Light-weight primitive, feather-weight security – a cryptanalytic knock-out

(Preliminary results)

V. Banciu
University of Bristol, Dept. of Computer Science,
Merchant Venturers Building
Woodland Road
Bristol, United Kingdom
Valentina.Banciu@bristol.ac.uk

S. Hoerder
University of Bristol, Dept. of Computer Science,
Merchant Venturers Building
Woodland Road
Bristol, United Kingdom
hoerder@cs.bris.ac.uk

D. Page
University of Bristol, Dept. of Computer Science,
Merchant Venturers Building
Woodland Road
Bristol, United Kingdom
page@cs.bris.ac.uk

ABSTRACT

In [12], the authors present a new light-weight cryptographic primitive which supports an associated RFID-based authentication protocol. The primitive has some structural similarities to AES, but is presented as a keyed one-way function using a 128-bit key. Although a security analysis is included, this is at a high-level only. To provide a more concrete idea as to the security of this primitive, we therefore make three contributions: first, a structural attack requiring $\mathcal{O}(2^5)$ plaintext/ciphertext pairs (and hence effort online) plus $\mathcal{O}(2^{21})$ effort offline, second algebraic attacks on round reduced versions of the primitive which requires only a single plaintext/ciphertext pair, and, third debunk the claimed attack of [36] on the same primitive. Our structural attack completely breaks the primitive and the algebraic attack highlights a crucial weakness of the primitive; we conclude that although one can consider countermeasures against these specific attacks, the design in general is questionable and should therefore be avoided.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Code breaking; C.3 [**Special-Purpose And Application-Based Systems**]: Real-time and embedded systems; D.4.6 [**Security and Protection**]: Authentication

Keywords

Light-weight block cipher, structural attack, algebraic attack, RFID authentication

1. INTRODUCTION

The ubiquity of modern information systems highlights the importance of all enabling technologies, and their ability to keep pace with the demands of associated applications. An archetypal example is that of Radio Frequency IDentification (RFID) tags, which interact with a reader (or terminal); such combinations form the basis for applications such as supply chain management and access control, where security and privacy [19] form central requirements. Although traditional cryptographic technologies offer many solutions, application-specific development and analysis of appropriate security models (see, e.g., [39]) is often still necessary.

This challenge is non-trivial, but not unique to RFID. Other exacerbating factors can be identified however: among the most important are the strict constraints on computational and storage capability placed on RFID tags, which impact greatly on the underlying protocols and primitives. Such constraints are driven by manufacturing cost in part, but also by a requirement for energy efficiency [21]. For example, active tags may include an on-board but low-capacity battery, while passive tags are even more constrained as a result of being powered remotely via electrical current induced in the antenna (i.e., by the reader). Since obvious candidates such as AES may not be suitable therefore, the research field of light-weight cryptography[1] has emerged. Related work broadly encompasses two strategies, namely

1. retro-fitting an existing primitive or protocol to meet constraints, e.g., through efficient implementation or parameter selection, and
2. specific design of new primitives and protocols with the goal of light-weightness in mind, e.g., to make best use of the platform characteristics.

Among related work, pertinent examples include the (standardised) block cipher PRESENT [5] and the hash function PHOTON [16]. Despite an increasing volume of such designs, precise definition of what the term light-weight means is more difficult than one might imagine; one might suggest low-footprint for instance, but Knezevic et al. [23] explore low-latency as a possible alternative. Either way, it should be clear that *security* light-weightness, i.e., insecurity, is highly unattractive: arguments such as [25, 34] posit that security should instead represent a first-class design goal.

[1]For an excellent overview, see [32, Chapters 1 and 2].

Finding appropriate trade-offs which meet this requirement while simultaneously catering for any resource constraints is difficult however. In common with other examples, this is highlighted by the EC-RAC family [27, 28, 26] of ECC-based protocols, each of which was broken [6, 38, 14].

While each attack on EC-RAC can be broadly characterised as relating to the security model and/or protocol, one might also ask a question of any given underlying primitive: how light-weight is *too* light-weight? Along such lines, this paper examines an RFID-based authentication protocol due to Dusart and Traoré [12]. Said protocol makes use of a light-weight keyed one-way function; following [12], we use

$$C \leftarrow h_K(M)$$

to denote this "Dusart-Traoré primitive" where a key K is used to process some message M. With respect to the Dusart-Traoré primitive, we answer the question above via contributions in Section 3. More specifically, we first demonstrate an efficient structural attack in Section 3.2, then, in Section 3.3, algebraic attacks that give a far more precise security estimate regarding the feasibility of such attacks against the primitive than the claims in [12]. In Section 3.4.3 we debunk the claim made in [36] of having broken the Dusart-Traoré primitive; in fact we demonstrate that the attack in [36] is not an attack at all. In addition to this analysis, we try to answer the question whether components of the Dusart-Traoré primitive can be salvaged for the design of future light-weight primitives by suggesting potential countermeasures in Section 3.4.4 and conducting a comparative efficiency estimation in Section 3.4.5.

2. BACKGROUND AND NOTATION

This Section describes the authentication protocol and supporting primitive presented in [12, Sections 4 and 5]; for clarity we adopt the same notation, with specific additions introduced where required.

The protocol and primitive. The protocol assumes two pre-shared secrets ID and K are held by the reader and tag. The reader sends the a challenge c to the tag, which computes and returns the pair

$$[\; ID \oplus h_K(c),\; h_{ID}(c)\;]. \tag{1}$$

Since the reader can (re)compute $h_K(c)$ and $h_{ID}(c)$, it can check whether the tag produced the correct result for c and thus infer knowledge of ID and K.

A central component is clearly h, the Dusart-Traoré primitive. The design goals [12, Section 5] (specifically that it is one-way, and produces random output) mean h can be described as a keyed one-way function. Figure 1 illustrates the internal structure, which is essentially a 4-round substitution-permutation network, with no key scheduling, that operates on a 128-bit state (which, in common with AES for instance, is organised as a 16-element array of 8-bit bytes). Although the AES S-box is used for the substitution layer, a bespoke function F realises the permutation layer: within the j-th round, this is defined as

$$\begin{aligned} F^j(M^j) \;=\; & f(M^j_0, M^j_{0+2^{j-1} \bmod 16}) \;\| \\ & f(M^j_1, M^j_{1+2^{j-1} \bmod 16}) \;\| \\ & \ldots \;\| \\ & f(M^j_{15}, M^j_{15+2^{j-1} \bmod 16}) \end{aligned} \tag{2}$$

where $\|$ denotes concatenation, M^j_i denotes the i-th state element within the j-th round, and

$$f(x,y) = \Big[\big[x \oplus ((255-y) \ll 1)\big] + \big[((255-x) \oplus (y \gg 1))_{\bmod 16}\big] \cdot 16\Big]_{\bmod 256}. \tag{3}$$

Note that we later use $M^j_i[a]$ to denote the a-th bit of said state element, extending the notation to $M^j_i[a \ldots b]$ so ranges of bits (between the a-th and b-th inclusive) can be specified.

Security model. Although [12, Section 2] lists some high-level assumptions, a concrete security model is lacking.

Peculiarly, although the assumptions state K must be shared between tag and reader, clearly ID must also remain secret. As such, the reader is implicitly assumed to know which tag it communicates with. However, the application-specific utility of the protocol for authentication (and indeed the protocol itself) are outside our scope.

That said, a key recovery attack on h would clearly allow an attacker to clone a tag. After collecting a (set of) authentication tuple(s), i.e., examples of Equation 1 for known (and controlled) c, the attacker proceeds as follows: first it recovers ID by targeting the $h_{ID}(c)$ component, then recovers K by targeting the $h_K(c)$ component (after removing each XOR with the now known ID). Once ID and K are known, the attacker can fool the reader into authenticating something other than the associated tag. This strategy forms the basis for our attacks within Section 3.

Provided security analysis. Security analysis of the primitive [12, Section 7.3] offers no exact security estimates, except for brute-force attacks. Instead, the analysis states

> "Since each round of the algorithm operations are performed modulo 16 or modulo 256 and the results from these transactions are processed by substitution tables, the [algorithm] is very difficult to analyze algebraically."

Any resilience against differential [4] and linear [29] cryptanalysis is limited to discussion of the S-box, and culminates in the statement:

> "Our function is well resistant to this [differential] attack. [...] We know that [...] the equation has

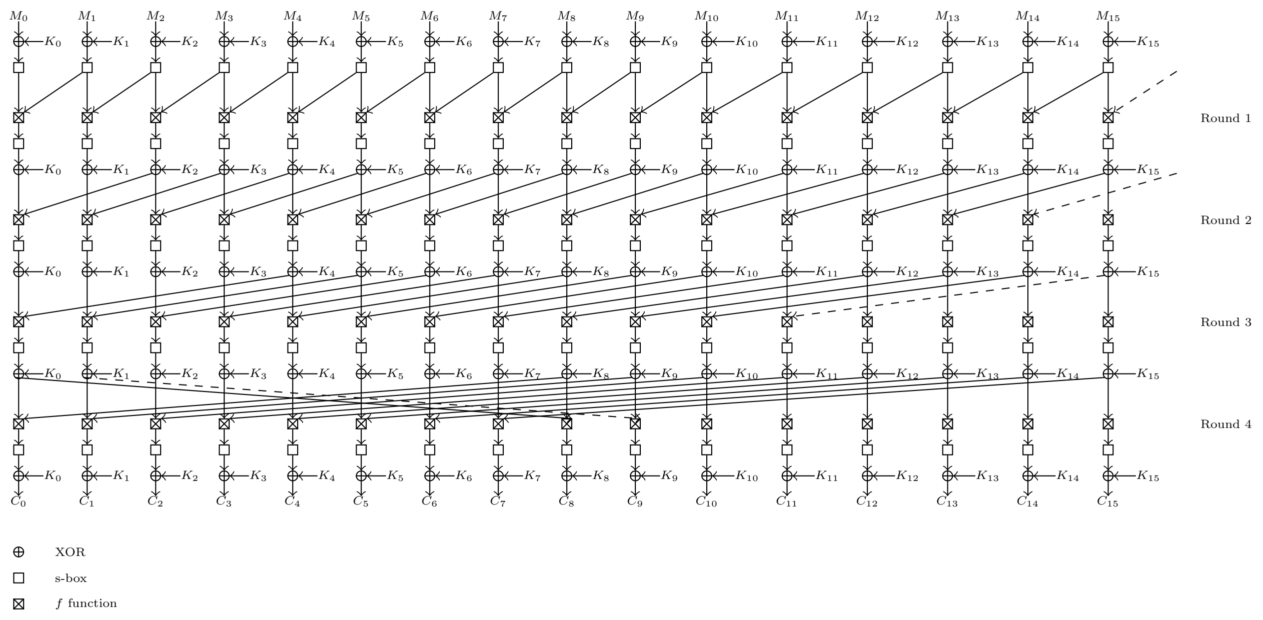

Figure 1: Structure of the Dusart-Traoré primitive (some input arrows to the f function are omitted for clarity).

a number of solutions close to 128 which makes expensive the linear attack."

Additional security considerations [12, Sections 7.1 and 7.2] are presented in the paper, focusing in particular on the authentication protocol; examples include man-in-the-middle and side-channel style approaches.

3. SECURITY ANALYSIS

Within this Section we split our security analysis of the Dusart-Traoré primitive into four parts:

- Section 3.1 demonstrates what the primitive is not, namely a one-way function, by demonstrating how to invert it.
- In Section 3.2 we demonstrate an efficient, structural chosen plaintext attack that provides an asymptotic upper bound on the security of the primitive.
- In Section 3.3 we provide detailed security analysis of the cipher against algebraic attacks which requires only a single observed plaintext/ciphertext pair, and can therefore be executable by a passive adversary.
- We conclude the Section by considering what the Dusart-Traoré primitive actually is (since it cannot be a one-way function), debunking the claimed attack of [36], and assessing whether further study of some components may lead to secure light-weight primitives in the future.

3.1 Inversion of F^j

Since the Dusart-Traoré primitive is supposed to be a one-way function, no inversion algorithm is given in [12]. However, superficial inspection of the internal structure suggests it *may* be invertible *if* the key K is known: of the three components (key addition via XOR, substitution layer, and permutation layer F^j), only the bespoke F^j lacks a known inversion algorithm. Put another way, *if* we can show F^j is invertible, an inversion algorithm for the Dusart-Traoré primitive becomes trivial.

Inverting F^j is essentially the same as inverting f, but one has to account for the permutation in round j. For $j = 4$, i.e., the last round, inversion is particularly simple since the permutation pairs two symmetric input bytes used via f, i.e.,

$$\big[f(M_0^4, M_8^4), f(M_8^4, M_0^4)\big],\ \big[f(M_1^4, M_9^4), f(M_9^4, M_1^4)\big],\ \ldots$$

This means instead of considering

$$F^{-1}: \qquad \mathbb{Z}_2^{8^{16}} \to \mathbb{Z}_2^{8^{16}},$$

we only have to consider

$$f^{-1}: \qquad \mathbb{Z}_2^8 \times \mathbb{Z}_2^8 \to \mathbb{Z}_2^8 \times \mathbb{Z}_2^8.$$

Both F^{-1} and f^{-1} can be computed using the same Boolean equations, but the latter requires far less of them. In the following, we explain how this can be achieved, focusing wlog. on the example of $f^{-1}(\tilde{M}_0^4, \tilde{M}_8^4)$ which yields M_0^4 and M_8^4 as output given the pair

$$\big[\tilde{M}_0^4 = f(M_0^4, M_8^4), \tilde{M}_8^4 = f(M_8^4, M_0^4)\big]$$

as input.

In [12, Section 6.2], the authors explain that

$$\begin{aligned} (255 - x) &\equiv (255 \oplus x) \\ x \bmod 16 &\equiv x \odot 15 \\ x \cdot 16 &\equiv x \ll 4 \end{aligned}$$

where $\ll$, $\odot$ and $\oplus$ denote logical left-shift, AND and XOR respectively, and that for addition modulo 256 any overflow

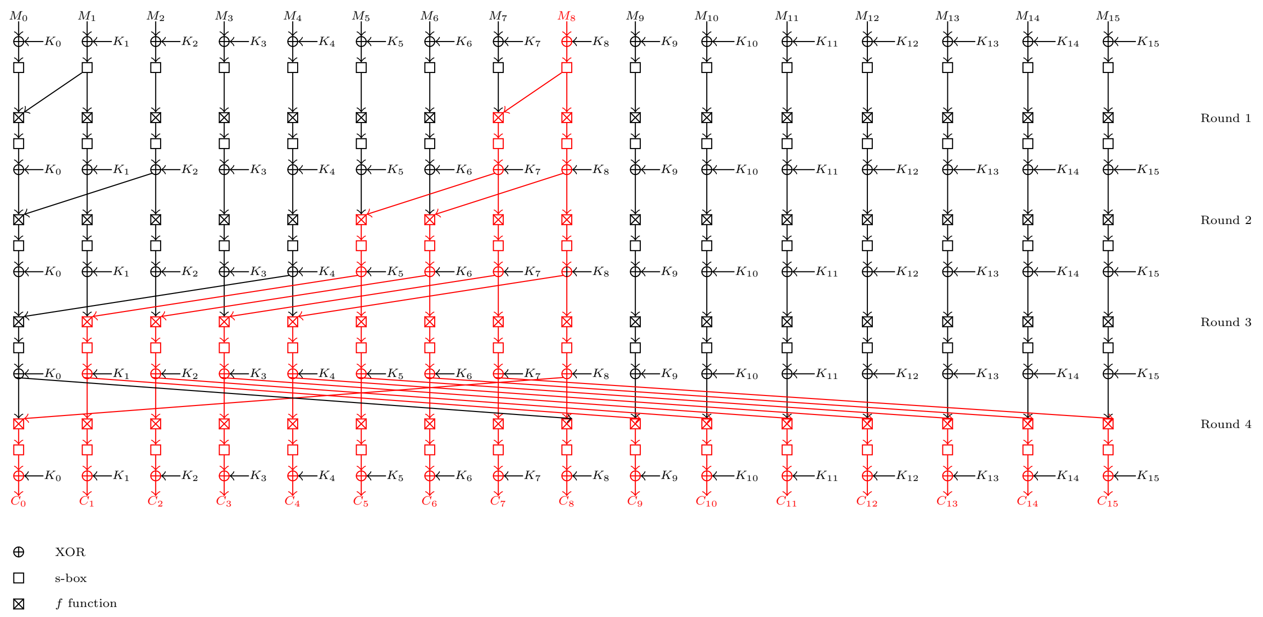

Figure 2: Exemplary differential propagation of the Dusart-Traoré primitive for a single differential occurring in M_8. Differentials are shown in red.

can be ignored. Using these equivalences, Equation 3 can be rewritten as

	$\tilde{M}_0^4$	$\leftarrow$	$f(M_0^4, M_8^4)$		
#0	$\tilde{M}_0^4[0]$	$\leftarrow$	$M_0^4[0]$	$+$	0
#1	$\tilde{M}_0^4[1]$	$\leftarrow$	$M_0^4[1] \oplus 1 \oplus M_8^4[0]$	$+$	0
#2	$\tilde{M}_0^4[2]$	$\leftarrow$	$M_0^4[2] \oplus 1 \oplus M_8^4[1]$	$+$	0
#3	$\tilde{M}_0^4[3]$	$\leftarrow$	$M_0^4[3] \oplus 1 \oplus M_8^4[2]$	$+$	0
#4	$\tilde{M}_0^4[4]$	$\leftarrow$	$M_0^4[4] \oplus 1 \oplus M_8^4[3]$	$+$	$1 \oplus M_0^4[0] \oplus M_8^4[1]$
#5	$\tilde{M}_0^4[5]$	$\leftsquigarrow$	$M_0^4[5] \oplus 1 \oplus M_8^4[4]$	$+$	$1 \oplus M_0^4[1] \oplus M_8^4[2]$
#6	$\tilde{M}_0^4[6]$	$\leftsquigarrow$	$M_0^4[6] \oplus 1 \oplus M_8^4[5]$	$+$	$1 \oplus M_0^4[2] \oplus M_8^4[3]$
#7	$\tilde{M}_0^4[7]$	$\leftsquigarrow$	$M_0^4[7] \oplus 1 \oplus M_8^4[6]$	$+$	$1 \oplus M_0^4[3] \oplus M_8^4[4]$

and

	$\tilde{M}_8^4$	$\leftarrow$	$f(M_8^4, M_0^4)$		
#0	$\tilde{M}_8^4[0]$	$\leftarrow$	$M_8^4[0]$	$+$	0
#1	$\tilde{M}_8^4[1]$	$\leftarrow$	$M_8^4[1] \oplus 1 \oplus M_0^4[0]$	$+$	0
#2	$\tilde{M}_8^4[2]$	$\leftarrow$	$M_8^4[2] \oplus 1 \oplus M_0^4[1]$	$+$	0
#3	$\tilde{M}_8^4[3]$	$\leftarrow$	$M_8^4[3] \oplus 1 \oplus M_0^4[2]$	$+$	0
#4	$\tilde{M}_8^4[4]$	$\leftarrow$	$M_8^4[4] \oplus 1 \oplus M_0^4[3]$	$+$	$1 \oplus M_8^4[0] \oplus M_0^4[1]$
#5	$\tilde{M}_8^4[5]$	$\leftsquigarrow$	$M_8^4[5] \oplus 1 \oplus M_0^4[4]$	$+$	$1 \oplus M_8^4[1] \oplus M_0^4[2]$
#6	$\tilde{M}_8^4[6]$	$\leftsquigarrow$	$M_8^4[6] \oplus 1 \oplus M_0^4[5]$	$+$	$1 \oplus M_8^4[2] \oplus M_0^4[3]$
#7	$\tilde{M}_8^4[7]$	$\leftsquigarrow$	$M_8^4[7] \oplus 1 \oplus M_0^4[6]$	$+$	$1 \oplus M_8^4[3] \oplus M_0^4[4]$

The addition of zero in the lines #0 to #3 has no effect and can be ignored entirely. The least-significant bits of M_0^4 and M_8^4, namely $M_0^4[0]$ and $M_8^4[0]$, are produced directly from the equations in line #0; this allows successive recovery of $M_{\{0,8\}}^4[1\ldots3]$ by solving equations in lines #1 to #3. In lines #4 to #7 we must accommodate the effect of addition, doing so by using equations that describe a full-adder:

$$\begin{aligned} s &\leftarrow a \oplus b \oplus c_{in} \\ c_{out} &\leftarrow (a \odot b) \oplus (c_{in} \odot (a \oplus b)) \end{aligned}$$

In line #4 we have no carry-in yet, i.e. $c_{in}[4] = 0$ but all other operands are known. Thus we can easily obtain $M_{\{0,8\}}^4[4]$, and compute $c_{out}[4]$. For the lines #5 to #7 we proceed accordingly but have to take the carry-out of the preceding line into account, e.g., $c_{in}[5] = c_{out}[4]$: this is highlighted by the $\leftsquigarrow$ notation. The final carry-out $c_{out}[7]$ can be discarded, since the addition is modulo 256.

Crucially, notice that the process of inverting f^{-1} is both deterministic and efficient.

3.2 A structural attack

The key recovery attack demonstrated in this Section is similar to (but simpler than) structural attacks on AES (see, e.g., [10, Chapter 10.2] for a discussion of the latter). Usually, block ciphers use a relatively large number of rounds to defeat such structural attacks; AES-128 and PRESENT have 10 and 31 rounds respectively for instance, whereas 4 rounds as used by the Dusart-Traoré primitive signal potential for this approach to be successful. Indeed, if one looks at the differential trail shown in Figure 2 (noting that all input bytes M_i except M_8 are constant) it becomes obvious that, based on Equation 2 we can exploit the symmetry of $f(M_0^4, M_8^4)$ and $f(M_8^4, M_0^4)$ where M_0^4 is constant whereas M_8^4 changes for each M_8.

As such, the basic idea is to first generate a sufficient number of (M, C) pairs that differ in M_8 only. This requires the attacker to send a (set of) challenges $c = M$ to the reader and record each response $h_K(M) = C$. Then, for all possible key byte candidates $[K_0, K_8]$, we test whether inversion of the final round, i.e., computing

$$\begin{aligned} \tilde{M}_0^4 &\leftarrow S^{-1}(C_0 \oplus K_0) \\ \tilde{M}_8^4 &\leftarrow S^{-1}(C_8 \oplus K_8) \\ \left[M_0^4, M_8^4\right] &\leftarrow f^{-1}(\tilde{M}_0^4, \tilde{M}_8^4) \end{aligned}$$

yields $\left[M_0^4, M_8^4\right]$ such that M_0^4 is constant for all (M, C) pairs. This process can then be repeated for differentials in $M_9, \ldots, M_{15}$ to recover the remaining key bytes. Algorithm 1 outlines the concrete attack; note that S^{-1} denotes inversion of the AES S-box while f^{-1} represents application

Algorithm 1: The structural attack.

Input: A bound n on the number of plaintext/ciphertext pairs used.
Output: A list of candidates for each key byte.
Data: An array $\mathbf{C}$ to temporarily store n ciphertexts, an array $\mathbf{M_i^4}$ to temporarily store n bytes M_i.

```
   // Iterate through all key byte tuples
1  for $i = 0$ upto 7 do
       // Online:
2      for $j = 0$ upto $n-1$ do
           // Choose suitable $M$, store corresponding $C$
3          $M_{0,\dots,15} \leftarrow \{0,\dots,0\}$
4          $M_{i+8} \leftarrow j$
5          $\mathbf{C}[j] \leftarrow$ challengeDevice($M$)
6      end
       // Offline: Determine candidates for $[K_i, K_{i+8}]$
7      for $k_A = 0$ upto 255 do
8          for $k_B = 0$ upto 255 do
9              for $j = 0$ upto $n$ do
10                 $C \leftarrow \mathbf{C}[j]$
11                 $\tilde{M}_i^4 \leftarrow S^{-1}(C_i \oplus k_A)$
12                 $\tilde{M}_{i+8}^4 \leftarrow S^{-1}(C_{i+8} \oplus k_B)$
13                 $[M_i^4, M_{i+8}^4] \leftarrow f^{-1}(\tilde{M}_i^4, \tilde{M}_{i+8}^4)$
14                 $\mathbf{M_i^4}[j] \leftarrow M_i^4$
15             end
               // Do all $M_i$ have the same value?
16             isCandidate ← true
17             for $j = 1$ upto $n-1$ do
18                 if $\mathbf{M_i^4}[j-1] \neq \mathbf{M_i^4}[j]$ then
19                     isCandidate ← false
20                 end
21             end
22             if isCandidate then
                   // We found candidates for $[K_i, K_{i+8}]$:
23                 printCandidates($i, [k_A, k_B]$)
24             end
25         end
26     end
27 end
```

of the aforementioned approach to inversion of f.

The run-time complexity of the algorithm is determined by the loops in line #1 (number of key bytes divided by two), lines #7 and #8 (number of possible key byte values) and line #9 (number of (M, C) pairs being used). This gives an overall run-time complexity of

$$\mathcal{O}(2^{3+8+8+\log_2(n)}) = \mathcal{O}(2^{19+\log_2(n)})$$

and memory complexity of $\mathcal{O}(n)$.

We implemented the attack[2] and experimentally tested it against 1000 randomly chosen keys: the results are shown in Table 1. The second row shows the number of keys that could be uniquely identified using at most n (M, C) pairs per key byte tuple. The last row shows the average number

[2]Code will be published online once the paper has been published.

n	successful key recoveries	avg. remaining candidates per tuple
2	0	260.034
3	0	2.043
4	968	1.006
5	31	1.031
6	1	1.000

Table 1: Experimental results using 1000 randomly sampled keys.

of key candidate tuples $[K_i, K_{i+8}]$ remaining after using n (M, C) pairs. The mean of the minimum number of pairs required to successfully recover a secret key is $n = 4.033$: this means an average run-time complexity of $\mathcal{O}(2^{21})$, requiring and average $8 \cdot 4.033 \approx 2^5$ of chosen (M, C) pairs. The results also show that the dispersion around the mean value is very small, with a standard deviation $\sigma \approx 0.184$. Furthermore, the reduction of possible candidates for a tuple $[K_i, K_{i+8}]$ happens rapidly; this means that even in cases where the number of available (M, C) pairs is insufficient to uniquely identify the correct tuple, the complexity of an exhaustive search through the remaining tuples is much improved.

We ran our experiment on a Intel Core i5-480M processor with 6GB of RAM, which performed the entire experiment in just over 700 seconds. Thus, a single key can comfortably be recovered in less than a second (without considering RFID communication overhead).

3.3 An algebraic attack

Algebraic attacks have been applied in the context of stream ciphers [7, 2] and block ciphers [9] alike. The idea is to express the underlying algorithm as a system of Boolean equations (involving the plaintext and ciphertext bits as known inputs and the key bits as unknowns) which can then be solved using automated tools. For bijective functions, and we have already shown the Dusart-Traoré primitive to be bijective in Section 3.1, a single observed (M, C) pair is sufficient to uniquely identify the secret key in an algebraic attack. Ideally, a modern cipher will lead to a system of equations which is as computationally hard to solve as performing a brute-force search on the key space [35].

In order to evaluate the resistance of the Dusart-Traoré primitive against algebraic attacks, we chose to use the CryptoMiniSat2 [37] SAT solver, which is one of the best available non-commercial solvers. Put simply, the underlying 3SAT problem is NP-complete and thus has an exponential worst-case run-time. CryptoMiniSat2 achieves a comparatively good average run-time using a probabilistic Conflict-Driven Clause-Learning (CDCL) approach. More precisely, as the system of equations has a unique solution, i.e., is solvable, only the run-time of CryptoMiniSat2 is probabilistic.

Intuitively, one has to wonder why anyone would want to translate a problem that is, as per Section 3.2 efficiently solvable into a NP-complete problem. Indeed, the authors of [8] for example question whether any block cipher will ever be broken using algebraic attacks and SAT solvers. From our point of view, it is interesting to test the resilience of a primitive against algebraic attacks for a range of reasons:

- They can be performed by passive adversaries who need to observe only a single (M, C) pair. Algebraic attacks are thus very useful in cases where the secret key gets updated after each invocation of the block cipher.
- In some cases an algebraic attack step using SAT solvers is added to side-channel or fault injection attacks in order to improve results (see, e.g., [30, 33, 40, 18]). Thus, it is important to know a primitive's security margin against algebraic attacks when estimating its resilience against side-channel or fault-injection attacks.
- They do not require any specific cryptanalytic knowledge and use only well established methods (i.e., equation translation and SAT solving) common to many areas of computer science. In addition, for common building blocks such as the AES S-box, optimized formulas ([9]) have been published and are readily available. Thus experts in other research areas, e.g., above mentioned side-channel and fault-injection attacks, will find it easy (albeit somewhat tedious) to integrate them into their attacks.

For the algebraic attack we consider round-reduced variants of the Dusart-Traoré primitive, each starting with the first round. In order to obtain the input equations for CryptoMiniSat2, we used the two step translation approach from [3]

$$\begin{tikzcd}[row sep=small] \text{polynomial} \arrow[d, Rightarrow] \\ \text{linear system} \arrow[d, Rightarrow] \\ \text{conjunctive normal form} \end{tikzcd}$$

which returns Conjunctive Normal Form (CNF) (see, e.g., [1]) equations with exactly 3 literals per clause as required by 3SAT (The CNF equations will be published online once the paper has been published). As the S-box is identical to the AES S-box, we used the system of 23 quadratic equations from [9] to describe it in the first step. We used one of the testvectors given by the authors of [12] as (M, C) pair and, due to submission deadlines, aborted CryptoMiniSat2 for run-times greater than two days. The experiment was run single threaded on a server with an Intel Xeon E5620 processor and 49 GB memory. The results we obtained are shown in Table 2. For 1 and 2 round versions, the algebraic attack was successful and fast whereas for 3 and 4 rounds the attack did not succeed within our imposed time limit. This is due to a combination of the cipher's diffusion property which increases exponentially in the number of rounds r, achieving full diffusion in the last round which makes it more difficult for the SAT solver to identify partial solutions as well as the exponential nature of the SAT problem.

In comparison, AES-128 achieves full diffusion, i.e., all key bytes affect each byte of the round output, in only two rounds and has far more rounds providing a far better security margin. This indicates that, for the Dusart-Traoré primitive, side-channel and fault injection attacks with an algebraic attack enhancement have a comparatively large number of rounds that can be exploited. Still, our experiments suggest that the Dusart-Traoré primitive is at least able to resist pure algebraic attacks.

r	Input		Resources		Solved
	clauses	variables	time	RAM	
1	114256	15712	0.55 s	65.26 MB	✓
2	172064	23600	453.14 s	261.91 MB	✓
3	229872	31488	> 2 d	$\gtrsim$ 3537.96 MB	–
4	287680	39376	> 2 d	$\gtrsim$ 7292.31 MB	–

Table 2: Results of our algebraic attack on round reduced versions of the Dusart-Traoré primitive. r denotes the number of rounds. For the aborted attacks($r \in \{3, 4\}$), CryptoMiniSat2 did not produce RAM usage statistics so we used the `VmPeak` measure of linux's `/proc/PID/status` instead.

3.4 Additional comment and analysis

3.4.1 Keyed hash function or block cipher?

In the conference presentation and in private communication, the authors of [12] describe h as a hash function while the paper only refers to it as a one-way function; clearly, some confusion about what h is supposed to be exists. Based on the results of the preceding sections, we challenge the claim that h is a keyed hash function and, as part of this argument, the claim that h is a one-way function:

- A basic characteristic of any hash function is that it compresses a message of arbitrary length into a fixed length digest. In contrast, the Dusart-Traoré primitive is specified as a mapping of fixed length inputs onto fixed length outputs, namely
$$\mathbb{Z}_2^{8^{16}} \times \mathbb{Z}_2^{8^{16}} \to \mathbb{Z}_2^{8^{16}}$$
It furthermore requires that one of the $\mathbb{Z}_2^{8^{16}}$ inputs has high entropy as it is used as secret key.
- There is no analysis of pre-image or collision resistance, nor of attack strategies such as length extension. More specifically, a hash function should be one-way functions, i.e., inverting it to compute a pre-image should be infeasible. Given our results in Section 3.1, it is clear that the Dusart-Traoré primitive can be inverted once the secret key input is known; our results from Section 3.2 and Section 3.3 also demonstrate that it is practically feasible to recover secret keys. As a result, the primitive can not be considered to be a one-way function.

As a result of the above, we propose the primitive is considered a (broken) block cipher. Given this classification, note the effectiveness of our two attacks prompted us *not* to further investigate potential weakness of the primitive in block cipher based MAC constructions such as CBC-MAC (see, e.g., [20, Chapter 4.5]) or other standard attacks against block ciphers such as differential [4] and linear [29] cryptanalysis.

3.4.2 Security analysis via statistical tests of randomness

The most concrete security analysis of the Dusart-Traoré primitive is in [12, Section 6.4], where the authors assess the randomness of output from h using the NIST test suite [31] for random number generators. Using a short-hand for the (binary) inputs to h, the sequence

$$
\begin{array}{l}
h_{00\ldots01}\,(00\ldots00) \\
h_{00\ldots01}\,(h_{00\ldots01}\,(00\ldots00)) \\
h_{00\ldots01}\,(h_{00\ldots01}\,(h_{00\ldots01}\,(00\ldots00))) \\
\cdots
\end{array}
$$

is used as input to the test suite, and obtain largely positive results from the various randomness tests; the conclusion is that the primitive meets this aspect of the design criteria.

However, randomness tests use a black-box model of the block cipher: their ability to detect differential vulnerabilities depends strongly on the black-box inputs, i.e., initialisation of the input generation sequence. In [12, Section 4], use of $K = 0 = 00\ldots00$ is explicitly avoided. Had this case been considered in the analysis, it would have been clear that each ciphertext with $M_i = M_j$ for all $0 \leqslant i, j < 16$ will produce outputs where all bytes have the same value. More generally, it can be easily seen that for all $0 \leqslant i, j < 16$ and (M, K) with $M_i = M_j$ and $K_i = K_j$ the ciphertext will have $C_i = C_j$. This is a crucial indicator of a design weakness and would have been detected by the NIST test suite if the inputs had been chosen differently.

However, to the best of our knowledge, there is no feasible strategy to detect all possible differential weaknesses using such a test suite: completely ignoring traditional, white-box cryptanalytic techniques in preference for automated black-box testing does not give an adequate result therefore.

3.4.3 *A broken attack*

In [36] the authors claim to have found a working attack on the Dusart-Traoré primitive. Unlike our attacks, they target the first round of the primitive by iterating over key candidates K_i, K_{i+1} and challenges that differ only in byte M_{i+1}, looking for collisions of the form

$$
\begin{array}{c}
f\Big(S(M_i \oplus K_i), S(M_{i+1} \oplus K_{i+1})\Big) \\
\shortparallel \\
f\Big(S(M_i \oplus K_i), S(M^*_{i+1} \oplus K_{i+1})\Big)
\end{array}
\qquad (4)
$$

where S denotes the S-box and M^*_{i+1} the byte containing the differential.

The first observation to make is that this equation will produce trivial collisions for all key byte candidates K_i, K_{i+1}. Having $M^0_{i+1} = S(M_{i+1} \oplus K_{i+1})$ and $M^{0*}_{i+1} = S(M^*_{i+1} \oplus K_{i+1})$ the trivial collisions occur for

$$
\begin{array}{rcl}
M^0_{i+1}[0] & = & M^{0*}_{i+1}[0] \\
M^0_{i+1}[1] & = & M^{0*}_{i+1}[1] \\
M^0_{i+1}[2] & = & M^{0*}_{i+1}[2] \\
M^0_{i+1}[3] & = & M^{0*}_{i+1}[3] \\
M^0_{i+1}[4] & = & M^{0*}_{i+1}[4] \\
M^0_{i+1}[5] & = & M^{0*}_{i+1}[5] \\
M^0_{i+1}[6] & = & M^{0*}_{i+1}[6]
\end{array}
$$

as one can easily see from the bitwise equations in Section 3.1. The example values presented by the authors of [36] of $K_1 = \texttt{0x92}$, $M_1 = \texttt{0xb7}$ and $M^*_1 = \texttt{0x66}$ produce $M^0_1 = \texttt{0x3f}$ and $M^{0*}_1 = \texttt{0xbf}$ which is just such a trivial collision. In addition to the non-trivial collisions, collisions can also occur for suitable values due to the carry of the integer addition in lines #4 to #7 of the bitwise equations in Section 3.1.

The second, more general observation is that the challenges used in Equation 4 have no relation to the key under attack. Iterating over all possible key byte candidates should thus produce a number of trivial collisions for all key byte candidates. In other words, the attack presented in [36] is no attack at all as it can not distinguish between correct and wrong key candidates. Instead it only highlights a structural property of the Dusart-Traoré primitive. In contrast to this, our structural attack focuses on the last round of the cipher using the ciphertext outputs of the Dusart-Traoré primitive which actually depend on the secret key that is being attacked.

3.4.4 *Potential countermeasures*

A first step to repair the primitive would be to add at least 4 more rounds, and to tweak Equation 3 into

$$
\begin{aligned}
f(x, y) = \Big[&\big[x \oplus ((\texttt{0xFF} \oplus y) \lll 1)\big] + \\
&\big[((\texttt{0xFF} \oplus x) \oplus (y \ggg 1))\big] \lll 4\Big]_{\bmod 256},
\end{aligned}
\qquad (5)
$$

i.e., using rotate operations instead of shifts and omitting the existing reduction modulo 16. The tweaks to Equation 3 will increase the complexity of inverting the f function, while the additional rounds will ensure that a single input difference affects all input bytes of the last round. In addition, one needs to update the secret key to a new random value after every single use in order to prevent attacks that require more than a single (M, C) pair. The additional overhead of the key update makes all claims regarding the light-weight property of the function highly questionable.

However, attack-specific "patching" of the primitive, which the above amounts to, seems fraught with danger; the result certainly provides no strong guarantee of a security level matching that intended. The lack of key schedule and existence of meet-in-the-middle attacks on other reduced round block ciphers (e.g., [11], which targets 8-round AES) seem fundamentally worrying for instance, even before necessary application of standard cryptanalysis (e.g., [29, 4]).

3.4.5 *Comparison of light-weightness wrt. implementation*

Considering that the primitive is broken, one has to ask whether the efficiency advantages are sufficient to warrant any attempt to salvage individual components. To explore this question, we make a rough comparison between the Dusart-Traoré primitive, AES and PRESENT on 8-bit microprocessors; see, e.g., [13] for a more detailed comparison of AES, PRESENT and other light-weight primitives and [17, 24] for even more resource constrained architectures.

The first question that arises is that of implementation strategy:

- A hardware implementations of some light-weight block cipher will typically trade-off efficiency in favour of area by instantiating a single round over which it then iterates. Where this approach is employed, PRESENT has a clear efficiency advantage over both AES and the Dusart-Traoré primitive; this stems from a simpler permutation layer (which requires no gates) and smaller S-box.

- A throughput-focused software implementation will use bit-slicing, although in context this approach is moot due to the small number of blocks processed per authentication round. Even so, PRESENT again enjoys an advantage due to the same reasons above.
- Finally, a vanilla software implementations using look-up tables[3] will typically impose a penalty on PRESENT because of the permutation layer.

For vanilla software implementations it makes sense to compare AES and the Dusart-Traoré primitive in greater detail. Obviously, the two major differences between AES and the Dusart-Traoré primitive are a) the reduced number of rounds and b) the lack of key scheduling. Initially, both are efficiency gains, but since they are also crucial security weaknesses for the Dusart-Traoré primitive, neither is salvageable.

What remains is the permutation layer.

AES The permutation layer consists of the `ShiftRow` operation which can be implemented for free, and the `MixColumn` operation which can be implemented using 3 XOR operations and 2 table look-ups per state byte (for details, see, e.g., [10, Section 4.1.1]).

Dusart-Traoré The f function described in Equation 3 requires 4 XOR operations, 3 shift operations and 1 integer addition per state byte.

Put simply, the bespoke permutation layer of the Dusart-Traoré primitive only becomes competitive if the 2 table look-ups used by AES require more cycles than 5 ALU-type operations.

While the relative efficiency of f depends somewhat on the platform therefore, the more rigorous study of properties pertaining to the AES S-box (or indeed alternative mappings [22] based on the combination of logical and arithmetic operations) arguably still makes it the more attractive choice.

4. CONCLUSIONS

Our results from Sections 3.2 clearly demonstrate that the Dusart-Traoré primitive is insecure for all practical purposes, and should therefore never be used; given the requirement for a light-weight block cipher, we suggest use of a standardised alternative such as PRESENT [5, ISO/IEC 29192-2:2011] with a more robust security analysis. In addition, our efficiency analysis in Section 3.4.5 strongly indicates that any attempt to salvage parts of the Dusart-Traoré primitive for a redesign is unlikely to result in a block cipher design that is both secure and competitive against AES. In addition to our results regarding the Dusart-Traoré primitive we have demonstrated the uselessness of the claimed attack of [36].

From a research perspective, two open questions remain. In some applications of light-weight block ciphers, the criteria for “light-weight” can be low round complexity. Some examples of broken block ciphers such as this with low round complexity exist and cryptanalysis of reduced-round variants is a common tool to establish the security margin of a given block cipher yet the minimum round complexity to achieve specific security levels remains unknown. In addition, the the aforementioned question of automated black box security analysis remains unanswered as well. A final question, that considers efficient design of masked “light-weight” block-ciphers, is already being addressed by current research, see, e.g., [15].

Acknowledgements. The work described in this paper has been supported by EPSRC grants EP/H001689/1 and EP/H049606/1. The authors would like to thank Pierre Dusart for additional information and test vectors provided during private communication.

5. REFERENCES

[1] Satisfiability Suggested Format. `ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/doc/satformat.dvi`. accessed on October 1st, 2012.

[2] F. Armknecht. A linearization attack on the Bluetooth key stream generator. Cryptology ePrint Archive, Report 2002/191, 2002.

[3] G. Bard, N. Courtois, and C. Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers. Cryptology ePrint Archive, Report 2007/024, 2007.

[4] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology (CRYPTO)*, pages 2–21. Springer-Verlag LNCS 537, 1990.

[5] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 450–466. Springer-Verlag LNCS 4727, 2007.

[6] J. Bringer, H. Chabanne, and T. Icart. Cryptanalysis of EC-RAC, a RFID identification protocol. In *Cryptology and Network Security (CANS)*, pages 149–161. Springer-Verlag LNCS 5339, 2008.

[7] N. Courtois. Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt. In *Information Security and Cryptology (ICISC)*, pages 182–199. Springer-Verlag LNCS 2587, 2003.

[8] N. Courtois, G. Bard, and D. Wagner. Algebraic and slide attacks on keeloq. In *Fast Software Encryption (FSE)*, pages 97–115. Springer-Verlag LNCS 5086, 2008.

[3]For RFID tags we consider this a possible scenario if the micro-controller does not have a cache, i.e., if cache timing attacks can be ignored, and if the application profile only requires occasional use of the block cipher on one or two message blocks.

[9] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology (ASIACRYPT)*, pages 267–287. Springer-Verlag LNCS 2501, 2002.

[10] J. Daemen and V. Rijmen. *The Design of Rijndael.* Springer-Verlag Berlin Heidelberg, 2002.

[11] H. Demirci and A.A.Selçuk. A Meet-in-the-Middle Attack on 8-Round AES. In *Fast Software Encryption (FSE)*, pages 116–126. Springer-Verlag LNCS 5086, 2008.

[12] P. Dusart and S. Traoré. Lightweight Authentication Protocol for Low-Cost RFID Tags. In *Information Security Theory and Practice (WISTP)*, pages 129–144. Springer-Verlag LNCS 7886, 2013.

[13] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24(6):522–533, 2007.

[14] J. Fan, J. Hermans, and F. Vercauteren. On the claimed privacy of EC-RAC III. In *Radio Frequency Identification: Security and Privacy Issues (RFIDSec)*, pages 66–74. Springer-Verlag LNCS 6370, 2010.

[15] B. Gérard, V. Grosso, M. Naya-Plasencia, and F.-X. Standaert. Block ciphers that are easier to mask: How far can we go? Cryptology ePrint Archive, Report 2013/369, 2013. To appear in proc. of Cryptographic Hardware and Embedded Systems (CHES) 2013.

[16] J. Guo, T. Peyrin, and A. Poschmann. The PHOTON family of lightweight hash functions. In *Advances in Cryptology (CRYPTO)*, pages 222–239. Springer-Verlag LNCS 6841, 2011.

[17] N. Jacob, S. Saetang, C.-N. Chen, S. Kutzner, S. Ling, and A. Poschmann. Feasibility and practicability of standardized cryptography on 4-bit micro controllers. In *Selected Areas in Cryptography (SAC)*, pages 184–201. Springer-Verlag LNCS 7707, 2013.

[18] P. Jovanovic, M. Kreuzer, and I. Polian. An algebraic fault attack on the led block cipher. Cryptology ePrint Archive, Report 2012/400, 2012.

[19] A. Juels. RFID security and privacy: a research survey. *IEEE Selected Areas in Communications*, 24(2):381–394, 2006.

[20] J. Katz and Y. Lindell. *Introduction To Modern Cryptography.* Chapman & Hall/CRC, 2008.

[21] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F.-X. Standaert. Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 390–407. Springer-Verlag LNCS 7428, 2012.

[22] A. Klimov and A. Shamir. A new class of invertible mappings. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 470–483. Springer-Verlag LNCS 2523, 2002.

[23] M. Knezevic, V. Nikov, and P. Rombouts. Low-latency encryption - is "lightweight = light + wait"? In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 426–446. Springer-Verlag LNCS 7428, 2012.

[24] L. Knudsen, G. Leander, A. Poschmann, and M. Robshaw. PRINTcipher: A Block Cipher for IC-Printing. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 16–32. Springer-Verlag LNCS 6225, 2010.

[25] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi. Security as a new dimension in embedded system design. In *Design Automation Conference (DAC)*, pages 753–760, 2004.

[26] Y. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-cost untraceable authentication protocols for RFID. In *ACM Conference on Wireless Nnetwork Security (WiSec)*, pages 55–64, 2010.

[27] Y. Lee, L. Batina, and I. Verbauwhede. EC-RAC (ECDLP based Randomized Access Control): provably secure RFID authentication protocol. In *IEEE International Conference on RFID*, pages 97–104, 2008.

[28] Y. Lee, L. Batina, and I. Verbauwhede. Untraceable RFID authentication protocols: revision of EC-RAC. In *IEEE International Conference on RFID*, pages 178–185, 2009.

[29] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology (EUROCRYPT)*, pages 386–397. Springer-Verlag LNCS 765, 1993.

[30] M. Mohamed, S. Bulygin, and J. Buchmann. Using sat solving to improve differential fault analysis of trivium. In *Information Security and Assurance*, volume 200 of *Communications in Computer and Information Science*, pages 62–71. Springer, 2011.

[31] NIST. Random Number Generation Toolkit. `http://csrc.nist.gov/groups/ST/toolkit/rng/index.html`, 2013.

[32] A. Poschmann. *Lightweight cryptography: cryptographic engineering for a pervasive world.* PhD thesis, Ruhr-University Bochum, 2009.

[33] N. Potlapally, A. Raghunathan, S. Ravi, N. Jha, and R. Lee. Aiding side-channel attacks on cryptographic software with satisfiability-based analysis. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(4):465–470, 2007.

[34] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady. Security in embedded systems: design challenges. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(3):461–491, 2004.

[35] C. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[36] W. Shao-Hui, X. Fu, C. Dan-wei, and W. Ru-chuan. Security analysis of lightweight authentication protocol from wistp 2013. Cryptology ePrint Archive, Report 2013/411, 2013.

[37] M. Soos. CryptoMiniSat2 v2.9.5. `http://www.msoos.org/cryptominisat2/`. accessed on June 1st, 2013.

[38] T. van Deursen and S. Radomirović. EC-RAC: enriching a capacious RFID attack collection. In *Radio Frequency Identification: Security and Privacy Issues (RFIDSec)*, pages 75–90. Springer-Verlag LNCS 6370, 2010.

[39] S. Vaudenay. On privacy models for RFID. In *Advances in Cryptology (ASIACRYPT)*, pages 68–87. Springer-Verlag LNCS 4833, 2007.

[40] X. Zhao, S. Guo, F. Zhang, T. Wang, Z. Shi, and

K. Ji. Algebraic differential fault attacks on led using a single fault injection. Cryptology ePrint Archive, Report 2012/347, 2012.