End-to-end data security for IoT: from a cloud of encryptions to encryption in the cloud

Anne Canteaut[1], Sergiu Carpov[2], Caroline Fontaine[3], Jacques Fournier[4], Benjamin Lac[5,6], María Naya-Plasencia[1], Renaud Sirdey[2], and Assia Tria[5]

[1] Inria, France, {anne.canteaut,maria.naya_plasencia}@inria.fr
[2] CEA LIST, France, {sergiu.carpov,renaud.sirdey}@cea.fr
[3] CNRS/Lab-STICC and IMT-Atlantique and UBL, France, caroline.fontaine@imt-atlantique.fr
[4] Univ. Grenoble Alpes, CEA LETI, DSYS/LSOSP, F-38000 Grenoble, jacques.fournier@cea.fr
[5] CEA TECH, France, {benjamin.lac,assia.tria}@cea.fr
[6] ENSM-SE, France

Abstract. This paper surveys the symmetric primitives suitable for the unusually strong versatility requirements of end-to-end encryption in the context of IoT. For example, on the objects side, criteria such as lightweightness as well as efficient software-based amenability to resist side channels or fault attacks must be considered. At the other end of the spectrum, on the cloud side, the same crypto primitives must be able to efficiently reach the very high performances required to handle large numbers of objects on higher-end processors. Cherry on the cake, it would also be desirable for the primitives that are deployed in today's IoT infrastructures to interface as seamlessly and as efficiently as possible with the future encrypted-domain services which will be built on top of emerging crypto primitives such as homomorphic encryption or multi-party computation. Based on recent works on both the implementation security as well as the homomorphic execution of some lightweight IV-based stream ciphers, this paper argues that it may be possible to have the cake, the cherry and the cream on top of it.

Keywords: lightweight cryptography, end-to-end encryption, IoT.

Category: specialized.

1 Introduction

Today's information systems are prone to an avalanche of cyber attacks threatening our strategic infrastructures, industries and even public safety, all this often by targeting the data churned by those systems. The origin of what seems to become an uncontrollable situation comes from the fact that such systems are getting more and more complex with the massive deployment of connected devices (through the Internet of Things —

IoT), with more and more data being collected, exchanged, stored and analysed and with the growing bargaining value of those data which relate to our personal lives. To mitigate the attacks against our systems, and hence the data within those systems, several critical aspects have to be further strengthened: the access to those data must be more securely managed, their availability must be guaranteed along with their authenticity, integrity and confidentiality. Each of the latter objectives is in itself a daunting task when considering the complexity and heterogeneity of IoT infrastructures being currently deployed.

In this paper we focus on one of the aforementioned aspects, that is how to ensure end-to-end data confidentiality in a complex IoT system, from the moment the data is collected by a low-end sensor node to the moment it is used on a cloud server for, say, data analysis or some artificial intelligence algorithm. Our paradigm here is to address how we move *from a cloud of encryptions to encryptions in the cloud*. In other words, fast, low power and securely implemented (against physical attacks) ciphers deployed in the resource-limited IoT nodes have to be compatible with (even) the more complex homomorphic schemes that can be deployed on the server side. In this paper, we propose a way of achieving such a goal based on recent works on both the implementation security as well as the homomorphic execution of some lightweight IV-based stream ciphers.

This paper surveys the symmetric primitives suitable for the unusually strong versatility requirements of end-to-end encryption in the context of IoT. For example, on the objects side, criteria such as lightweightness as well as efficient software-based amenability to resist side channels or fault attacks must be considered. At the other end of the spectrum, on the cloud side, the same cryptographic primitives must be able to efficiently reach the very high performances required to handle large numbers of objects on higher-end processors. Furthermore, the primitives deployed in today's IoT infrastructures have to interface as seamlessly and as efficiently as possible with the future encrypted-domain services which will be built on top of emerging crypto primitives such as homomorphic encryption or multiparty computation.

2 Lightweight stream-ciphers

2.1 Stream Cipher Basics

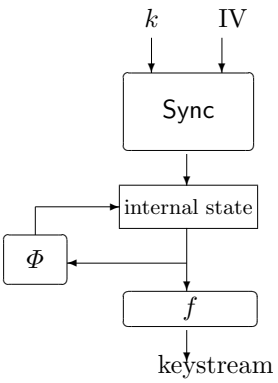
Stream ciphers are encryption schemes: they include an encryption function and a decryption function which can handle messages of an arbitrary length. In this sense, they cannot be compared with block ciphers, which

handle fixed-length inputs only. For instance, block ciphers with some particular modes of operation (like the CTR mode) are stream ciphers.

We here focus on the so-called *synchronous additive* stream ciphers, which capture the vast majority of stream ciphers used today. The encryption function in a synchronous additive stream cipher consists in adding bitwise to the plaintext a binary sequence of the same length, named the *keystream*. The keystream is generated independently from the plaintext and from the ciphertext. In most applications, the ciphertext cannot be sent as a very long stream since it is impossible to guarantee the synchronization of the decryption process. Therefore a resynchronization mechanism is included based on an initial value (IV), typically a frame number, which is used for initializing the keystream. Such stream ciphers are then known as IV-based stream ciphers [8].

An IV-based keystream generator is decomposed into:

- a resynchronization function, Sync, which takes as input the IV and the key (possibly expanded by some precomputation phase), and outputs some n-bit initial state;
- a transition function Φ which computes the next state of the generator;
- a filtering function f which computes a keystream segment from the internal state.



Several parameters affect the security of a synchronous additive stream ciphers. Most importantly, there exist some generic attacks, like time-memory-data-tradeoff attacks in [7, 40, 11], recovering the n-bit internal state of the generator with complexity $\mathcal{O}(2^{n/2})$, implying that the internal state must be at least twice larger than the secret key.

2.2 Lightweight Stream Ciphers

Due to their functionalities, stream ciphers are well-adapted to noisy or low-bandwidth communications. Indeed, they do not require any padding and decryption does not propagate transmission errors. Also, they have a low latency since encryption or decryption can start as soon as a single bit of the message has been received. Block-cipher-based stream ciphers, such as the AES with CTR mode, can be used as a first choice in many applications. However, in some particular applications, dedicated stream

ciphers must be privileged, for instance if a high encryption/decryption throughput or a very low latency is needed, or in resource-constrained environments. For these reasons, the most recent international effort for recommending stream ciphers, namely the eSTREAM project [29] initiated by the European Network of Excellence ECRYPT, focused on two profiles: stream ciphers for software applications with high throughput requirements, and stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption. This international competition received more than 30 new stream ciphers in 2005. Among them, 7 ciphers have been included in a portfolio of recommended stream ciphers: HC-128, Rabbit, Salsa20/12 and Sosemanuk for software profile, and Grain v1, MICKEY 2.0 and Trivium for hardware profile[1]. Most notably, Trivium has been specified as an International Standard under ISO/IEC 29192-3.

2.3 Trivium

Trivium is a synchronous stream cipher with a key and an IV of 80 bits each. Its internal state is composed of 3 registers of sizes 93, 84 and 111 bits, corresponding to a size of 288 bits in total. We use the notation introduced by the designers: the leftmost bit of the 93-bit register is s_1, and its rightmost one is s_{93}; the leftmost bit of the register of size 84 is s_{94} and the rightmost s_{177}; the leftmost bit of register of size 111 is s_{178} and the rightmost s_{288}. The initialization and the generation of an N-bit keystream are described below, and depicted on Fig. 1.

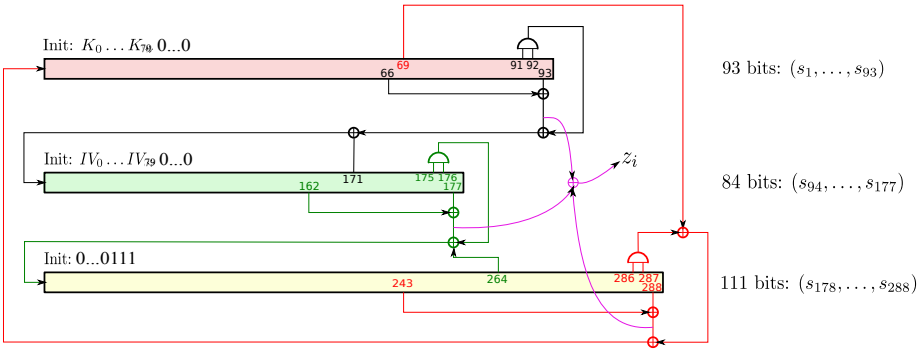


Fig. 1. Trivium.

[1] An 8th cipher, named F-FCSR, was included in the original eSTREAM portfolio but has been removed from the list after some cryptanalytic work.

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_0, \ldots, K_{79}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_0, \ldots, IV_{79}, 0, \ldots, 0)$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
for $i = 1$ to $1152 + N$ **do**
  $t_1 \leftarrow s_{66} + s_{93}$
  $t_2 \leftarrow s_{162} + s_{177}$
  $t_3 \leftarrow s_{243} + s_{288}$
    if $i > 1152$ **do**
      output $z_{i-1152} \leftarrow t_1 + t_2 + t_3$
    end if
  $t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
  $t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
  $t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

No attack better than an exhaustive key search is known so far on full Trivium. It can then be considered as secure. The family of attacks that seems to provide the best result on round-reduced versions is the cube attack and its variants [27, 6, 34, 73, 56]. They recover some key bits (resp. provide a distinguisher on the keystream) if the number of initialization rounds is reduced to 799 (resp. 885) rounds out of 1152. The highest number of initialization rounds that can be attacked is 961: in this case, a distinguisher exists for a class of weak keys [48].

2.4 Hardware vs software profile

Although initially included as part of the hardware profile of the eSTREAM portfolio, Trivium also allows efficient sofware implementations.

On high-end platforms, the so-called bitslicing parallelization technique can be used to boost the algorithm's performances by multiplexing either as many IV or as many keys (depending on the use-case) as there are bits in the processor registers. Table 1 provides typical performances of bitsliced implementation of Trivium on an Intel processor. Most notably, using AVX registers and instructions, the algorithm can reach 22.4 Gbits/s (when AES using AES-NI ISA extensions runs at around 5.6 Gbits/s on these platforms) translating in 0.79 cycle per keystream byte (when e.g. SOSEMANUK requires 4 to 9 cycles per keystream byte). On

these kinds of platforms, Trivium is therefore a solution of choice to either achieve very high throughput (through IV-multiplexing) or serve many users or devices (through key-multiplexing).

# slices	state size	throughput
8	288 bytes	1 Gbit/s
64	2304 bytes	7.5 Gbit/s
256	≈1 Mo	22.4 Gbit/s

Table 1. Software performances of bitsliced implementations of Trivium on *one* Intel i5-5200U core 2.2 Ghz.

Trivium also admits fairly efficient compact 8-bits and 32-bits software implementations suitable for more constrained platforms. Indeed, since it takes 64 (bit-level) cycles for the reinjection of t_3 to have an effect on t_1, up to 64 cycles can be performed in parallel leaving the possibility for byte-oriented, 32-bits-word-oriented and 64-bits-word-oriented implementations requiring only 36 bytes of memory for the internal state (slightly more for 64-bits implementations as 288 is not a mutliple of 64) when AES-128 (say in CTR mode) would require 192 bytes (counter plus key schedule). Note that 36 bytes is quite small but above the 160 bits minimum for 80-bits security, as discussed in Sect. 2.1. For example, an 8-bits implementation of Trivium can reach a throughput of around 150 kbit/s on an Arduino Nano running at 16 MHz and is also cost-effectively amenable to hardening against physical attacks as discussed in the next section.

3 Protecting stream ciphers against physical attacks

Usually, stream-ciphers are constructed to resist black box mathematical cryptanalysis. However, most of them do not take into account implementation related issues such as the vulnerability to physical attacks which is of the utmost importance in the context of IoT. Then, it is common to use one or more countermeasures in order to thwart such attacks but it is not straightforward to devise such countermeasures because of the diversity of the possible attack paths and because the usually deployed countermeasures have a serious impact on performances.

3.1 Physical attacks

Physical attacks can be divided into three categories: invasive like reverse engineering, non-invasive like side-channel analysis and semi-invasive like fault attacks. The first one involves specific hardware-related phenomena,

very much related to the way the integrated circuits running the ciphers are implemented. The second one is a particularly high threat to the way cryptographic algorithms are implemented. The third one has been the trickiest so far, as detailed below, due to the complexity of the different fault attack routes and the expensive countermeasures.

Side-channel attacks. Side-channel attacks [51, 61] exploit the dependencies that exist between some physical values or "side channels" of an integrated circuit like power consumption [50], electromagnetic radiation [35, 68] or calculation time [49] and operations and data manipulated during a given computation. Information about the internal processes of the chip and the data it is manipulating can then be derived by analyzing such dependencies using statistical tools like correlation [18], mutual information [39], variance [59] or entropy [60] or using architecture-dependent behaviours like cache accesses [9, 66, 67] or branch predictions [1, 2]. Such analyses can be quickly mounted with relatively cheap equipment, without altering the physical integrity of the circuit.

Many side-channel attacks target the S-boxes used by some ciphers: it is the case for most of block ciphers [3, 57]. Since stream ciphers generally do not use S-boxes, it is often more complex for an attacker to obtain an attack path using side-channels, although such attacks do exist [33].

Fault attacks. Fault attacks consist in disturbing the behaviour of the circuit in order to alter the correct execution of the cipher. The faults are injected into the device by various means such as light pulses [72], laser [71], clock glitches [4], spikes on the voltage supply [12] or electromagnetic (EM) perturbations [26]. They can have different effects on the encryption (or decryption) process like an instruction-skip or an n-bit set, reset or flip — commonly $n = 1$, 8 or 32 according to the chosen injection means and to the targeted implementation. An instruction-skip can for example allow to bypass the last key addition layer, common to many ciphers, and thus to retrieve the key which is equal to the difference between the correct and the faulty ciphertexts. A bit set, reset or flip can allow to make a safe-error analysis [47]. It consists in disturbing the content of a conditional loop dependent on a key bit in order to retrieve its value by comparing the obtained ciphertext with the correct one: in case of equality, the condition is false, otherwise it is true. A bit set, reset or flip can also allow to perform Differential Fault Analysis (DFA). DFA, originally described in [10, 13], consists in retrieving a secret key by comparing correct ciphertexts with faulty ones. DFA techniques have

been described and applied to most publicly known ciphers going from symmetric-key ciphers like DES [10] or AES [69] to asymmetric ones like RSA [13] or even more complex schemes like Pairings [55]. In the particular field of lightweight cryptography, DFA have been proposed against ciphers like PRESENT [76], SPECK [74], PRIDE [52] or Trivium [65].

Yet, these references are not exhaustive in terms of fault effects and in terms of their exploitation in real-life attacks.

3.2 Attacking Trivium

Electromagnetic radiations analysis. In order to propose a practical example of what can be achieved by observing the side channels of an integrated circuit, we implemented and executed the 8-bit implementation of Trivium given in [54] on an chip embedding an ARM Cortex-M3 micro-controller. We used this micro-controller since it is quite representative of the off-the-shelf devices used for IoT. We performed an electromagnetic (EM) radiation analysis which consists in identifying the operations made by the cipher. Such an analysis can allow for example to precisely target a chosen operation of the implementation in order to more easily perform a fault attack [53]. Figure 2 shows the obtained curves which allow us to identify all the steps of the Trivium execution.

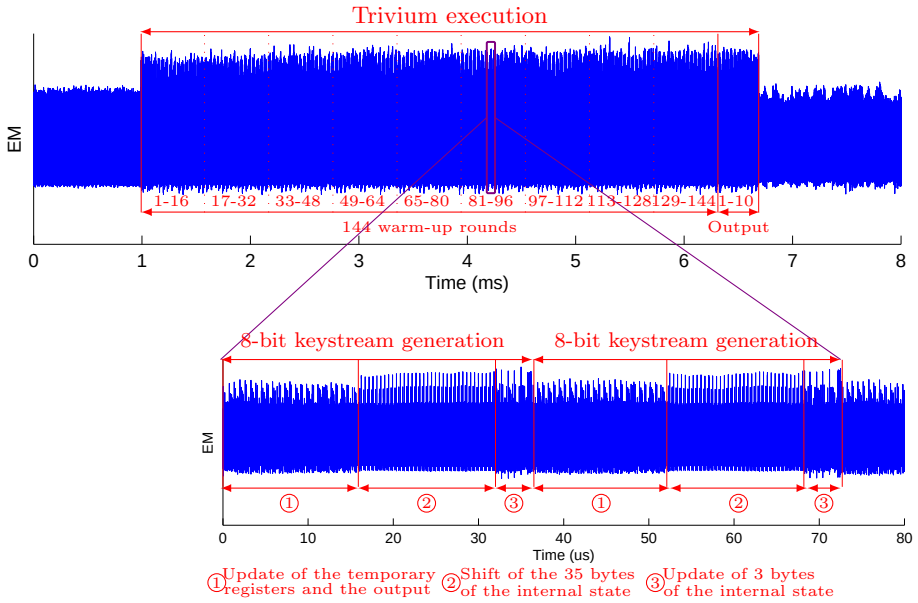


Fig. 2. Electromagnetic radiations analysis of Trivium

Differential fault analysis. The first DFA on Trivium was proposed by Hojsk and Rudolf in 2008 [43]. The aim is to retrieve a chosen inner state IS_{t_0} after the warm-up rounds (on which the attacker is able to inject faults) from the keystream it generates. Once the attacker obtains an inner state IS_{t_0}, she can run Trivium backwards until she obtains the initial inner state and so the secret key. The DFA consists in executing a first time the cipher without any disturbance. Then, each bit of the keystream generated from IS_{t_0} allows the attacker to obtain an equation with 288 variables (the bits of IS_{t_0}). However, the obtained systems of equations more variables than equations. It is therefore impossible for the attacker to retrieve the inner state. The idea is then to execute several more times the cipher with the same input parameters, i.e. the same secret key and nonce, by corrupting a single bit of IS_{t_0}. Such a fault model can be achieved by using a laser if well manipulated. The obtained faulty keystreams allow the attacker to obtain several other equations until having enough equations to retrieve the 288 variables. Another paper proposed by Mohamed & al. in 2011 [65] improved this attack using SAT solving with the same prerequisites and fault model. The attacker must be able to control the nonce to proceed with the DFA, which is rarely possible in the context of IoT.

Safe-error analysis. It is possible to perform a safe-error analysis on Trivium by targeting each secret key bit during the first initialization rounds. For each, the attacker must be able to perform a set (resp. a reset) of the targeted bit and to precisely know its position. She also must know the value of the correct execution of the cipher. Then, if the obtained faulty ciphertext is equal to the correct one, the value of the bit is 1 (resp. 0) and vice versa. Once again, the attacker must be able to control the nonce, which is rarely possible in the context of IoT.

3.3 Countermeasures

Thwarting side-channel attacks. Generally, to prevent side-channel attacks, the aim is to decrease leakage and/or increase noise. On the one hand, in order to decrease leakage, it is possible to balance the power of differential signals [32], to balance the processing of cryptographic computations [44] or to limit the number of operations per key. On the second hand, in order to increase noise, it is possible to use operation shuffling [58] which consists in randomizing the order of the operations performed by the cipher. Another possibility is to use masking [3, 45] which consists

in applying logic or arithmetic masks to the data. It is generally not expensive to generate and apply a mask. It is however more complex to remove it after the area it protects: it highly depends on the operations performed by the targeted cipher.

Thwarting fault attacks. Fault attacks can be circumvented using hardware or software countermeasures [46]. Regarding hardware countermeasures, passive shields, which are metal layers over the chip, allows to prevent optical fault injections [75]. However, it is possible to remove passive shields using chemical means and fault injections using EM pulses cannot be blocked by such shields neither. Active shields, which consist of wire meshes that run signals over the chip surface and detect any interruption on a wire, are thus a very effective mean to thwart many fault attacks. Furthermore, the use of light sensors [30] allows to detect anomalies in the circuit behavior. The main drawback of such hardware countermeasures is their cost and that even though simulations may show that they are efficient, there is no absolute guarantee that this will be the case on the final chip where other considerations like final place and route, manufacturing processes etc. may have a huge impact on the countermeasures efficiency. And, by the time the final chips are obtained, if a security flaw is identified, it is expensive to have another iteration of the design cycle to "patch" the design in hardware (in some cases metal fixes may be used but this would not apply to any part of the circuit). It is hence highly recommended to use such hardware countermeasures along with software ones. One of the basic principle behind most software countermeasures is to make sure that all the calculations timings are independent from the data or key being manipulated, and to "hide" the internal calculations of the cipher so that the attacker has no control over the data being manipulated and no means to understand what is happening. Masking has been shown to offer protection against some fault attacks like DFA [53]. However, such an approach cannot thwart fault attacks like statistical fault attacks [28] or safe-error analyses. Redundancy is thus a very efficient means to thwart this latter kind of attacks. It consists in computing the same operations on one or several copies of the data, providing either a spatial or a temporal redundancy, and then in comparing the obtained results. Hence, to perform a fault attack, an attacker must obtain the same fault on both computations. However, each spatial (resp. temporal) copy costs a memory (resp. time) overhead equal to that of an additional operations [64]. Therefore, more and more research focus on trying to perform such redundancies at lower costs.

3.4 Internal Redundancy Countermeasure

Recently, a countermeasure based on redundancy called the Internal Redundancy Countermeasure (IRC) [54] was proposed to thwart fault attacks. It consists in using an efficient 8-bit implementation of a given cipher (which is usually the preferred option for lightweight ciphers) simultaneously applied on 4 blocks on a 32-bit architecture (which is the most widely used architecture in IoT devices). It is possible by replacing each 8-bit operator by means of a single stream of 32-bit instructions corresponding to the same operation performed independently on each byte in a SIMD [2] fashion. IRC uses reference blocks, which are constant inputs (plaintexts and keys) for which the corresponding ciphertexts are known, to increase the countermeasure's efficiency (mainly to thwart instruction skips). The manipulated words are thus composed of one data byte interleaved with the corresponding byte of the reference block and two copies depending on the used cipher. Figure 3 shows a typical example of a 32-bit word as used in IRC.

Fig. 3. IRC's 32-bit word structure

More details are provided in [54] on how to efficiently protect stream-ciphers with IRC including a practical implementation on Trivium.

4 FHE transciphering

Fully Homomorphic Encryption (FHE) is a relatively recent [36, 63] kind of cryptographic techniques, which on top of allowing the scrambling of data in order to protect their confidentiality, also provides the necessary mathematical building blocks for the execution of general algorithms directly over encrypted data. As such, FHE is an emerging software-only technology allowing to enforce the confidentiality of data when they are manipulated by untrusted servers which avoids disclosing any secret to those servers.

4.1 FHE overview

The ability to compute directly over encrypted data results in the ability for a cloud computer (say Charlie) to do "something useful" with the data

[2] SIMD: Single Instruction Multiple Data

of an end user (say Alice), eventually using additional data from one or more providers (say, possibly many Bobs). In doing so, both Alice's and Bob's data remain confidential with respect to Charlie who manipulates them only in encrypted form and, thus, has neither access to these data in clear form nor is provided with any decryption capability. Indeed, in this setting, any by-product of Charlie's calculations (be it intermediate or final calculation results) remains sealed under Alice's FHE scheme who, as the owner of that cryptosystem secret key, is the only party able to retrieve the intelligibility of Charlie's outputs.

This capability allows to imagine a number of settings where users can benefit from cloud services taking into account their privacy-critical data, still without effectively giving them away. Among these are:

1. Undisclosed cross-valorization of data (and algorithms): where it becomes possible for an algorithm to interact with some data with this interaction implying neither the disclosure of the algorithm to the data owner, nor the disclosure of the data to the algorithm owner.
2. Intrinsic data protection on vulnerable platforms: where it becomes possible to store sensitive data (e.g. medical [23, 70] or biometric data [15]) on e.g. connected (hence intrinsically vulnerable) computing platforms while keeping a permanent protection layer on their confidentiality.
3. Privacy-preserving outsourcing: where it becomes possible to store data on an untrusted server (with respect to confidentiality i.e. in the honest-but-curious threat model) while still preserving an ability to do more than just retrieving them.

At this stage, it is important to recall the security model underlying the use of FHE. Indeed, in the most basic settings, two parties are involved: the user (owner of some private data) and the server (owner of an algorithm and possibly some data which it is willing to inject in the calculation). The issue that is addressed by FHE is that of protecting the confidentiality of user data with respect to threats coming from the server. As such, in proper cryptographic terminology, we are in the so-called honest-but-curious server setting.

It should also be emphasized that FHE schemes are necessarily probabilistic, and this means that some noise components is added in the encryption process to make sure that each possible cleartext message has a large number of possible ciphertexts. This is a fundamental property that is necessary for provable semantic security. Still, one of the issues is that with FHE, ciphertexts are intrinsically significantly larger than

plaintexts and practically coping with this issue is one of the key issues that need to be dealt with if FHE is ever to be practical (this is precisely why the present paper discusses FHE aspects, wait until Sect. 4.2 below). Primarily for this reason, all known FHE are intrinsically unstable: the noise amplitude grows with the homomorphic calculations until decryption is no more possible. Usually, noise growth is faster with multiplications than with additions. Part of the intrinsic complexity of FHE schemes is due to the necessity of noise amplitude management, in order to ensure that the homomorphic calculation results can be decrypted.

There exists two blueprints for building homomorphic encryption systems: one is based on the self-reference trick of bootstrapping [36], the other is based on somewhat fully homomorphic encryption, which is (until a recent new hope [25]) the only practical option. So the approach is to use cryptosystems that can be rendered homomorphic-enough to execute an a priori given (class of) algorithms, and this dimensioning can automatically be done "at compile time" [22]. Furthermore, we now have several reasonably efficient such cryptosystems: BGV [17] (implemented in the well-known open source library HELIB [41, 42]), Fan-Vercauteren [31], GSW [38], and a few others. Some of them also have the nice property of allowing some kind of bitslicing-type parallelism known as batching [16] which, when applicable, allows to better amortize the cost of FHE calculations.

Relatively to encrypted-domain data, the "FHE machine" can only execute so-called static control structure programs (i.e., programs which can always be turned into a linear sequence of instructions). Static control structure programs are formally equivalent to boolean circuits (i.e. networks of logical gates modelled by acyclic directed graphs) and, when it comes to efficient encrypted domain execution, *the* key parameter that needs to be optimized is the multiplicative depth of the circuit which is the largest number of AND gates on any path from an input to an ouput of the circuit. Indeed, with FHE schemes such as BGV [17] and FV [31], increasing the multiplicative depth results in a nonlinear increase in both ciphertext size and mul operator cost (it is even sometimes advantageous to perform much more AND gates albeit at lower depth). This is illustrated in Table 2.

4.2 Efficient communication towards FHE-enabled servers

As just emphasized, there are a number of issues with respect to transmitting FHE-encrypted data mainly: FHE-encryption is a computationally

depth	kb/bits	ms/AND
0	3	-
1	10	4
2	21	8
5	83	28
7	148	66
10	282	150
15	601	376
20	1039	680

Table 2. Illustration of the ciphertext size and AND gate computation time in terms of the multiplicative depth (FV scheme with $t = 2$ and $\lambda \approx 128$.)

heavy operation and FHE-encrypted data are much larger than their associated plaintexts.

However, FHE is powerful enough to allow a trick known as transciphering, by means of which it is possible to switch from some data encrypted with some (possibly non homomorphic) cryptosystem (e.g. a "classic" overhead-free symmetric cryptosystem) to the same data encrypted under FHE (hence on which we can now compute by means of homomorphic operations), without this data to ever be in clear form. Stated informally, using the AES as the underlying symmetric cipher, if

$$\mathsf{AES}^{-1}([x]_{\mathsf{AES}}, k) = x$$

then, by homomorphically executing AES^{-1} on a FHE-superencryption of $[x]_{\mathsf{AES}}$, we get,

$$\mathsf{AES}^{-1}([[x]_{\mathsf{AES}}]_{\mathsf{FHE}}, [k]_{\mathsf{FHE}}) = [x]_{\mathsf{FHE}}$$

where the encryption of the secret symmetric key under the FHE, $[k]_{\mathsf{FHE}}$ can be *known to the FHE computer* and is referred to as a *transciphering token*.

Still, homomorphically executing an AES decryption remains prohibitively heavy and this is intrinsic as the algorithm as a multiplicative depth of 40 [37, 22]. Hence, more "homomorphically-friendly" symmetric systems are required.

In this context, stream ciphers have emerged as better suited to FHE execution [20, 62]. Indeed, when a block-cipher usually is a relatively low degree function iterated a significant number of times (e.g. 10 times for AES-128 or even more, to the notable exception of PRINCE [14] and the more recent LOWMC [5]), a design which is intrinsically not FHE-friendly, stream ciphers (when not based on block-ciphers) follow different design patterns, some of them "friendlier" for efficient FHE execution.

So what is needed is a stream cipher where keystream bits are multiplicatively bounded. This is the case if keystream bits are independent by chuncks (which is good for parallelism and batching). Also, when using a stream cipher, keystream bits can be homomorphically computed independently of the data. Hence, transciphering induces almost no latency (it gets down to just an homomorphic XOR) as long as keystream calculations have been done in advance. It turns out that the basic pattern of using an IV-based (FHE-friendly) stream cipher in counter mode fullfills these requirements.

4.3 Homomorphic execution of Trivium

It turns out that, among the finalists of the recent eSTREAM stream cipher design competition, Trivium (our running example in this paper) was a good candidate as a respected 80-bits key lightweight stream cipher. Still, in order to increase the overall keylength to a larger 128-bits while retaining the FHE-friendlyness of Trivium, the Kreyvium variant has also been proposed [20] (further note that another 128-bits variant has also been proposed as part of the CESAR competition [24]). Table 3 provides the characteristics of homomorphically running Trivium at depth 12 (57 bits of keystream per IV) and 13 (136 bits of keystream per IV). Also, Table 4 provides some performance results when running Trivium over the FV FHE scheme on a mid-end 48-core server (4 x AMD Opteron 6172 processors with 64 GB of RAM) and for various multiplicative depths e.g. evaluating Trivium-12 at depth 12 leaves no "depth budget" for further FHE calculations whereas evaluating it at depth 19 leaves a multiplicative depth budget of 7 for further FHE operations.

Algorithm	λ	# ANDs	# XORs	$\times$ depth	# key stream
Trivium-12	80	3237	15019	12	57 bits
Trivium-13	80	3474	16537	13	136 bits

Table 3. Number of AND and XOR gates to homomorphically evaluate Trivium.

Algo	λ	N	$\times$-depth	1 core	48 cores	speedup
Trivium-12	80	57	12	681.5	26.8	$\times$ 25.4
			19	2097.1	67.6	$\times$ 31.0
Trivium-13	80	136	13	888.2	33.9	$\times$ 26.2
			20	2395.0	77.2	$\times$ 31.0

Table 4. Example computation timings for the homomorphic execution of TRIVIUM.

As such, the algorithm has been implemented in standard C++ and fed into an FHE compiler toolchain [22, 21, 19], which automatically turned it into an optimized boolean circuit, extracted parallelism and generated parallel code able to transparently obtain the speedups shown in Table 4.

Thus, if FHE is ever to be practical, dealing with the transmission overhead was critical for the following reasons: to avoid the computational burden of FHE-encryption on client devices; to avoid the intrinsic bandwidth inflation of transmitting FHE-encrypted data from devices; to (almost) transparently interface client devices with a remote "crypto-computer" as well as to use (almost) standard crypto on client devices.

Tranciphering also allows to perform FHE calculations (by a cloud platform towards an end user, owner of the FHE secret key) involving multiple data providers, each using their own symmetric secret keys (and providing the platform with the appropriate transciphering tokens). If encrypted with the kind of FHE-friendly symmetric cryptography we have just discussed, data encrypted today may also be later on used as part of future FHE-based services without any need for reencryption or reformatting.

5 Conclusion

Using the Trivium stream cipher as a running example, it is this paper's intent to argue that IV-based lightweight stream ciphers initially designed for hardware are versatile enough primitives to address the many needs and constraints of end-to-end encryption in both existing and future IoT system architectures. This is perhaps in slight opposition to conventional wisdom which tends to favor (lightweight) block ciphers in such applications. Of course, using 80-bits keys may seem insufficient in an era when the bitcoin miners pool takes 2.7 days to evaluate 2^{80} (partial) hash functions (roughly spending 13.5 M$ in electricity to do so) and attackers becomes (theoretically) empowered with Grover's algorithm. Still, although 80-bits keys can still be argued to provide acceptable security for IoT applications in the foreseable future, most of the arguments developped in this paper apply to other algorithms having larger key sizes and Trivium itself now has larger key size siblings.

References

1. O. Aciiçmez, Ç. K. Koç, and J.-P. Seifert. On the power of simple branch prediction analysis. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, ASIACCS '07, pages 312–320, New York, NY, USA, 2007. ACM.
2. O. Aciiçmez, Ç. K. Koç, and J.-P. Seifert. Predicting secret keys via branch prediction. pages 225–242, 2007.

3. A. Adomnicai, B. Lac, A. Canteaut, J. J. A. Fournier, L. Masson, R. Sirdey, and A. Tria. On the importance of considering physical attacks when implementing lightweight cryptography. In *NIST Lightweight Cryptography Workshop 2016*, Gaithersburg, Maryland, October 2016.
4. M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria. When clocks fail: On critical paths and clock faults. In D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, editors, *CARDIS 2010*, volume 6035, pages 182–193, Passau, Germany, April 14-16, 2010.
5. M. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In *EUROCRYPT, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, 2015.
6. J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir. Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In *FSE*, volume 5665 of *LNCS*, pages 1–22. Springer, 2009.
7. S. Babbage. A space/time trade-off in exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection*, number 408. IEEE, 1995.
8. C. Berbain and H. Gilbert. On the Security of IV Dependent Stream Ciphers. In *FSE*, volume 4593 of *LNCS*, pages 254–273. Springer, 2007.
9. G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero, and G. Palermo. AES power attack based on induced cache miss and countermeasure. In *ITCC 2005, Volume 1*, pages 586–591, Las Vegas, Nevada, USA, April 4-5, 2005. IEEE Computer Society.
10. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. pages 513–525, 1997.
11. A. Biryukov and A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In *ASIACRYPT*, volume 1976 of *LNCS*, pages 1–13. Springer, 2000.
12. J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the advanced encryption standard (AES). pages 162–181, 2003.
13. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). pages 37–51, 1997.
14. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications. In *ASIACRYPT*, volume 7658 of *LNCS*, pages 208–225. Springer, 2012.
15. N. Bouzerna, R. Sirdey, O. Stan, T.-H. Nguyen, and P. Wolf. An architecture for practical confidentiality-strengthened face authentication embedding homomorphic cryptography. In *Proceedings of the 8th IEEE International Conference on Cloud Computing Technology and Science*, pages 399–406, 2016.
16. Z. Brakerski, C. Gentry, and S. Halevi. Packed Ciphertexts in LWE-Based Homomorphic Encryption. In *Public Key Cryptography*, pages 1–13, 2013.
17. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325, 2012.
18. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. pages 16–29, 2004.
19. S. Canard, S. Carpov, D. Nokam Kuate, and R. Sirdey. Running compression algorithms in the encrypted domain: a case-study on the homomorphic execution of rle. In *Proceedings of Privacy, Security and Trust*, 2017.

20. A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. In *Proceedings of the 23rd International Conference on Fast Software Encryption*, pages 313–333, 2016.
21. S. Carpov, S. Aubry, and R. Sirdey. A multi-start heuristic for multiplicative depth minimization of boolean circuits. In *Proceedings of the 28th International Workshop on Combinatorial Algorithms*, 2017.
22. S. Carpov, P. Dubrulle, and R. Sirdey. Armadillo: A Compilation Chain for Privacy Preserving Applications. In *Proceedings of the 3rd International Workshop on Security in Cloud Computing*, pages 13–19, 2015.
23. S. Carpov, T.-H. Nguyen, R. Sirdey, G. Constantino, and F. Martinelli. Practical privacy-preserving medical diagnosis using homomorphic encryption. In *Proceedings of the 9th IEEE Conference on Cloud Computing*, pages 593–599, 2016.
24. A. Chakraborti, A. Chattopadhyay, M. Hassan, and M. Nandi. TriviA: a fast and secure authenticated encryption scheme. In *Proceedings of CHES'15*, volume 9293 of *LNCS*, pages 330–353, 2015.
25. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachne. Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In *Proceedings of ASIACRYPT*, pages 3–33, 2017.
26. A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In G. Bertoni and B. Gierlichs, editors, *FDTC 2012*, pages 7–15, Leuven, Belgium, September 9, 2012. IEEE Computer Society.
27. I. Dinur and A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. In *EUROCRYPT*, volume 5479 of *LNCS*, pages 278–299. Springer, 2009.
28. C. Dobraunig, M. Eichlseder, T. Korak, V. Lomné, and F. Mendel. Statistical fault attacks on nonce-based authenticated encryption schemes. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 369–395, 2016.
29. ECRYPT - European Network of Excellence in Cryptology. The eSTREAM Stream Cipher Project. http://www.ecrypt.eu.org/stream/, 2005.
30. D. El-Baze, J. B. Rigaud, and P. Maurine. A fully-digital EM pulse detector. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 439–444, March 2016.
31. J. Fan and F. Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
32. X. Fang, P. Luo, Y. Fei, and M. Leeser. Balance power leakage to fight against side-channel analysis at gate level in fpgas. In *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 154–155, July 2015.
33. W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten. *Differential Power Analysis of Stream Ciphers*, pages 257–270. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
34. P.-A. Fouque and T. Vannet. Improving Key Recovery to 784 and 799 Rounds of Trivium Using Optimized Cube Attacks. In *FSE*, volume 8424 of *LNCS*, pages 502–517. Springer, 2013.
35. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. pages 251–261, 2001.
36. C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, 2009.

37. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic Evaluation of the AES Circuit. In *CRYPTO*, volume 7417 of *LNCS*, pages 850–867. Springer, 2012.
38. C. Gentry, A. Sahai, and B. Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In *CRYPTO*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, 2013.
39. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual information analysis. pages 426–442, 2008.
40. J. D. Golic. Cryptanalysis of alleged A5 stream cipher. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 239–255. Springer-Verlag, 1997.
41. S. Halevi and V. Shoup. Algorithms in HElib. In *CRYPTO*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571, 2014.
42. S. Halevi and V. Shoup. Bootstrapping for HElib. In *EUROCRYPT*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, 2015.
43. M. Hojsík and B. Rudolf. *Differential Fault Analysis of Trivium*, pages 158–172. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
44. J. M. Jaffe, P. C. Kocher, and B. C. Jun. Balanced cryptographic computational method and apparatus for leak minimizational in smartcards and other cryptosystems, January 21 2003. US Patent 6,510,518.
45. A. Journault and F.-X. Standaert. Very high order masking: Efficient implementation and security evaluation. In *Conference on Cryptographic Hardware and Embedded Systems - CHES 2017*, LNCS. Springer, 2017. To appear.
46. D. Karaklajic, J.-M. Schmidt, and I. Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Trans. Very Large Scale Integr. Syst.*, 21(12):2295–2306, December 2013.
47. C. H. Kim, J. H. Shin, J.-J. Quisquater, and P. J. Lee. Safe-Error Attack on SPA-FA Resistant Exponentiations Using a HW Modular Multiplier. In K.-H. Nam and G. Rhee, editors, *Information Security and Cryptology - ICISC 2007*, volume 4817 of *LNCS*, pages 273–281. Springer, 2007.
48. S. Knellwolf, W. Meier, and M. Naya-Plasencia. Conditional Differential Cryptanalysis of Trivium and KATAN. In *SAC*, volume 7118 of *LNCS*, pages 200–212. Springer, 2011.
49. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. pages 104–113, 1996.
50. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. pages 388–397, 1999.
51. F. Koeune and F.-X. Standaert. A tutorial on physical security and side-channel attacks. In *Foundations of Security Analysis and Design III, FOSAD 2004/2005 Tutorial Lectures*, volume 3655, pages 78–108, 2005.
52. B. Lac, M. Beunardeau, A. Canteaut, J. J. A. Fournier, and R. Sirdey. A First DFA on PRIDE: from Theory to Practice. In *International Conference on Risks and Security of Internet and Systems - CRiSIS 2016*, volume 10158 of *LNCS*, pages 214–238. Springer, September 2016.
53. B. Lac, A. Canteaut, J. J. A. Fournier, and R. Sirdey. DFA on LS-Designs with a Practical Implementation on SCREAM. In *Constructive Side-Channel Analysis and Secure Design - COSADE 2017*, LNCS. Springer, 2017. To appear.
54. B. Lac, A. Canteaut, J. J. A. Fournier, and R. Sirdey. Thwarting fault attacks using the internal redundancy countermeasure (irc). Cryptology ePrint Archive, Report 2017/910, 2017. http://eprint.iacr.org/2017/910.
55. R. Lashermes, J. Fournier, and L. Goubin. Inverting the final exponentiation of Tate pairings on ordinary elliptic curves using faults. pages 365–382, 2013.

56. M. Liu. Degree evaluation of NFSR-based cryptosystems. In *CRYPTO*, volume 10402 of *LNCS*. Springer, 2017.
57. O. Lo, W. J. Buchanan, and D. Carson. Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa). *Journal of Cyber Security Technology*, 1(2):88–107, 2017.
58. P. Luo, L. Zhang, Y. Fei, and A. A. Ding. Towards secure cryptographic software implementation against side-channel power analysis attacks. In *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 144–148, July 2015.
59. H. Maghrebi, J. L. Danger, F. Flament, S. Guilley, and L. Sauvage. Evaluation of countermeasure implementations based on Boolean masking to thwart side-channel attacks. In *International Signals, Circuits and Systems Conference - SCS 2009*, pages 1–6, 2009.
60. H. Maghrebi, S. Guilley, J.-L. Danger, and F. Flament. Entropy-based power attack. In J. Plusquellic and K. Mai, editors, *HOST 2010*, pages 1–6, Anaheim Convention Center, California, USA, June 13-14, 2010. IEEE Computer Society.
61. S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
62. P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In *EUROCRYPT*, volume 9665, pages 311–343. Springer, 2016.
63. C. Aguilar Melchor, P. Gaborit, and J. Herranz. Additively homomorphic encryption with d-operand multiplications. In *Proceedings of CRYPTO*, 2010.
64. H. Mestiri, N. Benhadjyoussef, M. Machhout, and R. Tourki. A robust fault detection scheme for the advanced encryption standard. *IJCNIS*, 5(6):49–55, 2013.
65. M. S. E. Mohamed, S. Bulygin, and J. A. Buchmann. Using SAT solving to improve differential fault analysis of trivium. In T.-H. Kim, H. Adeli, R. J. Robles, and M. O. Balitanas, editors, *ISA 2011*, volume 200 of *Communications in Computer and Information Science*, pages 62–71, Brno, Czech Republic, August 15-17, 2011.
66. D. Page. Theoretical use of cache memory as a cryptanalytic side-channel. Cryptology ePrint Archive, Report 2002/169, 2002. http://eprint.iacr.org/2002/169.
67. D. Page. Defending against cache based side-channel attacks. *Information Security Technical Report*, 8(1):30–44, April 2004.
68. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart 2001*, volume 2140, pages 200–210, Cannes, France, September 19-21, 2001.
69. K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta. Information-theoretic approach to optimal differential fault analysis. *IEEE Transactions on Information Forensics and Security*, 7(1):109–120, 2012.
70. K. Singh, R. Sirdey, F. Artiguenave, D. Cohen, and S. Carpov. Towards confidentiality-strengthened personalized genomic medicine embedding homomorphic cryptography. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, pages 423–430, 2017.
71. S. Skorobogatov. Semi-invasive attacks - A new approach to hardware security analysis. Technical Report 630, University of Cambridge, April 2005.
72. S. P. Skorobogatov and R. J. Anderson. Optical fault induction attacks. pages 2–12, 2003.
73. Y. Todo, T. Isobe, Y. Hao, and W. Meier. Cube attacks on non-blackbox polynomials based on division property. In *CRYPTO*, volume 10402 of *LNCS*. Springer, 2017.

74. H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. Differential fault analysis on the families of SIMON and SPECK ciphers. Cryptology ePrint Archive, Report 2014/267, 2014. http://eprint.iacr.org/2014/267.
75. J. G. J. van Woudenberg, M. F. Witteman, and F. Menarini. Practical optical fault injection on secure microcontrollers. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 91–99, Sept 2011.
76. X. Zhao, T. Wang, and S. Guo. Improved side channel cube attacks on PRESENT. Cryptology ePrint Archive, Report 2011/165, 2011. http://eprint.iacr.org/2011/165.