All-or-Nothing Transforms as a Countermeasure to Differential Side-Channel Analysis

Robert P. McEvoy[1], Michael Tunstall[2], Claire Whelan[3], Colin C. Murphy[1], and William P. Marnane[1]

[1] Department of Electrical & Electronic Engineering, University College Cork, Cork, Ireland.
{robertmce,cmurphy,liam}@eleceng.ucc.ie
[2] Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol BS8 1UB, United Kingdom.
tunstall@cs.bris.ac.uk
[3] Department of Computer Science, Trinity College Dublin, Ireland.
claire.whelan@cs.tcd.ie

Abstract. All-or-Nothing Encryption was introduced by Rivest as a countermeasure to brute force key search attacks. This work identifies a new application for All-or-Nothing Transforms, as a protocol-level countermeasure to Differential Side-Channel Analysis (DSCA). We describe an extension to the All-or-Nothing protocol, that strengthens the DCSA resistance of the cryptosystem. The resultant scheme is a practical alternative to Boolean and arithmetic masking, used to protect implementations of encryption and decryption operations on electronic devices.

Keywords: Side-Channel Analysis, DSCA, DPA, Masking, All-or-Nothing Transforms.

1 Introduction

Electronic devices, such as smart cards used in electronic payment systems, naturally leak information about the data they are processing. For example, this leakage can manifest itself in the execution time of a cryptographic algorithm on the device [1], via its power consumption [2], or through electromagnetic emanations from the device [3]. Side-Channel Analysis (SCA) encompasses the class of implementation attacks which passively monitor these physical manifestations, then manipulate them to compromise the system's integrity. Inevitably, along with the development of an array of powerful attacks, there has also been active research to find appropriate, effective and efficient countermeasures [4–8].

All-or-Nothing Transforms (AONTs) are unkeyed probabilistic transformations, with numerous applications in cryptography. They were originally proposed by Rivest [9], as a means of hindering brute force key searches on block ciphers with short key lengths. By applying an AONT to a plaintext message, a 'pseudo-message' is formed. In order to invert the AONT, possession of the entire pseudo-message is required. This pre-processing stage is not considered encryption, however, as the AONT does not use a secret key. An All-or-Nothing Encryption (AONE) mode is formed when a pseudo-message is encrypted using a symmetric or asymmetric cipher.

In this paper, we consider All-or-Nothing cryptosystems from the point of view of side-channel attacks. We demonstrate that AONE possesses properties that inherently deter first-order Differential Side-Channel Analysis (DSCA). The AONE mode is extended, to further strengthen encryption systems against attacks based on either the plaintext or the ciphertext. The proposed system provides protection against DSCA while maintaining efficiency, in contrast to current DSCA countermeasures, which are often inefficient from the perspectives of speed and area.

This paper is organised as follows. Section 2 provides background on Differential Side-Channel Analysis, whilst Section 3 discusses the use of masking transformations as a DSCA countermeasure. In Section 4, we consider All-or-Nothing Transforms, and describe the DSCA resistance that is inherent to All-or-Nothing Encryption. Section 5 proposes an extension to the All-or-Nothing Encryption protocol to complete its DSCA resistance. Section 6 describes how the performance of the new encryption mode can be enhanced, by using the properties of the All-or-Nothing Transform. Finally, Section 7 presents further considerations, and we conclude in Section 8.

2 Differential Side-Channel Analysis

Differential Side-Channel Analysis (DSCA) exploits the fact that the instantaneous side-channel leakage from a cryptographic device, such as a smart card, depends on the data that it processes and the operations that it performs. For example, Differential Power Analysis (DPA) is a type of differential side-channel attack that has been well studied in the academic literature, and exploits data-dependent variations in the instantaneous power consumption of a cryptographic device [2, 10]. Similarly, Differential Electromagnetic Analysis (DEMA) is a type of DSCA that is based on electromagnetic (EM) radiation [3].

DSCA involves identifying an intermediate operation in a computation involving data that is known to an adversary (e.g. the plaintext or ciphertext), and data that is related to the secret (e.g. a portion of the secret key). This operation is referred to as a selection function. For example, let $\beta_i = S(\alpha_i, K)$ denote a selection function, where α_i is known by the adversary, and K is related to the secret. The algorithm is executed a number of times on the target device, with input α_i, where $1 \leq i \leq T$. For each execution, a trace t_i of the side-channel information (e.g. power consumption) is captured. In general, the selection function accepts n-bit inputs, where n may be the target device processor word size, or the number of bits input to an S-box, in the case of a symmetric-key algorithm. Therefore, it is feasible to calculate 2^n hypotheses $\beta_{i,j} = S(\alpha_i, K_j)$ for $0 \leq K_j < 2^n$, since K only relates to an n-bit portion of the secret.

The process of determining which hypothesis K_j corresponds to the actual secret K depends on the attack. For example, classical DPA categorises the power traces into two sets depending on a single bit b of $\beta_{i,j}$, where the same bit is used to categorise the power traces for all hypotheses of K_j, then calculates the distance of means between the two sets [2]. In contrast, Correlation Power Analysis (CPA) considers a processor's entire word [11]. CPA estimates the power consumption of $\beta_{i,j}$, which is calculated depending on a power model, where the power model is chosen to most accurately describe the target device's underlying architecture. The estimated power consumption is then compared to the actual power consumption, using a correlation test. Attacks such as these are termed first-order DSCA attacks, since the side-channel leakage from a single operation within the target device is exploited.

3 Masking as a DSCA Countermeasure

The goal of countermeasures against DPA attacks is to make the power consumption of a cryptographic device independent of the intermediate values of a cryptographic algorithm. Such countermeasures can be classified into two categories: hiding and masking. Hiding countermeasures remove the dependence of the side-channel leakage of a cryptographic device on the intermediate variables processed within that device. For example, hiding countermeasures against DPA either attempt to randomise the power consumption or to make it constant. This paper focuses on masking countermeasures.

3.1 Boolean and Arithmetic Masking

Masking countermeasures work by rendering the intermediate variables processed within a device independent of the intermediate variables in a cryptographic algorithm, so that the intermediate variable β_i processed by the device is not equal to $S(a_i, K)$ [4, 5]. This independence is achieved by combining each intermediate variable in the cryptographic algorithm with a random value (a 'mask'), which changes on each execution of the algorithm. For example, an intermediate variable x can be masked with a random value r via: $x_r = x \oplus r$. This is termed Boolean masking, since the masking operation is Boolean XOR, denoted by $\oplus$. Similarly, arithmetic masking applies a mask using an arithmetic function, such as modular addition or modular multiplication. Subsequently, it is the masked value x_r that appears on the device during the execution of an algorithm, rather than x. Since x_r is unpredictable to an attacker, first-order DSCA attacks are prevented.

Masking can be applied at the architecture level, where random masks are incorporated into high-level functions, such as multiplication or key addition [12]. Masking can also be implemented at the gate level, where basic cells, such as AND and OR gates, are modified to incorporate random inputs as well as masked inputs [13]. When classical masking is applied at either the architecture or the gate level, the implementation should also include mask correction circuitry. This circuitry tracks the modifications to the mask that happen when the masked data is processed by the cryptographic algorithm, so that the mask can be removed at the end of the cryptographic computation. Mask correction typically leads to increases in the implementation area and execution time of the device.

3.2 Masking using Cryptographic Transformations

To date, few publications in the literature have considered using cryptographic transformations to protect encryption (and decryption) algorithms against DSCA [7, 8, 14]. As illustrated in Figure 1, such countermeasures operate by applying cryptographic transformations T_1 and T_2

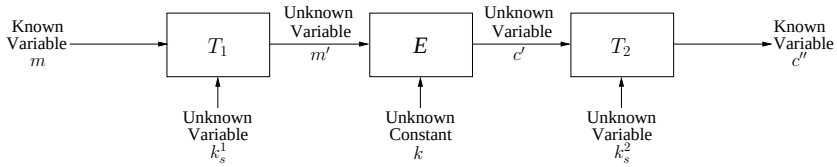

Fig. 1. Using cryptographic transformations as a DSCA countermeasure

before and after the encryption operation $E(\cdot, k)$, respectively. The cryptographic transformations are such that the input m' and output c' of the encryption function are obscured from a side-channel attacker. Therefore, hypotheses for the value of an intermediate variable β_i cannot be computed based on knowledge of the plaintext m or the ciphertext c'', and first-order DSCA is prevented. Since the transformations use additional variables k_s^1 and k_s^2 unknown to the attacker (e.g. random strings), this countermeasure may be considered to be a type of protocol level masking.

Giraud and Prouff proposed a construction of this countermeasure, to counteract DSCA of block ciphers [7]. Their solution was to add a layer $P_{k'}^0$ before the encryption operation and a layer $P_{k'}^1$ after, parameterised by a shared secret key k'. Each layer consisted of a fixed, linear, involutive diffusion function L and a non-linear permutation $\pi_{k'}$. However, the countermeasure proposed by Giraud and Prouff was withdrawn, after weaknesses were discovered in the design. In particular, the output of the $P_{k'}^0$ layer (i.e. m', the input to the encryption function) was found to be predictable by an attacker, for inputs of low Hamming weight.

In [8], Tiri, Schaumont and Verbauwhede proposed two 'side-channel leakage tolerant architectures'. As in [7], the aim of these designs was to provide DSCA protection, whilst leaving the encryption and decryption operations unmodified. The first architecture proposed the application of permutations P_1 and P_2 before and after encryption, respectively. These permutations are dependent on secret permutation keys K_{p1} and K_{p2}. However, specific constructions of the permutations P_1 and P_2 were not provided. Furthermore, the problem of distributing the secret permutation keys K_{p1} and K_{p2} was not discussed, nor how often these keys should be updated. The second architecture considered in [8] uses two intertwined Cipher Block Chaining (CBC) mode encryptions to prevent DSCA based on knowledge of the plaintext or the ciphertext. A disadvantage of this scheme is that the encryption function must be invoked twice for each message block that is to be encrypted.

Recently, Coron proposed the use of permutation tables as a DSCA countermeasure [14]. A randomised permutation P is applied to the message and the key prior to encryption, and it is these permuted variables that are used by the encryption algorithm. At each stage of the encryption algorithm, the intermediate variables are corrected, so that they remain in the permuted form described by P. This correction is time-consuming; the permutation table countermeasure is approximately four times slower than classical masking [14]. Once the encryption algorithm has completed, the inverse permutation P^{-1} is applied. In contrast, the countermeasures

described by [7] and [8] do not implement correction; therefore, the resultant ciphertext differs from that which would have been produced by an unprotected implementation. It is this type of scheme, with no mask correction, that is the focus of this paper. In a sense, the 'mask correction' required to retrieve the message m from the schemes of [7] and [8] is applied at the receiver side. The receiver uses the corresponding decryption protocol to invert T_2, decrypt with secret key k, and invert T_1, thereby retrieving the message m.

3.3 Properties of Masking Transformations

In general, cryptographic transformations T_1 and T_2 should possess certain properties, if they are to be used in the DSCA countermeasure shown in Figure 1. These properties are listed below.

Property 1: *Pre-encryption transformation T_1 and post-encryption transformation T_2 should be dependent on randomly chosen inputs k_s^1 and k_s^2, respectively.*
If k_s^1 is a random string generated within the device, then the input m' to the encryption function is rendered unpredictable to an attacker. Similarly, the entropy of k_s^2 ensures that c' is unpredictable, and 'reverse' DSCA cannot be mounted.

Property 2: *Post-encryption transformation T_2 should be dependent on a shared secret k_s^2.*
Transformation T_2 should only be invertible to the legitimate receiver, so that it cannot be inverted by an attacker. This requires that T_2 should include some element, k_s^2, known exclusively to the sender and receiver. For example, k_s^2 could be equal to, or derived from, the shared secret key k. As will be shown in the following section, it is not required that a receiver must possess k_s^1 in order to invert T_1 and recover the message m.

Property 3: *Post-encryption transformation T_2 should not linearly combine the output of the encryption stage c' with a constant.*
If the output of the encryption function is linearly combined with some constant; then this constant can be derived by DSCA in the same way in which one would attempt to derive the key. Another attack could include the constant as part of the final functions of the encryption algorithm. This attack could then proceed by firstly deriving the effect of the constant, and then deriving some information about the secret key k.

All-or-Nothing Encryption	All-or-Nothing Decryption
Inputs: plaintext m, random value r	Inputs: ciphertext c'
1. $m' = \text{AONT}_r(m)$;	1. $m' = D_k(c')$;
2. $c' = E_k(m')$;	2. $(m, r) = \text{Inv-AONT}(m')$;

Table 1. All-or-Nothing Encryption and Decryption

Property 4: *Pre- and post-encryption transformations T_1 and T_2 should themselves be DSCA-resistant.*
This final property may seem obvious; however, it is important to emphasise. When constructing T_2, it may seem reasonable to combine the shared secret k_s^2 (required by Property 2) with some variable data d. However, if this data is known to an attacker, then the focus of the DSCA attack shifts from the encryption operation E to the transformation T_2. Ideally, Property 4 can be attained by using random values for k_s^1 and k_s^2, that change for each new message m that is encrypted.

In the following section, we present new constructions for transformations T_1 and T_2 based on All-or-Nothing Transforms, that fulfil all of the above properties.

4 All-or-Nothing Transforms

All-or-Nothing Transforms (AONTs) were proposed by Rivest [9] in 1997, for the purpose of deterring exhaustive key searches on block ciphers. An All-or-Nothing Transform uses a random string r to map a plaintext message m (of variable bit length) to a pseudo-message m'. The pseudo-message can, subsequently, be encrypted with an encryption function E and key k; for example, using a symmetric block cipher such as AES [15]. The entire process is referred to as All-or-Nothing Encryption (AONE).

All-or-Nothing Decryption allows the receiver to recover the message m from the transmitted ciphertext c'. The receiver applies the decryption function D with key k to recover m'. Subsequently, the inverse AONT is applied, to recover r and m. The All-or-Nothing Encryption and Decryption protocols are described in Table 1.

An attacker attempting to conduct an exhaustive key search on an All-or-Nothing encryption mode is required to decipher *all* of the ciphertext blocks, in order to test a single key hypothesis. Therefore, the average execution time of a brute force attack on an cryptosystem using AONE is increased in proportion to the number of blocks in the ciphertext.

4.1 AONT Properties and Constructions

As put forward in [9], AONTs should possess four properties:

1. The transformation should be invertible. Given the entire pseudo-message, one can invert the transformation to retrieve the plaintext.
2. Both the AONT and its inverse should be efficiently computable.
3. All AONTs should be randomised, in order to avoid chosen-message and known-message attacks on the encryption mode.
4. If any l (or more) bits of the pseudo-message are unknown, it should be computationally infeasible to invert the AONT, or determine any function of the plaintext bits. This is called the "All-or-Nothing" property. The value of l is AONT-dependent, but large enough to deter brute force attacks on the unknown bits of the pseudo-message.

Property 3 implies that pseudo-message m' is not a deterministic function of the plaintext m. In order to fulfil this requirement, an AONT requires a user-generated random string r when computing m' from m.

Many candidate AONT constructions have been proposed in the literature. In this paper, we focus on OAEP (Optimal Asymmetric Encryption Padding) [16, 17] and the CTR Transform [18] as case studies. Other proposed transforms include: Package Transform [9], Exposure-Resilient Function-based Transforms [19], Quasigroup-based AONTs [20], 'Extended-Indistinguishable' AONTs [21], and Error-Correcting Code-based AONTs [22].

Both OAEP and CTRT conform to a general model for AONTs, shown in Figure 2. In our general AONT model, an expansion function G uses the random value r to produce an "All-or-Nothing keystream", having the same bit length as the message m. Since r changes for each new message m, the AON keystream $G(r)$ can be viewed as a type of one-time pad [23]. Subsequently, this AON keystream is used to mask the message m. In both OAEP and CTRT, the masking function is simply a Boolean XOR. This generates the majority of the pseudo-message bits. We refer to this sequence of operations as a "partial AONT", which is effectively equivalent to Boolean or arithmetic masking.

However, the partial AONT does not possess the All-or-Nothing property, nor is it invertible. To effectuate these properties, the sender must communicate the value of the random value r to the receiver. This is achieved by compressing the output of the partial AONT, and using it to mask the random value r. This final step completes the AONT, ensuring that the entire pseudo-message is required by the receiver, in order to retrieve r and subsequently invert the transformation.

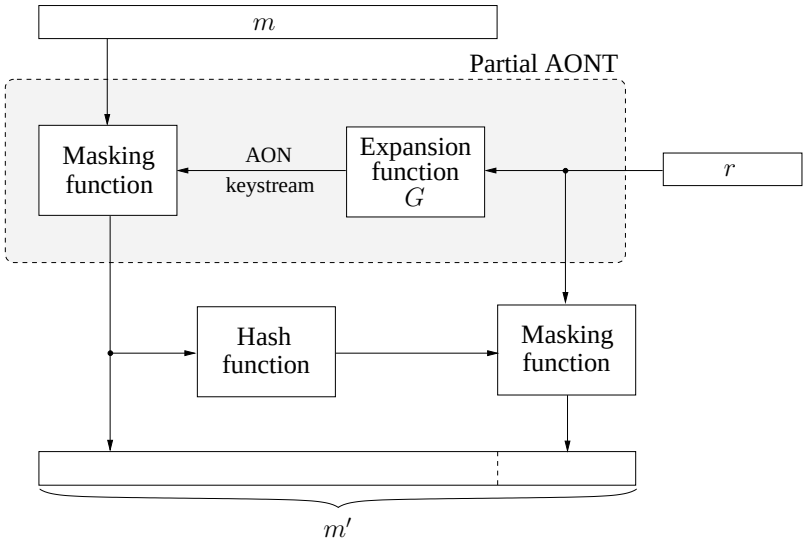

Fig. 2. General model for All-or-Nothing Transforms

The operations used by OAEP are recalled in Table 2, where $s||t$ denotes concatenation of the strings s and t. Functions G and H are defined as random oracles [16, 17]. OAEP forms the basis of a routine called Encoding Method for Encryption #1 (EME1), which is standardised in [24]. According to the standard, functions G and H should be instantiated using mask generation function MGF1, which has a user-specified output length. MGF1 is based on repeated iterations of a secure one-way hash function, such as SHA-256 [25]. In general, G can be regarded as expanding its input, whilst H compresses its input. The CTR Transform (CTRT) is described in Table 3. In CTRT, the expansion function is instantiated using a block cipher in CTR mode, essentially turning the block cipher E into a stream cipher, using random key k'. The final pseudo-message block is formed by simply XOR-ing together all of the other pseudo-message blocks, along with the random key.

4.2 DSCA Resistance of AONE systems

Aside from hindering brute force key searches on cryptosystems with short key lengths, AONTs have found other useful applications in cryptography. For example, AONTs are an important part of remotely keyed encryption [17, 26] and protocols for efficient block cipher encryption [27]. Here,

OAEP		**Inverse-OAEP**	
Inputs:	message m, random value r	Input:	pseudo-message $m' = s\|\|t$
Output:	pseudo-message $m' = s\|\|t$	Outputs:	random value r, message m
1.	$s = m \oplus G(r)$;	1.	$r = t \oplus H(s)$;
2.	$t = r \oplus H(s)$;	2.	$m = s \oplus G(r)$;

Table 2. OAEP and Inverse-OAEP Transformations

CTRT		**Inv-CTRT**	
Inputs:	message blocks m_i, $1 \le i \le n$ random key k'	Inputs:	pseudo-message blocks m'_i, $1 \le i \le n+1$
Outputs:	pseudo-message blocks m'_i, $1 \le i \le n+1$	Outputs:	random key k' message blocks m_i, $1 \le i \le n$
1.	$m'_i = m_i \oplus E_{k'}(i)$; $1 \le i \le n$	1.	$k' = m'_1 \oplus m'_2 \oplus \cdots \oplus m'_{n+1}$;
2.	$m'_{n+1} = k' \oplus m'_1 \oplus m'_2 \oplus \cdots \oplus m'_n$;	2.	$m_i = m'_i \oplus E_{k'}(i)$; $1 \le i \le n$

Table 3. CTR Transform (CTRT) and Inverse CTRT

it is shown that All-or-Nothing Encryption can be used to protect against Differential Side-Channel Attacks.

As described in Section 2, to perform DSCA on an encryption algorithm, an attacker is required to predict an intermediate state within the algorithm. In All-or-Nothing Encryption, however, the plaintext m is pre-processed along with a random string r. It is the resulting m' that is operated upon by the encryption algorithm E and key k. The attacker cannot predict m', without knowing r. Moreover, the random input r changes on each application of the AONT, mapping m to a different m' each time. Therefore, the hypothesis phase of the attack cannot be performed. Hence, All-or-Nothing Encryption (AONE) exhibits inherent resistance to DSCA. With reference to Figure 1, it can be seen that the AONT is equivalent to applying a pre-encryption transformation T_1. The random input r to the AONT is equivalent to the parameter k_s^1 of T_1.

5 Extended All-or-Nothing Encryption

All-or-Nothing Encryption uses a cryptographic transformation to mask the plaintext prior to encryption. However, AONE does not prevent hypotheses based on the resultant ciphertexts, and so a post-encryption transformation T_2 is required, to complete the DSCA protection. A construction for T_2 is presented in this section, based on a partial AONT (from the general AONT model defined above). We call the resultant protocol "Extended All-or-Nothing Encryption", and it is illustrated in Figure 3. The steps of the protocol are as follows:

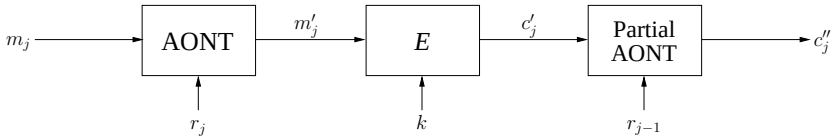

Fig. 3. Extended All-or-Nothing Encryption protocol, with Partial AONT.

Encryption protocol

1. At the beginning of the communication session, the sender and receiver must agree on a secret key k and a secret initial random value r_0. This part of the protocol is discussed later in this section.
2. The message m to be transmitted is divided into packets m_j, $1 \leq j \leq N$. The size of the packets is arbitrary and may, for example, depend on the particular transmission protocol in use by the communications system.
3. The value of j is set to 1.
4. Packet m_j is transformed using an AONT with random value r_j, producing a pseudo-message packet m'_j.
5. The pseudo-message packet m'_j is encrypted with the encryption algorithm E and secret key k. If E is a block cipher, then it may be applied several times, as the length of m'_j may exceed the block size of the cipher. The result is an encrypted pseudo-message packet c'_j.
6. A partial AONT is then applied to c'_j. However, a new random value is not used; instead, the random value r_{j-1} is re-used. Clearly, it is advantageous to base the partial AONT on the same AONT that is used in the pre-encryption transformation. In this way (for $2 \leq j \leq N$), the sender does not need to fully re-compute $G(r_{j-1})$, the All-or-Nothing keystream that was generated by using the expansion function on r_{j-1}, because it was already computed while m'_{j-1} was being generated. An extra block of $G(r_{j-1})$ does need to be generated, however, since the bit length of m' is greater than that of m.
7. The output of this post-encryption transformation is ciphertext packet c''_j, which is transmitted to the receiver. The value of j is incremented.
8. Steps 4 – 7 are repeated, until all N blocks in the message have been processed.

At the receiver side, the steps of the corresponding decryption protocol are as follows:

Decryption protocol

1. The secret key k and secret initial random value r_0 are agreed upon with the sender at the start of the communications session.
2. The value of j is set to 1.
3. Packet c''_j is received, and is processed using the (inverse) partial AONT with random value r_{j-1}, to produce c'_j.
4. Packet c'_j is decrypted using secret key k and D, the decryption algorithm corresponding to E. Depending on the size of c'_j, several calls to D may be required. The result of the decryption stage is packet m'_j.
5. Once all of the bits of m'_j have been determined, the inverse AONT is applied to it, generating m_j and r_j. The value of j is incremented.
6. Steps 3 – 5 are repeated, until all N packets of the received message have been processed. Finally, the m_j blocks are reassembled to form the message m.

Establishing the initial random value r_0

It is important that the initial random value r_0 is not made public, otherwise an attacker could use c''_1 to retrieve c'_1. By continually resetting a target device, the attacker could gather many different values of c'_1, and use this knowledge and corresponding side-channel information to conduct a differential side-channel attack on the encryption algorithm.

In public-key cryptosystems, r_0 can be determined as part of the normal key setup phase, e.g. using a protocol based on Diffie-Hellman key exchange [28]. However, ubiquitous cryptographic devices such as smart cards are often not part of a public-key infrastructure. Instead, these devices are often configured with secret keys at the time of their manufacture. The secret key value can only be accessed by the device itself, and by trusted entities such as banks. In this type of symmetric-key scenario, it is more difficult to exchange an initial random value r_0 securely.

A natural solution is to use the fact that the secret key k is known to both the sender and the receiver. The initial random value r_0 may be some derivative of the key; for example, the result of applying a hash function i times to the key, where the value of i is transmitted in clear from the sender to the receiver. It is a non-trivial problem to safely communicate the value of r_0, because any public information transmitted from the sender to the receiver could be used to perform a side-channel attack on the calculation from which r_0 is derived, leading to exposure of the secret key k. Whatever solution is chosen will ultimately depend on the

nature of the sender and receiver (e.g. smart cards, card readers, cryptographic processors etc.), their computational capabilities and the security protocols for key setup that already exist within the cryptosystem.

6 Combining with Efficient Encryption

In the previous section, we described how Extended All-or-Nothing Encryption can be used as a DSCA countermeasure. The performance of the Extended AONE scheme can be enhanced by combining it with another application of AONTs, namely "Efficient Encryption" [27, 29]. In the Efficient Encryption scheme, an AONT is applied to a plaintext message, but only some (as opposed to all) of the pseudo-message blocks are subsequently encrypted. The AONT ensures that all of the pseudo-message blocks are required in order to invert the transformation and regain the plaintext. Therefore, fewer secret-key encryptions and decryptions are required. The transmitted ciphertext comprises both encrypted and unencrypted pseudo-message blocks.

To combine the Efficient Encryption with the Extended AONE scheme, the plaintext message packet m_j is transformed with an AONT, as described above. A certain set of pseudo-message blocks is chosen to be encrypted with encryption function E and secret key k; knowledge of which blocks are in the set can be made public. The presence of the AONT

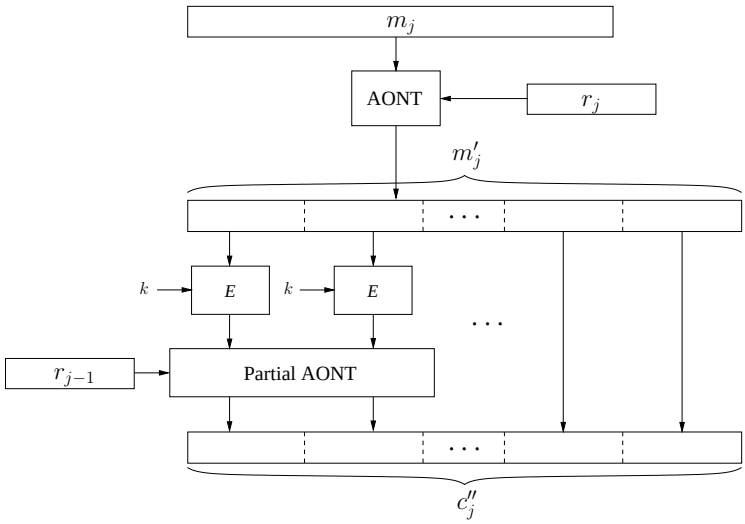

Fig. 4. Extended All-or-Nothing Encryption combined with Efficient Encryption.

as a pre-encryption transformation T_1 prevents DSCA of the encryption function, based on known plaintext messages. A post-encryption transformation T_2 (in the form of a partial AONT) must then be applied to those blocks that were encrypted, to prevent reverse DSCA. The transmitted ciphertext packet c_j'' consists of unencrypted pseudo-message blocks, alongside blocks that are output from T_2. An example of the combined scheme is shown in the block diagram in Figure 4. The system has the advantage of being resistant to DSCA, whilst requiring fewer encryption operations to be performed with the secret key. Therefore, depending on the particular constructions used for the AONT and partial AONT, and the message size, the overall latency and power consumption of the cryptosystem may be reduced.

In ordinary encryption modes (including AONE), the security of the ciphertext blocks relies on the security of the encryption function E, which, in turn, relies on the secrecy of the key k. However, in this Efficient Encryption mode, not all of the transmitted ciphertext blocks are operated on by the encryption function E. The encryption function E imparts security to the encrypted ciphertext block(s). If one has confidence in the security of the encryption function, then one can be confident that the missing pseudo-message block(s) cannot be recovered without knowledge of the secret key k. Because of the All-or-Nothing property of the pseudo-message, the security properties of E are bestowed on the entire pseudo-message. If the AONT is not invertible without possession of the missing pseudo-message block(s), the plaintext should not be recoverable by an attacker. Therefore, the security of the scheme shifts from relying entirely on the encryption function E when all blocks are encrypted, to relying almost entirely on the AONT when fewer blocks are encrypted.

7 Further Considerations

The main advantage of Extended All-or-Nothing Encryption over Boolean and arithmetic masking is that mask correction terms do not have to be computed. Calculation of mask correction terms typically impacts upon the area and speed of the cryptosystem, and such masking schemes must be designed specifically for the particular encryption algorithm being used [30]. Clearly, extra area and time are required in order to compute the pre- and post-encryption transformations. However, this is not necessarily a disadvantage if a fast AONT is employed. Additionally, in the above section it was shown that, due to the All-or-Nothing property, it is unnecessary to encrypt the entire AONT output. If just one pseudo-

message block is encrypted, then the latency of the encryption protocol approaches the latency of the AONT.

In describing our general AONT model, it was noted that application of the partial AONT may be regarded as Boolean or arithmetic masking. This implies that the new countermeasure is susceptible to the same types of strong side-channel attacks as the first-order masking countermeasure, namely higher-order DSCA [31] and template attacks [32]. It has been shown that significant protection against these attacks can be achieved by combining first-order masking with a hiding countermeasure such as randomisation of the order of operations ('shuffling') in the encryption function [33]. Therefore, Extended All-or-Nothing Encryption should be combined with shuffling, to deter stronger side-channel attacks.

Extended All-or-Nothing Encryption does not have the same weakness as the scheme of Giraud and Prouff [7], regarding inputs of low Hamming weight. The AONT input m is combined with the pseudo-random sequence $G(r)$ before being operated upon by the encryption function. Even if m is of low Hamming weight, the pseudo-randomness of the pseudo-message m' is unaffected.

Extended AONE is comparable to a side-channel leakage tolerant architecture of Tiri et al. [8]. Here, we instantiate P_1 and P_2 using an AONT and partial AONT. The session keys K_{p1} and K_{p2} correspond to the random inputs r_j and r_{j-1}, which update for each new message packet that is encrypted. Once the initial 'session key' r_0 has been securely shared between the sender and receiver, subsequent session keys (for subsequent packets) do not need to be explicitly relayed to the receiver. Because the pre- and post encryption transformations are based on AONTs, the receiver uncovers the value of r_j when decoding packet m_j. Therefore, updates of the session keys happen as a by-product of the usage of AONTs.

8 Conclusion

In this paper, we presented a novel countermeasure to first-order Differential Side-Channel Analysis, based on All-or-Nothing Transforms. The proposed countermeasure can be viewed as a type of masking countermeasure at the protocol level, since for each message packet m_j to be encrypted, the AONT uses a new random 'mask' r_j. The AONT expands the random value r_j, and uses it to mask all of the bits of the message m_j. Therefore, the masked message m'_j cannot be predicted by an attacker. Similarly, the output of the encryption operation c'_j is unpredictable to an attacker, because it is concealed by a mask derived from r_{j-1}, using a

partial AONT. This DSCA countermeasure differs from previously proposed masking schemes in the literature, because a mask correction is not applied. By using fast AONT constructions (e.g. hash function based), the new countermeasure can out-perform the classical masking countermeasure. In conjunction with the Efficient Encryption technique of [29], the latency of a DSCA-protected encryption can be reduced to the latency of the All-or-Nothing Transform. Future work will evaluate the performance of this new countermeasure on the smart card platform, for various AONT constructions, and compare it with the masking countermeasure, which is currently widely used in the electronic payments industry.

Acknowledgements

The authors would like to thank Emmanuel Prouff and Christophe Giraud of Oberthur Card Systems, for the helpful discussions on their work [7]. This research was supported in part by the Embark Initiative, operated by the Irish Research Council for Science, Engineering and Technology (IRCSET), and the support of the Informatics Commercialisation Initiative of Enterprise Ireland is gratefully acknowledged.

References

1. Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
2. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
3. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001, Third International Workshop*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
4. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology — CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
5. Louis Goubin and Jacques Patarin. DES and differential power analysis — the duplication method. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 1999, First International Workshop*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

6. Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Design, Automation and Test in Europe, DATE 2004*, pages 246–251. IEEE Computer Society, 2004.
7. Christophe Giraud and Emmanuel Prouff. A new approach to counteract DPA attacks on block ciphers. Private Communication. Available online at the IACR ePrint Archive, `http://eprint.iacr.org/2005/340`, September 2005.
8. Kris Tiri, Patrick Schaumont, and Ingrid Verbauwhede. Side-channel leakage tolerant architectures. In *Third International Conference on Information Technology: New Generations (ITNG 2006)*, pages 204–209. IEEE Computer Society, 2006.
9. Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *FSE '97, 4th International Workshop on Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218. Springer, 1997.
10. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.
11. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems — CHES 2004, 6th International Workshop*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
12. Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably secure masking of AES. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2004.
13. Jovan Dj. Golić. Techniques for random masking in hardware. *IEEE Transactions on Circuits and Systems — I*, 54(2):291–300, February 2007.
14. Jean-Sébastien Coron. A new DPA countermeasure based on permutation tables. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008*, volume 5229 of *Lecture Notes in Computer Science*, pages 278–292. Springer, 2008.
15. National Institute of Standards and Technology. FIPS PUB 197. Advanced Encryption Standard, November 2001.
16. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.
17. Victor Boyko. *On All-or-Nothing Transforms and Password-Authenticated Key Exchange Protocols*. PhD thesis, Massachusetts Institute of Technology, June 2000.
18. Anand Desai. The security of all-or-nothing encryption: Protecting against exhaustive key search. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000, 20th Annual International Cryptology Conference*, volume 1880 of *Lecture Notes in Computer Science*, pages 359–375. Springer, 2000.
19. Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, volume 2045 of *Lecture Notes in Computer Science*, pages 301–324. Springer, 2001.
20. Stelios I. Marnas, Lefteris Angelis, and George L. Bleris. All-or-nothing transforms using quasigroups. In *Proceedings of 1st Balkan Conference on Informatics*, pages 183–191, November 2003.
21. Rui Zhang, Goichiro Hanaoka, and Hideki Imai. On the security of cryptosystems with all-or-nothing transform. In Markus Jakobsson, Moti Yung, and Jianying

Zhou, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2004.
22. John Byers, Jeffrey Considine, Gene Itkis, Mei Chin Cheng, and Alex Yeung. Securing Bulk Content Almost for Free. *Journal of Computer Communications, Special Issue on Internet Security*, 29:290–290, Feb 2006.
23. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
24. IEEE. 1363a IEEE Standard Specifications for Public-Key Cryptography—Amendment 1: Additional Techniques, 2004.
25. National Institute of Standards and Technology. FIPS PUB 180-2. Secure Hash Standard, August 2002.
26. Matt Blaze. High-bandwidth encryption with low-bandwidth smartcards. In Dieter Gollmann, editor, *Fast Software Encryption, 3rd International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 33–40. Springer, 1996.
27. Donald B. Johnson, Stephen M. Matyas, and Mohammad Peyravian. Encryption of long blocks using a short-block encryption procedure. Submitted for inclusion in the IEEE P1363a standard, November 1996.
28. Whitfield Diffie and Martin Edward Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
29. Donald Byron Johnson and Stephen Michael Matyas, Jr. (IBM). Method and apparatus for encrypting long blocks using a short-block encryption procedure. US Patent # 5,870,470, February 1999.
30. Elisabeth Oswald, Stefan Mangard, and Norbert Pramstaller. Secure and efficient masking of AES — a mission impossible? IACR ePrint Archive, `http://eprint.iacr.org/2004/134`, June 2004.
31. Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000, Second International Workshop*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.
32. Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
33. Stefan Tillich and Christoph Herbst. Attacking state-of-the-art software countermeasures — a case study for AES. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems — CHES 2008, 10th International Workshop*, volume 5154 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2008.