# Multipurpose Cryptographic Primitive ARMADILLO3

Petr Sušil* and Serge Vaudenay

EPFL, Lausanne, Switzerland
{petr.susil,serge.vaudenay}@epfl.ch

**Abstract.** This paper describes a new design of the multipurpose cryptographic primitive ARMADILLO3 and analyses its security. The ARMADILLO3 family is oriented on small hardware such as smart cards and RFID chips. The original design ARMADILLO and its variants were analyzed by Sepehrdad et al. at CARDIS'11, the recommended variant ARMADILLO2 was analyzed by Plasencia et al. at FSE'12 and by Abdelraheem et al. at ASIACRYPT'11. The ARMADILLO3 design takes the original approach of combining a substitution and a permutation layer. The new family ARMADILLO3 introduces a reduced-size substitution layer with $3 \times 3$ and $4 \times 4$ S-boxes, which covers the substitution layer from 25% to 100% of state bits, depending on the security requirements. We propose an instance ARMADILLO3-A1/4 with a pair of permutations and S-boxes applied on 25% of state bits at each stage.

## 1 Introduction

Tiny computing devices such as smart cards, sensor networks and RFID tags are becoming more and more widespread. The implementation of standardized cryptographic algorithms such as the block cipher AES [21] or the hash functions SHA [10] are very expensive in terms of the number of gates and power consumption. Moreover, the security requirements of these tiny devices are often weaker than which of algorithms such as AES or SHA. The widespread usage of the constrained devices triggered a spontaneous competition for the tiniest and the most secure designs. There have been several designs of such primitives [5,8,9,14,15,17,25].

Since these devices communicate over an insecure channel, usually a wireless channel, there is a threat of an attacker trying to listen to the communication or trying to impersonate a server or another device. Therefore, there is a need for an authentication protocol to provide authenticity of the device, and an encryption to provide the confidentiality. However, as we want to reduce the implementation cost as much as possible, it is important to find a universal design, which can be used in many different applications. This allows to further reduce the

* Supported by a grant of the Swiss National Science Foundation, 200021_134860/1. This work has been supported in part by the European Commission through the ICT program under contract ICT-2007-216646 ECRYPT II.

S. Mangard (Ed.): CARDIS 2012, LNCS 7771, pp. 203–218, 2013.

implementation cost, as it is not necessary to implement multiple algorithms on the small device. Some recent designs deal with this issue by reusing some parts of the implementation, for instance the hash function QUARK [2] and the message authentication code SQUASH-128 [23] use some components of the stream cipher GRAIN [14]. This approach is the first step towards a multipurpose cryptographic primitive, that can be used in all applications.

We introduce a new primitive ARMADILLO3 which is designed to be used as a message authentication code (MAC), a hash function and a pseudo-random number generator. The ARMADILLO3 is the third generation of the multipurpose cryptographic function ARMADILLO [3] introduced at CHES'10. The new version ARMADILLO3 prevents all known attacks against the ARMADILLO [22] design and the attack against ARMADILLO2 based on parallel matching [1], and Hamming weight preservation in PRNG mode [19]. We provide a security analysis against known types of attacks and discuss some dedicated attacks and counter-measures. We support our security claims using the security analysis based on properties of the underlying expander graph of ARMADILLO3.

The ARMADILLO is a family of cryptographic functions based on data dependent permutations. That is, we use an internal function $P$ defined by $P(p\|b, Z) = P(p, S(Z_{\sigma_b}))$ iteratively, where $b$ is the tailing bit of the first operand $p\|b$, $S$ is a substitution layer, $\sigma_b$ is a permutation ($\sigma_0$ or $\sigma_1$) and $Z_{\sigma_b}$ denotes the transposition of $Z$ based on permutation $\sigma_b$. The extension ARMADILLO3 adopts a preprocessing to prevent the known attacks against ARMADILLO1 reported in [22], and it introduces a reduced-size S-box layer to improve the confusion of ARMADILLO2 which lead to a practical low complexity attack reported in [19].

The ARMADILLO3 internal function generalizes the SPN structure by introducing a second permutation. In every round, we choose one of the two permutations based on a pseudorandom value.The internal function is then followed by an XOR with the input and the control register value similar to the Davies-Meyer construction.

The ARMADILLO3 reduces the number of S-boxes due to both the higher number of rounds and the pseudorandom selection of the permutation. This means that only some bits of the internal state go through the S-boxes in a single round. The pair of permutations for ARMADILLO3 has to be selected in such a way that even when the attacker controls the selection of the permutation at every round, which is the case for hash functions, she should be unable to prevent the diffusion of the input. Therefore, the selection of the two permutations is a non-trivial task. We introduce a notion of Hierarchical Permutations which ensure that every bit goes through an S-box in a minimum number of steps for all possible sequences of data-dependent permutations, while making no significant restrictions on other properties of these permutations. The selection of the final pair is based on the diffusion properties of both permutations and the expansion properties of the expander graph corresponding to the pair of the permutations.

## 2 The ARMADILLO3 Function

The ARMADILLO3 is based on a recursive function $P$ which takes two parameters $P(Y, X)$. The register $Y$ is used as a control register for selecting the permutation in each step of the function $P$, and the $i$th step of $P$ consists of applying permutation $\sigma_0$ or $\sigma_1$, depending on the value $Y[i]$, and the S-boxes on specified bits. Since the value $Y$ has to be pseudo-random which is difficult to control for an attacker, we set $Y = P(X, X)$ for an input $X$. Therefore, the ARMADILLO3 consists of two steps: preprocessing step for computing the value $Y$ and the computation of $P(Y, X)$ followed by an XOR with the input $X$ and the control register $Y$. We give recursive definition of ARMADILLO3 followed by the pseudo-code. The parameters for the algorithm are: the type of S-boxes, the number and placement of S-boxes, and the permutation pair.

The ARMADILLO3 algorithm on input $\mathsf{W} = \mathsf{H} \| \mathsf{X}$ is defined as follows.

$$\begin{aligned} \mathsf{ARMADILLO3}(\mathsf{W}) &= P(\mathsf{Y}, \mathsf{W}) \oplus \mathsf{W} \oplus \mathsf{Y}, \text{ for} \\ \mathsf{Y} &= P(\mathsf{W}, \mathsf{W}) \\ P(p \| b, Z) &= P(p, S(Z_{\sigma_b})) \\ P(\lambda, Z) &= Z \end{aligned}$$

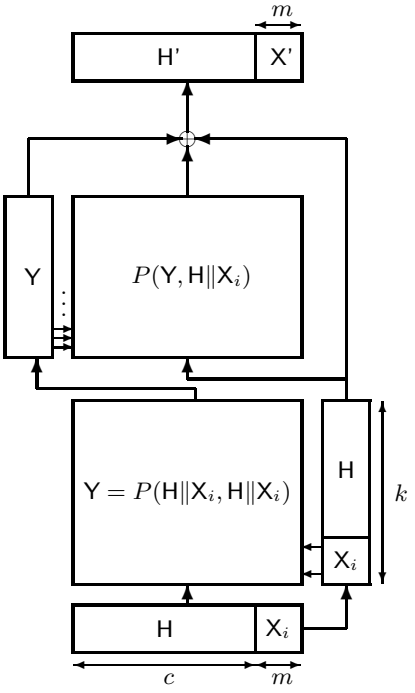


**Fig. 1.** Scheme of ARMADILLO3

where $p$ denotes a bit string, $b$ denotes a bit, $S$ denotes the substitution layer which is defined separately, and $\lambda$ denotes the empty string. The substitution layer of ARMADILLO3 consists of $r$ identical $t \times t$ S-boxes. In our example, i.e., ARMADILLO3-A1/4, we build the permutation pair $\sigma_0$, $\sigma_1$ for the placement of $3 \times 3$ S-boxes at positions 0-32 (so we have 11 $3 \times 3$ S-boxes), which gives the "coverage rate" $= \frac{rt}{k}$, where $k$ is the size of register Y.

**Algorithm 1.** ARMADILLO3 pseudo-code

```
input X, H
W = H||X
R = H||X
for i=0 to |W| do
   b = W[i]
   R ← S(R_{σ_b})
end for
for i=0 to |W| do
   b = P[i]
   W ← S(W_{σ_b})
end for
return W
```

The substitution layer $S$, i.e., the S-boxes and the bits covered by S-boxes, together with permutations $\sigma_0$, $\sigma_1$ are defined later for ARMADILLO3-A1/4.

The function ARMADILLO3 differs from the original design ARMADILLO1 [3] in several ways. Like ARMADILLO2 [3], it has an internal register of size $k$ instead of $2k$, which makes the design more compact, and as ARMADILLO2 it also includes a preprocessing step, i.e., $\mathsf{Y} = P(\mathsf{H}\|\mathsf{X}, \mathsf{H}\|\mathsf{X})$. The preprocessing prevents the attacker from controlling the permutation $P(\mathsf{Y}, \mathsf{H}\|\mathsf{X})$, since it is difficult for an attacker to predict $\mathsf{Y} = P(\mathsf{H}\|\mathsf{X}, \mathsf{H}\|\mathsf{X})$ without a knowledge of H. In the case of the hash function, when the attacker knows the value H or is allowed to choose this value to find a pseudocollision, the attacker can only control the register in the preprocessing phase. The ARMADILLO3 differs from ARMADILLO2 [3] by removing the XOR with a constant and adding a reduced-size substitution layer.

### 2.1 Modes of Operations in ARMADILLO3

*FIL-MAC.* The fixed input-length message authentication code is required in RFID applications. The output X′ is used for authentication of the tag. In applications such as Pathchecker [20], the secret key of the RFID tag is renewed with the value H′.

*Hashing.* For a variable-length input message we use the strengthened Merkle-Damgård construction [18,7]. The ARMADILLO3 is used as a compression function. The value H is taken as the IV and the compression function ARMADILLO3

produces H' which is the new IV. The value X is a message block to be processed. Such construction is similar to a sponge construction proposed in [4]. The inner function of ARMADILLO3 could also be used in a sponge construction as an alternative to our construction.

*PRNG, PRF.* In this mode we use the $j$ most significant bits of the output value $(\mathsf{H'}\|\mathsf{X'}) = \mathsf{ARMADILLO3}(\mathsf{H}, \mathsf{X})$, where $j$ is a parameter. The input value X is chosen sequentially, and can be sent in clear for the resynchronization purposes for a self-synchronizing stream cipher.

### 2.2 The Permutation Pair for ARMADILLO3

We introduce a concept of Hierarchical Permutation which ensures the avalanche effect in small number of rounds even if the selection of permutations is under full control of the attacker. Given the set of indices $X$ and a set of indices $S$, covered by S-boxes, we define a height of $i \in X$ for the permutation $\pi$ as $h(i) = \min_j\{j : \pi^j(i) \in S\}$. We now explain how to build the Hierarchical permutation for $t \times t$ S-boxes, and give a concrete example for the case $t = 3$. We suppose that the layer of S-boxes covers bits $[0, tr-1]$. We define sets of indices $A_i$, $B_i$, and $C_i$ so that $\sum_{i=0}^{h} |A_i| + \sum_{i=0}^{h-1} |B_i| + \sum_{i=0}^{h-2} |C_i| = k$, i.e., $A_i$, $B_i$, $C_i$ are partitions of $[0, k-1]$. In the case $t = 3$ let $a$, $b$ and $c$ be integers such that $a + b + c = 3r$, and similarly in the case $t = 4$ let $a$, $b$, $c$ and $d$ be integers such that $a + b + c + d = 4r$. Ideally, we would have $a = b = c = r$, but this is not always possible. So, we target $a \approx b \approx c \approx r$. We define several sets $A_i$, $B_i$ and $C_i$ to partition $\{0, 1, 2, \ldots, k-1\}$: $A_h = \{a+b+c, \ldots, 2a+b+c-1\}$, $A_{h-1} = \{2a+b+c, \ldots, 3a+b+c-1\}$, $B_{h-1} = \{3a+b+c, \ldots, 3a+2b+c-1\}$, $A_{h-2} = \{3a+2b+c, \ldots, 4a+2b+c-1\}$, $B_{h-2} = \{4a+2b+c, \ldots, 4a+3b+c-1\}$, $C_{h-2} = \{4a+3b+c, \ldots, 4a+3b+2c-1\}$, $A_{h-3} = \{4a+3b+2c, \ldots, 5a+3b+2c-1\}$, $\ldots$, $A_0 = \{0, 1, 2, \ldots, a-1\}$, $B_0 = \{a, \ldots, a+b-1\}$, $C_0 = \{a+b, \ldots, a+b+c-1\}$. In what follows, $AB_i$ denotes the union of $A_i$ and $B_i$. $ABC_i$ denotes the union of $A_i$, $B_i$, and $C_i$. We further define pairwise disjoint sets $A_{h+1}$, $B_h$, and $C_{h-1}$ so that $A_{h+1} \cup B_h \cup C_{h-1} = S = ABC_0$ and that output bits from an S-box fall into different sets $A_{h+1}$, $B_h$ and $C_{h-1}$ (with very few exceptions). In the case when $|A_{h+1}| = |B_h| = |C_{h-1}|$ we set $A_{h+1} = \{3i : i \in [0, r-1]\}$, $B_h = \{3i+1 : i \in [0, r-1]\}$ and $C_{h-1} = \{3i+2 : i \in [0, r-1]\}$ or in case of $4 \times 4$ S-boxes $A_{h+1} = \{4i : i \in [0, r-1]\}$, $B_h = \{4i+1 : i \in [0, r-1]\}$, $C_{h-1} = \{4i+2 : i \in [0, r-1]\}$ and $D_{h-2} = \{4i+3 : i \in [0, r-1]\}$. In the case when the sets $A_{h+1}$, $B_h$ and $C_{h-1}$ are not balanced, we select the excess elements to be far from each other. We construct $\sigma$ such that $A_{h+1}$ is mapped to $A_h$. $B_h$ is mapped to $B_{h-1}$. $C_{h-1}$ is mapped to $C_{h-2}$. $A_h$ is mapped to $A_{h-1}$. $AB_{h-1}$ is mapped to $AB_{h-2}$. $ABC_{h-2}$ is mapped to $ABC_{h-3}$. $ABC_{h-3}$ is mapped to $ABC_{h-4}$. Etc. These constraints are depicted in Fig. 2. Note that

$$\mathcal{T} = \{A_{h+1}, B_h, C_{h-1}, A_h, AB_{h-1}, ABC_{h-2}, ABC_{h-3}, \ldots, ABC_1\}$$

is a partition of $\{1, \ldots, k\}$, since $A_{h+1} \cup B_h \cup C_{h-1} = ABC_0$. From the construction of Hierarchical Permutation, we have that every $i$ in $A_i$ $(i \leq h+1)$ and $B_i$ $(i < h)$, $C_i$ $(i < h-1)$ has height $i$. The unbalanced-height structure makes it such that the output bits of the S-box will meet the S-box layer every $h+1$, $h$, or $h-1$ iterations. That is, two bits of the same height are likely to have different heights after going through their respective S-box. When the structure is balanced with $a = b = c$, we can take $A_{h+1}$ to the list of the first output bits of S-boxes, $B_h$ to the list of the second output bits of the S-boxes, and $C_{h-1}$ to the list of the third output bits of the S-boxes. This way, two bits going out from the same S-box cannot meet in the same S-box the next time since they have different height. When the structure is unbalanced, it should be close to the same situation. Exceptions to this rule are called "collisions". In the case of ARMADILLO3-A1/4 we have $a = 9$, $b = 11$, and $c = 13$ for $r = 11$, $h = 4$. This gives coverage $\frac{33}{128} \approx \frac{1}{4}$, and $A_5$, $B_4$, $C_3$ as follows

- $A_5 = \{0, 3, 6, 9, 12, 18, 21, 24, 27\}$
- $B_4 = \{1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31\}$
- $C_3 = \{2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 15, 30\}$

Additionally, we check that for $\sigma'(x) = \sigma^{h(x)}(x)$ we have

$$\left\lfloor \frac{\sigma'(15)}{3} \right\rfloor \neq \left\lfloor \frac{\sigma'(17)}{3} \right\rfloor \text{ and } \left\lfloor \frac{\sigma'(30)}{3} \right\rfloor \neq \left\lfloor \frac{\sigma'(32)}{3} \right\rfloor .$$

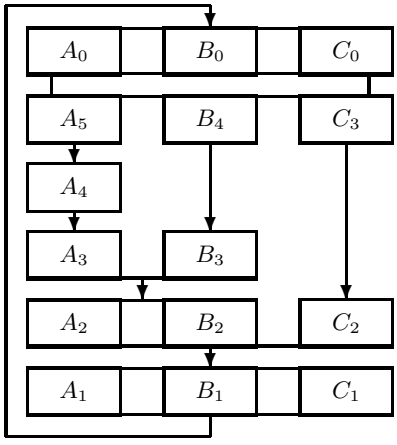


**Fig. 2.** Hierarchy for Permutations with $h = 4$ and $t = 3$

We provide additional criteria to achieve the highest possible diffusion for a pair of Hierarchical permutations.

**Definition 1 (Distance).** *Let $\sigma_0$, $\sigma_1$ be a pair of the permutations on a set $\{1,\ldots,k\}$. We say that they have a distance $d_\ell$ at level $\ell$ where*

$$d_\ell(\sigma_0,\sigma_1) = \min_{U,V\in\{0,1\}^\ell, U\neq V} |\{j : \sigma_U(j) \neq \sigma_V(j)\}|$$

For the selection of $\sigma_0$ and $\sigma_1$ we maximize $d_2(\sigma_0,\sigma_1)$. The high distance $d_1$ is associated with small number of fixed points of permutation $\sigma_0\sigma_1^{-1}$. Similarly, the high distance $d_2$ is associated with small number of fixed points of the permutation $\sigma_U\sigma_V^{-1}$, for all $U,V\in\{0,1\}^2$. Otherwise, for some values $U$ and $V$ of the control register, the bit from position $i$ is mapped to $\sigma_U(i)=\sigma_V(i)=j$ for some $j$. This allows the attacker to predict the behavior of the unknown permutation. In ARMADILLO3-A1/4, which is defined in Section 4, we require $d_2(\sigma_0,\sigma_1)=127$, which means there is at most one index $i$ which is mapped to the same index $j$ by the permutations $\sigma_U$ and $\sigma_V$, where $U,V\in\{0,1\}^2$.

**Definition 2 (Graph $\Omega_{\sigma_0\sigma_1,U}$).** *Let $U$ be a bitstring and $r$, $t$ be integers (representing the number and the types of S-boxes respectively). We define a multigraph $\Omega_{\sigma_0\sigma_1,U}=(V,E)$ for permutations $\sigma_0$, $\sigma_1$ and parameters $r$ and $t$ as follows:*

$$V=\bigcup_{i=0}^{|U|} V^i \qquad V^i=\{v_{i,j} : j\in\{1,\ldots,k\}\} \qquad E=\bigcup_{i=1}^{|U|}\left(E^i\cup S^i\right)$$

$$E^i=\left(\underbrace{\left(v_{i-1,j}\ ,\ v_{i,\sigma_{U_i}(j)}\right),\ldots,\left(v_{i-1,j}\ ,\ v_{i,\sigma_{U_i}(j)}\right)}_{t};\ rt<j\le k\right)$$

$$S^i=\left(\left(v_{i-1,j*t+a+1}\ ,\ v_{i,\sigma_{U_i}(j*t+b+1)}\right);\ j<r \text{ and } a,b\in\{0,\ldots,t-1\}\right)$$

*where $U_i$ denotes the $i$th bit of $U$.*

The set $E^i$ is a multiset of edges between level $i-1$ and level $i$ where every edge is taken $t$ times, and the set $S^i$ is a set of edges representing the S-boxes, i.e., for every S-box we have a complete bipartite graph $t\times t$. Therefore, the definition 2 gives a $t$-regular multigraph (since some edges are repeated $t$ times), i.e., $\Omega_{\sigma_0\sigma_1,U}$ is an expander graph. Combinatorically, the expander graphs are highly connected sparse graphs, probabilistically expander graphs behave like random graphs. Let $\lambda_0$ denote the second largest eigenvalue of adjacency matrix of graph $G$. We now introduce a new criterion which measures the randomness of the graph $\Omega_{\sigma_0\sigma_1,U}$. This criterion is based on the expander graph theory, the reader is referred to [24] for details. We recall that an expander graph is a $\tau$-regular graph $G$ with expansion factor $D(G)>c$ for some constant $c>0$ and some $\tau\in\mathbb{N}$, where the expansion factor $D(G)$ is given by the following formula. Let $\delta(S)$ denote a set of edges neighboring of $S$, then

$$D(G)=\min_{0<|S|\le\frac{|V|}{2}}\frac{|\delta(S)|}{|S|}$$

Let $\sigma_0$, and $\sigma_1$ be permutations. We say that the graph $\Omega_{\sigma_0\sigma_1,U}$ diffuses if for all $v \in V^0$ and $w \in V^{|U|}$ there exists an oriented path from $v$ to $w$ in graph $\Omega_{\sigma_0\sigma_1,U}$. We say that the pair $(\sigma_0, \sigma_1)$ has diffusion level $\mathrm{dif}_{\sigma_0,\sigma_1}$ where

$$\mathrm{dif}_{\sigma_0,\sigma_1} = \min\{h : \forall U \in \{0,1\}^h \text{ graph } \Omega_{\sigma_0\sigma_1,U} \text{ diffuses }\}.$$

For the selection of $\sigma_0$, $\sigma_1$ we minimize $\mathrm{dif}_{\sigma_0,\sigma_0}$, $\mathrm{dif}_{\sigma_1,\sigma_1}$, and $\mathrm{dif}_{\sigma_0,\sigma_1}$.

Additionally, we verify the randomness of selected pair permutations. Let $G = (V, E)$ be a $4t$-regular multigraph where $V = V(\Omega_{\sigma_0\sigma_0,0})$ and $E = E(\Omega_{\sigma_0\sigma_0,0}) \cup E(\Omega_{\sigma_1\sigma_1,0}) \cup E(\Omega_{\sigma_0^{-1}\sigma_0^{-1},0}) \cup E(\Omega_{\sigma_1^{-1}\sigma_1^{-1},0})$. We require the second largest eigenvalue $\lambda_0$ of adjacency matrix of multigraph $G$ to be small. According to the Expander mixing lemma, we have

$$\left| E(S,T) - \frac{d|S||T|}{n} \right| \leq \lambda_0 \sqrt{|S||T|}.$$

This criterion helps us to select pair of permutations which minimizes $\mathrm{dif}_{\sigma_0,\sigma_1}$, where $E(S,T)$ denotes the number of edges between $S$ and $T$, and $d$ is the degree of each vertex (in our case $4t$), and $n$ is the total number of vertices (in our case $2k$). It allows to quantify the diffusion coming from the the data dependent permutation layer, as the high number of edges means the higher diffusion. The Expander mixing lemma gives an estimate, on how far we are from an optimum ($\Omega_{\sigma_0\sigma_1,U}$ behaving like a random d-regular graph). We refer the reader to [3] for further analysis of ARMADILLO family based on expander graphs.

## 3 The Security Analysis of ARMADILLO3 Function

### 3.1 Differential and Linear Cryptanalysis

The differential and linear cryptanalysis is complicated by the fact, that the attacker does not know the sequence $Y$, i.e., the sequence in which permutations $\sigma_0$ and $\sigma_1$ are selected. In the differential cryptanalysis, the attacker looks for differentials which propagate with a high probability through the cipher. Since the permutation Y is not fixed while it varies according to the input X and the input H, the input Y is hard to predict, i.e., it is hard for an attacker to find a good differential path and mount differential cryptanalysis. Similarly, the linear relations between input and output of ARMADILLO3-A1/4 depend on the value $Y$ which is unpredictable, and therefore obtaining a good linear characteristic is hard. Moreover, the S-boxes are selected to provide good security guarantees against both differential and linear cryptanalysis, therefore even if the value Y is known to the attacker, the differential/linear cryptanalysis should be impossible. From LAT, resp. DDT we can see that any linear characteristic resp. differential have probability at most $\frac{1}{4}$. Therefore, any differential/linear characteristic over $(h+1)$ rounds will have a probability at most $\frac{1}{4}$ from the construction of the S-box. Consequently, any $(h+1) \cdot g$ round differential characteristic will have

probability $2^{-2g}$ and any $(h+1) \cdot g$ round linear characteristic will have a correlation of at most $2^{-2g}$. In the case of ARMADILLO3-A1/4, we have $k = 128$, and $h = 4$. Therefore, the best differential/linear characteristic has probability probability at most $2^{-2\frac{128}{5}} \approx 2^{-51}$. The security margin is obtained from the fact that the attacker have to know the values in the control register to be able to use such differential/linear characteristics.

### 3.2 High Order Differentials and Algebraic Attacks

The number of S-boxes and the structure of the selected permutations ensure that the degree of underlying ANF equations is close to maximum and the data-dependency of the design ensures that these equations do not have any simple structural properties.

### 3.3 Statistical Saturation Attacks

The statistical saturation attacks were introduced in [6] against PRESENT [5]. Such attack is based on low diffusion trails in the linear layer of PRESENT. However, as the low diffusion trails are not constant and the permutations are selected in such a way that the distance of $\sigma_0$, $\sigma_1$ and their compositions $\sigma_0\sigma_0$, $\sigma_0\sigma_1$, $\sigma_1\sigma_0$, and $\sigma_1\sigma_1$ are maximal, the low diffusion trace changes substantially for different sequences $Y$. As the value $Y$ depends both on the secret key $C$ and the challenge $U$, and since $\mathsf{Y} = P(\mathsf{X}, \mathsf{H} \| \mathsf{X})$, we expect that it would be difficult for an attacker to control the low diffusion paths and to utilize them at the same time.

### 3.4 The Internal Collision Attack

The attacker can try to force an internal collision. The internal collision can appear during the computation, since it is possible to find a pair $(Y, Y')$ such that $Y_{\sigma_0} = Y'_{\sigma_1}$. Let consider a single step of ARMADILLO3, i.e., $P(p\|b, Z) = P(p, S(Z_{\sigma_b}))$ and let consider how can we obtain an internal collision $S(Z_{\sigma_b}) = S(Z'_{\sigma_{b'}})$. We need to have $Z' = Z_{\sigma_b^{-1}\sigma_{b'}}$ as the substitution layer is bijective. This means that either $Z = Z'$ for $b = b'$ or we have a prescribed relation between $Z$ and $Z'$. This allows the attacker to force the value $Y$ (from the preprocessing phase) to be the same for different inputs $U$ and $U'$. However, the attacker has then no control over the propagation of the difference in the computation $\mathsf{ARMADILLO3}(\mathsf{H} \| \mathsf{X})$ and $\mathsf{ARMADILLO3}(\mathsf{H} \| \mathsf{X}')$.

**3.4.1 Invariant States.** The relation $Z' = Z_{\sigma_b^{-1}\sigma_{b'}}$ also allows the attacker to find invariant internal state, i.e, a state $W$ such that $W_{\sigma_0} = W_{\sigma_1}$. Therefore, the invariant state has to follow an equation $W = W_{\sigma_b^{-1}\sigma_{b'}}$ which means that bits of $W$ are constant for each cycle of permutation ${\sigma_0}^{-1}\sigma_1$. Therefore, selecting $\sigma_0$ and $\sigma_1$ so that ${\sigma_0}^{-1}\sigma_1$ has long cycles is a good protection against these types of

attacks. We note that a cycle of permutation ${\sigma_0}^{-1}\sigma_1$ is a subset of the set $A$, $B$ or $C$. Therefore, we cannot obtain a pair of permutations, such that ${\sigma_0}^{-1}\sigma_1$ has less than $h+1$ cycles. Moreover, as the S-boxes takes input/output bits from different cycles and changes the parity and therefore if $Z'^i = Z^i_{\sigma_b^{-1}\sigma_{b'}} \Longrightarrow Z'^{i+1} \neq Z^{i+1}_{\sigma_b^{-1}\sigma_{b'}}$ which means there is no invariant state in ARMADILLO3.

# 4 ARMADILLO3-A1/4

This section gives a concrete proposal of ARMADILLO3-A1/4 with $k = 128$ $t = 3$ and $r = 11$ and "coverage rate" $\approx \frac{1}{4}$. This instance has only 11 $3 \times 3$ S-boxes, which makes it an interesting design for study by cryptographic community. We give a description of S-boxes and the pair of permutations. We argue about the security of ARMADILLO3 based on the properties of Hierarchical Permutations and the low second largest eigenvalue of the selected pair of permutations.

## 4.1 The S-box Layer of ARMADILLO3

The function $S$ is defined as follows:

$$S(z_1\|z_2\|z_3\|\ldots\|z_{31}\|z_{32}\|z_{33}\|\ldots\|z_{128}) = s(z_1,z_2,z_3)\|\ldots\|s(z_{31},z_{32},z_{33})\|z_{34}\|\ldots\|z_{128}$$

The reader should notice that the indices covered by S-boxes correspond to the sets $A_0$, $B_0$, and $C_0$ of the Hierarchical Permutation.

**Table 1.** ARMADILLO3 variants with "coverage" $\frac{1}{4}$

| version | $k$ | $c$ | $m$ | $r$ | $t$ |
|---|---|---|---|---|---|
| ARMADILLO3-A1/4 | 128 | 80 | 48 | 11 | 3 |
| ARMADILLO3-B1/4 | 192 | 128 | 64 | 16 | 3 |
| ARMADILLO3-C1/4 | 240 | 160 | 80 | 20 | 3 |
| ARMADILLO3-D1/4 | 288 | 192 | 96 | 24 | 3 |
| ARMADILLO3-E1/4 | 384 | 256 | 128 | 32 | 3 |

**Table 2.** Implementation results with throughput at 1MHz, using 0.35$\mu$m

| Algorithm | Block (bits) | Key (bits) | Area (GE) | Time (cycles/block) | Cell power (mW) |
|---|---|---|---|---|---|
| ARMADILLO3-A1/4 | 48 | 80 | 4048 | 176 | 60 |
| ARMADILLO3-B1/4 | 64 | 128 | 6065 | 256 | 89 |
| ARMADILLO3-C1/4 | 80 | 160 | 7576 | 320 | 110 |
| ARMADILLO3-D1/4 | 96 | 192 | 9133 | 384 | 134 |
| ARMADILLO3-E1/4 | 128 | 256 | 12239 | 512 | 177 |

**Table 3.** Implementation comparison for hash functions with throughput at 100 kHz

| Algorithm | Digest (bits) | Block (bits) | Area (GE) | Time (cycles/block) | Throughput (kb/s) | Logic ($\mu$m) | FOM (nanobit/cycle.GE$^2$) |
|---|---|---|---|---|---|---|---|
| ARMADILLO2-A | 80 | 48 | 4 030 | 44 | 109 | 0.18 | 67.17 |
| PHOTON [13] | 80 | 20 | 1168 | 132 | 775 | 0.18 | 59.27 |
| ARMADILLO3-A1/4 | 80 | 48 | 4 302 | 44 | 109 | 0.35 | 58.95 |
| ARMADILLO2-A | 80 | 48 | 2 923 | 176 | 27 | 0.18 | 31.92 |
| ARMADILLO3-A1/4 | 80 | 48 | 2 991 | 176 | 27 | 0.35 | 30.49 |
| PHOTON [13] | 80 | 20 | 865 | 708 | 144 | 0.18 | 20.12 |
| KECCAK-f[400][16] | 128 | 128 | 5 090 | 32 | 200 | 0.18 | 110.41 |
| H-PRESENT-128[5] | 128 | 128 | 4 256 | 32 | 200 | 0.18 | 110.41 |
| ARMADILLO2-B | 128 | 64 | 6 025 | 64 | 1000 | 0.18 | 27.55 |
| ARMADILLO3-B1/4 | 128 | 64 | 6 409 | 64 | 1000 | 0.35 | 24.34 |
| PHOTON [13] | 128 | 16 | 1 708 | 156 | 422 | 0.18 | 15.06 |
| ARMADILLO2-B | 128 | 64 | 4 353 | 256 | 250 | 0.18 | 13.19 |
| ARMADILLO3-B1/4 | 128 | 64 | 4 449 | 256 | 250 | 0.35 | 12.62 |
| MD5 [12] | 128 | 512 | 8 400 | 612 | 83.66 | 0.13 | 11.86 |
| U_QUARK[2] | 136 | 8 | 2 392 | 68 | 476 | 0.18 | 8.51 |
| PHOTON [13] | 128 | 16 | 1 122 | 996 | 66 | 0.18 | 5.48 |
| U_QUARK[2] | 136 | 8 | 1 379 | 544 | 87 | 0.18 | 3.20 |
| ARMADILLO2-C | 160 | 80 | 7 492 | 80 | 100 | 0.18 | 17.81 |
| PHOTON [13] | 160 | 36 | 2 117 | 180 | 731 | 0.18 | 17.01 |
| ARMADILLO3-C1/4 | 160 | 80 | 7 972 | 80 | 100 | 0.35 | 15.72 |
| ARMADILLO2-C | 160 | 80 | 5 406 | 320 | 250 | 0.18 | 8.55 |
| ARMADILLO3-C1/4 | 160 | 80 | 5 526 | 320 | 250 | 0.35 | 8.18 |
| D_QUARK[2] | 176 | 16 | 2 819 | 88 | 616 | 0.18 | 8.08 |
| PHOTON [13] | 160 | 36 | 1 396 | 1332 | 98 | 0.18 | 5.28 |
| SHA-1 [12] | 160 | 512 | 8 120 | 1 274 | 40.18 | 0.35 | 6.10 |
| D_QUARK[2] | 176 | 16 | 1 702 | 704 | 76 | 0.18 | 2.77 |
| ARMADILLO2-D | 192 | 96 | 8 999 | 96 | 100 | 0.18 | 12.35 |
| ARMADILLO3-D1/4 | 192 | 96 | 9 575 | 96 | 100 | 0.35 | 10.90 |
| C-PRESENT-192[5] | 192 | 192 | 8 048 | 108 | 59.26 | 0.18 | 9.15 |
| ARMADILLO2-D | 192 | 96 | 6 554 | 384 | 25 | 0.18 | 5.82 |
| ARMADILLO3-D1/4 | 192 | 96 | 6 698 | 384 | 25 | 0.35 | 5.37 |
| MAME [26] | 256 | 256 | 8 100 | 96 | 266.67 | 0.18 | 40.64 |
| ARMADILLO2-E | 256 | 128 | 11 914 | 128 | 100 | 0.18 | 7.05 |
| ARMADILLO3-E1/4 | 256 | 128 | 12 682 | 128 | 100 | 0.35 | 6.22 |
| SHA-256 [12] | 256 | 512 | 10 868 | 1 128 | 45.39 | 0.35 | 3.84 |
| ARMADILLO2-E | 256 | 128 | 8 653 | 512 | 25 | 0.18 | 3.34 |
| ARMADILLO3-E1/4 | 256 | 128 | 8 845 | 512 | 25 | 0.35 | 3.19 |
| PHOTON [13] | 256 | 32 | 4 362 | 156 | 650 | 0.18 | 2.94 |
| PHOTON [13] | 256 | 32 | 2 177 | 996 | 1034 | 0.18 | 1.85 |

**Table 4.** Implementation comparison for encryption with throughput at 100 kHz

| Algorithm | Key (bits) | Block (bits) | Area (GE) | Time (cycles/block) | Throughput (kb/s) | Logic ($\mu$m) | FOM (nanobit/cycle.GE$^2$) |
|---|---|---|---|---|---|---|---|
| PRESENT-80 [5] | 80 | 64 | 1 570 | 32 | 200 | 0.18 | 811.39 |
| Grain [14] | 80 | 1 | 1 294 | 1 | 100 | 0.13 | 597.22 |
| KTANTAN64 [8] | 80 | 64 | 927 | 128 | 50 | 0.13 | 581.85 |
| KATAN64 [8] | 80 | 64 | 1 269 | 85 | 75 | 0.13 | 467.56 |
| ARMADILLO2-A | 80 | 128 | 4 030 | 44 | 291 | 0.18 | 179.12 |
| ARMADILLO3-A1/4 | 80 | 128 | 4 302 | 44 | 291 | 0.35 | 157.19 |
| Trivium [9] | 80 | 1 | 2 599 | 1 | 100 | 0.13 | 148.04 |
| PRESENT-80 [5] | 80 | 64 | 1 075 | 563 | 11 | 0.18 | 98.37 |
| ARMADILLO2-A | 80 | 128 | 2 923 | 176 | 73 | 0.18 | 85.12 |
| ARMADILLO3-A1/4 | 80 | 128 | 2 991 | 176 | 73 | 0.35 | 81.30 |
| PRESENT-128 [5] | 128 | 64 | 1 886 | 32 | 200 | 0.18 | 562.27 |
| HIGHT [15] | 128 | 64 | 3 048 | 34 | 189 | 0.25 | 202.61 |
| TEA [25] | 128 | 64 | 2 355 | 64 | 100 | 0.18 | 180.31 |
| ARMADILLO2-B | 128 | 192 | 6 025 | 64 | 300 | 0.18 | 82.64 |
| ARMADILLO3-B1/4 | 128 | 192 | 6 409 | 64 | 300 | 0.35 | 73.03 |
| ARMADILLO2-B | 128 | 192 | 4 353 | 256 | 75 | 0.18 | 39.58 |
| ARMADILLO3-B1/4 | 128 | 192 | 4 449 | 256 | 75 | 0.35 | 37.89 |
| AES-128 [11] | 128 | 128 | 3 400 | 1 032 | 12 | 0.35 | 10.73 |

**Table 5.** Permutation $\sigma_0$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 33 | 61 | 92 | 34 | 52 | 86 | 36 | 54 | 89 | 41 | 59 | 93 | 39 | 53 | 84 | 94 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 55 | 88 | 35 | 57 | 90 | 37 | 58 | 85 | 38 | 56 | 82 | 40 | 51 | 91 | 83 | 60 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 87 | 50 | 45 | 43 | 49 | 42 | 47 | 44 | 48 | 46 | 78 | 69 | 70 | 73 | 79 | 63 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 72 | 75 | 67 | 81 | 71 | 64 | 76 | 66 | 77 | 62 | 65 | 80 | 68 | 74 | 118 | 119 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 100 | 122 | 127 | 107 | 108 | 117 | 109 | 121 | 111 | 105 | 110 | 98 | 97 | 96 | 120 | 103 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 99 | 115 | 116 | 123 | 126 | 124 | 114 | 113 | 125 | 95 | 106 | 104 | 101 | 102 | 112 | 0 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 24 | 29 | 2 | 13 | 6 | 25 | 16 | 10 | 32 | 21 | 15 | 18 | 1 | 27 | 7 | 11 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 17 | 22 | 19 | 31 | 9 | 30 | 4 | 8 | 12 | 28 | 5 | 20 | 26 | 3 | 23 | 14 |

**Table 6.** Permutation $\sigma_1$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 34 | 53 | 88 | 37 | 61 | 82 | 35 | 51 | 86 | 36 | 58 | 85 | 41 | 55 | 94 | 90 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 57 | 87 | 40 | 52 | 89 | 38 | 59 | 83 | 33 | 60 | 84 | 39 | 56 | 92 | 93 | 54 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 91 | 46 | 49 | 42 | 47 | 48 | 44 | 43 | 50 | 45 | 64 | 65 | 67 | 80 | 75 | 76 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 66 | 71 | 68 | 63 | 73 | 70 | 72 | 74 | 79 | 77 | 62 | 78 | 69 | 81 | 104 | 116 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 113 | 106 | 126 | 105 | 95 | 119 | 127 | 124 | 100 | 122 | 117 | 114 | 112 | 123 | 96 | 102 |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 125 | 120 | 103 | 110 | 98 | 99 | 97 | 111 | 121 | 115 | 109 | 118 | 108 | 101 | 107 | 25 |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 5 | 18 | 22 | 21 | 12 | 16 | 23 | 4 | 26 | 32 | 11 | 0 | 7 | 30 | 17 | 29 |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| 13 | 15 | 8 | 24 | 6 | 20 | 9 | 14 | 19 | 31 | 1 | 3 | 10 | 27 | 28 | 2 |

The proposed instance of ARMADILLO3 has $3 \times 3$ S-boxes. The S-box satisfies the following equations. For an input $(x_0, x_1, x_2)$ and output $(y_0, y_1, y_2) = S(x_0, x_1, x_2)$. We have

$$\begin{cases} y_0 = x_0 + x_1 + x_2 + x_0 * x_1 + 1 \\ y_1 = x_0 + x_1 + x_0 * x_2 + 1 \\ y_2 = x_0 + x_1 * x_2 + 1 \end{cases}$$

The permutation defined by the S-box expressed in decimal is a non-linear cycle: (0 7 5 3 2 4 6 1). Thus, with 3 AND, 6 XOR and 2 NOT we can implement a single ARMADILLO S-box in hardware.

### 4.2 The Permutation Pair

For a selection of a good pair of permutations, we create a pool of Hierarchical Permutations with low diffusion $\text{dif}_\sigma$ and a small number of long cycles. Afterwards we select a pair which achieves a full diffusion in 25 rounds and the second largest eigenvalue of graph $\Omega_{\sigma_0\sigma_1,U}$ (Def. 2) is $\lambda_0 = 9.36$. The permutation $\sigma_0$ is given in Table 5 and the permutation $\sigma_1$ is given in Table 6.

### 4.3 Test Vectors and Implementation

We give a test vector for the ARMADILLO3-A1/4 with the S-boxes above and the permutation pair $\sigma_0$ in Table 5 and $\sigma_1$ in Table 6.

$$\text{ARMADILLO3-A1/4}(0^k) = \text{0xF89FCBAB 0x47D36AF6 0xDC51602D 0x31C3EEA1}$$
$$\text{ARMADILLO3-A1/4}(1^k) = \text{0x7C7A0E1F 0xBA9214DF 0x5FC3CD65 0x374EB994}$$

The synthesis results at 1MHz with typical 0.35$\mu$m library and 2.2V voltage supply can be found in Table 2. We give the details for several instances with "coverage" $\frac{1}{4}$. We give the figures for several variants of ARMADILLO3 depending on the size of internal state, see Table 1 for details on the variants with coverage "$\frac{1}{4}$". Concrete proposals for variants ARMADILLO3-B1/4, ARMADILLO3-C1/4, ARMADILLO3-D1/4, and ARMADILLO3-E1/4 are omitted due to the lack of space.

## 5 Conclusion

We introduced a new hardware oriented class of cryptographic primitives ARMADILLO3. Our design of ARMADILLO3 is based on data-dependent permutations and a reduced size substitution layer. To meet the criteria for good confusion and diffusion layers, we introduce the concept of Hierarchical Permutations. Such permutations give guarantees, that the diffusion is fast despite the reduced substitution layer. The applications for ARMADILLO3 include MACs, hashing and PRNG. We propose an instance ARMADILLO3-A1/4 to encourage the study of ARMADILLO3. The ARMADILLO3-A1/4 consists of a pair of carefully selected Hierarchical Permutations and 11 $3 \times 3$ S-boxes.

## References

1. Abdelraheem, M.A., Blondeau, C., Naya-Plasencia, M., Videau, M., Zenner, E.: Cryptanalysis of ARMADILLO2. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 308–326. Springer, Heidelberg (2011)
2. Aumasson, J.-P., Henzen, L., Meier, W., Naya-Plasencia, M.: QUARK: A lightweight hash. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 1–15. Springer, Heidelberg (2010)
3. Badel, S., Dağtekin, N., Nakahara Jr., J., Ouafi, K., Reffé, N., Sepehrdad, P., Sušil, P., Vaudenay, S.: ARMADILLO: A Multi-purpose Cryptographic Primitive Dedicated to Hardware. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 398–412. Springer, Heidelberg (2010)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
5. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash Functions and RFID Tags: Mind the Gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)

6. Collard, B., Standaert, F.X.: A Statistical Saturation Attack against the Block Cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
7. Damgård, I.B.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
8. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
9. De Cannière, C., Preneel, B.: Trivium - A Stream Cipher Construction Inspired by Block Cipher Design Principles. eSTREAM, ECRYPT Stream Cipher (2005)
10. Eastlake, D.E., Jones, P.E.: US Secure Hash Algorithm 1 (SHA1), `http://www.ietf.org/rfc/rfc3174.txt?number=3174`
11. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong authentication for RFID systems using the AES algorithm. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 357–370. Springer, Heidelberg (2004)
12. Feldhofer, M., Rechberger, C.: A Case Against Currently Used Hash Functions in RFID Protocols. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM Workshops 2006. LNCS, vol. 4277, pp. 372–381. Springer, Heidelberg (2006)
13. Guo, J., Peyrin, T., Poschmann, A.: The `PHOTON` Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011)
14. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)
15. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.-S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
16. Kavun, E.B., Yalcin, T.: A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 258–269. Springer, Heidelberg (2010)
17. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
18. Merkle, R.C.: A Fast Software One-Way Hash Function. J. Cryptology 3(1), 43–58 (1990)
19. Naya-Plasencia, M., Peyrin, T.: Practical cryptanalysis of `ARMADILLO2`. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 146–162. Springer, Heidelberg (2012)
20. Ouafi, K., Vaudenay, S.: Pathchecker: An RFID application for tracing products in Supply-chains. In: Batina, L. (ed.) Proceedings of RFIDSec 2009 (2009)
21. Federal Information Processing Standards Publications. Advanced Encryption Standard. Technical Report FIPS PUB 197, National Institute of Standards and Technology (November 2001)
22. Sepehrdad, P., Sušil, P., Vaudenay, S.: Fast Key Recovery Attack on ARMADILLO1 and Variants. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 133–150. Springer, Heidelberg (2011)
23. Shamir, A.: SQUASH – A new MAC with provable security properties for highly constrained devices such as RFID tags. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 144–157. Springer, Heidelberg (2008)

24. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bulletin of the AMS 43(4), 439–561 (2006)
25. Wheeler, D., Needham, R.: TEA, a Tiny Encryption Algorithm (1995)
26. Yoshida, H., Watanabe, D., Okeya, K., Kitahara, J., Wu, H., Küçük, Ö., Preneel, B.: MAME: A Compression Function with Reduced Hardware Requirements. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 148–165. Springer, Heidelberg (2007)