FIDES: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware

Begül Bilgin[1,2], Andrey Bogdanov[3], Miroslav Knežević[4], Florian Mendel[5], and Qingju Wang[1,6]

[1] KU Leuven, ESAT/COSIC and iMinds, Belgium
[2] University of Twente, EEMCS-DIES, The Netherlands
[3] Technical University of Denmark, Department of Mathematics, Denmark
[4] NXP Semiconductors, Belgium
[5] Graz University of Technology, IAIK, Austria
[6] Department of Computer Science and Engineering, Shanghai Jiao Tong University, China

Abstract. In this paper, we present a novel lightweight authenticated cipher optimized for hardware implementations called FIDES. It is an online nonce-based authenticated encryption scheme with authenticated data whose area requirements are as low as 793 GE and 1001 GE for 80-bit and 96-bit security, respectively. This is at least two times smaller than its closest competitors Hummingbird-2 and Grain-128a. While being extremely compact, FIDES is both throughput and latency efficient, even in its most serial implementations. This is attained by our novel sponge-like design approach. Moreover, cryptographically optimal 5-bit and 6-bit S-boxes are used as basic nonlinear components while paying a special attention on the simplicity of providing first order side-channel resistance with threshold implementation.

Keywords: Lightweight cryptography, authenticated encryption, keyed sponge, glitch-free masking, APN permutation, almost bent permutation.

1 Introduction

Motivation. Lightweight cryptography is a rapidly growing field, being motivated by real-world applications with limited budget to spend on cryptographic mechanisms but rather essential demands for security. Though numerous lightweight ciphers have been proposed (including the ISO/IEC standard PRESENT as well as more recent designs such as KATAN [14], LED [20], Piccolo [30]), extended security functionalities are being addressed much more rarely in the lightweight context. Indeed, apart from the cryptographic hash functions (with the domain quite densely covered by the notable designs of Quark [2],

G. Bertoni and J.-S. Coron (Eds.): CHES 2013, LNCS 8086, pp. 142–158, 2013.

Photon [19] and SPONGENT [7]), almost no other security functionalities have been intensively analyzed for lightweight applications[1].

This situation is rather surprising though, since non-encryption security functionalities are often of much higher value than secrecy, authenticity and *authenticated encryption* ranking highest among them — emphasized by the recently announced NIST-funded *CAESAR competition* for authenticated encryption [10]. Cryptographically speaking, it is rather straightforward to deploy a lightweight block cipher in a mode of operation to implement an authenticated encryption scheme. However, this usually requires multiple additional memory states, additional operations on top of a single block cipher call, or both.

Yet, the landscape of *dedicated* authenticated encryption targeting lightweight scenarios remains unexplored to a large extent. While ALE [8] has been recently proposed to address the issue of more lightweight authenticated encryption across various platforms, it is based on the AES round operation and the AES-128 key schedule that per se confines its lightweight properties in hardware, though facilitating a high performance in parallel software implementations, especially with the Intel AES instruction set. At the same time, Grain-128a [1] and Hummingbird-2 [17] are among the very small number of truly dedicated designs aimed at attaining the traditional lightweight design goals such as low area and low power, yielding estimated area requirements of 2770 GE and 2159 GE, respectively. Though Hummingbird-2 has been recently broken in the related-key model [29], Grain-128a remains unaffected so far. However, Grain-128a leaves a lot of room for improvement in terms of area consumption, being comparable to software-optimized AES-based ALE in this crucial parameter.

In this paper, we aim to address this lack of dedicated lightweight-optimized authenticated ciphers.

Our Contributions. We propose FIDES — an online single-pass nonce-based authenticated encryption algorithm with either 80-bit or 96-bit key, FIDES-80 and FIDES-96. We report the area consumption of 793 GE and 1001 GE correspondingly, which is about 2 times smaller than Hummingbird-2 and about 3 times more compact than Grain-128a, though for a slightly different security level. FIDES has a highly competitive throughput, even in most serial implementations. It comes with a built-in efficient dedicated masking scheme to thwart basic side-channel attacks. The gate count for the protected ASIC implementation of FIDES-80 and FIDES-96 is 2876 and 4792, respectively, which is comparable to the plain implementation of AES-based authenticated encryption schemes such as ALE.

While basing upon well-established security principles to account for security, FIDES attains its efficiency by a bunch of innovative means including:

– *Novel Design Approach:* Like SHA-3, FIDES alternates message input and unkeyed permutations. However, unlike sponge, it inputs message chunks in every round. As opposed to ASC-1 and ALE though, the rounds in our

[1] All these successful and sound lightweight primitives mentioned here – with the sole exception for Photon – have been proposed at CHES from 2007 to 2011.

construction are not keyed. The original sponge construction is rather redundant which is needed for the so called *hermetic sponge* claim. So we employ an automated technique for lower-bounding the number of active S-boxes which allows us to choose the positions and number of message injections in a way being both efficient and secure, by taking exactly as many security as we need.
- *Usage of Optimal S-Boxes With Respect to Differential and Linear Cryptanalysis:* FIDES is the first symmetric-key design — to the best of our knowledge — to use S-boxes optimal with respect to differential and linear cryptanalysis. Namely, in two variants of our design, we use the 5-bit AB (almost bent) and the 6-bit APN (almost perfect nonlinear) invertible S-boxes. The AB permutations have the optimal differential and linear properties for S-boxes and exist only in odd dimensions. The 6-bit APN permutation is optimal towards differential properties in even dimensions. The permutation we use is the only (up to extended affine equivalence) permutation in even dimension known to be APN and is due to Dillon [15].
- *Off-the-Shelf Glitch-Free Side-Channel Masking:* FIDES offers off-the-shelf glitch-free secret-sharing based masking. This is also the first effort as regards the side-channel resistant sharing of optimal S-boxes. Moreover, we offer the first systematic treatment of shared S-box implementations in dimensions larger than 4-bit. In fact, we searched in the class of 5-bit AB and 6-bit APN permutation for the S-box instances with lowest area requirements. So the efficient side-channel resistance is offered by the very design of our construction.

Thus, following these approaches, we are able to construct FIDES — an authenticated encryption scheme particularly suitable for constrained hardware implementations. It is the authenticated encryption design with the smallest footprint at both around 80 and 100 bits of security level available. At the same time, more in the spirit of the recent low-latency considerations [9,22], we have made every effort to ensure its time efficiency at the same time. It is the advantage of our novel design approach that allows us to attain both – a highly competitive footprint and a time-efficient implementation – simultaneously.

Organization. Section 2 specifies the design of FIDES and provide some basic design rationale. Section 3 provides a more detailed security analysis of FIDES. In Section 4, both lightweight and protected threshold implementations of FIDES are elaborated and a detailed comparison to the existing designs is given.

2 The Design

FIDES is an online single-pass nonce-based authenticated encryption algorithm. Its structure is similar to the duplex sponge construction [4] and follows the design principles of the Rijndael block cipher [13]. As Rijndael-256, FIDES is designed according to the wide trail strategy and operates on 4×8 internal state. We propose two variants of FIDES with two different security levels:

	b	k	n	t	r	security(bit)		
	(bit)	(bit)	(bit)	(bit)	(bit)	key recovery	state recovery	forgery
FIDES-80	160	80	80	80	10	80	80	80
FIDES-96	192	96	96	96	12	96	96	96

The encryption/authentication procedure of FIDES accepts a key K with k bits, a message m, associated data a and a nonce N with n bits. The encryption/authentication procedure outputs the ciphertext c of exactly the same bit length as the message m and the authentication tag T of t bits for both the message m and associated data a. Its decryption/verification procedure accepts key K, ciphertext c, associated data a, nonce N and tag T. It returns the decrypted message m if the tag is correct or $\perp$ otherwise.

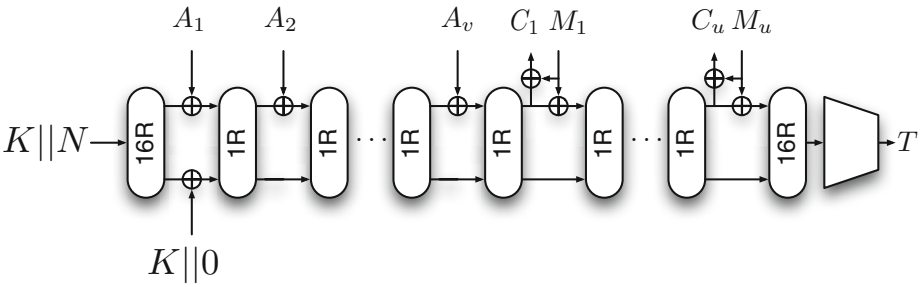


Fig. 1. The encryption/authentication operation of FIDES

The encryption/authentication operation of FIDES is given in Figure 1 and can be described in five steps:

Padding: The padding of FIDES is very simple. It appends a single "1" and the smallest number of zeroes to the message m such that the length of the result is a multiple of the required block length. The resulting padded message is split into u blocks of r bits each, $M_1 \| \ldots \| M_u$. Note that for associated data the same padding method is used and the padded associated data is split into v blocks of again r bits each, $A_1 || \ldots || A_v$.

Initialization: The initialization of FIDES is based on the Even-Mansour construction [18]. The 4×8 internal state is initialized with the key K and the nonce N. Then the internal state of $b = k + n$ bits is updated by applying the FIDES round transformation 16 times. Finally, the key K is xored to the internal state again. Now the internal state is initialized.

Processing Associated Data: If there is only one padded associated data block, then A_1 is xored to the internal state in row 3 at positions $0, 2$ and one proceeds with processing the padded message immediately. Otherwise, if there are at least two padded associated data blocks, associated is processed block by block: The internal state is updated using the FIDES round

transformation and then the next block is xored to the internal state in row 3 at positions 0 and 2.

Processing Message: The padded message is processed block by block: The internal state is updated using the FIDES round transformation. Then two elements of the internal state in row 3 at positions 0 and 2 are xored to the current block of the message to produce the according ciphertext block. Finally, the current block of the message is also xored to the internal state at the same positions.

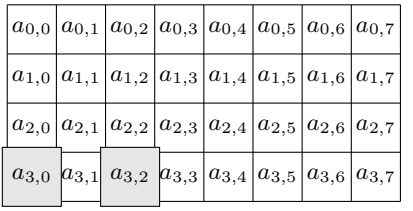


Fig. 2. The injection layer of FIDES

Finalization: The internal state is updated by applying the FIDES round transformation 16 times. The output is truncated to 80 (resp. 96) bits and returned as the authentication tag T for the message and associated data.

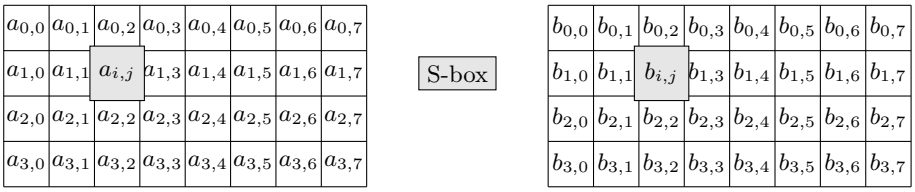
The decryption/verification procedure is defined correspondingly. The only two differences are that one works with the ciphertext $c = C_1||\ldots||C_u$ instead of the message m while xoring with the stream and that the supplied tag value T is compared to the one computed by the algorithm. We want to stress that only if the tag is correct the decrypted message is returned.

2.1 The Round Transformations of FIDES

In the following, we briefly describe the round transformations of FIDES. It is designed according to the wide trail strategy [12] and its structure is very similar to the Rijndael block cipher [13]. It operates on a 4×8 state of 5 (resp. 6) bits and updates the internal state by means of the sequence of transformations

$$\mathsf{CA} \circ \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB} \ .$$

SubBytes (SB). The SubBytes step is the only non-linear transformation of the algorithm. It is a permutation consisting of an S-box applied to each element of the 4×8 state. This permutation is an almost bent (AB) permutation (Table 2) in FIDES-80 and almost perfect nonlinear (APN) permutation (Table 1) in FIDES-96.



AB permutations which are a subset of APN permutations provide optimum security against linear and differential cryptanalysis [11]. Unfortunately, they only exist if the size of the S-box is odd and there are only four 5-bit vectorial AB function known so far. On the other hand, APN permutations exist even if the size is even but they provide optimum security only against differential cryptanalysis and there is only one vectorial function known so far. For both S-boxes the differential and linear probability is 2^{-4}, which is optimal.

Table 1. 6-bit S-box

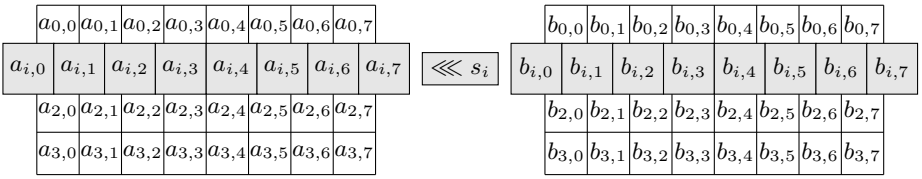
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	54	0	48	13	15	18	35	53	63	25	45	52	3	20	33	41
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S(x)	8	10	57	37	59	36	34	2	26	50	58	24	60	19	14	42
x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
S(x)	46	61	5	49	31	11	28	4	12	30	55	22	9	6	32	23
x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
S(x)	27	39	21	17	16	29	62	1	40	47	51	56	7	43	38	44

Table 2. 5-bit S-box

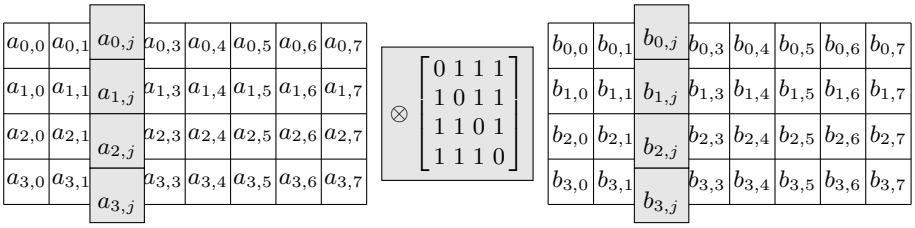
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	1	0	25	26	17	29	21	27	20	5	4	23	14	18	2	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S(x)	15	8	6	3	13	7	24	16	30	9	31	10	22	12	11	19

For this work, we exhaustively searched through the affine equivalent class of quadratic AB and APN permutations paying a special attention to fix points. We synthesized possible candidates with and without threshold implementation to see their area requirements. We chose FIDESS-boxes so that the area of both plain and shared implementation provide a good tradeoff.

ShiftRows (SR). The ShiftRows step is a byte transposition that cyclically shifts the rows of the state over different offsets. Row i is shifted left by $s_i = \{0, 1, 2, 7\}$ positions. Since ShiftRows is only wiring in hardware, its overall cost is negligible.



MixColumns (MC). The MixColumns step is operating on the state column by column. To be more precise, it is a left-multiplication by a 4×4 matrix over $\mathbb{F}_{2^5}$ (resp. $\mathbb{F}_{2^6}$). The main design goal of the MixColumns transformation is to follow the wide trail strategy and that it can be implemented efficiently. On one hand by restricting the coefficients of the matrix to 0 and 1 MixColumns can be implemented with only a few XOR operations, but on the other hand there does not exist a matrix of this form that is also MDS. Therefore, we use in FIDES a matrix that is almost-MDS and has a *branch number* (the smallest nonzero sum of active inputs and outputs of each column) of 4.



ConstantAddition (CA). In this transformation the state is modified by combining it with a predefined constant by a bitwise `xor` operation. The purpose of adding round constants is to make each round different and to break the symmetry of the other transformations. Furthermore, it provides a natural opportunity to make the parts for processing associated data and message different from each other. The hardware implementation of ConstantAddition is in fact very cheap since it consists of wires and invertors only.

2.2 Security Assumptions and Claims

The security analysis of the algorithm starts from the following assumptions.

Assumption 1 (Nonce-respecting adversary). *A nonce value is only used once with the same master key for encryption.*

This assumption is quite common among nonce-based designs. Note that on most platforms, this assumption can be easily satisfied by implementing the nonce as a counter.

Assumption 2 (Abort on verification failure). *If the verification step of the algorithm reveals that the ciphertext has been tampered with, then the algorithm returns no information beyond the verification failure. In particular, no plaintext blocks are returned.*

This assumption significantly reduces the impact of chosen-ciphertext attacks, since the adversary obtains very little information from a chosen-ciphertext query. We feel that this assumption is quite natural for authenticated encryption modes. After all, when the verification fails, we know that the integrity of the plaintext has been jeopardized, and there is no reason to output it.

Under these assumptions, the security claims for the FIDES are as follows.

Claim 1 (Resistance against key recovery). *Any key recovery with complexity equivalent to processing Z data blocks has a success probability at most $Z2^{-k}$, even if the internal state has been recovered.*

Claim 2 (Resistance against state recovery). *Any internal state recovery with complexity equivalent to processing Z data blocks not involving key recovery has a success probability at most $Z2^{-t}$.*

Claim 3 (Resistance against forgery w/o state recovery). *Any forgery attack not involving key recovery/internal state recovery has a success probability at most 2^{-t}.*

3 Security Analysis

3.1 Differential and Linear Cryptanalysis

Bounds for the Initialization and Finalization. The round transformation of FIDES has diffusion properties according to the wide trail design strategy. Since the MixColumns transformation has branch number 4, and ShiftRows is diffusion optimal (moves the elements in each column to four different columns), it is guaranteed that there are at least $4^2 = 16$ active S-boxes in any four-round differential trail (see the left side of Table 3). Note that this bound is tight. To obtain better bounds for FIDES we adopt the mixed-integer linear programming (MILP) technique proposed in [6] and [24] to find the minimum number of differentially and linearly active S-boxes of the target ciphers. Using this technique and the optimizer CPLEX [21], we obtained the differentially and linear bound up to 8 rounds Initialization and Finalization of FIDES. The results are listed in the left part of Table 3.

As shown in the table, there are at least 48 active S-boxes for eight-round differential and linear trail, therefore for sixteen-round of Initialization and Finalization, there are at least $2 \cdot 48 = 96$ active S-boxes. This, combined with the maximum differential and linear probability of the S-box of 2^{-4} for both FIDES-80 and FIDES-96, means that the probabilities of any differential and linear trail (assuming independent rounds) is 2^{-384} for any sixteen-round differential (and linear) trail. Therefore, there is only a very small chance that a standard differential or linear attack would lead to a successful attack on the Initialization or Finalization of FIDES.

Table 3. Bounds for differential and linear trails in FIDES. On the left side the bounds are shown for trails in the Initialization/Finalization and on the right side the bounds are shown for collision producing trails in the message processing part.

Round	Active S-box
1	0
2	4
3	7
4	16
5	22
6	32
7	42
8	48

Round	Active S-box
1	-
2	-
3	-
4	-
5	-
6	52
7	49
8	48

Bounds for Collision Producing Trails. Assume we have a certain difference for the message that may result in a zero difference in the state with a high probability after the difference has been injected. Then this can be used in a forgery attack on FIDES. Note that a linear trail of a similar shape might be used for a distinguish attack on the keystream of FIDES.

However, the simple design of FIDES allows to prove also good bounds against this kind of differential and linear attacks. In more detail, using again the mixed-integer linear programming (MILP) technique and the optimizer CPLEX we could show that any collision producing differential or linear trail for FIDES has at least 48 active S-boxes. In more detail, we found that for 5 and less rounds, there does not exist such trails. For 6, 7 rounds, only trails with at least 52 respectively 49 active S-boxes can result in a collision. For 8 and more rounds, only trails with at least 48 active S-boxes can result into a collision, resulting in an upper bound for the differential probability of 2^{-384}.

Note that these bounds depends on the choice of the injection layer. For the design of FIDES we have tested several different injection layers and choose the one that resulted in the best bound.

3.2 Impossible Differential Cryptanalysis

In this section, we will discuss the application of impossible differential cryptanalysis to FIDES. However, first we will introduce some properties of the matrix M used in MixColumns we need in the analysis. In the following let "$*$" denote the nonzero element (difference) and "0" denote a zero element.

Property 1. If there is only one nonzero element in the input vector X, then after the MixColumns operation there will be three nonzero elements in the output vector $Y = MX$. Additionally, the positions of the nonzero elements are determined by the matrix M.

Assume that the input vector is $X = (*, 0, 0, 0)^T$, the output vector is determined as $Y = (0, *, *, *)^T$. Similarly, we get $M(0, *, 0, 0)^T \rightarrow (*, 0, *, *)^T$, $M(0, 0, *, 0)^T \rightarrow (*, *, 0, *)^T$ and $M(0, 0, 0, *)^T \rightarrow (*, *, *, 0)^T$.

Property 2. If there are two nonzero elements in the input vector X, then the number of the nonzero elements in the output vector Y will be 2 or 4, and the positions of the nonzero elements are again fixed by the matrix M.

Assume the input vector is $X = (*, *, 0, 0)^T$, then the output vector Y can be $(*, *, 0, 0)^T$ or $(*, *, *, *)^T$. The other five patterns can also result in the outputs in a similar way.

Property 3. If there are three nonzero elements in the input vector X, then the number of the nonzero elements in the output vector Y might be 1, 3 or 4, and the positions of the nonzero elements are fixed by the matrix M in some cases.

Assume the input vector is $X = (*, *, *, 0)^T$, and if there is only one nonzero element in the output, from Property 1, we already know the output vector Y is $(0, 0, 0, *)^T$. If the value is 3, then any three of the elements in the output are possible. The case for 4 is obvious.

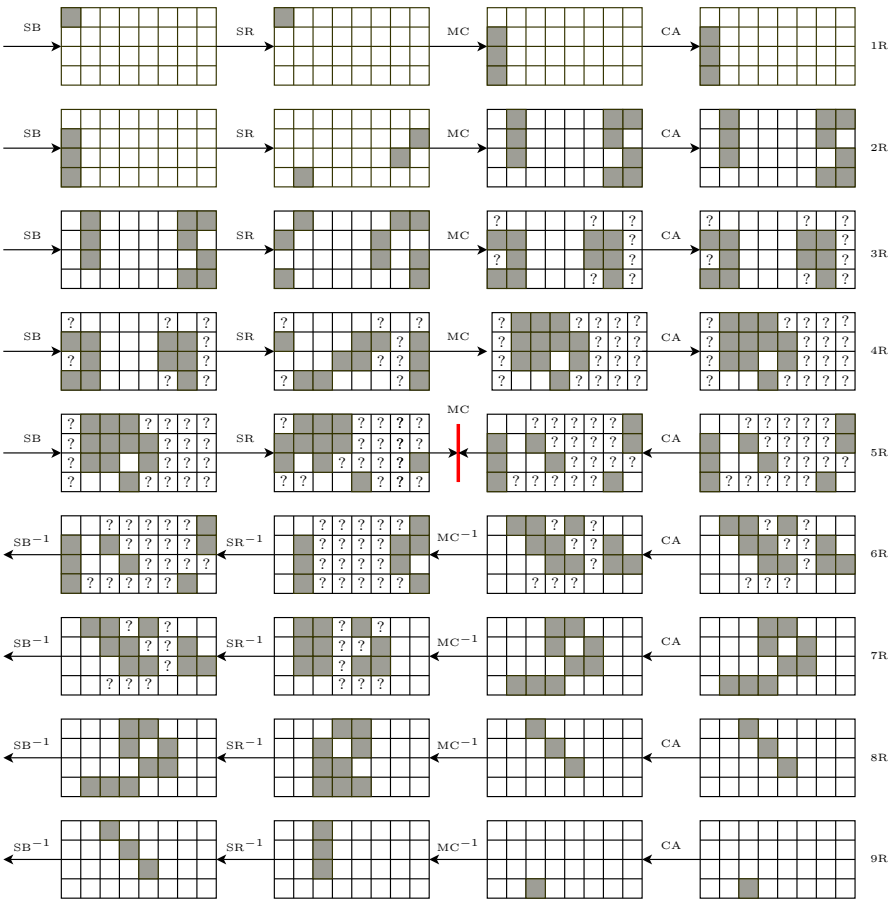


Fig. 3. 9 Rounds Impossible Differential

Property 4. If all the four elements in the input vector X are nonzero, then the number of the nonzero elements in the output vector Y might be 2, 3 or 4 at arbitrary positions.

Note that the number of nonzero elements before and after the MixColumns operation can never be five. Based on these properties, we constructed an impossible differential for 9 rounds of FIDES, which is depicted in Figure 3. Assume we start from the first round, if the difference is at position (0,0) of the state, then after 4.5 rounds transformation of FIDES encryption, the vector in column 1 before the MixColumns operation in the fifth round is $(*, *, 0, ?)^T$, whereas "?" denotes an indeterminate difference. Given the difference at position (3,2) at the bottom of the distinguisher, decrypt 4.5 rounds of the transformation of FIDES the output vector in column 1 after MixColumns in the fifth round is $(0, 0, 0, ?)^T$. This means that $M(*, *, 0, ?)^T \rightarrow (0, 0, 0, ?)^T$, from Property 3 in the above, there is an contradiction before and after the MixColumns operation. Therefore, a 9-round impossible differential has been constructed.

Therefore, for FIDES-80 and FIDES-96, based on Claim 1, it should be difficult to recover the key using these impossible differentials even if the internal state (right after the state initialization) has been recovered.

4 Hardware Implementations and Comparison

In this section, we describe four different architectures of FIDES-80 and FIDES-96. Firstly, we explore a round-based implementation, which completes one round in a single clock cycle. This architecture is straightforward for implementation and its area is mainly occupied by the 32 instances of S-box, 4 instances of MixColumns and the state register file.

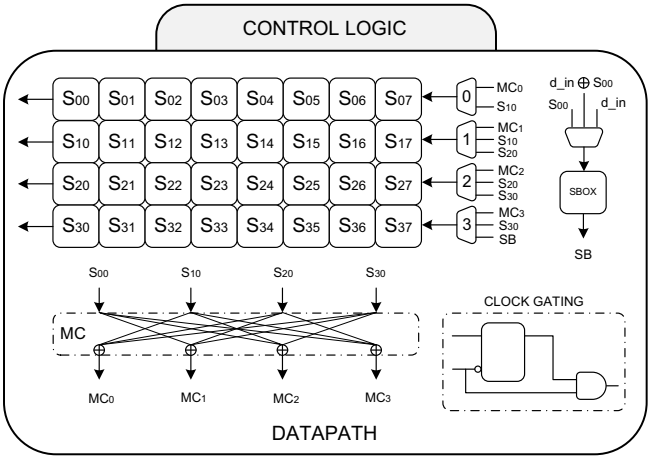


Fig. 4. Serial architecture

The second architecture is implemented in a serial fashion (see Figure 4). Its control logic comprises a simple finite state machine, which orchestrates the data flow within the datapath. The state is stored inside a fully-serial register file and its 32 elements are wired in a single shift register. When SubBytes operation is performed, the output from SBOX is through MUX_3 fed into S_{37} and inputs S_{30}, S_{20}, and S_{10} are active in MUX_2, MUX_1, and MUX_0, respectively. After 32 clock cycles the finite state machine enters the ShiftRows operation, which consumes 7 clock cycles in total. The inputs S_{30}, S_{20}, and S_{10} are active in MUX_3, MUX_2, MUX_1, respectively. Except the first one, other rows of the register file are shifted simultaneously, while the glitch-free clock gating logic ensures the correct schedule. Namely, when the registers in a single row need to keep their value, the clock gating logic disables their clock, which is a considerably cheaper solution than the usage of additional feedback multiplexers or scan registers. Finally, when performing MixColumns, our architecture receives an input column-wise which is, from the MC block, injected into the state through S_{07}, S_{17}, S_{27}, and S_{37}. Including 8 cycles of the MixColumns operation, one round of FIDES consumes 47 clock cycles in total.

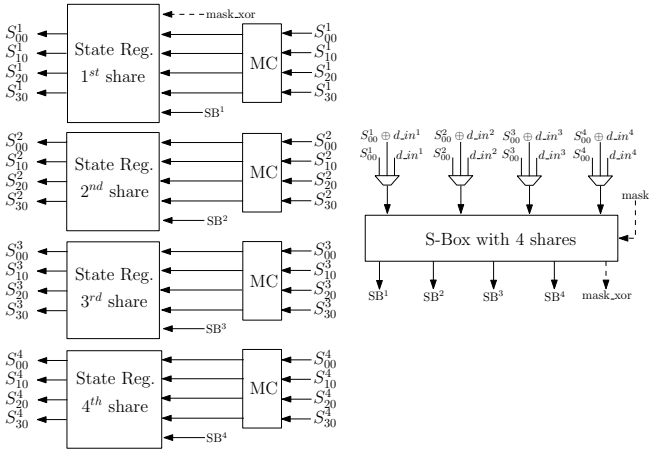


Fig. 5. Threshold implementation

To have a more complete overview of the overall hardware performance, we also implement an architecture with 4 S-boxes. The only difference from the fully serial version is the SubBytes operation which now is performed within 8 clock cycles only. At the expense of some additional hardware, this way we manage to reduce the latency for more than two times.

Finally, the fourth explored architecture is a threshold implementation (TI) [26], which is depicted in Fig. 5. It benefits from a secret-sharing based masking countermeasure against first order side-channel analysis. Being secure even against the leakage caused by the presence of the glitches, TI provides a relatively cheap countermeasure. While protecting linear functions is trivial [27], it becomes

a challenging task to properly address the security of non-linear functions such as S-boxes [5, 23]. As mentioned in Section 2.1, we pay special attention while choosing the S-box such that it can be securely implemented in a single clock cycle, yet having a small area footprint. In order to have a threshold sharing of a 5-bit S-box, fulfilling all the properties, it is sufficient to use 4 shares.

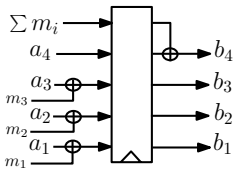


Fig. 6. Re-masking of 4 shares

We further observe that for any 6-bit optimal S-box, which is affine equivalent of our selection, uniformity property is satisfied with more than 5 shares which contradicts with the lightweight philosophy of FIDES. Therefore, we aim to use re-masking over 6-bits as suggested in [23] in order to achieve uniformity at the output of the S-box. Details of the re-masking are shown in Fig. 6 as well as in Fig. 5 (dotted lines). We are given 4 uniform shares where a simple XOR provides the unshared value and we store each share in a different storage element. The threshold implementation of MixColumns or ShiftRows can be simply seen as 4 instances of those functions working in parallel, each using one share only. The S-box absorbs all shares and outputs 4 shares such that the each output share is independent of one input share.

Table 4 gives a complete overview of our results. The smallest amongst all is a serial architecture of FIDES-80 (denoted as FIDES-80-S), which consumes only 793 GE in 90 nm CMOS library. We furthermore implement a round based architecture, which at the cost of 3.5 times larger area achieves 47 times higher throughput. Note here that due to the initialization phase, the additional latency per message is 16 clock cycles for round-based and 752 clock cycles for serial implementation, respectively. TI consumes roughly $3.5 - 4.5$ times more area than the ordinary serial implementation.

The RTL code of our architectures has been written in Verilog and the synthesis carried out in Cadence RTL Compiler version 11.10-p005. For that purpose, we used three different libraries, including an open-cell 45 nm NANGATE [25] library, version PDKv1_3_v2010_12. The power consumption has been measured using a High-Speed UMC 130 nm CMOS generic process provided by Faraday Technology Corporation. Note that the power estimates are obtained after synthesis and as such are not accurate enough to be used for comparison with other designs available in the literature. Their purpose is rather to have a relative comparison of our own implementations. Finally, we provide additional hardware figures using an advanced NXP 90 nm CMOS process, outlining the performance of our design when implemented using an industry compliant technology.

Table 4. Hardware performance of the implemented FIDES architectures (synthesis results). Latency is defined as the number of clock cycles per round while the throughput is observed at 100 kHz assuming very long messages.

Design	Security (bits)	Area (GE)	Frequency (kHz)	Latency	Throughput (kb/s)	Power (μW)
	Advanced NXP 90 nm CMOS process, typical case PVT (25° C, 1.2 V)					
FIDES-80-S	80	793	100	47	10.64	N/A
FIDES-80-4S	80	1178	100	23	21.74	N/A
FIDES-80-R	80	2922	100	1	500	N/A
FIDES-80-T	80	2876	100	47	10.64	N/A
FIDES-96-S	96	1001	100	47	12.77	N/A
FIDES-96-4S	96	1305	100	23	26.09	N/A
FIDES-96-R	96	6673	100	1	600	N/A
FIDES-96-T	96	4792	100	47	12.77	N/A
	NANGATE 45 nm CMOS process, typical case PVT (25° C, 1.1 V)					
FIDES-80-S	80	1244	100	47	10.64	N/A
FIDES-80-4S	80	1819	100	23	21.74	N/A
FIDES-80-R	80	4023	100	1	500	N/A
FIDES-80-T	80	4696	100	47	10.64	N/A
FIDES-96-S	96	1584	100	47	12.77	N/A
FIDES-96-4S	96	2023	100	23	26.09	N/A
FIDES-96-R	96	9180	100	1	600	N/A
FIDES-96-T	96	7541	100	47	12.77	N/A
	UMC 130 nm CMOS process, typical case PVT (25° C, 1.2 V)					
FIDES-80-S	80	1153	100	47	10.64	1.97
FIDES-80-4S	80	1682	100	23	21.74	2.82
FIDES-80-R	80	4175	100	1	500	7.90
FIDES-80-T	80	4267	100	47	10.64	7.47
FIDES-96-S	96	1453	100	47	12.77	2.49
FIDES-96-4S	96	1870	100	23	26.09	3.12
FIDES-96-R	96	8340	100	1	600	14.82
FIDES-96-T	96	6812	100	47	12.77	11.84
	[8] ST 65 nm CMOS LP-HVT process, typical case PVT conditions.					
ALE	128	2579	20×10^3	105	121.9	94.87
ALE e/d	128	2700	20×10^3	105	121.9	102.32
ASC-1 A	128	4793	20×10^3	370	34.59	169.11
ASC-1 A e/d	128	4964	20×10^3	370	34.59	193.71
ASC-1 B	128	5517	20×10^3	235	54.47	199.02
ASC-1 B e/d	128	5632	20×10^3	235	54.47	207.13
AES-CCM	128	3472	20×10^3	452	28.32	128.31
AES-CCM e/d	128	3765	20×10^3	452	28.32	162.15
	[3] TSMC 90 nm CMOS process, typical case PVT conditions.					
C-QUARK	128	3125	100	768	8.33	N/A
C-QUARK	128	7100	100	24	266.67	N/A
	[31] NANGATE 45 nm CMOS process, typical case PVT conditions.					
KECCAK-200-MD	80	7400	50×10^3	18	200	N/A
PHOTON–196-MD	80	11000	50×10^3	N/A	N/A	N/A
QUARK-176-MD	80	5900	50×10^3	N/A	N/A	N/A
SPONGENT-176-MD	80	6500	50×10^3	N/A	N/A	N/A
	[16] TSMC 180 nm CMOS process, unknown PVT conditions.					
HB2-ee4c	128	3220	100	4	400	5.10
HB2-ee16c	128	2332	100	16	100	4.70
HB2-ee20c	128	2159	100	20	80	4.36

FIDES-xy-S – Serial architecture (1 S-box).
FIDES-xy-4S – Architecture with 4 S-boxes.
FIDES-xy-R – Round-based architecture (32 S-boxes).
FIDES-xy-T – Threshold implementation (1 S-box).
ABC-xyz-MD – MONKEYDUPLEX scheme (area is estimated from the graphs reported in [31]).

For the purpose of comparison, at the bottom of Table 4, we add figures of the recent designs of ALE, c-Quark, ASC-1 and Hummingbird-2. Note that the performance of ALE is given for the frequency of 20 MHz using a low-power 65 nm advanced CMOS library. Additionally, although not providing the exact hardware figures, the authors of Grain-128a estimate that the smallest implementation of their design consumes 2770 GE. For the sake of completeness, we also include the figures of the AES-CCM mode. Note that the performance of designs reported in [31] is actually the performance of unrolled architectures and as such is not directly comparable to our implementations. We further note here that the security level of all the designs we compare Fides to is different and needs to be taken into account when considering the possible trade-offs between security, area, and speed.

What can be observed further from Table 4 is a substantial influence of the technology choice on the overall hardware performance. A difference in the relative size of designs synthesized in the advanced NXP 90 nm technology and the open-cell NANGATE library, for instance, spans between 35 % and 65 %. This affirms the difficulty of such one-to-one comparison, which is often seen in the literature. We therefore opt for making future comparisons to our designs easier by including hardware figures obtained using the freely available open-cell technology [25].

5 Conclusion

We have presented Fides, a very lightweight authenticated cipher especially suitable for constrained hardware environments. The results achieved in this work, including amongst others a compact implementation of only 793 GE for 80-bit and 1001 GE for 96-bit security, significantly outperform any previous design known by the authors. Based on the cryptographically optimal 5-bit and 6-bit S-boxes, we have built a very compact threshold implementation whose area requirements are as low as 2876 GE for a design attaining an 80-bit security level.

Acknowledgments. This work has been supported in part by the Austrian Government through the research program COMET, project SeCoS (project number 836628) and by the Austrian Science Fund (FWF), project TRP 251-N23, and is funded by the Major State Basic Research Development Program of China (973 Plan) (No. 2013CB338004), National Natural Science Foundation of China (No. 61073150), and Chinese Major Program of National Cryptography Development Foundation (No. MMJJ20110201).

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. IJWMC 5(1), 48–59 (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. Aumasson, J.P., Knellwolf, S., Meier, W.: Heavy Quark for secure AEAD. In: DIAC - Directions in Authenticated Ciphers (2012)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012)
5. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3×3 and 4×4 s-boxes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 76–91. Springer, Heidelberg (2012)
6. Bogdanov, A.: On unbalanced feistel networks with contracting mds diffusion. Des. Codes Cryptography 59(1-3), 35–58 (2011)
7. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: Spongent: A Lightweight Hash Function. In: Preneel and Takagi [28], pp. 312–325
8. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-Based Lightweight Authenticated Encryption. In: 20th International Workshop on Fast Software Encryption – FSE (2013)
9. Borghoff, J., et al.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
10. CAESAR. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, `http://competitions.cr.yp.to/caesar.html`
11. Carlet, C., Charpin, P., Zinoviev, V.: Codes, bent functions and permutations suitable for des-likecryptosystems. Des. Codes Cryptography 15(2), 125–156 (1998)
12. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 222–238. Springer, Heidelberg (2001)
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
14. 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)
15. Dillon, J.F.: APN polynomials: an update. In: International Conference on Finite Fields and Applications - Fq9 (2009)
16. Engels, D., Saarinen, M.-J.O., Schweitzer, P., Smith, E.M.: The hummingbird-2 lightweight authenticated encryption algorithm. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 19–31. Springer, Heidelberg (2012)
17. Engels, D., Saarinen, M.-J.O., Schweitzer, P., Smith, E.M.: The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 19–31. Springer, Heidelberg (2012)
18. Even, S., Mansour, Y.: A Construction of a Cioher From a Single Pseudorandom Permutation. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 210–224. Springer, Heidelberg (1993)
19. 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 (2011)
20. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel and Takagi [28], pp. 326–341
21. IBM. IBM ILOG CPLEX Optimizer, `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`

22. Knežević, M., Nikov, V., Rombouts, P.: Low-latency encryption – is lightweight = light + wait? In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 426–446. Springer, Heidelberg (2012)
23. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
24. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012)
25. NANGATE. The NanGate 45nm Open Cell Library, `http://www.nangate.com`
26. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
27. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
28. Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011)
29. Saarinen, M.-J.O.: Related-key Attacks Against Full Hummingbird-2. In: Moriai, S. (ed.) Fast Software Encryption. LNCS. Springer (to appear, 2013)
30. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: Preneel and Takagi [28], pp. 342–357
31. Yalçın, T., Kavun, E.B.: On the Implementation Aspects of Sponge-based Authenticated Encryption for Pervasive Devices. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 141–157. Springer, Heidelberg (2013)