if there is a method to calculate a round key/encryption key from another round key, this means that

TABLE V. Our proposed AES's key scheduling.

```

Enhanced-Expanded-Key (EKey)
ExKey=Expand-Key (EKey)
Counter= 0
for i=0 to n-1
    OUTi=Encrypt-AES (ExKey, Counter)
    inc32 (Counter)
end for
return OUT

```

the AES is broken. Our proposed key schedule protects the current AES implementation from many attacks like related key attacks [25] and cache timing attacks [19], [20], [21]. It is worth to mention that this method increase the time complexity of many attacks, even the exhaustive search, in the sense that to try a key 11 to 15 encryption steps are needed.

The only drawback of our proposed solution is its performance, as for example to generate the expanded keys of AES-128 the AES encryption function should be called 11 times, and for AES-256 it should be called 15 times. So this method is not suitable for the applications that uses on-the-fly subkeys computations, and is more suitable to the applications that does not change the expanded key a lot (e.g. for DIM and SIM models, where the primary key has a long lifetime).

In the rest of this paper, we assume that the Enhanced AES Key Scheduling is used instead of the original AES key scheduling.

VIII. Security analysis

A. Definitions

We refer to a modified cipher that uses our proposed models as a two keys block cipher, where (1) presents encrypting the plaintext P using the block cipher B , where PK is the primary key and SK is the secondary key to produce the ciphertext C . The decryption process is represented by (2).

$$C = B(PK, SK)(P) \quad (1)$$

$$P = B^{-1}(PK, SK)(C) \quad (2)$$

Variability is the ability of a two keys block cipher to vary its output depending on its secondary key, more formally:

Variability: For a two keys block cipher if $C1=B(PK,SK1)(P)$ and $C2=B(PK,SK2)(P)$ then $C1 \neq C2$ as long as $SK1 \neq SK2$.

Strong variability is the ability of a two keys block cipher to protect its secondary key, in the sense that by knowing

a set of plaintext/ciphertext pairs, it is not easy to recover neither the secondary key nor the primary key, more formally:

Strong Variability: By encrypting the same plaintext with different instances of a two keys block cipher with the same primary key and different secondary keys, it is hard to recover the secondary keys (totally or partially).

Differentiability is the ability to have strong variability, even if the primary key is shared, more formally:

Differentiability: In the differentiability model, the primary key is shared among (m) parties and each party has its unique secondary key. A two keyed block cipher $B(PK,SK)$ offers differentiability, if by knowing the secondary key of a party, the secondary keys of the other parties can not be easily recovered.

In the following text, we provide the guidelines to construct secure two keys block ciphers using DIM. The same guidelines apply to SIM, as it is a special case of DIM.

1) *The guidelines for single injection:* We present the guidelines to inject one transformation function using DIM model, where $m=1$ and $I[0]=x$. This construction can be used to provide variability and strong variability.

Here are the guidelines:

- *The secondary key MUST be unique, random and not predictable by the attacker*, using a secure cipher like AES in counter mode [2], to generate secondary keys, can archive this at a low cost.
- *The secondary key MUST NOT be controlled by the attacker*, by controlling the values of the secondary key, the attacker can introduce a difference in the middle of the cipher, which can lead to new attacks (e.g. the attacks in [26], [27]).
- *The transformation function should not weaken the cipher*. For example if a transformation function is an inverse of the ciphers round function, and the secondary key inverts the effect of a round function (totally or partially), this will weaken the construction and should not be used.
- $\mu \leq x \leq n - \mu$, where $\mu = \max(l, d)$, l =the number of rounds the cipher is resistant to linear cryptanalysis [28] and d is the number of rounds the cipher is resistant to differential cryptanalysis [29]. This constrain is to avoid a successful attack using linear/differential cryptanalysis, where a linear/differential propagation is avoided.
- *Full confusion and full diffusion are expected in both the encryption and decryption directions*. This to provide strong variability.
- $x < n - \nu$, where ν =the number of rounds the cipher is resistant to chosen plaintext attacks. If the attacker

can encrypt the same plaintext with the same primary key and different secondary keys, she can introduce a difference in the intermediate encrypted text, this difference can be used in a chosen plaintext attack (if the attacker can choose/find secondary keys to share a certain pattern and mount a chosen plaintext attack).

- $x > \theta$, where θ =the number of rounds the cipher is resistant to chosen ciphertext attacks. If the attacker can decrypt the same ciphertext with the same primary key and different secondary keys, she can introduce a difference in the intermediate decrypted text, this difference can be used in a chosen ciphertext attack (if the attacker can choose/find secondary keys to share a certain pattern and mount a chosen ciphertext attack).
- *The injected function should modify all the block.* Modifying a partial block increases the probability of certain attacks, like Square attack [18], where the attacker should find intermediate texts that vary in certain positions and are constant in the other positions.

2) *The guidelines for double injection:*

We present the guidelines to inject two transformation functions using DIM model, where $m=2$, $I[0]=x$ and $I[1]=y$. This construction can be used to provide differentiability together with strong variability.

Here are the guidelines:

- *The two secondary keys MUST be unique, random and not predictable by the attacker*, using a secure cipher like AES in counter mode [2] can archive this at a low cost.
- *The secondary keys MUST NOT be controlled by the attacker*, by controlling the values of the secondary key, the attacker can introduce a difference in the middle of the cipher, which can lead to new attacks (e.g. the attacks in [26], [27]).
- *The transformation functions should not weaken the cipher.* For example if a transformation function is an inverse of the ciphers round function, and the secondary key inverts the effect of the last round function (totally or partially), this will weaken the construction and should not be used.
- *Full confusion and full diffusion are expected after injecting the first function.* As the keys consumed by the injection functions are unique and random, any difference in the first secondary keys will be destroyed before applying the second injection function, the same in decryption direction.
- *The injected functions should modify all the block.* Modifying a partial block increases the probability of certain attacks, like Square attack [18], where the attacker should find intermediate texts that vary in certain positions and are constant in the other positions.

Note that, if more than 2 round keys are needed to be injected, the above guidelines apply to the first and last rounds, where $I[0]=x$ and $I[m-1]=y$.

B. Facts about AES

- 1) μ (defined in Sect. VIII-A1=4) [30].
- 2) AES need only 4 rounds to achieve full confusion and full diffusion properties [31].
- 3) No feasible chosen plaintext attack, exists to attack more than 6 rounds.

C. AESI1

For an attacker that does not know either the primary key and the secondary key, she tries to attack an AES-256 with an extra round. AESI1 follows the guidelines in Sect. VIII-A1. After the injection, 7 rounds of the AES are performed, these rounds assures that not only full confusion and diffusion after the injection are achieved, but also that any differential and linear propagation is completely destroyed. To our knowledge there is no feasible 7 round chosen plaintext attack exists on AES, and by using random secondary keys the probability of finding secondary keys that share a certain pattern is considered very low, the same applies to chosen ciphertext attacks were there is no known 7 round ciphertext attack on AES. Thus, the attacker can not mount linear, differential, chosen plaintext or chosen ciphertext attacks and AESI1 is considered secure.

D. AESI2

For an attacker that does not know either the primary key and the secondary key, she tries to attack a full round AES-256. AESI2 follows the guidelines in Sect. VIII-A2. After the first injection, 9 rounds of the AES are performed (when encrypting), these rounds assures that not only full confusion and diffusion after the injection are achieved, but also that any differential and linear propagation is completely destroyed in the encryption direction (as both the secondary keys are unique and random). After the second injection, 10 inverse rounds of the AES are performed (when decrypting), these rounds assures that not only full confusion and diffusion after the injection are achieved, but also that any differential and linear propagation is completely destroyed in the decryption direction as both the secondary keys are unique and random). To our knowledge there is no feasible 9 round chosen plaintext attack exists on AES, and by using random secondary keys the probability of finding secondary keys that share a certain pattern is considered very low, the same applies to chosen ciphertext attacks were there is no known 10

round ciphertext attack on AES. Thus, the attacker can not mount linear, differential, chosen plaintext or chosen ciphertext attacks and AES2I is considered secure.

AES2I offers differentiability, so when the primary key is shared among m parties, recovering the secondary of another party is considered hard. There are two kind of attackers on AES2I:

- 1) An outside attacker **A** that watches the ciphertext. For an external attacker (that does not possess the primary key), she needs to attack AES-256 with two random subkeys.
- 2) An inside attacker **B**, that posses the primary key:
 - a) The attacker decrypts the known ciphertext with the known primary key until the 10^{th} round to produce the intermediate state.
 - b) By knowing the input to AES2I, the attacker can reduce AES2S to an Even-Mansour construction [32], where K1 and K2 are the key and the reduced AES (5 rounds) is considered as the (Pseudo)random permutation.
 - c) The security of Even-Mansour is:
 - i) About 2^{255} , using exhaustive search over the key space (K1 and K2 are both of size 128-bits), which is considered large enough by todays standards.
 - ii) Daemen demonstrated in [33] that a known plaintext attack, will take on average 2^{127} calculations, which has the same complexity as attacking AES with 128-bits key, which is considered secure with todays technology.
 - iii) Daemen also demonstrated in [33] that a chosen plaintext attack, will take on average 2^{64} calculations using 2^{64} stored blocks.
 - iv) Biryukov-Wagner demonstrated in [34], that a "sliding with a twist" attack allows an adversary to recover the key using $\sqrt{2} \times 2^{64}$ known plaintexts and $\sqrt{2} \times 2^{64}$ work.

E. Applications and Recommendations

The DIM/SIM models appeal for applications that require tweakable block ciphers, like disk encryption applications. Any variant of a block cipher constructed using DIM/SIM model, should be analyzed in the used scenario/application.

IX. Conclusion

In this paper, we present the Dynamic Injection Model (DIM), that allows any iterative block cipher to accept variable length secondary key, this is achieved by injecting

the block cipher with keyed transformation functions. Then we presented the Static Injection Model (SIM), which can be implemented more efficiently. We used the SIM model to construct two variant of the AES. We also proposed an enhanced key scheduling for AES. We provided guidelines to construct secure two keys block ciphers using DIM/SIM models.

References

- [1] A. Menezes, P. Van Oorschot., and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [2] D. McGrew. Counter Mode Security: Analysis and Recommendations. <http://citeseer.ist.psu.edu/mcgrew02counter.html>, 2002.
- [3] P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3), 2003.
- [4] R. Schroepel. The Hasty Pudding cipher. The first AES conference, NIST, 1998.
- [5] P. Crowley. Mercy: a fast large block cipher for disk sector encryption. In *Bruce Schneier, editor, Fast Software Encryption: 7th International Workshop, FSE 2000*, 2001.
- [6] M. Liskov, R. Rivest, and D. Wagner. Tweakable Block Ciphers. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, 2002.
- [7] P. Rogaway. Efficient Instantiations of Tweakable Block ciphers and Refinements to Modes OCB and PMAC. <http://citeseer.ist.psu.edu/rogaway03efficient.html>, 2003.
- [8] S. Halevi and P. Rogaway. A parallelizable enciphering mode. Cryptology ePrint Archive, Report 2003/147.
- [9] S. Halevi and P. Rogaway. A tweakable enciphering mode. Cryptology ePrint Archive, Report 2003/148.
- [10] D. McGrew and S. Fluhrer. The Extended Codebook (XCB) Mode of Operation. Cryptology ePrint Archive, Report 2004/278, 2004.
- [11] S. Halevi. Invertible Universal Hashing and the TET Encryption Mode. Cryptology ePrint Archive, Report 2007/014, 2007.
- [12] P. Rogaway. The Security of DESX. *RSA Laboratories Cryptobytes*, Vol. 2, No. 2, 1996.
- [13] NIST. Announcing the ADVANCED ENCRYPTION STANDARD (AES). Technical Report 197, Federal Information Processing Standards Publication, 2001.
- [14] M. El-Fotouh and K. Diepold. Dynamic Substitution Model. In *The Fourth International Conference on Information Assurance and Security (IAS'08)*, Naples, Italy, September 2008.
- [15] M. El-Fotouh and K. Diepold. Galois Substitution Counter Mode (GSCM). In *International Workshop on Security and Privacy in Enterprise Computing (InSPEC 2008) in conjunction with the 12th IEEE International EDOC Conference (EDOC 2008)*, Munich, Germany, September 2008.
- [16] M. El-Fotouh and K. Diepold. A Fast Encryption Scheme for Networks Applications. In *SECURITY 2008*, Porto, Portugal, July 2008.
- [17] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting. Improved Cryptanalysis of Rijndael. In *FSE '00: Proceedings of the 7th International Workshop on Fast Software Encryption*, 2001.
- [18] J. Daemen, L. Knudsen, and V. Rijmen. The Block Cipher Square. In *FSE '97: Proceedings of the 4th International Workshop on Fast Software Encryption*, 1997.
- [19] M. Neve, J. Seifert, and Z. Wang. A refined look at Bernstein's AES side-channel analysis. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 2006.
- [20] M. Neve and J. Seifert. Advances on access-driven cache attacks on AES. In *SAC 2006*, 2006.

- [21] J. Bonneau and I. Mironov. Cache-collision timing attacks against AES. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems—CHES 2006*, October 2006.
- [22] G. Carter, E. Dawson, and L. Nielsen. Key schedules of iterative block ciphers. In *ACISP '98: Proceedings of the Third Australasian Conference on Information Security and Privacy*, 1998.
- [23] G. Carter, E. Dawson, and L. Nielsen. Key schedule classification of the AES candidates. <http://citeseer.ist.psu.edu/carter99key.html>, 1999.
- [24] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., 2002.
- [25] E. Biham. New types of cryptanalytic attacks using related keys. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, 1994.
- [26] M. El-Fotouh and K. Diepold. Cryptanalysis of Substitution Cipher Chaining mode (SCC). In *2009 IEEE International Conference on Communications (ICC 2009)*, Dresden, Germany, June 2009.
- [27] M. El-Fotouh and K. Diepold. Breaking the Substitution Cipher Chaining mode (SCC-256). In *International Conference on Cryptography, Coding and Information Security (ICCCIS)*, Paris, France, June 2009.
- [28] M. Matsui. Linear cryptanalysis method for DES cipher. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, 1994.
- [29] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, 1991.
- [30] F. Sano, K. Ohkuma, H. Shimizu, and S. Kawamura. On the Security of Nested SPN Cipher against the Differential and Linear Cryptanalysis (Special Section on Cryptography and Information Security). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 86(1), 2003.
- [31] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson. Strengthening the Key Schedule of the AES. In *ACISP '02: Proceedings of the 7th Australian Conference on Information Security and Privacy*, 2002.
- [32] S. Even and Y. Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 10(3), 1997.
- [33] J. Daemen. Limitations of the Even-Mansour Construction. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, 1991.
- [34] A. Biryukov and D. Wagner. Advanced Slide Attacks. In *Advances in Cryptology—Eurocrypt '00 Proceeding*, 2000.

Mohamed Abo El-Fotouh received his B.Sc. and M.Sc. degrees in computer science from Ain Shams University, Cairo, Egypt. He is currently pursuing his Ph.D. degree at the Technische Universität München (TUM), Munich, Germany. His research is mainly in the field of cryptography, cryptanalysis and computer security.

Klaus Diepold received a Diplom-Ingenieur and a Doktor-Ingenieur degree both in Electrical Engineering from TUM in 1987 and 1992, respectively. He worked 10 years in high-tech industry dealing with digital TV and advanced video processing algorithms, and was actively involved in MPEG standardization. Since 2002, Klaus Diepold is a Professor for Data Processing at the Department for Electrical Engineering and Information Technology, Technische Universität München (TUM), Germany. His research interests include computational modeling, network theory and system theory, audio-visual signal processing and compression. More recently he works on algorithms for sub symbolic machine learning and cognitive

information processing.